Fully Automated Parameter Estimation for Mixtures of Factor Analyzers

John Davey

October 23, 2022

Thesis submitted for the degree of
Master of Philosophy
in
Statistics
at The University of Adelaide
Faculty of Sciences, Engineering, and Technology
School of Mathematical Sciences

THE UNIVERSITY
of ADELAIDE

Contents

Signed Statement

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Signed: .... Date: .....

Abstract

Mixture models are a family of statistical models that can model datasets with underlying sub-population structures effectively. This thesis focuses on one particular mixture model, called the Mixtures of Factor Analyzers (MFA) model [Ghahramani et al., 1997], which is a multivariate clustering model more parsimonious than the well known Gaussian mixture model (GMM). The MFA model has two hyperparameters, g, the number of components, and q, the number of factors per component. When these are assumed to be known in advance, approximate maximum likelihood estimates for the remaining model parameters can be obtained using Expectation Maximisation (EM)-type algorithms [Dempster et al., 1977] [Ghahramani et al., 1997] [McLachlan and Peel, 2000] [Zhao and Yu, 2008].

This work reviews methods for fitting the MFA model in the more realistic case where its two hyperparameters are not known *a priori*. A systematic comparison of seven methods for fitting the MFA model when its hyperparameters are unknown is conducted. The methods are compared based on their ability to infer the two hyperparameters accurately, as well as general model fit, clustering accuracy and the length of time taken to fit the model. The results suggest that a naïve grid search over both hyperparameters performs the best on all of the metrics except for the time taken to fit the models. The Infinite Mixtures of Infinite Factor Analyzers (IMIFA) algorithm [Murphy et al., 2020] also performs well on most of the metrics. However, like the naïve search, IMIFA is also very computationally intensive. The Automatic Mixture of Factor Analyzers (AMFA) algorithm [Wang and Lin, 2020] is a viable alternative when available computation time is limited, as it often performs comparably to the naïve search and IMIFA, but with greatly reduced computation times. To facilitate the comparison, the `R` package `autoMFA` is created, which implements five methods for the automated fitting of the MFA model and is available on the Comprehensive R Archive Network (CRAN).

A limitation of the MFA model is its inability to deal with asymmetrical cluster shapes, which is a consequence of using multivariate Gaussian component densities. The Mixtures of Mean-Variance Mixture of Normal Distribution Factor Analyzers (MMVMNFA) family is proposed as a generalisation of the MFA model, which permits asymmetrical component densities. A new EM-type algorithm for parameter estimation of MMVMNFA models is

developed. Based on its performance in the comparison, the AMFA algorithm is selected and generalised to the MMVMNFA family. Six specific instances of the MMVMNFA family are considered, and the steps for the EM-type algorithm are derived for each. The Julia package FactorMixtures is created, which contains implementations of each of these algorithms. The six instances are tested on two synthetic datasets and two real world datasets, where their superior ability to capture heavy-tailed data and data exhibiting multivariate skewness is demonstrated in comparison to the standard MFA model, which cannot effectively capture either of these properties.

Acknowledgements

My heartfelt thanks to Associate Professor Gary Glonek and Dr Sharon Lee for their invaluable guidance over the course of my candidature, and to my friends and family for their ongoing support.

Dedication

For G. W. J. C, M. H and B. P.

Publications

The following publications are in preparation as a result of this work:

J. Davey, G. Glonek, and S. Lee. The `autoMFA` Package for Automatically Fitting Mixtures of Factor Analyzers Models. *In Preparation*, 2022a.

This paper introduces the `autoMFA` `R` package from Chapter 3.

J. Davey, G. Glonek, and S. Lee. A Survey of Automated Methods for the Mixtures of Factor Analyzers Model. *In Preparation*, 2022b.

This paper summarises the results of the systematic comparison performed in Chapter 4.

J. Davey, G. Glonek, and S. Lee. Modelling Multivariate Data Using the Mixtures of Mean-Variance Mixture of Normal Distribution Factor Analyzers Model Family. *In Preparation*, 2022c.

This paper introduces the MMVMNFA model family and the EM-type algorithm derived for parameter estimation of that family from Chapter 5. It also demonstrates how the MMVMNFA models can achieve better model fits and clustering results compared to the regular MFA model using the examples in Chapter 6.

Acronyms and Abbreviations

Table 1 contains the various acronyms and abbreviations used in this thesis, along with a page reference to their definition and/or description.

Acronym or Abbreviation	Meaning	Page
AECM	Alternating Expectation Conditional Maximisation	19
AFA	Automatic Factor Analysis	109
AIS	Australian Institute of Sport	109
AMFA	Automatic Mixtures of Factor Analyzers	41
AMoFA	Adaptive Mixtures of Factor Analyzers	43
ARD	Automatic Relevance Detection	48
ARI	Adjusted Rand Index	17
BIC	Bayesian Information Criterion	9
BICM	Bayesian Information Criterion Monte (Carlo)	53
CRAN	Comprehensive R Archive Network	8
ECM	Expectation Conditional Maximisation	18
EM	Expectation Maximisation	18
FA	Factor Analysis	5
GMM	Gaussian Mixture Model	13
IMIFA	Infinite Mixtures of Infinite Factor Analyzers	53
MCMC	Markov Chain Monte Carlo	53
MCNFA	Mixtures of Contaminated Normal Factor Analyzers	91
MFA	Mixtures of Factor Analyzers	6
MFA-BS	Mixtures of Factor Analyzers using Birnbaum-Saunders scaling variables	102
MFA-L	Mixtures of Factor Analyzers using Lindley scaling variables	104
MGHFA	Mixtures of Generalised Hyperbolic Factor Analyzers	100
MLE	Maximum Likelihood Estimate	1
MML	Minimum Message Length	43
MMVMNFA	Mixtures of Mean-Variance Mixture of Normal Distribution Factor Analyzers	94
MSLFA	Mixtures of Slash Factor Analyzers	87
MSMNFA	Mixtures of Scale Mixtures of Normal Distributions Factor Analyzers	78
MtFA	Mixtures of t Factor Analyzers	85
MVMN	Mean-Variance Mixture of Normal	94
NI	Normal Independent	77
OMIFA	Overfitted Mixtures of Infinite Factor Analyzers	53
PX-EM	Parameter Expanded Expectation Maximisation	19
RI	Rand Index	16
SMN	Scale Mixtures of Normal Distributions	77
VaR	Value at Risk	7
VBEM	Variational Bayesian Expectation Maximisation	47
VBMFA	Variational Bayesian Mixtures of Factor Analyzers	47

Table 1: The acronyms and abbreviations used in this work, their meanings and the page where they are defined and/or described. The acronyms and abbreviations are sorted in alphabetical order.

Chapter 1

Introduction

Mixture models are statistical models which are able to model datasets with underlying sub-population structures effectively. They are also a powerful unsupervised learning technique which can be applied to classification problems where the predictor variables are all numeric. However, often there are challenging issues with the use of these models in practice. This thesis studies some of the issues related to the model fitting process, in particular, the choice of hyperparameters.

Example 1. To motivate the idea of a mixture model, we will consider the "faithful" dataset from Härdle [1991] which is available in the statistical programming language `R` [R Core Team, 2021]. The faithful dataset measures the length of eruptions of the Old Faithful geyser in Yellowstone National Park in the USA, as well as how long it took for the geyser to erupt again. Both measurements are in minutes. Figure 1.1 is a scatter plot of the dataset. Visually, there appears to be two reasonably well separated sub-populations present in the data.

If we were careless in our analysis of this dataset, we might ignore the presence of these sub-populations. For example, we could try to model the data using a single bivariate Gaussian distribution, with the mean and covariance matrix given by their respective Maximum Likelihood Estimates (MLEs). Figure 1.2 shows, perhaps unsurprisingly, that fitting a bivariate Gaussian distribution will not produce a reasonable model for this data. In fact, this model predicts that the data should be densest in the region between the two sub-populations while in reality, the data is actually very sparse in this region.

Given that in this example we can clearly observe two sub-populations in the data, we will instead model the data using a mixture model. Figure 1.3 shows the contours of a two component bivariate Gaussian mixture model fitted to the faithful dataset[1]. The

[1]More information on how models such as this can be fitted will be provided later.

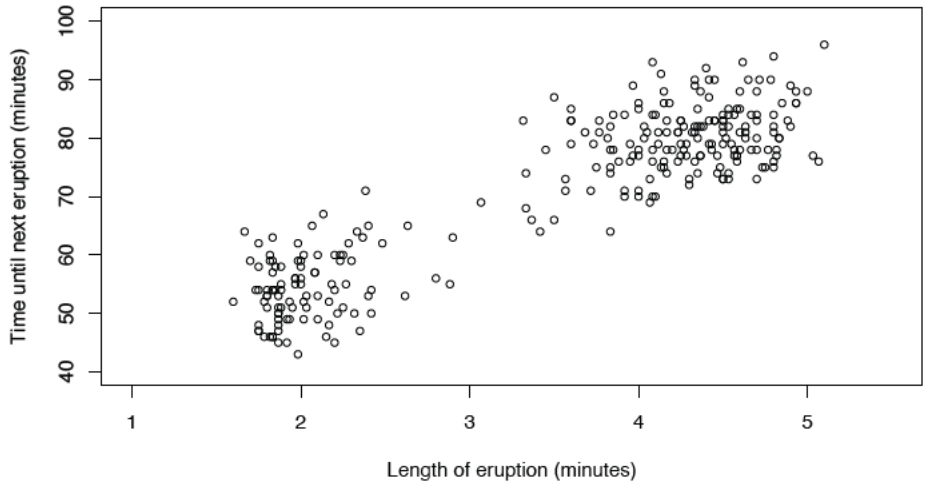


Figure 1.1: The faithful dataset from R.

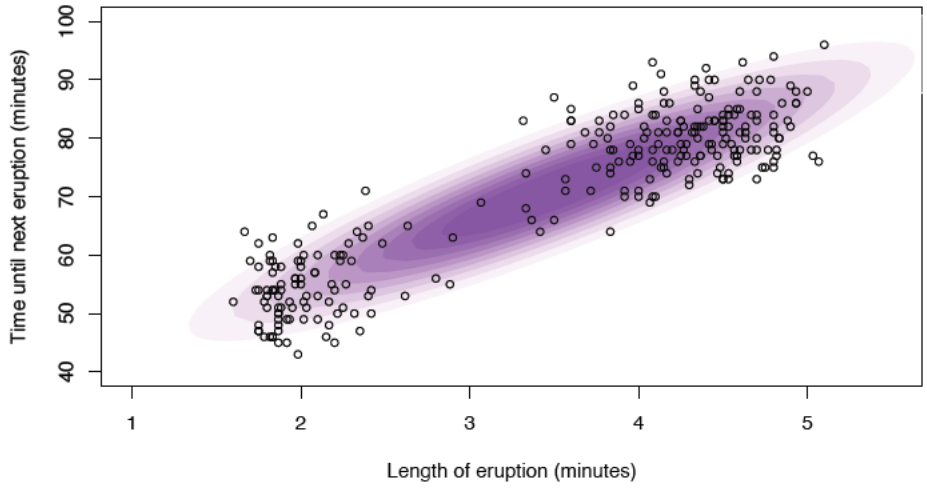


Figure 1.2: The faithful dataset from R, with the contours of the bivariate Gaussian distribution defined by the MLEs for its mean vector and covariance matrix shaded in purple.

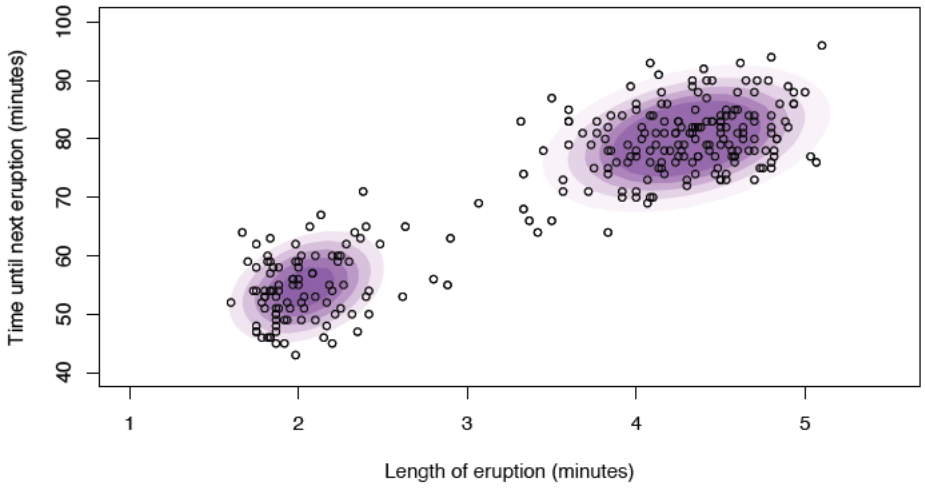


Figure 1.3: The faithful dataset from R, with the contours of a two component bivariate Gaussian mixture model shaded in purple.

name "two component bivariate Gaussian mixture model" is due to the form of its density function, which is

$$f(\mathbf{y}) = \pi\phi_1(\mathbf{y}) + (1-\pi)\phi_2(\mathbf{y})$$

for $\mathbf{y} \in \mathbb{R}^2$, where ϕ_1 and ϕ_2 are bivariate Gaussian density functions (possibly with different mean vectors and/or covariance matrices) and $0 \leq \pi \leq 1$.

Under this model, very little probability density is present between the two sub-populations, while the majority of the probability density lies within the two sub-populations. While this is certainly not a perfect model for this dataset, it clearly captures the structure of the data much better than a single Gaussian distribution, and as a result, illustrates how mixture models can be useful tools for analysing some datasets.

■

1.1 Mixture Models

We say that a real-valued random p-vector $\mathbf{Y}$ follows a mixture distribution if

$$\mathbf{Y} \sim f_i(\mathbf{y}; \boldsymbol{\theta}_i) \text{ with probability } \pi_i$$

for $i = 1, \ldots, g$. Here we are using the slight abuse of notation $\mathbf{Y} \sim f_i(\mathbf{y}; \boldsymbol{\theta}_i)$ to mean that $\mathbf{Y}$ is a random vector whose density function is given by $f_i(\mathbf{y}; \boldsymbol{\theta}_i)$. We call

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 & \ldots & \pi_g \end{bmatrix}$$

the vector of mixing proportions, where $\pi_i \geq 0$ for each i, and $\sum_{i=1}^{g} \pi_i = 1$.

In other words, we say that the random vector $\mathbf{Y}$ follows a mixture distribution if it is generated by first randomly selecting a density function $f_i(\mathbf{y}; \boldsymbol{\theta}_i)$ with probability π_i from a set of g possible density functions $\{f_1, \ldots, f_g\}$ and then randomly drawing $\mathbf{Y}$ from $f_i(\mathbf{y}; \boldsymbol{\theta}_i)$.

A mixture model is a statistical model of the form

$$\mathbf{Y}_j \sim f_i(\mathbf{y}_j; \boldsymbol{\theta}_i) \text{ with probability } \pi_i, \tag{1.1}$$

independently for $j = 1, \ldots, n$ and for $i = 1, \ldots, g$. In other words, under a mixture model, we postulate that each $p \times 1$ data point $\mathbf{Y}_j$ is drawn independently from the same mixture distribution for $j = 1, \ldots, n$.

The formulation of a mixture model in Equation (1.1) is quite intuitive. Recall that in Example 1, the scatter plot Figure 1.1 appears to show two clear sub-populations. The number of points in each sub-population does not appear to be exactly equal. In fact, the number of points which appear to belong to the sub-population corresponding to shorter eruptions with a smaller gap until the next eruption is less than the number belonging to the other sub-population. As a result, a reasonable model for this dataset might suggest that some proportion $p < 0.5$ of the points belong to the sub-population with smaller eruptions and a smaller gap until the next eruption while the remaining points must belong to the other sub-population. The model should also allow for the distribution of points in each sub-population to be different so that we can capture the potentially very different behavior of the two sub-populations. This is what the formulation of a mixture in Equation (1.1) describes.

While in Example 1 there appear to be two clearly separate sub-populations produced by the physical process driving the eruptions, mixture models can also be applied to datasets where there are no separate physical populations. Indeed, mixture models may contain highly overlapping components, which makes them a very flexible family of models that are often used as a tool for density estimation.

It is mathematically convenient to encode the "with probability π_i" statement from Equation (1.1) into the mixture model by instead conditioning on a multinomial random variable, which leads to the following (equivalent) formulation of a mixture model:

$$\begin{aligned} \mathbf{Z}_j &\sim \text{Multinomial}(1, \boldsymbol{\pi}), \\ \mathbf{Y}_j \mid Z_{ij} = 1 &\sim f_i(\mathbf{x}; \boldsymbol{\theta}_i). \end{aligned} \tag{1.2}$$

The formulation in Equation (1.2) is a hierarchical model with two levels. At the first level of the hierarchy, $\mathbf{Z}_j$ is drawn from a multinomial distribution with a single trial with probability vector $\boldsymbol{\pi}$. This means that

$$\mathbf{Z}_j = \begin{bmatrix} Z_{1j} & \dots & Z_{gj} \end{bmatrix}$$

will be a vector of zeros, except for a single entry which will be equal to one. The position of the one indicates which sub-population the data point $\mathbf{Y}_j$ belongs to. Accordingly, $Z_{ij} = 1$ if and only if $\mathbf{Y}_j$ belongs to sub-population i. Also, since $\mathbf{Z}_j$ is assumed to follow a multinomial distribution, it follows that $\Pr(Z_{ij} = 1) = \pi_i$, which makes the equivalence between Equation (1.1) and Equation (1.2) clear.

At the top layer of the hierarchy, conditioned on data point j belonging to sub-population i, the model states that $\mathbf{Y}_j$ is distributed according to the density function for sub-population i.

1.2 The Factor Analysis Model

Perhaps the most well known multivariate probability distribution is the multivariate Gaussian distribution. The p-dimensional multivariate Gaussian distribution is parameterised by the $p \times 1$ mean vector $\boldsymbol{\mu}$ and the $p \times p$ covariance matrix $\boldsymbol{\Sigma}$. A simple model for multivariate data is just to use the multivariate Gaussian distribution, i.e.

$$\mathbf{Y}_j \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

independently for $j = 1, \dots, n$. We call this the multivariate Gaussian model, under which closed-form MLEs exist for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

The simplicity of the multivariate Gaussian model and the existence of MLEs for its parameters are both appealing properties. However, the number of scalar parameters required to define the model is given by[2] $k^*_{\mathrm{G}}(p) = p + p(p+1)/2$. When p is large, this has two major implications. First, we will require at least $k^*_{\mathrm{G}}(p)$ data points to obtain unique parameter estimates for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, which may not always be possible. Second, the estimation of the full $p \times p$ covariance matrix $\boldsymbol{\Sigma}$ will become increasingly computationally expensive, and possibly prohibitively so, even if we have enough data points to guarantee a unique solution.

The factor analysis (FA) model is one way of addressing these problems. This model postulates that $\mathbf{Y}_j$ is formed as the sum of an underlying mean $\boldsymbol{\mu}$, a linear combination of $q < p$ independent underlying factors (after which the model is named) and an additive error vector $\mathbf{e}_j$.

[2] Justification of the expressions for $k^*_{\mathrm{G}}(p)$ and $k^*_{\mathrm{FA}}(p, q)$ is provided in Appendix A.1.

A special case of the model was first used by Spearman [1904] to find an underlying "general intelligence factor" which could explain the intra-student correlations in the results of tests in seemingly unrelated fields. Spearman employed the simplest case of the FA model, i.e. using a single underlying factor ($q = 1$). Thurstone [1931] later formalised and generalised Spearman's work so any integer number of factors $q \geq 1$ could be considered.

The FA model is

$$\mathbf{Y}_j = \boldsymbol{\mu} + \mathbf{B}\mathbf{U}_j + \mathbf{e}_j, \tag{1.3}$$

where

$$\mathbf{U}_j \sim \mathcal{N}_q(\mathbf{0}, \mathbf{I}_q)$$

and

$$\mathbf{e}_j \sim \mathcal{N}_p(\mathbf{0}, \mathbf{D})$$

independently, for $j = 1, \ldots, n$. Here, $\mathbf{I}_q$ denotes the $q \times q$ identity matrix, $\mathbf{Y}_j$ is one of a set of n independent and identically distributed p-dimensional random variables, the $p \times 1$ vector $\boldsymbol{\mu}$ is the underlying mean of the data, the $p \times q$ matrix $\mathbf{B}$ (where $q < p$) is called the factor loading matrix and $\mathbf{D}$ is called the error-variance matrix. The $q \times 1$ vector $\mathbf{U}_j$ are called the factors and the $p \times 1$ vector $\mathbf{e}_j$ are called the errors.

So, as mentioned above, Equation (1.3) shows that under the FA model, each $p \times 1$ observation $\mathbf{Y}_j$ is a sum of the underlying mean $\boldsymbol{\mu}$, a vector composed of linear combinations of the $q < p$ underlying factors specific to each individual, $\mathbf{U}_j$ (with the coefficients of the linear combinations stored in the factor loading matrix $\mathbf{B}$), and an additive error vector $\mathbf{e}_j$.

We can show that under the FA model,

$$\mathbf{Y}_j \sim \mathcal{N}_p(\boldsymbol{\mu}, \mathbf{B}\mathbf{B}^\top + \mathbf{D}).$$

Hence, the FA model is just a multivariate Gaussian model with a restricted covariance structure. The number of scalar parameters required to define the FA model is $k^*_{\text{FA}}(p, q) = pq - q(q-1)/2 + 2p$, which scales linearly with p as opposed to the quadratic scaling of $k^*_{\text{G}}(p)$. This shows that the FA model is one way of avoiding the issues encountered when fitting full-covariance multivariate Gaussian models.

1.3 The Mixtures of Factor Analyzers Model

The Mixtures of Factor Analyzers (MFA) model combines a mixture model with the FA model. More specifically, it is a mixture model where the component densities are all FA models and was initially proposed by Ghahramani et al. [1997]. As a result, it can perform both clustering and local dimension reduction, which can be useful in some

applications. For example, MFA models have been applied to the clustering of cell lines on the basis of gene expressions from microarray experiments [McLachlan et al., 2003], in image processing, where it has been used for face detection [Yang et al., 1999] and in finance, where MFA models have been used to model Value at Risk (VaR) [Ko and Baek, 2018].

The MFA has the hierarchical stochastic formulation

$$\begin{aligned}\mathbf{Z}_j &\sim \text{Multinomial}(1, \boldsymbol{\pi})\\ \mathbf{e}_j \mid Z_{ij} = 1 &\sim \mathcal{N}_p(\mathbf{0}, \mathbf{D}_i)\\ \mathbf{U}_j \mid Z_{ij} = 1 &\sim \mathcal{N}_{q_i}(\mathbf{0}, \mathbf{I}_q)\\ \mathbf{Y}_j \mid \mathbf{U}_j, \mathbf{e}_j, Z_{ij} = 1 &= \boldsymbol{\mu}_i + \mathbf{B}_i\mathbf{U}_j + \mathbf{e}_i,\end{aligned} \tag{1.4}$$

independently, for $j = 1, \ldots, n$ and where $\mathbf{e}_j \mid Z_{ij} = 1$ and $\mathbf{U}_j \mid Z_{ij} = 1$ are also assumed to be independent.

Here, the parameters $\boldsymbol{\mu}_i$, $\mathbf{B}_i$ and $\mathbf{D}_i$ are the mean, factor loading matrix and error-variance matrix of the i^{th} FA model, respectively. The number of latent factors in the i^{th} FA model is given by q_i. For simplicity, unless stated otherwise we will assume a common latent dimensionality q across all of the FA models in the mixture.

1.4 Motivation

In practice, when attempting to fit a mixture model to a dataset, we will generally only observe the $\mathbf{y}_j$'s. Notably, this means that we will either need to specify g *a priori* or infer it while fitting the mixture model. While in Example 1 the bivariate scatter plot showed two well separated sub-populations and hence we chose to use $g = 2$, in general the "best" choice of g may not be clear following visualisation of the data alone. As mentioned earlier, it is also possible to fit mixture models to datasets where no clear sub-population structure is present. In such cases, it is necessary to infer g via some other means.

In a similar manner, when we attempt to fit an FA model to a dataset, we would also generally only observe the $\mathbf{y}_j$'s. As a result, we either need to specify q *a priori*[3] or infer it during the model fitting process.

Since the MFA model is a mixture model with component densities that all follow the FA model, to fit an MFA model we will, in general, need to infer both g and q. Several authors have suggested methods for determining these two hyperparameters automatically.

Even though several methods for automatically inferring g and q in the MFA model have been proposed, the degree to which implementations of these methods are available varies.

[3]Such as Spearman using $q = 1$ in an attempt to determine the "general intelligence factor" of students.

Some of the methods have implementations which can be found relatively easily, while others have no publicly available implementations at all. Additionally, of the publicly available implementations, they are not all implemented in the the same language, with some being implemented in R and others in MATLAB [MATLAB, 2022].

This thesis fills three gaps in the existing literature. First, it provides a common framework through which a number of existing methods for automatically fitting the MFA model can be accessed in the form of the R package `autoMFA`, which we have produced and made available on the Comprehensive R Archive Network (CRAN). Second, it performs a systematic comparison of seven different methods for fitting the MFA model in the case where its two hyperparameters are unknown. The methods are used to fit MFA models to 1,200 synthetically generated datasets and compared based on their ability to infer the hyperparameters of the MFA model correctly, as well as clustering accuracy, general model fit and the amount of time they took to fit the models. Finally, it proposes the Mixtures of Mean-Variance Mixture of Normal Distribution Factor Analyzers (MMVMNFA) model family as a generalisation of the MFA model which can also model multivariate skewness. A new Expectation Maximisation (EM)-type algorithm is derived for the MMVMNFA model family, and six specific instances are given, along with all of the steps needed for the EM-type algorithms in each case. In addition, it generalises one of the methods from the systematic comparison so that the number of factors, q, can be inferred automatically for MMVMNFA models. Implementations of the six examples can be found in our Julia [Bezanson et al., 2017] package FactorMixtures[4], which we use to test the examples of MMVMNFA models on two synthetic datasets and two real world datasets.

1.5 Aims and Objectives

The aims and objectives of this thesis are as follows:

I Produce a publicly available R package which contains implementations of automatic MFA model fitting methods for which no pre-existing R implementation is available.

II Perform a systematic comparison of the current methods for automatically fitting the MFA model.

III Investigate extensions of the MFA model and its model fitting procedures.

1.6 Outline of Thesis

In Chapter 2, we provide background information on methods for fitting the MFA model and some of the techniques which we will use in the systematic comparison. In Chap-

[4]Available at https://github.com/john-c-davey/FactorMixtures

ter 3, we provide a literature review of the existing methods for automatically determining the hyperparameters of the MFA model. Each of the methods discussed in this chapter is available in the `R` package `autoMFA` which we developed to facilitate the systematic comparison. Then, in Chapter 4, we perform a systematic comparison of the existing methods. They are assessed in terms of the average time taken to fit each model, their average Adjusted Rand Index[5] (compared to the true sub-population structure which generated the data sets), the proportion of the time where they correctly inferred the number of components (g) and the number of factors (q) and the average Bayesian Information Criterion (BIC) [Schwarz, 1978] of each model. In Chapter 5 we generalise a parameter estimation technique which was initially proposed for the MFA model to a much broader family of models which maintains the parsimonious mixture model formulation of the MFA model whilst also having the ability to capture heavy-tails and multivariate skewness. We call this the Mixtures of Mean-Variance Mixture of Normal Distribution Factor Analyzers (MMVMNFA) family. We also generalise one of the methods from Chapter 3 to the MMVMNFA family so that the number of factors, q, can be automatically inferred. Six instances of the MMVMNFA family are provided, with a complete EM-type algorithm for parameter estimation given in each case. Finally, in Chapter 6 we compare the performance of the six MMVMNFA models on two synthetic datasets and two real world datasets.

[5]This quantity will be defined in Chapter 2.

Chapter 2

Background Information

In this chapter, we will provide the background information required to cover the content contained in the remainder of this thesis. We will begin by revisiting mixture models, the FA model and then the MFA model in more detail. We will also introduce the Adjusted Rand Index as a method for comparing the similarity of two partitions. Then, we will introduce the EM algorithm as the most common parameter estimation approach for the MFA model in the case where g and q are assumed to be known. Finally, we will consider two existing EM-type algorithms for estimating the parameters of the MFA model.

2.1 Mixture Models

Using the law of total probability, it follows from Equation (1.2) that the density of $\mathbf{y}_j$ can be expressed as

$$f(\mathbf{y}_j; \boldsymbol{\Theta}) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}_j; \boldsymbol{\theta}_i), \tag{2.1}$$

where $\boldsymbol{\Theta} := (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_g)$[1] is the set of model parameters. From this, we can observe that the density function of a mixture model is just a weighted sum of the density functions of the mixture components, with weights given by the mixing proportions.

Strictly speaking, the number of components, g, is also a model parameter. However, since both the number of mixing proportions and the number of component-density-function-specific parameter vectors both depend on g, we choose not to include it in $\boldsymbol{\Theta}$ for simplicity. However, g is not generally known *a priori* and will need to be estimated from the data.

[1] Note that the condition that $\sum_i \pi_i = 1$ means that π_g can be deduced from $\pi_1, \ldots, \pi_{g-1}$.

The indicator vectors are also not typically observed, which means that they need to be inferred should we wish to examine the sub-population structure of a dataset using a mixture model. Finally, to fit a mixture model, we also need to choose a form for the component densities $f_i(\mathbf{y}_j; \boldsymbol{\theta}_i)$ *a priori*.

The identifiability of a statistical model is a very important property that impacts on how well we can recover the model parameters from the observed data. The following definition, due to McLachlan and Basford [1988], describes when identifiability will hold for mixture models.

Definition 2.1.1. Mixture model identifiability [McLachlan and Basford, 1988]

Let $f(\mathbf{y}; \boldsymbol{\Theta})$ define a class of finite mixture densities according to Equation (2.1). Such a class of finite mixtures is said to be identifiable for $\boldsymbol{\Theta} \in \boldsymbol{\Omega}$ if for any two members

$$f(\mathbf{y}; \boldsymbol{\Theta}) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}; \boldsymbol{\theta}_i)$$

and

$$f(\mathbf{y}; \boldsymbol{\Theta}^*) = \sum_{i=1}^{g^*} \pi_i f_i(\mathbf{y}; \boldsymbol{\theta}_i^*)$$

then

$$f(\mathbf{y}; \boldsymbol{\Theta}) \equiv f(\mathbf{y}; \boldsymbol{\Theta}^*)$$

if and only if $g = g^*$ and we can permute the component labels so that

$$\pi_i = \pi_i^* \text{ and } f_i(\mathbf{y}; \boldsymbol{\theta}) \equiv f_i(\mathbf{y}; \boldsymbol{\theta}^*)$$

for $i = 1, \ldots, g$. Here, $\equiv$ implies equality of the densities for almost all $\mathbf{y}$ relative to the underlying measure on $\mathbb{R}^p$ appropriate for $f(\mathbf{y}; \boldsymbol{\Theta})$.

McLachlan and Basford [1988] also explain that for mixture models where all of the component densities $f_i(\mathbf{y}; \boldsymbol{\theta}_i)$ belong to the same parametric family (which is true of all of the mixture models considered in this thesis), then $f(\mathbf{y}; \boldsymbol{\Theta})$ is invariant under the $g!$ permutations of the component labels in $\boldsymbol{\Theta}$. In practice, however, this is of little concern since if we use a maximum likelihood estimation technique, we can impose a suitable constraint on the resulting parameter estimate, such as

$$\pi_1 \geq \pi_2 \cdots \geq \pi_g,$$

as used by Aitkin and Rubin [1985].

We will now consider a simple example of a mixture model.

Example 2. Perhaps the best known mixture model is the Gaussian mixture model (GMM), defined as

$$\begin{aligned} \mathbf{Z}_j &\sim \text{Multinomial}(1, \boldsymbol{\pi}), \\ \mathbf{Y}_j \mid Z_{ij} = 1 &\sim \mathcal{N}_p(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \end{aligned} \tag{2.2}$$

independently for $j = 1, \ldots, n$ and $i = 1, \ldots, g$. ■

As mentioned in Example 1, Figure 1.3 shows the contours of a two component bivariate GMM. The GMM has seen extensive use in the field of cluster analysis, where the goal of a model is to try and identify groups (or clusters) of data points based on some measure of similarity. Using the faithful data from Example 1, we might expect that a cluster analysis model would be able to identify those points with shorter eruptions and shorter inter-eruption times as belonging to one group and the remaining points with longer eruptions and longer inter-eruption times as belonging to another group.

There are two types of cluster analysis models: hard classifiers and soft classifiers. Hard classifiers assign each data point a single label indicating which group it has been assigned to. Soft classifiers instead assign to each data point a vector containing the probability that the data point belongs to each group.

Mixture models are an example of a soft classifier because, as shown in Section 2.6, the most common method for fitting mixture models involves estimating the posterior probability that each data point belongs to each component of the mixture.

Mixture models have also seen usage outside of cluster analysis. For example, Karl Pearson used a two-component GMM to estimate a bimodal empirical density function as early as 1894 [Pearson, 1894].

2.2 The FA Model

As we mentioned in the introduction, under the FA model, the $q \times 1$ vectors $\mathbf{U}_j$ are called the factors or the factor vectors and the $p \times 1$ vectors $\mathbf{e}_j$ are called the errors, where $q < p$. The $p \times 1$ vector $\boldsymbol{\mu}$ is the underlying mean vector of the data[2]. The $p \times q$ matrix $\mathbf{B}$ is the matrix of factor loadings (the factor loading matrix) and the $p \times p$ matrix $\mathbf{D}$ is a diagonal matrix describing the magnitude of the errors $\mathbf{e}_j$ (the error-variance matrix). Note that the error-variance matrix $\mathbf{D}$ being constrained to be diagonal enforces the condition that each element of $\mathbf{Y}_j$ is independent from the other elements of $\mathbf{Y}_j$ when conditioned on the factors. In other words, the only dependence between the elements of $\mathbf{Y}_j$ is captured by the corresponding vector of factors $\mathbf{U}_j$.

[2]For simplicity, we often assume that the data has been centered before fitting the FA model so that $\boldsymbol{\mu} = \mathbf{0}$. This simplifies the log-likelihood of the model, which in turn simplifies the model fitting process.

Since by assumption $\mathbf{U}_j$ and $\mathbf{e}_j$ are independent, it follows from Equation (1.3) that

$$\mathbf{Y}_j \sim \mathcal{N}_p(\boldsymbol{\mu}, \mathbf{B}\mathbf{B}^\top + \mathbf{D}), \tag{2.3}$$

so

$$f(\mathbf{y}_j; \boldsymbol{\mu}, \mathbf{B}, \mathbf{D}) = \phi_p(\mathbf{y}_j; \boldsymbol{\mu}, \mathbf{B}\mathbf{B}^\top + \mathbf{D}), \tag{2.4}$$

where ϕ_p represents the p-dimensional Gaussian density function.

Equation (2.3) shows that the FA model is just a p-dimensional Gaussian model for the data with a restricted covariance matrix. While the stochastic formulation of Equation (2.3) is especially useful if we are seeking an explanatory model[3], the restricted-covariance Gaussian model interpretation is equally valid.

To maintain identifiability under the FA model, Ledermann [1937] showed that the number of factors, q, are required to obey the so called Ledermann bound,

$$q \leq p + \frac{1 - \sqrt{1 + 8p}}{2}. \tag{2.5}$$

This bound ensures that the number of parameters required to fit the FA model is less than the number required to fit a full-covariance p-dimensional Gaussian model to the data. The FA model can, therefore, be considered a dimension reduction technique so long as Equation (2.5) is satisfied.

Even when the Ledermann bound is satisfied, the FA model suffers from an identifiability issue, as the distribution of $\mathbf{Y}$ is invariant under orthogonal transformations of the factor loading matrix $\mathbf{B}$. To see this, for any orthogonal $q \times q$ matrix $\mathbf{V}$, we can write Equation (1.3) as

$$\begin{aligned}\mathbf{Y}_j &= \boldsymbol{\mu} + \mathbf{B}\mathbf{U}_j + \mathbf{e}_j \\ &= \boldsymbol{\mu} + \mathbf{B}\mathbf{V}\mathbf{V}^\top\mathbf{U}_j + \mathbf{e}_j \\ &= \boldsymbol{\mu} + \mathbf{B}'\mathbf{U}'_j + \mathbf{e}_j,\end{aligned}$$

where $\mathbf{B}' := \mathbf{B}\mathbf{V}$ and $\mathbf{U}'_j := \mathbf{V}^\top\mathbf{U}_j$.

Since $\mathbf{U}'_j$ is a linear combination of Gaussian random variables it must also have a Gaussian distribution, and using the usual formulæ,

$$\mathbb{E}\left[\mathbf{U}'_j\right] = \mathbf{V}^\top\mathbf{0} = \mathbf{0},$$

and

$$\mathrm{Cov}\left(\mathbf{U}'_j\right) = \mathbf{V}^T\,\mathrm{Cov}\left(\mathbf{U}_j\right)\mathbf{V} = \mathbf{V}^T\mathbf{V} = \mathbf{I}_q.$$

[3] Like Spearman, who wanted to use the underlying factors to measure the "general intelligence factor" of students.

Hence,

$$\mathbf{U}_j' \sim \mathcal{N}_q(\mathbf{0}, \mathbf{I}_q)$$

which means that

$$\mathbf{Y}_j = \boldsymbol{\mu} + \mathbf{B}'\mathbf{U}_j' + \mathbf{e}_j$$

also satisfies the requirements of the FA model.

Any $q \times q$ orthogonal matrix $\mathbf{V}$ will have $\frac{1}{2}q(q-1)$ free parameters, so to achieve identifiability, we need to apply $\frac{1}{2}q(q-1)$ constraints to $\mathbf{B}$. We can see this by noting that the number of free parameters in the matrix $\mathbf{V}$ is the same as the number of free parameters in an orthonormal basis of $\mathbb{R}^q$. For such a basis, we have $q-1$ parameters in the first vector in the basis (one is lost due to the normalisation constraint). For the second basis vector, we have $q-2$ free parameters because of normalisation and the constraint that it is orthogonal to the first basis vector. For the third basis vector, we have $q-3$ free parameters due to one normalisation constraint and two orthogonality constraints, and so on. So in total we have $\sum_{i=1}^{q-1} i = \frac{1}{2}q(q-1)$ free parameters.

There are many possible ways to apply these constrains to the factor loadings. One method which we will make use of is Varimax rotation [Kaiser, 1958], which applies a particular orthogonal rotation to factor loading matrices and has an existing implementation in `R`.

2.3 The MFA Model

Using the law of total probability, the density of the MFA model is given by

$$\begin{aligned} f(\mathbf{y}_j; \boldsymbol{\Theta}) &= \sum_{i=1}^{g} f(\mathbf{y}_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) \Pr(Z_{ij} = 1; \boldsymbol{\Theta}) \\ &= \sum_{i=1}^{g} \pi_i \phi_p(\mathbf{y}_j; \boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i). \end{aligned} \tag{2.6}$$

where $\boldsymbol{\Theta} = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_g)$ and $\boldsymbol{\theta}_i = (\boldsymbol{\mu}_i, \mathbf{B}_i, \mathbf{D}_i)$, for $i = 1, \ldots, g$.

As the MFA model is a probabilistic mixture of FA models, both of the identifiability issues from the FA model also apply to the MFA model. Accordingly, to achieve identifiability, the number of factors per component, q, must satisfy the Ledermann bound and $q(q-1)/2$ constraints must be applied to each of the loading matrices, leading to a total of $gq(q-1)/2$ constraints.

Let

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1^\top & \cdots & \mathbf{y}_j^\top \end{bmatrix}^\top.$$

Under the MFA model, the log-likelihood of $\mathbf{y}$ will be

$$\ell(\boldsymbol{\Theta} \mid \mathbf{y}) = \log \prod_{j=1}^{n} f(\mathbf{y}_j \mid \boldsymbol{\Theta})$$

Using Equation (2.6), it follows that

$$\ell(\boldsymbol{\Theta} \mid \mathbf{y}) = \sum_{j=1}^{n} \log \left(\sum_{i=1}^{g} \pi_i \phi_p(\mathbf{y}_j; \boldsymbol{\mu}_i, \mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i) \right). \tag{2.7}$$

2.4 The Adjusted Rand Index

Suppose now that we have a set of data $\{\mathbf{Y}_j\}$, where $j = 1, \ldots, n$ and two candidate partitions of the dataset indices, $\mathbb{P}_1 = \{p_{1,1}, p_{1,2}, \ldots, p_{1,\eta_1}\}$ and $\mathbb{P}_2 = \{p_{2,1}, p_{2,2}, \ldots, p_{2,\eta_2}\}$. Here, $p_{i,k} \cap p_{i,l} = \emptyset$ for each $i \in \{1, 2\}$ and each $k, l \in \{1, \ldots, \eta_i\}$ where $k \neq l$. We also assume that

$$\bigcup_{k=1}^{\eta_i} p_{i,k} = \{1, 2, \ldots, n\}$$

for $i \in \{1, 2\}$.

The partitions give us a way of representing potential sub-population structures. So, in Chapter 4, where we want to assess the quality of the MFA models being fitted, we can simulate datasets using a mixture model. By doing so, we have access to the "true" sub-population structure of each dataset, which we can represent using a partition. Then, each time we fit a model to these datasets, we will also obtain an inferred sub-population structure which gives us a second "inferred" partition. We can then compare the two partitions to measure the model's ability at inferring the sub-population structure of the dataset. However, the method we might use to go about comparing the two partitions is not immediately obvious.

Consider Table 2.1, a contingency table for the partitions $\mathbb{P}_1$ and $\mathbb{P}_2$. Here, we define $n_{i,k}$ to be the number of objects (in our case data point indices) common between subsets $p_{2,i}$ and $p_{1,k}$, for $i \in \{1, 2, \ldots, \eta_2\}$ and $k \in \{1, 2, \ldots, \eta_1\}$. Also, define $n_{\cdot,k} := \sum_{i=1}^{\eta_2} n_{i,k}$ and $n_{i,\cdot} := \sum_{k=1}^{\eta_1} n_{i,k}$.

The Rand Index (RI), proposed in Rand [1971], is defined as

$$\mathrm{RI}(\mathbb{P}_1, \mathbb{P}_2) := \frac{\binom{n}{2} + \sum_{i=1}^{\eta_2} \sum_{k=1}^{\eta_1} n_{ik}^2 - \frac{1}{2}\left\{\sum_{i=1}^{\eta_2} n_{i,\cdot}^2 + \sum_{k=1}^{\eta_1} n_{\cdot,k}^2\right\}}{\binom{n}{2}},$$

and is one possible method for measuring the similarity of the two partitions of the data point indices.

		Partition 1				
	Class	$p_{1,1}$	$p_{1,2}$	$\dots$	p_{1,η_1}	Sums
Partition 2	$p_{2,1}$	$n_{1,1}$	$n_{1,2}$	$\dots$	n_{1,η_1}	$n_{1,\cdot}$
	$p_{2,2}$	$n_{2,1}$	$n_{2,2}$	$\dots$	n_{2,η_1}	$n_{2,\cdot}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
	p_{2,η_2}	$n_{\eta_2,1}$	$n_{\eta_2,2}$	$\dots$	n_{η_2,η_1}	$n_{\eta_2,\cdot}$
	Sums	$n_{\cdot,1}$	$n_{\cdot,2}$		$n_{\cdot,\eta_1}$	$n_{\cdot,\cdot}$

Table 2.1: A contingency table for partitions $\mathbb{P}_1$ and $\mathbb{P}_2$.

The RI of two partitions will always lie between zero and one. Values close to zero indicate little agreement between the partitions while values close to one indicate that the partitions are similar. It is also worth noting that the number of subsets making up each partition does not need to be the same when calculating their RI. That is, we do not require $\eta_1 = \eta_2$. To see why this is desirable, consider comparing the partitions

$$\mathbb{P}_1 = \{\{1,2,3\},\{4,5,6\}\} \text{ and } \mathbb{P}_2 = \{\{1,2,3\},\{4,5\},\{6\}\}.$$

These two partitions appear to be very similar since the only difference is data point index 6 belonging to its own subset in $\mathbb{P}_2$. Thankfully, the Rand Index can still be used to compare these two datasets, and, as might be expected, produces the relatively high value of $0.8\overline{6}$ in this example.

The Rand Index takes the value one if the two partitions are equivalent, which is to say that it will take the one if and only if $\mathbb{P}_1 = \mathbb{P}_2$.

On the other hand, the RI will only take the value zero if the two partitions have nothing in common. More formally, for every possible pair of data point indices (up to reordering), $\mathbb{P}_1$ and $\mathbb{P}_2$ cannot both assign the pair to be in the same partition and they also cannot both assign the pair to be in different positions. This is not a trivial condition to meet. In fact, Hubert and Arabie [1985] showed that if the partitions $\mathbb{P}_1$ and $\mathbb{P}_2$ are picked at random from the generalized hypergeometric distribution, then its expected value is

$$\mathbb{E}\left[\mathrm{RI}(\mathbb{P}_1,\mathbb{P}_2)\right] = 1 + 2\sum_{i=1}^{\eta_2}\binom{n_{i,\cdot}}{2}\sum_{k=1}^{\eta_1}\binom{n_{\cdot,k}}{2}\Big/\binom{n}{2}^2 - \left[\sum_{i=1}^{\eta_2}\binom{n_{i,\cdot}}{2} + \sum_{k=1}^{\eta_1}\binom{n_{\cdot,k}}{2}\right]\Big/\binom{n}{2}.$$

That is, the RI will not, on average, take the value zero even if the partitions are randomly generated.

As a result, we will use a corrected-for-chance variation of the RI called the Adjusted Rand Index (ARI), proposed by Hubert and Arabie [1985]. It is defined as

$$\mathrm{ARI}(\mathbb{P}_1,\mathbb{P}_2) = \frac{\mathrm{RI}(\mathbb{P}_1,\mathbb{P}_2) - \mathbb{E}\left[\mathrm{RI}(\mathbb{P}_1,\mathbb{P}_2)\right]}{1 - \mathbb{E}\left[\mathrm{RI}(\mathbb{P}_1,\mathbb{P}_2)\right]}.$$

Hubert and Arabie [1985] also show that

$$\text{ARI}(\mathbb{P}_1, \mathbb{P}_2) := \frac{\sum_{i=1}^{\eta_2}\sum_{k=1}^{\eta_1}\binom{n_{ik}}{2} - \sum_{i=1}^{\eta_2}\binom{n_{i,\cdot}}{2}\sum_{k=1}^{\eta_1}\binom{n_{\cdot,k}}{2}/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i=1}^{\eta_2}\binom{n_{i,\cdot}}{2} + \sum_{k=1}^{\eta_1}\binom{n_{\cdot,k}}{2}\right] - \sum_{i=1}^{\eta_2}\binom{n_{i,\cdot}}{2}\sum_{k=1}^{\eta_1}\binom{n_{\cdot,k}}{2}/\binom{n}{2}}.$$

Unlike for the RI, if the data is randomly assigned into partitions in the manner described above, its ARI has an expected value of zero. It is still bounded above by one, but unlike the RI, the ARI can also take values less than zero if the RI is less than its expected value. The ARI is bounded below by minus one.

We will use the ARI to compare the sub-population structures inferred by the MFA fitting procedures with the true sub-population structure that generated the data while performing the systematic comparison of methods in Chapter 4.

2.5 Parameter Estimation for the MFA Model

For now, suppose that the two hyperparameters of the MFA model, g and q, are known. Under this assumption, consider the problem of performing maximum likelihood estimation on Equation (2.7). The standard approach to obtaining an estimate for a particular parameter would proceed by taking the derivative of Equation (2.7) with respect to the parameter, setting the resulting expression equal to zero and then solving this equation for the parameter.

For illustration purposes, suppose we want to find a MLE for the $\boldsymbol{\mu}_i$'s. From Equation (2.7), we find that

$$\begin{aligned}\frac{\partial \ell}{\partial \boldsymbol{\mu}_k} &= \frac{\partial}{\partial \boldsymbol{\mu}_k}\left[\sum_{j=1}^{n}\log\left(\sum_{i=1}^{g}\pi_i\phi_p(\mathbf{y}_j;\boldsymbol{\mu}_i,\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i)\right)\right] \\ &= \sum_{j=1}^{n}\left[\frac{\frac{\partial}{\partial\boldsymbol{\mu}_k}\left\{\pi_k\phi_p(\mathbf{y}_j;\boldsymbol{\mu}_k,\mathbf{B}_k\mathbf{B}_k^\top+\mathbf{D}_k)\right\}}{\sum_{i=1}^{g}\pi_i\phi_p(\mathbf{y}_j;\boldsymbol{\mu}_i,\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i)}\right].\end{aligned} \tag{2.8}$$

In fact, Equation (2.8) is sufficiently complicated that a closed form expression for the MLE of $\boldsymbol{\mu}_i$ is not available. The same is true of $\mathbf{B}_i$ and $\mathbf{D}_i$, meaning that using a direct approach to obtain MLEs is not feasible for the MFA model.

While we cannot derive analytic expressions for the maximum likelihood estimators of the MFA model, we can still obtain approximate maximum likelihood solutions via iterative methods. The most common method for producing approximate MLEs for the parameters of mixture models is the Expectation Maximisation (EM) algorithm of Dempster et al. [1977] and generalisations thereof, such as the Expectation Conditional Maximisation (ECM) algorithm of Meng and Rubin [1993], the Alternating Expectation Conditional

Maximisation (AECM) algorithm of Meng and van Dyk [1997] and the Parameter Expanded Expectation Maximisation (PX-EM) algorithm of Liu et al. [1998]. Accordingly, we will discuss the EM algorithm and its extensions (which we will refer to as EM-type algorithms) in more detail.

2.5.1 The EM Algorithm

The EM algorithm is a parameter estimation algorithm which is applicable to a broad range of problems [Dempster et al., 1977]. At a high level, the idea of the EM algorithm is to introduce some new (and typically unobserved) latent variables such that the joint likelihood of the observed data and the latent variables is of a simpler form than that of the observed data alone. We call this joint log-likelihood the complete-data log-likelihood.

The EM algorithm is iterative. We use k to track the number of iterations which have been completed, so $\boldsymbol{\Theta}^{(k)}$ is the estimate of $\boldsymbol{\Theta}$ given by the k^{th} iteration of the EM algorithm. Each iteration of the EM algorithm is composed of two steps. In the first step, which is usually called the E-step, we find the sufficient statistics of the complete-data log-likelihood, except that wherever functions of the latent variables appear, we replace them with their conditional expectation given the observed data and the current estimates of the model parameters. Then, in the second step, which is generally called the M-step, we find maximum likelihood estimates for the model parameters using the complete-data log-likelihood, in terms of the modified sufficient statistics from the E-step.

The original motivation of the EM algorithm was to develop a parameter estimation technique for missing data problems. In fact, Dempster et al. [1977] provides several examples of applying the EM algorithm to missing data problems. In some cases, data is explicitly missing from the design matrix, so the missing data is used as the latent variables. They also discuss the FA model, where the unobserved factors $\mathbf{U}_j$ constitute a natural choice of latent variable. However, the latent variables need not be explicitly missing or unobserved. The EM algorithm can be applied in any situation where augmenting the observed data with a set of latent variables makes a previously intractable likelihood function tractable.

More formally, suppose we have a model with parameter vector $\boldsymbol{\Theta}$. Let $\mathbf{y}$ be the observed data to which we wish to fit the model and let $\mathbf{w}$ be the set of latent variables. We define an observation of the complete data to be $\mathbf{x}_i = (\mathbf{y}_i, \mathbf{w}_i)$, the concatenation of the observed data and the latent variables. We also define the so called Q-function as

$$Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(k)}) = \mathbb{E}\left[\log f(\mathbf{x}; \boldsymbol{\Theta}) \mid \mathbf{y}; \boldsymbol{\Theta}^{(k)}\right], \tag{2.9}$$

where $f(\mathbf{x}; \boldsymbol{\Theta})$ is the complete-data likelihood function of the model. This means that the Q-function is just the conditional expectation of the complete-data log-likelihood given the observed data.

Calculating the Q-function is the first step of each iteration of the EM algorithm and is known as the expectation step or E-step. The second step in each iteration of the EM algorithm is the maximisation step or M-step, so called because in this step, we maximise the Q-function, Equation (2.9), with respect to $\boldsymbol{\Theta}$.

Suppose now that we have our observed data $\mathbf{y}$ and that we have chosen an appropriate set of latent variables $\mathbf{w}$ such that we can evaluate the complete-data likelihood, that is, we can compute $f(\mathbf{x}; \boldsymbol{\Theta})$. Under this pretense, Algorithm 2.1 defines the EM algorithm.

Algorithm 2.1: The EM Algorithm from Dempster et al. [1977]

Input: A set of observed data $\mathbf{y}$, an initial estimate of the parameters $\boldsymbol{\Theta}^{(0)}$, a convergence criterion and a maximum number of iterations k_{max}

Result: Approximate maximum likelihood estimates for the vector of parameters $\boldsymbol{\Theta}$

Set $k = 0$;
while *Convergence criterion not satisfied and* $k < k_{max}$ **do**
  The E-Step: Compute $Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(k)})$;
  The M-Step: Set $\boldsymbol{\Theta}^{(k+1)}$ to be the $\boldsymbol{\Theta}$ that maximizes $Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(k)})$ from the previous step;
  if *Convergence criterion satisfied* **then**
    Return $\boldsymbol{\Theta}^{(k+1)}$;
  else
    Set $k = k + 1$;
  end
end
Return $\boldsymbol{\Theta}^{(k_{\max})}$;

Algorithm 2.1 contains three objects which we have not yet discussed; a convergence criterion, the maximum number of iterations and an initial estimate of the parameters. Like most iterative methods, the EM algorithm requires a convergence criterion. That is, it needs to have a rule with which it can determine whether or not the sequence of parameter estimates which it has produced has converged to a final set of values. There are many possible choices for convergence criteria. Two common choices for EM algorithms are the absolute difference in log-likelihood,

$$|\ell(\boldsymbol{\Theta}^{(k+1)} \mid \mathbf{y}) - \ell(\boldsymbol{\Theta}^{(k)} \mid \mathbf{y})| < \varepsilon, \tag{2.10}$$

and the relative absolute difference in log-likelihood

$$\frac{|\ell(\boldsymbol{\Theta}^{(k+1)} \mid \mathbf{y}) - \ell(\boldsymbol{\Theta}^{(k)} \mid \mathbf{y})|}{|\ell(\boldsymbol{\Theta}^{(k)} \mid \mathbf{y})|} < \varepsilon, \tag{2.11}$$

for some pre-specified $\varepsilon > 0$. We usually also specify a maximum number of iterations because if the convergence criterion is strict enough, it may take a very large number of EM iterations before the convergence criterion is met. This maximum number of iterations acts as a second termination condition. If the maximum number of iterations is met, then the algorithm will terminate, even if the convergence criterion is not yet satisfied.

Dempster et al. [1977] showed that the sequence of observed-data log-likelihoods produced by the EM Algorithm are guaranteed to be non-decreasing. Whilst this is an extremely desirable property, the increase in log-likelihood from one iteration to the next may be very small. Consequently, the convergence of the algorithm may be slow. Dempster et al. [1977] note that the rate of convergence tends to decrease as the number of latent variables being used increases. As a result, minimising the number of latent variables which are introduced may be desirable to increase the efficiency of the algorithm. However, introducing less latent variables may lead to more complicated complete-data log-likelihoods.

It may happen that the log-likelihood function $\ell(\boldsymbol{\Theta} \mid \mathbf{y})$ is multimodal. In this case, the algorithm will be sensitive to the initial parameter estimates which are supplied. If the model is initialised close to a local maximum of $\ell(\boldsymbol{\Theta} \mid \mathbf{y})$ then it is likely that the EM iterations will converge to the local maximum, instead of the global maximum. As a result, it is usually advisable to perform several runs of the EM algorithm using different combinations of initial parameters. By doing so, we increase the chances of having the EM algorithm converge to the global maximum of $\ell(\boldsymbol{\Theta} \mid \mathbf{y})$ in at least one of the runs.

As mentioned previously, there are a number of extensions to the original EM algorithm presented by Dempster et al. [1977]. These include the ECM algorithm of Meng and Rubin [1993], the AECM algorithm of Meng and van Dyk [1997] and the PX-EM algorithm of Liu et al. [1998]. While we will not discuss these in great detail in this work, it is worth mentioning that the EM-type algorithms which we utilise in this work are technically ECM algorithms. This is because some of the maximisations in the M-step are actually conditional maximisations which use the current (i.e. $(k+1)^{st}$ time step) estimates of other parameters.

To demonstrate the EM algorithm, in the following example, we work through the derivation of an ECM algorithm for a simple GMM.

Example 3. Consider a set of independent scalar observations $y_1, \ldots, y_n$. Suppose that each observation has been generated by one of two univariate Gaussian distributions with mixing proportions π_1 and π_2 respectively. In other words, suppose the data were generated by the mixture model

$$\begin{aligned} \mathbf{Z}_j &\sim \text{Multinomial}(1, \boldsymbol{\pi}) \\ Y_j \mid Z_{ij} = 1 &\sim N(\mu_i, \sigma_i^2), \end{aligned}$$

independently for $j = 1, \ldots, n$, where $\boldsymbol{\pi} = (\pi_1, \pi_2)$.

Using the law of total probability, the log-likelihood of the observed data is given by

$$\begin{aligned}\ell(\boldsymbol{\Theta} \mid \mathbf{y}) &= \log \prod_{j=1}^{n} f(y_j; \boldsymbol{\Theta}) \\ &= \sum_{j=1}^{n} \log \sum_{i=1}^{2} f(y_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) \Pr(Z_{ij} = 1; \boldsymbol{\Theta}) \\ &= \sum_{j=1}^{n} \log \sum_{i=1}^{2} \pi_i \phi(y_j \mid \mu_i, \sigma_i^2), \end{aligned} \tag{2.12}$$

where $\boldsymbol{\Theta} = (\pi_1, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)^\top$. As would be expected, this is similar to the likelihood of the MFA model from Equation (2.7), and as in the MFA model, we cannot obtain closed form expressions for the MLEs of the parameters of this model directly by differentiating this log-likelihood.

So, instead, we can use the EM algorithm to obtain approximate MLEs. In order to apply the EM algorithm, we need to introduce suitable latent variables. In this case, the choice of latent variables seems fairly obvious from the formulation of the model; the only piece of unobserved data in the model are the indicator vectors $\mathbf{Z}_j$. So, we define the complete data as $\mathbf{x}_j = (y_j, \mathbf{z}_j)^\top$ where $\mathbf{z}_j = (z_{1j}, z_{2j})$. It follows that the complete-data log-likelihood is given by

$$\begin{aligned}\ell(\boldsymbol{\Theta} \mid \mathbf{x}) &= \log \prod_{j=1}^{n} f(\mathbf{x}_j; \boldsymbol{\Theta}) \\ &= \log \prod_{j=1}^{n} f(y_j \mid \mathbf{z}_j; \boldsymbol{\Theta}) f(\mathbf{z}_j; \boldsymbol{\Theta}) \\ &= \log \prod_{j=1}^{n} \prod_{i=1}^{2} (\pi_i \phi(y_j; \mu_i, \sigma_i^2))^{z_{ij}} \\ &= \sum_{j=1}^{n} \sum_{i=1}^{2} z_{ij} \log \pi_i \phi(y_j; \mu_i, \sigma_i^2). \end{aligned} \tag{2.13}$$

We obtain Equation (2.13) by applying the multinomial density, and by realising that since

$$f(y_j \mid \mathbf{z}_j; \boldsymbol{\Theta}) = \phi(y_j; \mu_{i^*}, \sigma_{i^*}^2)$$

where i^* is the index of the unique entry of $\mathbf{z}_j$ such that $z_{ij} = 1$, we can also write this as

$$f(y_j \mid \mathbf{z}_j; \boldsymbol{\Theta}) = \prod_{i=1}^{2} \phi(y_j; \mu_i, \sigma_i^2)^{z_{ij}}.$$

The next step in deriving the EM algorithm for this model is to find the Q-function, which in this case is given by

$$Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(k)}) = \sum_{j=1}^{n} \sum_{i=1}^{2} \mathbb{E}\left[Z_{ij} \mid \mathbf{y}; \boldsymbol{\Theta}^{(k)}\right] \log \pi_i \phi(y_j; \mu_i, \sigma_i^2). \tag{2.14}$$

In this example, we notice that the complete-data log-likelihood, Equation (2.13), is linear in the z_{ij}'s. As a result, because with respect to the conditional expectation only the Z_{ij}'s are random variables, we find that the only unknown quantity in Equation (2.14) is $\mathbb{E}\left[Z_{ij} \mid y_j; \boldsymbol{\Theta}^{(k)}\right]$. However, we will see in Section 2.6 and then again in Chapter 5 that sometimes multiple quantities need to be calculated in each E-step.

Next, we define the responsibilities[4] as

$$\tau_{ij}^{(k)} := \mathbb{E}\left[Z_{ij} \mid y_j; \boldsymbol{\Theta}^{(k)}\right].$$

To perform the E-step, we need to calculate the Q-function from Equation (2.14). The only unknowns in Equation (2.14) are the $\tau_{ij}^{(k)}$'s, so if we can calculate these then we can perform the E-step. Since each Z_{ij} is an indicator variable,

$$\tau_{ij}^{(k)} = \Pr\left(Z_{ij} = 1 \mid y_j; \boldsymbol{\Theta}^{(k)}\right), \tag{2.15}$$

and using Bayes' rule and the law of total probability, it follows from Equation (2.15) that

$$\tau_{ij}^{(k)} = \frac{\pi_i^{(k)} \phi(y_j; \mu_i^{(k)}, \sigma_i^{2(k)})}{\sum_{l=1}^{2} \pi_l^{(k)} \phi(y_j; \mu_l^{(k)}, \sigma_l^{2(k)})}. \tag{2.16}$$

So, given our set of observed data $\mathbf{y}$ and k-step estimates for the model parameters, we can calculate the responsibilities using Equation (2.16). This means that the E-step is to calculate the Q-function from Equation (2.14) by using Equation (2.16). The M-step, on the other hand, is to maximise the Q-function with respect to π_i, μ_i and σ_i.

We will maximise the Q-function with respect to the π_i's first. Note that the π_i's are also subject to the constraint that $\sum_{i=1}^{2} \pi_i = 1$. We can find a suitable maximiser by using Lagrange multipliers, forming the modified objective function

$$Q_1^*(\pi_i, \lambda) = \sum_{j=1}^{n} \sum_{i=1}^{2} \tau_{ij}^{(k)} \log \pi_i - \lambda \left(\sum_{i=1}^{2} \pi_i - 1 \right).$$

[4]The name *responsibilities* reflects the interpretation of the $\tau_{ij}^{(k)}$'s as the posterior probability that data point y_j belongs to sub-population i. In other words, $\tau_{ij}^{(k)}$ measures the degree of responsibility that sub-population i has for data point y_j.

Note that here, we have dropped the terms in Equation (2.14) that do not depend on π_i. Solving this in the usual way leads to the coupled system of equations

$$\sum_{i=1}^{2} \pi_i = 1 \tag{2.17}$$

$$\sum_{j=1}^{n} \frac{\tau_{ij}^{(k)}}{\pi_i} = \lambda. \tag{2.18}$$

Rearranging Equation (2.18) and substituting it into Equation (2.17) leads to $\lambda = n$ after noting that

$$\sum_{i=1}^{2} \sum_{j=1}^{n} \pi_i = n,$$

which follows almost immediately from Equation (2.16). Equation (2.18) then yields

$$\pi_i^{(k+1)} = \frac{1}{n} \sum_{j=1}^{n} \tau_{ij}^{(k)}. \tag{2.19}$$

Consider now the following modified Q-function, where only the terms which depend on μ_i and σ_i have been included:

$$Q_2^*(\mu_i, \sigma_i^2; \boldsymbol{\Theta}^{(k)}) = \sum_{j=1}^{n} \sum_{i=1}^{2} \tau_{ij}^{(k)} \log \left[(\sigma_i^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \frac{(y_j - \mu_i)^2}{2\sigma_i^2} \right) \right]. \tag{2.20}$$

The M-step estimate for μ_i can be obtained by taking the partial derivative of Equation (2.20) with respect to μ_i and setting the resulting expression equal to zero, which yields the updated estimate

$$\mu_i^{(k+1)} = \frac{1}{n_i} \sum_{j=1}^{n} \tau_{ij}^{(k)} y_j, \tag{2.21}$$

where $n_i = \sum_{j=1}^{n} \tau_{ij}^{(k)}$. The estimate for σ_i^2 is obtained by taking the partial derivative of Equation (2.20) with respect to σ_i^2, setting it equal to zero and solving it for σ_i^2, which produces the updated estimate

$$\sigma_i^{2(k+1)} = \frac{1}{n_i} \sum_{j=1}^{n} \tau_{ij}^{(k)} (y_j - \mu_i{}^{(k)})^2. \tag{2.22}$$

These three updates comprise the M-step of the algorithm. Algorithm 2.2 summarises the steps of the EM Algorithm for this example.

Algorithm 2.2: The EM Algorithm for Example 3

Input: An initial estimate of the parameters, $\pi_i^{(0)}, \mu_i^{(0)}$ and $\sigma_i^{2(0)}$, $i = 1, 2$, a convergence criterion and a maximum number of iterations k_{max}

Result: Approximate maximum likelihood estimates for π_i, μ_i and σ_i^2, $i = 1, 2$.

Set $k = 0$;
while *Chosen convergence criterion not satisfied and* $k < k_{max}$ **do**
  The E-Step: Compute $\tau_{ij}^{(k)}$ using Equation (2.16) for $i \in \{1, 2\}$ and $j \in \{1, \ldots, n\}$;
  The M-Step: Compute $\pi_i^{(k+1)}$, $\mu_i^{(k+1)}$ and $\sigma_i^{2(k+1)}$ using Equation (2.19), Equation (2.21) and Equation (2.22), respectively, for $i \in \{1, 2\}$;
  if *Convergence criterion satisfied* **then**
    Return $\pi_i^{(k+1)}, \mu_i^{(k+1)}$ and $\sigma_i^{2(k+1)}$ for $i \in \{1, 2\}$;
  else
    Set $k = k + 1$;
  end
end
Return $\pi_i^{(k_{\max})}, \mu_i^{(k_{\max})}$ and $\sigma_i^{2(k_{\max})}$ for $i \in \{1, 2\}$;

■

We demonstrate Algorithm 2.2 with the following example.

Example 4. Figure 2.1 shows the evolution of a mixture model density function defined by the parameter estimates produced by Algorithm 2.2. In this case, 1500 points were generated according to the two-component univariate Gaussian mixture model with the following parameters:

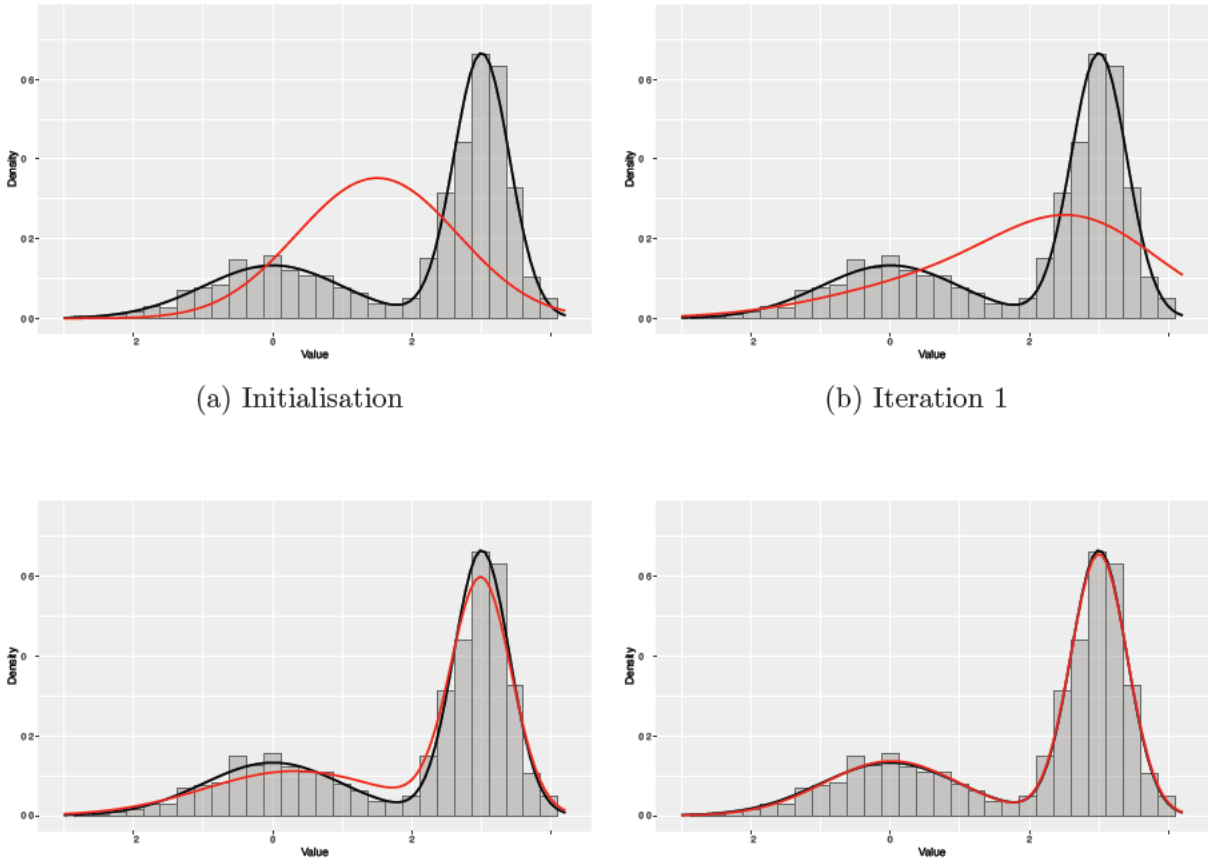
$$\boldsymbol{\pi} = \left(\frac{1}{3}, \frac{2}{3}\right)^\top, \ \mu_1 = 0, \ \mu_2 = 3, \ \sigma_1^2 = 1, \ \sigma_2^2 = 0.16.$$

The initial estimates used were

$$\boldsymbol{\pi}^{(0)} = \left(\frac{1}{2}, \frac{1}{2}\right)^\top, \ \mu_1^{(0)} = 1, \ \mu_2^{(0)} = 2, \ \sigma_1^{2(0)} = \sigma_2^{2(0)} = 1.$$

While the initial parameter estimates define a density that describes the data poorly, we see that the EM iterations gradually improve the estimated density until after ten iterations the estimated density is almost indistinguishable from the true density. The `R` code used to generate these figures is included in Appendix C.

■



(a) Initialisation

(b) Iteration 1

(c) Iteration 4

(d) Iteration 10

Figure 2.1: A normalised histogram of the 1500 data points randomly generated with the parameters given in Example 4. The black line shows the theoretical density of the data, the red line shows the density defined by the EM algorithm's estimates of the parameters.

2.6 EM-type Algorithms for the MFA Model

Several EM-type algorithms for the MFA model have been proposed. We will examine two of these in this work. The first, which we will call MFA-ECM-1 was proposed in Ghahramani et al. [1997]. It treats both the factor and the indicator vectors as latent variables. The second, which we will call MFA-ECM-2, was proposed in Zhao and Yu [2008]. In contrast to MFA-ECM-1, this algorithm only treats the indicator vectors as latent variables.

The derivation of MFA-ECM-1 is simpler than that of MFA-ECM-2, since we condition on more latent variables. This results in a simpler Q-function in the M-step. Most of the mathematical details in the derivation of this algorithm are included in Ghahramani et al. [1997]. As a result, in the derivation of MFA-ECM-1 we have not included full mathematical details as they are already available in the original paper.

On the other hand, the derivation of MFA-ECM-2 is more complicated. We have included more detail in the derivation of this algorithm, since some of the details were excluded in the original work. In both derivations, we assume that g and q are known *a priori* for now.

2.6.1 MFA-ECM-1

Recall the MFA model defined by Equation (1.4). Ghahramani et al. [1997] derived an ECM algorithm for this model as follows.

Recall that under Equation (1.4), the only pieces of observed data are the $\mathbf{y}_j$'s. We define the complete data as $\mathbf{x}_j = (\mathbf{y}_j, \mathbf{u}_j, \mathbf{z}_j)$. So the latent variables for observation $\mathbf{y}_j$ are both the corresponding factor vector $\mathbf{u}_j$ and the corresponding indicator vector $\mathbf{z}_j$.

The log-likelihood of the complete data is therefore

$$\begin{aligned}
\ell(\boldsymbol{\Theta} \mid \mathbf{x}) &= \log \prod_{j=1}^{n} f(\mathbf{x}_j; \boldsymbol{\Theta}) \\
&= \log \prod_{j=1}^{n} f(\mathbf{y}_j \mid \mathbf{z}_j, \mathbf{u}_j; \boldsymbol{\Theta}) f(\mathbf{z}_j, \mathbf{u}_j; \boldsymbol{\Theta}) \\
&\propto \log \prod_{j=1}^{n} \prod_{i=1}^{g} (\pi_i \phi_p(\mathbf{y}_j; \boldsymbol{\mu}_i + \mathbf{B}_i \mathbf{U}_j, \mathbf{D}_i))^{z_{ij}}, \qquad (2.23)
\end{aligned}$$

which we obtain by using a broadly similar argument to Example 3. We do, of course, need to adjust the conditional density of the observed data and also apply independence to break up the joint distribution of the indicators and factors into the product of their distributions. We also note that the distribution of the factors is the standard multivariate

Gaussian distribution, so we can ignore it and work with the expression in Equation (2.23), which is correct up to a constant of proportionality.

For ease of computation, we define

$$\widetilde{\mathbf{u}}_j := \begin{bmatrix} \mathbf{u}_j \\ 1 \end{bmatrix},$$

and

$$\widetilde{\mathbf{B}}_i := \begin{bmatrix} \mathbf{B}_i & \boldsymbol{\mu}_i \end{bmatrix},$$

such that

$$\widetilde{\mathbf{B}}_i \widetilde{\mathbf{u}}_j = \boldsymbol{\mu}_i + \mathbf{B}_i \mathbf{u}_j,$$

which is the location parameter in the Gaussian density function in Equation (2.23).

Let $\boldsymbol{\Theta} = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_g)^\top$ be a vector containing the parameters of the model, where $\boldsymbol{\theta}_i = (\widetilde{\mathbf{B}}_i, \mathbf{D}_i)^\top$. We define the responsibilities analogously to those of Example 3. That is,

$$\tau_{ij}^{(k)} := \mathbb{E}\left[Z_{ij} \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)} \right].$$

It follows that the Q-function for the $\boldsymbol{\theta}_i$'s is

$$\begin{aligned} Q(\boldsymbol{\theta}_i; \boldsymbol{\Theta}^{(k)}) \propto -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{g} \tau_{ij}^{(k)} \log |\mathbf{D}_i| - \sum_{j=1}^{n} \sum_{i=1}^{g} \Bigg[& \frac{1}{2} \mathbf{y}_j^\top \mathbf{D}_i^{-1} \mathbf{y}_j \\ & - \mathbf{y}_j^\top \mathbf{D}_i^{-1} \widetilde{\mathbf{B}}_i \mathbb{E}\left[Z_{ij} \widetilde{\mathbf{U}}_j \mid \mathbf{y}; \boldsymbol{\Theta}^{(k)} \right] \\ & + \frac{1}{2} \operatorname{tr}\left\{ \widetilde{\mathbf{B}}_i^\top \mathbf{D}_i^{-1} \widetilde{\mathbf{B}}_i \mathbb{E}\left[Z_{ij} \widetilde{\mathbf{U}}_j \widetilde{\mathbf{U}}_j^\top \mid \mathbf{y}; \boldsymbol{\Theta}^{(k)} \right] \right\} \Bigg] \end{aligned} \tag{2.24}$$

up to a constant of proportionality. The mixing proportions have not been included in Equation (2.24) since we can estimate them separately, in an almost identical manner to that of Example 3.

We now proceed with the E-step, which constitutes evaluating the conditional expectations in Equation (2.24). The $\tau_{ij}^{(k)}$'s can be calculated using

$$\tau_{ij}^{(k)} = \frac{\pi_i^{(k)} \phi(\mathbf{y}_j; \boldsymbol{\mu}_i^{(k)}, \mathbf{B}_i^{(k)} \mathbf{B}_i^{(k)\top} + \mathbf{D}_i^{(k)})}{\sum_{l=1}^{g} \pi_l^{(k)} \phi(\mathbf{y}_j; \boldsymbol{\mu}_l^{(k)}, \mathbf{B}_l^{(k)} \mathbf{B}_l^{(k)\top} + \mathbf{D}_l^{(k)})}, \tag{2.25}$$

which is derived in the same manner as Equation (2.16).

As a brief aside, if we are interested in using the MFA model to infer the sub-population structure of a dataset, then we can use the responsibilities from Equation (2.25). Recall that the $\tau_{ij}^{(k)}$'s satisfy

$$\tau_{ij}^{(k)} = \Pr(Z_{ij} = 1 \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}),$$

so they are the posterior probability that data point j belongs to component i. While these already give us "soft classifications" (i.e. probabilities), we can obtain hard classifications for each data point by assigning $\mathbf{y}_j$ to the component satisfying $\text{argmax}_l \, \tau_{lj}^{(k)}$. We generally do this at the end of the model fitting process, using the final estimates of the responsibilities.

By exploiting the binary nature of the Z_{ij}'s, it is not hard to see that

$$\mathbb{E}\left[Z_{ij}\widetilde{\mathbf{U}}_i \mid \mathbf{y}; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)} \mathbb{E}\left[\widetilde{\mathbf{U}}_i \mid \mathbf{y}, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] \tag{2.26}$$

and

$$\mathbb{E}\left[Z_{ij}\widetilde{\mathbf{U}}_j\widetilde{\mathbf{U}}_j^\top \mid \mathbf{y}; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)} \mathbb{E}\left[\widetilde{\mathbf{U}}_j\widetilde{\mathbf{U}}_j^\top \mid \mathbf{y}, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]. \tag{2.27}$$

Since $\mathbf{U}_j$ and $\mathbf{Y}_j$ are jointly Gaussian, we have that

$$\mathbb{E}\left[\widetilde{\mathbf{U}}_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] = \begin{bmatrix} \boldsymbol{\beta}_i(\mathbf{y}_j - \boldsymbol{\mu}_i) \\ 1 \end{bmatrix} \tag{2.28}$$

and

$$\mathbb{E}\left[\widetilde{\mathbf{U}}_i\widetilde{\mathbf{U}}_i^\top \mid \mathbf{y}_i, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] = \begin{bmatrix} \mathbf{W}_{ij} & \mathbb{E}\left[\mathbf{U}_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] \\ \mathbb{E}\left[\mathbf{U}_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]^\top & 1 \end{bmatrix} \tag{2.29}$$

where

$$\boldsymbol{\beta}_i := (\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1}\mathbf{B}_i$$

and

$$\mathbf{W}_{ij} := \mathbf{I}_q - \boldsymbol{\beta}_i^\top\mathbf{B}_i + \boldsymbol{\beta}_i(\mathbf{y}_j - \boldsymbol{\mu}_i)(\mathbf{y}_j - \boldsymbol{\mu}_i)^\top\boldsymbol{\beta}_i^\top.$$

The E-step is to calculate the Q-function, Equation (2.24), using Equations (2.25) to (2.29).

As mentioned earlier, the M-step estimate for $\pi_i^{(k+1)}$ can be derived in almost exactly the same way as Equation (2.19), leading to

$$\pi_i^{(k+1)} = \frac{1}{n}\sum_{j=1}^{n} \tau_{ij}^{(k)}. \tag{2.30}$$

Differentiating Equation (2.24) with respect to $\widetilde{\mathbf{B}}_i$ and $\mathbf{D}_i$ separately leads to the following two M-step updates:

$$\begin{aligned} \widetilde{\mathbf{B}}_i^{(k+1)} = &\left(\sum_{j=1}^{n} \tau_{ij}^{(k)}\mathbf{y}_j\mathbb{E}\left[\widetilde{\mathbf{U}}_j \mid \mathbf{y}, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]^\top\right) \\ &\times \left(\sum_{j=1}^{n} \tau_{ij}^{(k)}\widetilde{\mathbf{B}}_i^{(k)}\mathbb{E}\left[\widetilde{\mathbf{U}}_j\widetilde{\mathbf{U}}_j^\top \mid \mathbf{y}, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]\right)^{-1} \end{aligned} \tag{2.31}$$

and

$$\mathbf{D}_i^{(k+1)} = \frac{1}{n_i} \operatorname{diag}\left\{\sum_{j=1}^{n} \tau_{ij}^{(k)} \left(\mathbf{y}_j - \widetilde{\mathbf{B}}_i^{(k)} \mathbb{E}\left[\widetilde{\mathbf{U}}_j \mid \mathbf{y}, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]\right) \mathbf{y}_j^\top\right\}. \tag{2.32}$$

Algorithm 2.3: The MFA-ECM-1 Algorithm from Ghahramani et al. [1997]

Input: An initial estimate of the parameters, $\boldsymbol{\Theta}^{(0)}$, a convergence criterion and a maximum number of iterations k_{max}

Result: Maximum likelihood estimates for the vector of parameters $\boldsymbol{\Theta}$

```
Set k = 0;
while Chosen convergence criterion not satisfied and k < k_max do
    The E-Step: Compute the condition expectations in Equation (2.24) using
     Equation (2.25), Equation (2.26), Equation (2.27), Equation (2.28) and
     Equation (2.29) for i ∈ {1, ..., g} and j ∈ {1, ..., n};
    The M-Step: Compute Θ^(k+1) using Equation (2.30), Equation (2.31) and
     Equation (2.32), respectively;
    if Convergence criterion satisfied then
        Return Θ^(k+1);
    else
        Set k = k + 1;
    end
end
Return Θ^(k_max);
```

Algorithm 2.3 summarises MFA-ECM-1, which was the first EM-type scheme introduced for the fitting of the MFA model. Its derivation is appealing in its simplicity, as including the factors as latent variables means that the M-step estimates for the loading matrices and error-variance matrices can be determined by finding the appropriate derivative of the Q-function, setting it equal to zero and then rearranging. However, we will see in Section 2.6.2 that the inclusion of the factors as latent variables is not necessary to derive an EM-type scheme for the MFA model. Since it includes more latent variables than is strictly necessary, we can expect this method to converge more slowly that MFA-ECM-2 which we introduce next in Section 2.6.2.

2.6.2 MFA-ECM-2

More recently, Zhao and Yu [2008] addressed the issue of a potentially large amount of latent variables being used in MFA-ECM-1 by proposing an ECM algorithm for the MFA model which uses only the indicator vectors as latent variables. We call this algorithm

MFA-ECM-2. As alluded to at the end of Section 2.6.1, by only using the Z_{ij}'s as latent variables we would expect MFA-ECM-2 to generally achieve a faster rate of convergence than MFA-ECM-1. However, this comes at the cost of estimating the underlying factors $\mathbf{U}_j$.

To derive MFA-ECM-2, we first divide the model parameters into two sets, as follows. Let $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ with

$$\boldsymbol{\theta}_1 = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_g)$$

and

$$\boldsymbol{\theta}_2 = (\mathbf{B}_1, \ldots, \mathbf{B}_g, \mathbf{D}_1, \ldots, \mathbf{D}_g).$$

Let the complete data be given be $\mathbf{x}_j = (\mathbf{y}_j, \mathbf{z}_j)$. The complete-data log-likelihood satisfies

$$\begin{aligned}\ell\left(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}\right) \propto \sum_{j=1}^{n}\sum_{i=1}^{g}\Bigg[z_{ij}\log\pi_i - \frac{1}{2}z_{ij}(\mathbf{y}_j - \boldsymbol{\mu}_i)^\top\left(\mathbf{B}_i\mathbf{B}_i + \mathbf{D}_i\right)^{-1}(\mathbf{y}_j - \boldsymbol{\mu}_i) \\ - \frac{1}{2}\log\left|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right|\Bigg].\end{aligned} \tag{2.33}$$

The Q-function is for the parameters in $\boldsymbol{\theta}_1$ satisfies

$$Q\left(\boldsymbol{\theta}_1; \boldsymbol{\Theta}^{(k)}\right) \propto \sum_{j=1}^{n}\sum_{i=1}^{g}\tau_{ij}^{(k)}\left[\log\pi_i - \frac{1}{2}(\mathbf{y}_j - \boldsymbol{\mu}_i)^\top\left(\mathbf{B}_i^{(k)}\mathbf{B}_i^{(k)\top} + \mathbf{D}_i^{(k)}\right)^{-1}(\mathbf{y}_j - \boldsymbol{\mu}_i)\right] \tag{2.34}$$

where $\tau_{ij}^{(k)}$ is defined as usual, which means that they can be calculated using Equation (2.25). The update for the π_i's is identical to their update in MFA-ECM-1, being given by Equation (2.30).

The partial derivative of Equation (2.34) with respect to $\boldsymbol{\mu}_i$ is

$$\frac{\partial Q}{\partial \boldsymbol{\mu}_i} = \sum_{j=1}^{n}\tau_{ij}^{(k)}(\mathbf{B}_i^{(k)}\mathbf{B}_i^{(k)\top} + \mathbf{D}_i^{(k)})^{-1}(\mathbf{y}_j - \boldsymbol{\mu}_i),$$

which provides the update

$$\boldsymbol{\mu}_i^{(k+1)} = \frac{\sum_{j=1}^{n}\tau_{ij}^{(k)}\mathbf{y}_j}{\sum_{j=1}^{n}\tau_{ij}^{(k)}}. \tag{2.35}$$

Now consider the problem of updating $\mathbf{B}_i$ and $\mathbf{D}_i$. We show in Appendix A.2 that the Q-function for the parameters in $\boldsymbol{\theta}_2$ satisfies

$$Q\left(\boldsymbol{\theta}_2; \boldsymbol{\Theta}^{(k+1/2)}\right) \propto -\frac{n}{2}\sum_{i=1}^{g}\pi_i^{(k+1)}\left[\log\left|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right| + \operatorname{tr}\left\{\left(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right)^{-1}\mathbf{S}_i^{(k)}\right\}\right] \tag{2.36}$$

where

$$\mathbf{S}_i^{(k)} := \frac{1}{n\pi_i^{(k+1)}} \sum_{j=1}^{n} \tau_{ij}^{(k)} (\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})^\top,$$

and $\boldsymbol{\Theta}^{(k+1/2)} = (\boldsymbol{\theta}_1^{(k+1)}, \boldsymbol{\theta}_2^{(k)})$. Using the shorthand Q^* for $Q\left(\boldsymbol{\theta}_2 \mid \boldsymbol{\Theta}^{(k+1/2)}\right)$, we show in Appendix A.3 that

$$\frac{\partial Q^*}{\partial \mathbf{B}_i} = -n\pi_i^{(k+1)} \left[(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1}\mathbf{B}_i - (\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1}\mathbf{S}_i^{(k)}(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1}\mathbf{B}_i \right]. \quad (2.37)$$

Setting Equation (2.37) equal to zero leads to

$$\mathbf{B}_i = \mathbf{S}_i^{(k)}(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1}\mathbf{B}_i. \quad (2.38)$$

Notice that this is not an expression for the maximum likelihood solution for $\mathbf{B}_i$, but it is an expression which must be satisfied by any maximum likelihood solution for $\mathbf{B}_i$. Also, notice that for any $\mathbf{B}_i$ satisfying Equation (2.38), it must also be true that

$$\mathbf{B}_i\mathbf{V} = \mathbf{S}_i^{(k)}(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1}\mathbf{B}_i\mathbf{V}$$

for any $q \times q$ matrix $\mathbf{V}$. If, in addition, we have $\mathbf{V}\mathbf{V}^\top = \mathbf{I}_q$ then we know that $\mathbf{B}_i^* := \mathbf{B}_i\mathbf{V}$ must also satisfy Equation (2.38) since $\mathbf{B}_i^*\mathbf{B}_i^{*\top} = \mathbf{B}_i\mathbf{B}_i^\top$. In other words, given any solution $\mathbf{B}_i$ to Equation (2.38), we can always find an equivalent solution by postmultiplying $\mathbf{B}_i$ by any $q \times q$ matrix $\mathbf{V}$ satisfying $\mathbf{V}\mathbf{V}^\top = \mathbf{I}_q$.

Using Equation (2.38), Jöreskog [1967] showed that a local maximum of Q^* with respect to $\mathbf{B}_i$ is given by

$$\hat{\mathbf{B}}_{im_i}^{(k+1)} = \left[\mathbf{D}_i^{(k)}\right]^{1/2} \mathbf{U}_{m_i}(\boldsymbol{\Lambda}_{m_i} - \mathbf{I}_{m_i})^{1/2}\mathbf{V}_{m_i}, \quad (2.39)$$

for a fixed value of m_i, where

$$\tilde{\mathbf{S}}_i := \left[\mathbf{D}_i^{(k)}\right]^{-1/2} \mathbf{S}_i^{(k)} \left[\mathbf{D}_i^{(k)}\right]^{-1/2}, \quad (2.40)$$

$\boldsymbol{\Lambda}_{m_i}$ is a diagonal matrix containing the m_i largest eigenvalues of $\tilde{\mathbf{S}}_i$ and $\mathbf{U}_{m_i}$ is an orthogonal matrix composed of the corresponding m_i eigenvectors of $\tilde{\mathbf{S}}_i$. $\mathbf{V}_{m_i}$ is an $m_i \times q$ matrix satisfying $\mathbf{V}_{m_i}{\mathbf{V}_{m_i}}^\top = \mathbf{I}_{m_i}$. This solution will be real-valued provided that $\max\{\lambda_{i1}, \ldots, \lambda_{im}\} > 1$, where λ_{il} is the l^{th} eigenvalue of $\tilde{\mathbf{S}}_i$. In other words, when at least one of the eigenvectors of $\tilde{\mathbf{S}}_i$ is strictly greater than one. Since we want the global maximum likelihood estimates of $\mathbf{B}_i$ from Q^*, we want to find the value of m_i which maximises the likelihood of the corresponding $\hat{\mathbf{B}}_{im_i}^{(k+1)}$, for each possible i.

We will find the optimal value of m_i by expressing Q^* in terms of m_i. In Appendix A.4, we show that

$$\left|\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right| \cdot \left|\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top + \mathbf{D}_i^{(k)}\right| \cdot \left|\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right| = \prod_{l=1}^{m_i}\lambda_{il}, \tag{2.41}$$

where λ_{il} is the l^{th} eigenvalue of $\tilde{\mathbf{S}}_i$, sorted in descending order. Consequently,

$$\frac{\left|\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top + \mathbf{D}_i^{(k)}\right|}{|\tilde{\mathbf{S}}_i|} = c_0 \prod_{l=1}^{m_i}\lambda_{il}\left(\prod_{l=1}^{p}\lambda_{il}\right)^{-1},$$

where c_0 is a constant introduced by the two $\left|\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right|$ terms, which do not depend on m_i. Hence,

$$\log\left|\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top + \mathbf{D}_i^{(k)}\right| - \log|\tilde{\mathbf{S}}_i| \propto \sum_{l=1}^{m_i}\log\lambda_{il} - \sum_{l=1}^{p}\log\lambda_{il}. \tag{2.42}$$

Since $\log|\tilde{\mathbf{S}}_i|$ and $\pi_i^{(k+1)}$ are both constants with respect to $\boldsymbol{\theta}_2$, the maximum likelihood estimates from Q^* will be the same as those from

$$Q^* + \frac{n}{2}\sum_{i=1}^{g}\pi_i^{(k+1)}\log|\tilde{\mathbf{S}}_i| \propto -\frac{n}{2}\sum_{i=1}^{g}\pi_i^{(k+1)}\left[\log\frac{|\mathbf{B}_i\mathbf{B}_i{}^\top + \mathbf{D}_i|}{|\tilde{\mathbf{S}}_i|} + \operatorname{tr}\left\{\left(\mathbf{B}_i\mathbf{B}_i{}^\top + \mathbf{D}_i\right)^{-1}\mathbf{S}_i^{(k)}\right\}\right]. \tag{2.43}$$

Now consider the trace term in Equation (2.43). We show in Appendix A.5 that

$$\operatorname{tr}\left\{\left(\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top + \mathbf{D}_i^{(k)}\right)^{-1}\mathbf{S}_i^{(k)}\right\} = \sum_{l=1}^{p}\lambda_{il} - \sum_{l=1}^{m_i}(\lambda_{il}-1). \tag{2.44}$$

In Appendix A.6, we also show that in combination with Equation (2.42), Equation (2.44) leads to

$$Q^* + \frac{n}{2}\sum_{i=1}^{g}\pi_i^{(k+1)}\log|\tilde{\mathbf{S}}_i| \propto -\frac{n}{2}\left[\sum_{i=1}^{g}\pi_i^{(k+1)}\sum_{l=1}^{m_i}(\log\lambda_{il} - \lambda_{il} + 1)\right]. \tag{2.45}$$

Since the function $\log x - x + 1$ is negative and strictly decreasing for $x \in (1, \infty)$, it follows that Q^* is maximised if we take

$$m_i = \sum_{i=1}^{p}\mathbb{I}(\lambda_{il} > 1),$$

where the eigenvalues are sorted in order of decreasing magnitude. Our update for $\mathbf{B}_i$, therefore, is

$$\mathbf{B}_i^{(k+1)} = \left[\mathbf{D}_i^{(k)}\right]^{1/2} \mathbf{U}_{m_i}(\mathbf{\Lambda}_{m_i} - \mathbf{I}_{m_i})^{1/2}\mathbf{V}_i, \tag{2.46}$$

with $m_i := \sum_{i=1}^{p} \mathbb{I}(\lambda_{il} > 1)$ and where $\mathbf{V}_i$ is an $m_i \times q$ matrix satisfying $\mathbf{V}_i{\mathbf{V}_i}^\top = \mathbf{I}_{m_i}$.

The final parameter for which we need to derive an M-step estimate is $\mathbf{D}_i$. Recall that $\mathbf{D}_i$ is diagonal, so

$$\mathbf{D}_i^{(k)} = \text{diag}(d_{i1}^{(k)}, \ldots, d_{ip}^{(k)}).$$

Now, define

$$\mathbf{D}_{il}^{(k)} := \text{diag}(d_{i1}^{(k+1)}, \ldots, d_{i(l-1)}^{(k+1)}, d_{il}, d_{i(l+1)}^{(k)}, \ldots, d_{ip}^{(k)}),$$

$$\mathbf{\Sigma}_i := \mathbf{D}_i + \mathbf{B}_i{\mathbf{B}_i}^\top$$

and

$$\mathbf{\Sigma}_{il} := \mathbf{D}_{il}^{(k)} + \mathbf{B}_i^{(k+1)}\left[\mathbf{B}_i^{(k+1)}\right]^\top.$$

We show in Appendix A.3 that

$$\frac{\partial Q^*}{\partial \mathbf{D}_i} = -\frac{n\pi_i^{(k+1)}}{2}\left[\mathbf{\Sigma}_i^{-1} - \mathbf{\Sigma}_i^{-1}\mathbf{S}_i^{(k)}\mathbf{\Sigma}_i^{-1}\right]. \tag{2.47}$$

Define

$$\tilde{\mathbf{\Sigma}}_i := \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \mathbf{\Sigma}_i \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}.$$

Then pre-multiplying and post-multiplying Equation (2.47) by $\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}$ before setting the element-wise derivatives equal to zero produces the constraint on any MLE for d_{il} that

$$\left[\tilde{\mathbf{\Sigma}}_i^{-1} - \tilde{\mathbf{\Sigma}}_i^{-1}\tilde{\mathbf{S}}_i\tilde{\mathbf{\Sigma}}_i^{-1}\right]_{ll} = 0. \tag{2.48}$$

Now define

$$\tilde{\mathbf{B}}_i := \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \mathbf{B}_i^{(k+1)}$$

and

$$\tilde{\mathbf{D}}_{il} := \left[\mathbf{D}_{il}^{(k)}\right]\left[\mathbf{D}_i^{(k)}\right]^{-1} = \mathbf{I}_p + \text{diag}\left(\omega_{i1}^{(k+1)}, \ldots, \omega_{i(l-1)}^{(k+1)}, \omega_{il}, 0, \ldots, 0\right)$$

where

$$\omega_{il} := -1 + \frac{d_{il}}{d_{il}^{(k)}}$$

and

$$\omega_{jl}^{(k+1)} := -1 + \frac{d_{jl}^{(k+1)}}{d_{jl}^{(k)}}$$

for $j = 1, \ldots, l-1$.

With $\mathbf{e}_l$ as the l^{th} column of the $p \times p$ identity matrix, we have

$$\tilde{\boldsymbol{\Sigma}}_{il} = \tilde{\mathbf{D}}_{il} + \tilde{\mathbf{B}}_i \tilde{\mathbf{B}}_i^\top = \omega_{il} \mathbf{e}_l \mathbf{e}_l^\top + \mathbf{C}_{il},$$

where

$$\mathbf{C}_{il} := \mathbf{I}_p + \tilde{\mathbf{B}}_i \tilde{\mathbf{B}}_i^\top + \sum_{j=1}^{l-1} \omega_{ij}^{(k+1)} \mathbf{e}_j \mathbf{e}_j^\top.$$

Suppose that $\mathbf{C}_{il}$ is symmetric and positive definite. Then, via Proposition 1 of Zhao et al. [2007],

$$\tilde{\boldsymbol{\Sigma}}_{il}^{-1} = \mathbf{C}_{il}^{-1} - \frac{\omega_{il} \mathbf{C}_{il}^{-1} \mathbf{e}_l \mathbf{e}_l^\top \mathbf{C}_{il}^{-1}}{1 + \omega_{il} \mathbf{e}_l^\top \mathbf{C}_{il}^{-1} \mathbf{e}_l}. \tag{2.49}$$

As shown in Appendix A.7, applying Equation (2.49) to Equation (2.48) and rearranging produces

$$\omega_{il}^{(k+1)} = \frac{\mathbf{e}_l^\top \mathbf{C}_{il}^{-1} \tilde{\mathbf{S}}_i \mathbf{C}_{il}^{-1} \mathbf{e}_l - \mathbf{e}_l^\top \mathbf{C}_{il}^{-1} \mathbf{e}_l}{\left(\mathbf{e}_l^\top \mathbf{C}_{il}^{-1} \mathbf{e}_l \right)^2}. \tag{2.50}$$

Our update of $\mathbf{D}_i$ can, therefore, be performed component-wise by setting

$$d_{il}^{(k+1)} = (\omega_{il}^{(k+1)} + 1) d_{il}^{(k)},$$

where $\omega_{il}^{(t+1)}$ is defined by Equation (2.50) sequentially from $l = 1$ to p.

However, in practice, an update of this form does not guarantee that $\mathbf{C}_{il}$ will be positive definite. Zhao and Yu [2008] instead suggest that the update be performed by computing $\omega_{il}^{(k+1)}$ according to Equation (2.50) and then setting

$$d_{il}^{(k+1)} = \max \left\{ \eta, (\omega_{il}^{(k+1)} + 1) d_{il}^{(k)} \right\}, \tag{2.51}$$

for $l = 1, \ldots, p$ and some pre-specified $\eta > 0$, which acts as the smallest possible entry in any of the error-variance matrices. They show that this update will guarantee an increasing sequence of likelihoods and also benefits from additional numerical stability.

The inferred sub-population structure of a dataset can be obtained from the responsibilities estimated by MFA-ECM-2 in the same way as it was obtained from those of MFA-ECM-1, as discussed in Section 2.6.1.

Algorithm 2.4 summarises MFA-ECM-2.

Algorithm 2.4: MFA-ECM-2 from Zhao and Yu [2008]

Input: An initial estimate of the parameters, $\boldsymbol{\Theta}^{(0)}$, a convergence criterion and a maximum number of iterations k_{max}

Result: Maximum likelihood estimates for the vector of parameters $\boldsymbol{\Theta}$

Set $k = 0$;
while *Chosen convergence criterion not satisfied and* $k < k_{max}$ **do**
    The E-Step: Compute the responsibilities using Equation (2.25) for $i \in \{1, \ldots, g\}$ and $j \in \{1, \ldots, n\}$;
    The M-Step: Compute $\boldsymbol{\Theta}^{(k+1)}$ using Equation (2.30), Equation (2.35), Equation (2.46) and Equation (2.51), respectively;
    if *Convergence criterion satisfied* **then**
        Return $\boldsymbol{\Theta}^{(k+1)}$;
    else
        Set $k = k + 1$;
    end
end
Return $\boldsymbol{\Theta}^{(k_{\max})}$;

Chapter 3

Existing Methods

In this chapter, we will discuss existing methods which have been proposed to infer g, q or both g and q for the MFA model. We will also include information about their usage in our `R` package `autoMFA`, which is available on CRAN. This package provides a consistent framework for applying each of the methods discussed in this chapter, making it easy to apply one or all of them and compare the results.

We will discuss five different methods which aim to determine the optimal values of g and q with as little input from the user as possible. The parameter estimation routines for all but one of the methods are based on either MFA-ECM-1 or MFA-ECM-2. The only method which is not based on one of these two algorithms is based on a Bayesian formulation of the MFA model.

The simplest method for choosing g and q is to fit MFA models for a large number of combinations of g and q and then choose a final model according to some model selection criterion. In fact, we will regard this naïve grid search as the gold standard, as by searching over large enough ranges for g and q, we should be almost guaranteed to find the best possible values of g and q. We can view the other methods in this chapter as approximations of this method, since they are attempting to find the best values of g and q without exhaustively searching over all of the possible combinations. This is of practical importance because for some datasets, an exhaustive search over large ranges of g and q will be prohibitively computationally expensive.

Three of the five methods instead attempt to choose g using a hierarchical splitting approach. That is, by starting with a single sub-population and then allowing it to be split into two sub-populations if certain criteria are met, and so on. Some of the methods also include mechanisms by which an existing sub-population can be removed from the mixture.

Four out of the five methods also attempt to determine q without naïvely searching over a range of possible values. Two of them use an approach which is intrinsically linked to the derivation of MFA-ECM-2 in Section 2.6.2 and try to choose q using an approximation of the BIC. Another attempts to choose q by examining the sample covariance structure of each of the components in the mixture and then comparing it with the modeled covariance structure of that component. It then adds an extra factor to the component with the largest difference between the modeled and sample covariance structure. Finally, the method which uses the Bayesian formulation of the MFA model is designed to give very small factor loadings to some columns in the factor loading matrix if less factors are supported by the data than were specified by the user at the outset. As a result, we can automatically select q by retaining only the columns of the factor loading matrices with non-negligible loadings.

We have developed the aforementioned `R` package `autoMFA` so that each of the methods discussed in this chapter can be accessed via a standard framework in `R`. While some of the methods had existing implementations in other programming languages, this is (to our knowledge) the first such `R` package. The package also offers other advantages, such as a consistent output format across each of the methods, and additional diagnostic information which was not available in some of the original implementations.

We note that the code in `autoMFA` has not been professionally optimised. The code has, however, been structured as consistently as possible between the different methods.

3.1 Naïve Grid Search

The first method for automatically choosing the values of g and q is the simplest. We perform a naïve grid search over different combinations of g and q, fitting several MFA models for each combination using MFA-ECM-2[1] (with several different starting values) and then choose the best model amongst all of the candidate models according to some model selection criterion.

To perform this method, we assume that the true values of g and q lie within the ranges $g_{\min} \leq g \leq g_{\max}$ and $q_{\min} \leq q \leq q_{\max}$ respectively. Here $q_{\max}$ is still required to respect the Ledermann bound. If we choose inappropriate values for $q_{\min}$, $q_{\max}$, $g_{\min}$ or $g_{\max}$ such that the true values of g or q or both lie outside of the grid defined by those four values, then clearly this method will not be able to infer g or q or both correctly.

Ideally, by searching over large ranges of possible values for g and q, we should be able to

[1] We could instead use MFA-ECM-1 to fit each MFA model, or other similar schemes like the AECM algorithm for the MFA model from McLachlan et al. [2003]. We chose to use MFA-ECM-2 because of its improved rate of convergence compared to the aforementioned two algorithms, which was demonstrated in Zhao and Yu [2008].

find the best combination of parameters reliably. However, the total number of candidate models is given by

$$n_{\text{cand}} = n_{\text{s}}(q_{\max} - q_{\min} + 1)(g_{\max} - g_{\min} + 1),$$

where n_{s} is the number of initial values used for each parameter combination. Hence, the number of candidate models grows rapidly as $g_{\max}$ and $q_{\max}$ increase (assuming that we hold $g_{\min}$ and $q_{\min}$ fixed, often both at one). For example, searching over $1 \leq g \leq 10$ and $1 \leq q \leq 5$ with $n_{\text{s}} = 10$ produces 500 candidate models. In other words, we would need to perform MFA-ECM-2 in full, 500 times, just for this relatively small example.

Several different model selection criteria are possible. We have chosen to select the final model using the BIC. That is, the final model is the model which obtained the lowest BIC value amongst all of the n_{cand} models which will be fitted using MFA-ECM-2. The BIC is the most commonly applied model selection criterion for mixture models, however McLachlan et al. [2019] discuss some other possible methods.

To calculate the BIC of a model, we need to know how many parameters it contains. In the case of an MFA model with g components and q factors per component, the number of parameters is given by

$$k^*_{\text{MFA}}(p, g, q) = g\left(2p + pq + 1 - \frac{1}{2}q(q-1)\right) - 1. \tag{3.1}$$

We obtain this expression via a simple counting argument; we need to estimate $g - 1$ mixing proportions, with the final one constrained by unity, we also need to estimate g $p \times 1$ mean vectors, g $p \times p$ diagonal error-variance matrices (which only contain p parameters each) and finally g $p \times q$ factor loading matrices where $\frac{1}{2}q(q-1)$ constraints have been applied to each. Rearranging then leads to Equation (3.1).

The feasibility of this naïve approach will depend on the dataset being considered. For small datasets where each execution of MFA-ECM-2 is very fast, this approach is reasonable. However, if we were instead working with a dataset that took several hours to fit a single MFA model, then this method may be completely infeasible.

We have created implementations of this algorithm in both `R` and Julia. In `R`, the naïve search is available as the `MFA_ECM` function in our `R` package `autoMFA`, while in Julia it is available as the `MFA_ECM` function in our package FactorMixtures. Julia's superior performance may significantly improve the range of datasets where using this naïve implementation is feasible.

The `R` method `MFA_ECM` in the `autoMFA` package has the following inputs:

- `Y`, an $n \times p$ data matrix where each row represents a data point.

- `gmin`, the lower bound on the range of g values to be searched over. By default it is set to one.
- `gmax`, the upper bound on the range of g values to be searched over. By default it is set to ten.
- `qmin`, the lower bound on the range of q values to be searched over. By default it is set to one.
- `qmax`, the upper bound on the range of q values to be searched over. If it is unspecified, it will default to the Ledermann bound.
- `eta` represents η from Equation (2.51), which is the smallest possible entry allowed for any of the error-variance matrices. By default, it is set to 0.005, as in Zhao and Yu [2008].
- `itmax` controls the maximum number of ECM iterations which will be performed when fitting models. By default, `itmax` $= 500$, so if the convergence criterion has not been satisfied by the 500^{th} iteration, then the model will be returned with the parameters given by $\boldsymbol{\Theta}^{(500)}$.
- `nkmeans`, the number of model initialisations based on k-means clustering. By default it is set to five.
- `nrandom`, the number of random model initialisations. By default it is set to five.
- `tol` represents ε from Equation (2.10) and Equation (2.11). By default it is set to 1×10^{-5}.
- `conv_measure` sets the measure of convergence, with "`diff`" representing Equation (2.10) (the default), and "`ratio`" representing Equation (2.11).
- `varimax` is a boolean variable which controls whether or not the loading matrices from the final fitted model should be constrained using varimax rotation or not. The default is `FALSE`, but if set to `TRUE` then each loading matrix is passed through `stats::varimax` before the model is returned.

For each combination of g and q, a total of $n_s =$ `nrandom` $+$ `nkmeans` MFA models will be fitted. `nrandom` controls the number of random initialisations which will be used. That is, the dataset is randomly divided into g sub-populations (i.e. each data point is randomly allocated a label from 1 to g) and then the initial estimates for the parameters in $\boldsymbol{\Theta}$ are created using these sub-populations. `nkmeans` controls the number of initialisations which will be used based on the output of k-means clusterings. That is, the dataset is passed to the `stats::kmeans` function in `R` with g `centers`. The sub-population structure inferred by the k-means algorithm is then used to create the initial parameter estimates.

The initial parameter estimates are calculated using the method suggested by McLachlan et al. [2003]. For the i^{th} sub-population (given either by a random initialisation or as the result of a k-means clustering run), $\pi_i^{(0)}$ is just the fraction of the total dataset belonging to sub-population i. The initial value $\boldsymbol{\mu}_i^{(0)}$ is given by the sample mean of the data belonging to sub-population i and the initial value $\mathbf{D}_i^{(0)}$ is given by $\text{diag}(\hat{\mathbf{S}}_i)$, where $\hat{\mathbf{S}}_i$ is the sample covariance matrix of the data points belonging to sub-population i. Finally, $\mathbf{B}_i^{(0)}$ is given by

$$\mathbf{B}_i^{(0)} = \left[\mathbf{D}_i^{(0)}\right]^{-\frac{1}{2}} \mathbf{U}_{iq}^{(0)} \left(\boldsymbol{\Lambda}_{iq}^{(0)} - \left(\sigma_i^{(0)}\right)^2 \mathbf{I}_q\right),$$

where $\boldsymbol{\Lambda}_{iq}^{(0)}$ is the diagonal matrix containing the first q eigenvectors of

$$\mathbf{E}_i = \left[\mathbf{D}_i^{(0)}\right]^{-\frac{1}{2}} \hat{\mathbf{S}}_i \left[\mathbf{D}_i^{(0)}\right]^{-\frac{1}{2}},$$

sorted in descending order, and $\mathbf{U}_{iq}^{(0)}$ is the corresponding matrix of eigenvectors. Here, $\left(\sigma_i^{(0)}\right)^2$ is given by the mean of the remaining $p-q$ eigenvalues of $\mathbf{E}_i$.

The implementation of the Julia method `MFA_ECM` is very similar to that of the `MFA_ECM` method in the `R` package `autoMFA`. Notably, however, the Julia implementation does not have a `varimax` input, and the k-means clustering is performed using the `kmeans` function from the `Clustering.jl` package.

3.2 AMFA

A major drawback of the naïve approach to inferring g and q is the computational burden required to search over the whole grid of candidate (g,q) pairs. The Automatic Mixtures of Factor Analyzers (AMFA) algorithm by Wang and Lin [2020] attempts to reduce this burden by treating q as a parameter to be estimated during the execution of MFA-ECM-2. Wang and Lin [2020] based their idea on the work of Zhao and Shi [2014], who introduced the Automatic Factor Analysis (AFA) algorithm which aims to automatically determine q in the FA model. In the AMFA method, we let $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ with

$$\boldsymbol{\theta}_1 = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_g, q_1, \ldots, q_g)$$

and

$$\boldsymbol{\theta}_2 = (\mathbf{B}_1, \ldots, \mathbf{B}_g, \mathbf{D}_1, \ldots, \mathbf{D}_g).$$

The key step in the argument for deriving the M-step update for q used by AMFA is in examining the Q-function for $\boldsymbol{\theta}_2 = (\hat{\mathbf{B}}_{im}, \mathbf{D}_i)$ from the derivation of Algorithm 2.4. Recall

that after assuming the update for $\mathbf{B}_i$ takes the form of

$$\hat{\mathbf{B}}_{im}^{(k+1)} = \left[\mathbf{D}_i^{(k)}\right]^{1/2} \mathbf{U}_m(\boldsymbol{\Lambda}_m - \mathbf{I}_m)^{1/2},$$

we were able to show that

$$Q(\boldsymbol{\theta}_2; \boldsymbol{\theta}_1^{(k+1)}) \propto -\frac{n}{2}\left[\sum_{i=1}^{g} \pi_i^{(k+1)} \sum_{l=1}^{m_i} (\log \lambda_{il} - \lambda_{il} + 1)\right]. \tag{3.2}$$

Previously, we assumed that $m_i \leq q$ is a fixed integer and we wanted to choose the value of m_i which maximises the likelihood of $\hat{\mathbf{B}}_{im_i}$. However, when we choose m_i in Equation (3.2), we are actually choosing the number of columns in $\hat{\mathbf{B}}_{im_i}$, i.e. the number of factors. Of course, previously, if we chose $m_i < q$ we could always end up with a q-factor loading matrix by rotating $\hat{\mathbf{B}}_{im_i}$ by an orthogonal $m_i \times q$ matrix.

If, instead, we treat q as a variable to be estimated, then Wang and Lin [2020] suggest the update

$$q^{(k+1)} = \underset{q \leq q_{\max}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{g} n\pi_i^{(k+1)} \sum_{l=1}^{q} (\log \lambda_{il} - \lambda_{il} + 1) + k^*_{\text{MFA}}(g, q) \log n \right\}. \tag{3.3}$$

where $k^*_{\text{MFA}}(p, g, q)$ is the number of parameters being estimated in the MFA model with g components and q factors, given in Equation (3.1). The expression being minimised in Equation (3.3) is an approximation of the BIC. Recall that the BIC for a given model is

$$\text{BIC} := k^* \log n - 2\hat{\ell}$$

where k^* is the number of parameters being estimated in the model and $\hat{\ell}$ is the maximised value of the log-likelihood function for the model.

Since the Q-function from Equation (3.2) is by definition the expected value of the log-likelihood of the complete data, it is our "best guess" at the log-likelihood of the model. Hence, Equation (3.3) is an approximation of the BIC as claimed.

Functionally, Equation (3.3) acts as an additional M-step in Algorithm 2.4, to be performed after updating π_i and $\boldsymbol{\mu}_i$ but before updating $\mathbf{B}_i$. For simplicity, we have included Algorithm 3.1 which summarises the AMFA algorithm.

Since g still needs to be chosen in the AMFA algorithm, the initial grid search over (g,q) has been reduced to a line search over g. When determining g, the `AMFA` method in `autoMFA` behaves in the same way as `MFA_ECM`; it fits $n_s =$ `nrandom`+`nkmeans` MFA models for each value of g between `gmin` and `gmax` and then chooses the value of g corresponding to the model with the best BIC.

AMFA is implemented in `autoMFA` as the `AMFA` method. The inputs are mostly identical to those of `MFA_ECM`, except that `qmin` and `qmax` are not valid inputs for the `AMFA` method.

Algorithm 3.1: The AMFA algorithm from Wang and Lin [2020]

Input: An initial estimate of the parameters $\boldsymbol{\Theta}^{(0)}$, a number of components g and a maximum number of iterations k_{max}
Result: Maximum likelihood estimates for the vector of parameters $\boldsymbol{\Theta}$

Set $k = 0$;
while *Chosen convergence criterion not satisfied and* $k < k_{max}$ **do**
  The E-Step: Compute the responsibilities using Equation (2.25) for $i \in \{1, \ldots, g\}$ and $j \in \{1, \ldots, n\}$;
  The M-Step: Compute $\boldsymbol{\Theta}^{(k+1)}$ using Equation (2.30), Equation (2.35), Equation (3.3), Equation (2.46) and Equation (2.51), respectively;
  if *Convergence criterion satisfied* **then**
    Return $\boldsymbol{\Theta}^{(k+1)}$;
  else
    Set $k = k + 1$;
  end
end
Return $\boldsymbol{\Theta}^{(k_{\max})}$;

3.3 AMoFA

Kaya and Salah [2015] proposed the Adaptive Mixtures of Factor Analyzers (AMoFA) algorithm which aims to determine both g and q automatically, without performing naïve searches over either parameter, in contrast to the naïve search and AMFA methods. While we defer a full description of the algorithm to their work, a brief outline of the algorithm follows.

AMoFA begins by fitting a single component, single factor MFA model to the dataset using a slightly modified version of MFA-ECM-1. Then, the algorithm chooses between two potential actions: to add a component to the model or add a factor to an existing component. Unlike the other methods which we consider, AMoFA allows for the number of factors to differ between components. That is, instead of using a single value of q for all components in the mixture, each component has its own number of factors q_i.

The choice of whether to add a new component or to add a factor to an existing component is governed by the Minimum Message Length (MML) criterion, which is a model selection criterion proposed in Wallace and Boulton [1968] and applied to the MFA model by Kaya and Salah [2015]. Both of the actions are considered by the algorithm, and the action which results in the model with the smaller MML is retained.

To add a component to the model, an existing component is split into two smaller compo-

nents. The choice of which component to split is based on the multivariate kurtosis metric γ_j from Mardia [1970], which was adapted to mixture models by Salah and Alpaydin [2004] and is defined there as

$$\gamma_i^{(k)} := \left\{b_{2,p}^i - p(p+2)\right\} \left[\frac{8p(p+2)}{\sum_{j=1}^n \tau_{ij}^{(k)}}\right]^{-\frac{1}{2}} \tag{3.4}$$

where

$$b_{2,p}^i := \frac{1}{\sum_{j=1}^n \tau_{ij}^{(k)}} \sum_{j=1}^n \tau_{ij}^{(k)} \left[(\mathbf{y}_j - \boldsymbol{\mu}_i)(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1}(\mathbf{y}_j - \boldsymbol{\mu}_i)\right]^2. \tag{3.5}$$

Mardia [1970] showed that in the context of non-mixture models[2], under appropriate assumptions, as $n \to \infty$

$$\gamma \sim N(0, 1)$$

asymptotically. The AMoFA algorithm uses this result to assess the kurtosis of the components in the model, choosing to split the component with the largest $|\gamma_j^{(k)}|$. Intuitively, this aims to select the component with the largest kurtosis, which is the least likely to have been sampled from a multivariate Gaussian distribution.

Once a component has been selected for splitting, we then need to decide how to split it (for simplicity, suppose component i has been selected for splitting). AMoFA does this by specifying two new mean vectors, one for each "child" component. Then, these new mean vectors are used as the respective initial parameter estimates for a two-component MFA model, fitted only to the data currently assigned to component i. These assignments are based on the responsibilities and use the same method as the one discussed at the end of Section 2.6.1.

The mean vectors are defined as $\boldsymbol{\mu}_{\text{new},1} := \boldsymbol{\mu}_i + \mathbf{t}_i$ and $\boldsymbol{\mu}_{\text{new},2} := \boldsymbol{\mu}_i - \mathbf{t}_i$, where $\boldsymbol{\mu}_i$ is the sample mean of all points currently assigned to cluster i, and

$$\mathbf{t}_i := \sum_{i=1}^p \mathbf{u}_i \lambda_i$$

with $\mathbf{u}_i$ and λ_i respectively being the eigenvectors and eigenvalues of the sample covariance matrix of all of the data points currently assigned to cluster i, $\mathbf{S}_i$, where

$$\mathbf{S}_i := \frac{1}{\sum_{i=1}^n \mathbb{I}(\mathbf{y}_j, i)} \sum_{j=1}^n \mathbb{I}(\mathbf{y}_j, i)(\mathbf{y}_j - \boldsymbol{\mu}_i)(\mathbf{y}_j - \boldsymbol{\mu}_i)^\top.$$

$\mathbb{I}(\mathbf{y}_j, i)$ is an indicator variable for data point j belonging to component i, so

$$\mathbb{I}(\mathbf{y}_j, i) = 1 \iff \text{argmax}_k\{\tau_{kj}\} = i.$$

[2]Their definition of γ subtracts $p(p+2)\frac{n-1}{n+1}$ from $b_{2,p}$ and uses n instead of $\sum_{j=1}^n \tau_{ij}^{(k)}$

We point out here that the procedure for initializing new components is not well defined, at least as presented in Kaya and Salah [2015]. The vector $\mathbf{t}_i$ is defined in terms of the values of the eigenvectors, but eigenvectors are only ever identifiable up to a sign change. As a result, we suggest imposing a direction constraint on the eigenvectors, for example, that their first element is always positive. This constraint is necessary if we want to obtain equivalent results from cross-platform implementations of the algorithm, so we have included it in the `amofa` method in `autoMFA`.

As mentioned earlier, whenever AMoFA requires parameter estimates of an MFA model, it will produce them using a slightly modified version of the MFA-ECM-1. The only change is in the M-step for the mixing proportions, where the MML criterion dictates that any component satisfying

$$\sum_{j=1}^{n} \tau_{ij}^{(k)} < \frac{1}{2}\left(p(q_i+2) + L^*(q_i)\right) \tag{3.6}$$

should be removed, for $i = 1, \ldots, g$, where

$$L^*(q_i) := \log^*(q_i) + \log c,$$

with $c \approx 2.865064$ and

$$\log^*(q_i) = \log q_i + \log\log q_i + \cdots$$

where the sum contains only as many terms as are positive. After removing any components satisfying Equation (3.6), the remaining components have their mixing proportions updated as normal.

MLEs for the two-component MFA model are obtained using this method. Then, the two-component MFA model is re-combined into the original model (i.e. the model before splitting component i) by appropriately dividing up the original mixing proportion for component i between the two new components according to their mixing proportions from the 2 component model. The MML for the recombined model is calculated and stored, for comparison against the MML obtained by a factor addition.

AMoFA uses the following factor addition metric. Let

$$\boldsymbol{\Delta}_i := \text{offdiag}\left\{(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i) - \mathbf{S}_i\right\},$$

then with

$$\delta_i := \text{tr}\left\{\boldsymbol{\Delta}_i^\top \boldsymbol{\Delta}_i\right\},$$

AMoFA chooses to add a factor to the component with the largest value of δ_i, which is the element-wise sum of squares of the difference between the modelled covariance of component i and the sample covariance of component i (excluding the terms on the main

diagonal, i.e. the variances). The scheme for the initialisation of the new column in the factor loading matrix can be found in Salah and Alpaydin [2004]. Once the component has been selected for factor addition and the new column of the appropriate factor loading matrix has been initialised, the modified version of MFA-ECM-1 is used to find maximum likelihood estimates for this new model. Once convergence has been achieved, the final MML is calculated.

Whichever action resulted in the model with the lowest MML is retained. Then, the process of selecting one of the two actions is repeated on this model, and this performed iteratively until the decrease in the MML obtained by performing the best action falls below a pre-specified tolerance ε_1.

Finally, once the MML decrease is less than ε_1, the algorithm enters its decremental phase. AMoFA will now find the component with the smallest soft support, i.e. the smallest value of $\sum_{j=1}^{n} \tau_{ij}^{(k)}$ and remove it from the model. Then, the modified version of MFA-ECM-1 is used to find maximum likelihood estimates for this model. The process of deleting the weakest component continues until only one component remains, at which point the final model is chosen to be the model with the smallest MML found throughout the entire fitting process.

Algorithm 3.2: The AMoFA algorithm from Kaya and Salah [2015]

Input: An input dataset $\mathbf{Y}$
Result: Maximum likelihood estimates $\boldsymbol{\Theta}$ for an MFA model with g and q_i automatically chosen

repeat
  Split an existing component using the process described above and store the MML of the resulting model;
  Add a factor to an existing component using the process described above and store the MML of the resulting model;
  Keep the model with the lower MML
until *Decrease in MML is less than* ε_1;
while *While* $g > 1$ **do**
  Calculate $n_i = \sum_{j=1}^{n} \tau_{ij}^{(k)}$ for $i = 1, \ldots, g$;
  Delete component $i^* = \text{argmin}_i\, n_i$ and fit the resulting $g - 1$ component model. Check if this has a lower MML than any model encountered so far;
end
Return the model with the lowest MML encountered during the entire fitting process.

The AMoFA algorithm is implemented as the `amofa` method in `autoMFA`. It takes the following inputs:

- `Y`, an $n \times p$ data matrix as in `MFA_ECM` and `AMFA`.
- `itmax`, which defines the maximum number of iterations used whenever the slightly modified version of MFA-ECM-1 is applied during the fitting process, and is set to 100 by default.
- `verbose`, a boolean variable which controls whether detailed output should be printed during the fitting process. This defaults to `FALSE`.
- `varimax`, which behaves in the same way as in as in `MFA_ECM` and `AMFA`.

3.4 VBMFA

The Variational Bayesian Mixture of Factor Analyzers (VBMFA) algorithm from Ghahramani and Beal [1999] is another algorithm which aims to infer both g and q without requiring a naïve search over either parameter.

This method is based on a Bayesian formulation of the MFA model, making it unique in this respect among the techniques that we will consider in this chapter. This formulation, including specifications of the necessary prior distributions, can be found in Ghahramani and Beal [1999]. We will note here, though, that it assumes a common error-variance matrix $\mathbf{D}$ across all components.

We will not include a full description of the VBMFA algorithm, as it would require a detailed introduction to Variational Bayesian methods. Instead, we refer interested readers to Beal [2003] for details about Variational Bayesian methods more generally, and for full details on how they can be applied to the Bayesian MFA model.

However, we will point out that the model fitting process of VBMFA is similar to that of the frequentist models, as it makes use of the so-called Variational Bayesian Expectation-Maximisation (VBEM) algorithm, which retains the alternating series of E-steps and M-steps from the EM algorithm. We also note that during the execution of the VBEM algorithm for the MFA model derived in Ghahramani and Beal [1999] and Beal [2003], the mixing proportion of a component can be estimated as 0. In this circumstance, the component would be removed from the model. This means that VBEM for the Bayesian MFA model can perform automatic inference on the number of components. However, starting with a very large number of components and waiting for unnecessary components to be removed automatically is not very computationally efficient. Instead, Beal [2003] suggests an incremental approach which starts with a one component model and includes a mechanism for component birth. The fitting process terminates once a pre-specified number of attempts have been made to split each component and no-improvement to the model's fit has been achieved.

The particular heuristic that Beal suggests for splitting an existing component into two "child" components is as follows. Suppose component i is to be split. We first sample

$$\mathbf{d}_i \sim \mathcal{N}_p\left(\boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}\right).$$

Then, for all of the data points currently assigned to component i, we allocate any point satisfying

$$(\mathbf{y}_j - \boldsymbol{\mu}_i)^\top \mathbf{d}_i \geq 0$$

to one "child" component. The remaining points are assigned to the other "child" component.

Inference over the number of factors q is handled via the use of Automatic Relevance Detection (ARD) priors over the columns of the factor loading matrices. Let $\mathbf{B}_i^l$ denote the l^{th} column of $\mathbf{B}_i$. Then Beal [2003] suggests taking

$$p(\mathbf{B}_i^l \mid \nu_i^l) = \phi_p\left(\mathbf{B}_i^l; \mathbf{0}, \frac{\mathbf{I}_p}{\nu_i^l}\right).$$

The parameter ν_i^l is the precision on the l^{th} column of $\mathbf{B}_i$, which in turn is governed by the hyperprior

$$p(\nu_i^l) = \text{Gamma}\left(a^*, \frac{1}{b^*}\right).$$

The model fitting process allows for $\nu_i^l \to \infty$ in some circumstances. If this occurs, then the entries of $\mathbf{B}_i^l$ will necessarily become very small, so this dimension is effectively being ignored by the model. After the model fitting process has been completed, we can finish the inference on q by only including the columns of factor loading matrices that are not extremely close to the zero vector according to some heuristic.

Unfortunately, however, in our testing, we were unable to reproduce the behaviour shown in Beal [2003]. Neither the original MATLAB implementation provided with Ghahramani and Beal [1999] nor our equivalent `R` method `vbmfa` produced columns of factor loading matrices with extremely small loadings. As a result, we could not produce inferred values of q from the models fitted using `vbmfa`, as the number of columns in the output factor loading matrices is just the maximum number of columns allowed, which is a user input. Because of these issues, estimates of model BIC and the number of factors returned by VBMFA are unreliable.

Ghahramani and Beal [1999] suggest centering and scaling all datasets before applying the VBMFA algorithm. As a result, alongside the `vbmfa` method, we also include the `preprocess` method, which takes in an $n \times p$ data matrix and returns a centered and scaled version. The `vbmfa` method takes the following inputs:

- `Y`, which is expected to be an $n \times p$ data matrix returned by the `preprocess` method.

- `qmax`, which represents the number of columns which the final models factor loading matrices will contain[3]. The default number is $p - 1$.
- `numTries` controls how many times the algorithm will attempt to split each component. If `numTries` splits are attempted for each component and no improvement in the model fit is found then the fitting process will terminate.
- `verbose`, a boolean variable which controls whether detailed output should be printed during the fitting process. This defaults to `FALSE`.
- `varimax`, which behaves in the same way as in as in `MFA_ECM`, `AMFA` and `autoMFA`.

3.5 Incremental AMFA

The final algorithm included in `autoMFA` is our own contribution. It aims to infer g and q automatically without a naïve search over either variable. The inference over q will be performed using the M-step update from Algorithm 3.1, which only leaves g to be inferred.

Instead of naïvely searching over a range of possible values for g, we propose an incremental approach. The algorithm will begin by fitting a single component, single factor model using the same initialisation procedures described in Section 3.1. Then, the single component model will be split into a two component model. The points will be allocated to the new components in the same way as the "child" components in VBMFA. The two-component model will be fitted using Algorithm 3.1. If the BIC of the new model is higher than the BIC of the model before splitting, then the split is rolled back and a new split is attempted. If no improvement to the BIC is made after a specified number of split attempts, then the fitting process is terminated.

Otherwise, the first split which decreases the BIC is accepted. Then the splitting process is repeated. This time, the attempted splitting order for components will be in order of decreasing $|\gamma_j|$, with γ_j as in Equation (3.4). The model fitting process is only terminated once the algorithm has attempted to split each of the components the maximum specified number of times without an improvement in the BIC occurring.

This algorithm is incorporated into `autoMFA` as the `AMFA_inc` method. Most of the inputs are the same as those of the `MFA_ECM`, and `AMFA` methods: `Y`, `eta`, `itmax`, `tol`, `conv_measure` and `varimax` are the same as those from `AMFA`. The arguments `nkmeans` and `nrandom` behave in the same way as they do for `AMFA`, except that they only apply to the initial single component, single factor model which is fitted. The only other input

[3]Recall that the model should automatically give very low loadings to "unnecessary" columns in the factor loading matrix, resulting in a model that essentially contains less than or equal to `qmax` factors for each component. However, we were unable to observe this occurring in our experiments.

is `numTries`, the maximum number of times that the algorithm should attempt to split each component.

3.6 Additional Information About `autoMFA`

The output of models fitted using `autoMFA` has been standardised as much as possible. The returned object will be a list with several elements. The fitted model itself will be one of these elements, which itself will be a list containing the mixing proportion vector $\boldsymbol{\pi}$, the factor loading matrices $\mathbf{B}_i$, the error-variance matrices $\mathbf{D}_i$, the mean vectors $\boldsymbol{\mu}_i$ and a vector containing the numbers of factors for each component.

Another element will be the clustering information, which will include the posterior probabilities of each point belonging to each of the sub-populations in the model, and the clusterings implied by these posterior probabilities.

There will also be a diagnostics element, which contains information specific to the fitting process of each algorithm, but will always include the total time taken to fit the model.

Output list component	Object name	Description
`model`	`mu`	The mean vectors
	`B`	The loading matrices
	`D`	The error-variance matrices
	`pivec`	The mixing proportion vector
	`numFactors`	Number of factors for each component
`diagnostics`	`bic`	Fitted model BIC
	`logL`	Fitted model log-likelihood
	`totalTime`	Total time to fit model
`clustering`	`responsibilities`	Posterior probabilities
	`allocations`	Posterior probability hard allocations

Table 3.1: The output information common to all `autoMFA` models.

Table 3.1 summarises the structure of the output. For example, if our fitted model is called `MFAfit`, then we could obtain the loading matrices with `MFAfit$model$B` or the BIC with `MFAfit$diagnostics$bic`. All models in the `autoMFA` package will provide the information in Table 3.1.

3.7 Summary

We have introduced five different methods for automatically fitting the MFA model. Each of these methods has been implemented in our `R` package `autoMFA` which is available on

CRAN. The five methods available in `autoMFA` are summarised in Table 3.2. In addition, the naïve search method has also been implemented in our Julia package FactorMixtures so that it can benefit from Julia's better computational efficiency. We will use `autoMFA` to perform a systematic comparison of the automatic MFA model fitting methods in the following chapter.

Algorithm name	Name in `autoMFA`	Original Reference	Discussed in
Naïve search	`MFA_ECM`	-	Section 3.1
AMFA	`AMFA`	Wang and Lin [2020]	Section 3.2
AMoFA	`amofa`	Kaya and Salah [2015]	Section 3.3
VBMoFA	`vbmfa`	Ghahramani and Beal [1999]	Section 3.4
Incremental AMFA	`AMFA_inc`	Wang and Lin [2020]	Section 3.5

Table 3.2: A summary of the methods in autoMFA.

Chapter 4

Systematic Comparison

Each of the methods from the previous chapter have been implemented in our `R` package, `autoMFA`, which is available on CRAN.

The purpose of this chapter is to conduct a systematic comparison of automatic methods for fitting the MFA model. We will compare the five methods in `autoMFA`, as well as three additional methods with pre-existing `R` implementations.

The first additional method is GMMs fitted using the `mclust` package [Scrucca et al., 2016]. This will serve as a benchmark to compare the ARI and BIC of the fitted MFA models against. We expect that the unrestricted covariance structure of the GMMs may afford them higher log-likelihoods, but that the more parsimonious MFA models will achieve lower BICs.

The other two additional methods are the Infinite Mixtures of Infinite Factor Analyzers (IMIFA) and Overfitted Mixtures of Infinite Factor Analyzers (OMIFA) models, which were proposed in Murphy et al. [2020]. Both methods are available in the `R` package `IMIFA` [Murphy et al., 2021]. Both IMIFA and OMIFA are based on Bayesian formulations of the MFA model and are designed to infer both g and q without naïve searches. Both of these methods make use of Markov Chain Monte Carlo (MCMC). As a result, instead of providing the BIC as a model selection criterion, they provide a modified version of the BIC, the Bayesian Information Criterion Monte (Carlo) or BICM, as introduced by Raftery et al. [2007]. We cannot, therefore, directly compare them to the methods from `autoMFA` or `mclust` in terms of BIC.

The methods will be tested on a range of datasets generated by MFA models, defined by various combinations of the following six variables:

- the dimension of the data

- the number of factors, q
- the number of components, g
- the degree of separation of the components
- the number of points in the dataset
- whether the number of points is the same across all components.

Each of the six variables will have two possible values. Where possible, we have constructed these as "high" and "low" settings for each variable.

One hundred simulated replicates were used to evaluate the performance of the methods according to the following quantities:

- model fitting time
- the inferred number of factors, q
- the inferred number of components, g
- the BIC of the fitted model
- the clustering allocation of each data point.

While the importance of tracking the first four quantities is hopefully clear, tracking the clustering allocations of each model is useful as it allows us to calculate the ARI between the sub-population inferred by each model and the true sub-population structure of the dataset. We use this as a measure of the clustering accuracy of each model, that is, how well each model is able to infer the underlying sub-population structure of the datasets.

We also identified the `fabMix` package [Papastamoulis, 2020] which provides methods for fitting a model similar to OMIFA, except that it used finite factor analyzers as opposed to infinite factor analyzers [Papastamoulis, 2018]. However, this method, which relied on an naïve search for determining q, proved to be prohibitively slow and as a result, we were unable to include it in the comparison.

4.1 Experimental Design

With six variables and two possible values for each variable, there will be $2^6 = 64$ possible combinations of the variables. Rather than running the full 64 trials, we instead make use of a 2^{6-2} fractional factorial design[1], given by Box [2005]. This reduces the number

[1]Here, we make use of the fractional factorial experiment notation I^{k-p}. Here, $I = 2$ is the number of levels of each factor in the experiment, $k = 6$ is the number of factors in the experiment and $p = 2$ is the fraction of the full experiment which we conduct (i.e. $2^{-2} = \frac{1}{4}$ of the full factorial design).

of trials from 64 to 16, at the cost of introducing confounding from third and sixth order interactions.

Table 4.1 summarises the variable settings used in each of the 16 different experiment groups. Each of the 16 combinations has been given an experiment ID, which will be used in the results section as a group label when we summarise our findings for all of the experiments in that group.

The different settings for the 6 variables are as follows:

- For the dimension of the data, we chose the low setting to be $p = 3$ dimensional data and the high setting to be $p = 10$.
- The number of factors used to generate the data scale with p. The low value is $q = \lfloor \frac{1}{3}p \rfloor$ and the high value is $q = \lfloor \frac{2}{3}p \rfloor$. This choice means that the fraction q/p remains approximately constant, even as p increases. If we had instead made $q = 1$ the low setting regardless of p, then the $p = 3$ case has proportionally more factors than the $p = 10$ case. However, in abiding by the Ledermann bound, we run into an issue with the $p = 3$ case, because the maximum possible number of factors is $q = 1$. Our high value for the number of factors is $q = \lfloor \frac{2}{3}3 \rfloor = 2$ which cannot occur. Consequently, we have chosen to remove the 4 experiments which used the high setting for q when $p = 3$, leaving us with 12 experiments instead of the original 16. The affected combinations are shown in grey in Table 4.1.
- The number of sub-populations, like the dimensional of the data, varied between three and ten. These values were not chosen completely arbitrarily; clearly the minimum number of components required for a non-degenerate mixture model is $g = 2$, so having three mixture components is still considered to be small. On the other hand, a ten component model would generally be considered large, as this would imply that the data follows a highly multimodal density function.
- The separation of each component is defined in terms of the mean of each component. This is because the covariance matrices for each component will be approximately equal in magnitude, which we will elaborate on later. First, define the temporary mean of component i as $\boldsymbol{\mu}_{t,i}$. When $p = 3$ and $g = 3$, we have

$$\boldsymbol{\mu}_{t,1} = (1, 0, 0)^\top, \boldsymbol{\mu}_{t,2} = (0, 1, 0)^\top, \boldsymbol{\mu}_{t,3} = (0, 0, 1)^\top.$$

  Similarly, when $p = 10$ and $g = 3$, we have

$$\boldsymbol{\mu}_{t,1} = (1, 0, \ldots, 0)^\top, \boldsymbol{\mu}_{t,2} = (0, 1, 0, \ldots, 0)^\top, \boldsymbol{\mu}_{t,3} = (0, 0, 1, 0, \ldots, 0)^\top.$$

  When $p = 3$ and $g = 10$, we take

$$\boldsymbol{\mu}_{t,1} = (1, 0, 0)^\top, \boldsymbol{\mu}_{t,2} = (1, 0, 1)^\top, \boldsymbol{\mu}_{t,3} = (0, 0, 0)^\top, \boldsymbol{\mu}_{t,4} = (0, 0, 1)^\top,$$

$$\boldsymbol{\mu}_{t,5} = (0,-1,0)^\top,\ \boldsymbol{\mu}_{t,6} = (0,-1,1)^\top,\ \boldsymbol{\mu}_{t,7} = (-1,0,0)^\top,\ \boldsymbol{\mu}_{t,8} = (-1,0,1)^\top,$$
$$\boldsymbol{\mu}_{t,9} = (0,1,0)^\top,\ \boldsymbol{\mu}_{t,10} = (0,1,1)^\top.$$

Finally, when $p = 10$ and $g = 3$, we have

$$\boldsymbol{\mu}_{t,1} = (1,0,\ldots,0)^\top,\ \boldsymbol{\mu}_{t,2} = (0,1,0,\ldots,0)^\top,\ \ldots,\ \boldsymbol{\mu}_{t,10} = (0,\ldots,0,1)^\top.$$

For well separated data we set the mean of each component, $\boldsymbol{\mu}_i$, to be given by $\boldsymbol{\mu}_i = 3\boldsymbol{\mu}_{t,i}$. For not well separated data, we instead took $\boldsymbol{\mu}_i = 1.5\boldsymbol{\mu}_{t,i}$.

- The total number of points also scaled with g, the low setting was $60g$ and the high setting was $240g$.
- If the proportion of points in each component is the same, then each component is given an equal share of the total number of data points. Otherwise, we allocated the number of points as follows. First, to make sure that none of the components are too small to perform inference on, we assign 30 points to each. Then, defining n_{t} as the total number of points in the dataset, we calculate the total number of data points yet to be allocated, $n_t - 30g$. By design, we chose to allocate ten times more of these remaining data points to the largest cluster than the smallest, with the intermediate clusters obtaining a linearly interpolated fraction of this amount. In practice, this means that the remaining points are distributed according to the following proportion vectors $\boldsymbol{\pi}^*$. For $g = 3$,

$$\boldsymbol{\pi}^* = \left(0.\overline{60},\quad 0.\overline{3},\quad 0.\overline{06}\right)$$

and for $g = 10$

$$\boldsymbol{\pi}^* = \left(0.\overline{18},\quad 0.1\overline{63},\quad 0.1\overline{45},\quad 0.1\overline{27},\quad 0.1\overline{09},\quad 0.0\overline{9},\quad 0.0\overline{72},\quad 0.0\overline{54},\quad 0.0\overline{36},\quad 0.0\overline{18}\right).$$

Each dataset is generated by simulating

$$\mathbf{Y}_{i1}, \mathbf{Y}_{i2}, \ldots, \mathbf{Y}_{in_i} \overset{i.i.d}{\sim} \mathcal{N}_p\left(\boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right) \text{ for } i = 1,\ldots,g$$

and then concatenating them into one dataset. We keep each covariance matrix relatively small by taking $\mathbf{D}_i = 0.01\mathbf{I}_p$ and randomly generating $\mathbf{B}_i$ such that

$$\mathbf{B}_i = \begin{bmatrix} \sqrt{0.2}R_{11} & \cdots & \sqrt{0.2}R_{1q} \\ \vdots & \ddots & \vdots \\ \sqrt{0.2}R_{p1} & \cdots & \sqrt{0.2}R_{pq} \end{bmatrix}$$

where $R_{lm} \overset{i.i.d}{\sim} \mathcal{N}(0,1)$, for $l \in \{1,\ldots,p\}$ and $m \in \{1,\ldots,q\}$. In doing so, the elements of each covariance matrix $\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i$ will be small compared to the elements of the corresponding $\boldsymbol{\mu}_i$. As a result, the generated points will be tightly clustered around each $\boldsymbol{\mu}_i$ relative to the distance between the $\boldsymbol{\mu}_i$'s. Hence, the degree to which the components are separated will be dependent almost entirely on the separation of the means, which is the separation parameter we are controlling.

ID	p	Total Points	Well Separated	g	Proportion	q
1	3	$60g$	No	3	Unequal	$\lfloor \frac{1}{3}p \rfloor = 1$
2	10	$60g$	No	3	Equal	$\lfloor \frac{1}{3}p \rfloor = 3$
3	10	$240g$	No	3	Unequal	$\lfloor \frac{2}{3}p \rfloor = 6$
4	10	$60g$	Yes	3	Unequal	$\lfloor \frac{2}{3}p \rfloor = 6$
5	3	$240g$	Yes	3	Unequal	$\lfloor \frac{1}{3}p \rfloor = 1$
6	10	$240g$	Yes	3	Equal	$\lfloor \frac{1}{3}p \rfloor = 3$
7	10	$60g$	No	10	Equal	$\lfloor \frac{2}{3}p \rfloor = 6$
8	3	$240g$	No	10	Equal	$\lfloor \frac{1}{3}p \rfloor = 1$
9	10	$240g$	No	10	Unequal	$\lfloor \frac{1}{3}p \rfloor = 3$
10	3	$60g$	Yes	10	Equal	$\lfloor \frac{1}{3}p \rfloor = 1$
11	10	$60g$	Yes	10	Unequal	$\lfloor \frac{1}{3}p \rfloor = 3$
12	10	$240g$	Yes	10	Equal	$\lfloor \frac{2}{3}p \rfloor = 6$
13	3	$240g$	No	3	Equal	$\lfloor \frac{2}{3}p \rfloor = 2$
14	3	$60g$	Yes	3	Equal	$\lfloor \frac{2}{3}p \rfloor = 2$
15	3	$60g$	No	10	Unequal	$\lfloor \frac{2}{3}p \rfloor = 2$
16	3	$240g$	Yes	10	Unequal	$\lfloor \frac{2}{3}p \rfloor = 2$

Table 4.1: Variable settings for each of the 16 combinations. Note the last four rows are greyed out because they do not satisfy the Ledermann bound.

4.2 Results

We present the results of the comparison as the following set of tables. Table 4.2 and Table 4.3 show the mean fitting time (in seconds) of the methods within each experiment group. Table 4.4 and Table 4.5 show the mean BIC/BICM of the methods within each experiment group. Table 4.6 shows the mean ARI of the methods within each experiment group. Table 4.7 and Table 4.8 show the proportion of the experiments within each group that correctly inferred the number of components, and inferred the number of components to within a tolerance of two, respectively. Table 4.9 and Table 4.10 show the corresponding proportions for the inferred number of factors, q. Table 4.11 shows the proportion of experiments where each method correctly inferred q after stratifying on the correct identification of g. Finally, Table 4.12 shows the ability estimates of each model for predicting each of g, $g \pm 2$, q and $q \pm 2$ under the Rasch model [Rasch, 1960], which ere obtained using the `R` package `ltm` [Rizopoulos, 2006]. We will discuss these tables in the following section.

In addition to the tables described above, histograms for the inferred values of g and q can be found in Appendix B.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust
1	80.9881	4.0490	7.1489	1.4182	79.2914	0.1434
2	38.5470	2.2915	5.4293	0.5443	385.8049	0.0813
3	201.5613	14.7605	74.6669	5.4524	1417.7323	2.2793
4	40.1561	2.3887	6.7150	0.6172	312.0735	0.0691
5	206.8572	6.6270	1.7904	3.5234	202.7172	1.3102
6	172.4055	27.5839	8.7546	3.8093	1684.1250	2.2612
7	210.3839	13.6387	51.3788	4.2410	1659.7744	2.1338
8	527.8520	225.1070	16.4439	96.4536	521.6713	5.6754
9	385.6559	487.2741	368.8590	192.7831	4136.9533	8.1284
10	119.6726	29.9985	11.6425	4.5551	116.7766	0.3184
11	75.5418	30.1718	77.7980	9.7393	997.5532	0.5766
12	552.4807	196.8137	515.8520	231.5884	2748.2744	5.6644
Avg	217.6752	86.7254	95.5399	46.2271	1188.5623	2.3868

Table 4.2: Mean time taken (in seconds) to fit each model for each experiment group.

ID	IMIFA	OMIFA
1	403.7176	364.3704
2	877.5861	565.1778
3	1357.1503	757.5887
4	1020.3364	573.7155
5	864.7408	452.6027
6	968.9900	742.5656
7	1010.6108	891.7379
8	842.1925	767.8052
9	1810.3102	1476.7388
10	547.2551	376.6435
11	1184.2645	884.7914
12	1839.0619	1596.3306
Avg	1060.5180	787.5057

Table 4.3: Mean time taken (in seconds) to fit each model for each experiment group continued.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust
1	1050.7228	1054.4714	1294.3148	-	1050.7228	1050.5567
2	3507.5636	3606.5715	3956.2990	-	3495.2498	3820.6508
3	17812.1526	17849.2621	20141.9293	-	17792.6720	18050.6258
4	5148.7162	5386.0111	5557.2780	-	4985.5135	5121.6210
5	3829.9685	3831.7224	3895.5671	-	3829.9685	3829.5695
6	12407.7403	12703.1300	12586.6397	-	12407.7403	12725.9944
7	19095.5663	19088.6927	20775.5173	-	18797.9084	19110.6593
8	17626.3888	17635.4263	17821.8976	-	17626.3888	17647.1745
9	46659.6513	47633.2190	51111.2959	-	46655.6679	48378.1994
10	5321.7465	5311.2477	5863.8217	-	5321.7465	5285.2753
11	13674.8590	15603.8832	15440.3559	-	13599.5406	14580.4656
12	66760.3419	69780.2167	72589.8460	-	66644.2970	66636.2590
Avg	17741.2848	18290.3212	19252.8968	-	17683.9513	18019.7543

Table 4.4: Mean BIC for each model and each experiment group.

ID	IMIFA	OMIFA
1	-2118.3328	-2384.8549
2	-4203.4096	-4771.5617
3	-16965.5368	-18976.5741
4	-5293.6050	-5853.9074
5	-95533.4676	-97659.5807
6	-18319.8867	-20830.9932
7	-31415.2886	-25043.5271
8	-265393.8117	-87380.4818
9	-69184.5104	-75598.8663
10	-86117.5153	-74742.2687
11	-19758.4464	-22376.9821
12	-90264.2513	-101370.5933
Avg	-58714.0052	-44749.1826

Table 4.5: Mean BICM for the two methods from the `IMIFA` package.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust	IMIFA	OMIFA
1	0.8446	0.8314	0.5720	0.7370	0.8446	0.8481	0.7610	0.7210
2	0.9566	0.9403	0.6119	0.5335	0.9617	0.7888	0.9648	0.9653
3	0.9589	0.9555	0.3667	0.9598	0.9600	0.8223	0.9593	0.9593
4	0.9506	0.8574	0.5397	0.7354	0.9952	0.8833	0.9940	0.9938
5	0.9997	0.9984	0.9847	0.9769	0.9997	0.9997	0.9972	0.9972
6	1.0000	1.0000	0.9513	0.9998	1.0000	0.9931	0.9999	0.9999
7	0.0475	0.0651	0.1368	0.0222	0.6359	0.2013	0.5574	0.5610
8	0.5811	0.5877	0.4811	0.4927	0.5811	0.5829	0.3879	0.3601
9	0.9369	0.9348	0.6108	0.9232	0.9369	0.8882	0.9371	0.9371
10	0.9483	0.9570	0.6980	0.3530	0.9483	0.9838	0.4376	0.4079
11	0.9926	0.8376	0.6912	0.7258	0.9995	0.9914	0.9990	0.9990
12	0.9911	0.9272	0.7010	0.9920	0.9920	0.9918	0.9902	0.9901
Avg	0.8507	0.8244	0.6121	0.7043	0.9046	0.8312	0.8321	0.8243

Table 4.6: Mean ARI for each model and each experiment group.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust	IMIFA	OMIFA
1	0.8800	0.8300	0.0800	0.6500	0.8800	0.8900	0.6300	0.5200
2	0.9900	0.9300	0.0400	0.0800	1.0000	0.6300	1.0000	1.0000
3	1.0000	0.9900	0.0100	0.9900	1.0000	0.4800	1.0000	1.0000
4	0.8800	0.6700	0.0100	0.2700	1.0000	0.5500	1.0000	1.0000
5	1.0000	0.9800	0.6600	0.9700	1.0000	1.0000	0.9800	0.9800
6	1.0000	1.0000	0.6900	1.0000	1.0000	0.9400	1.0000	1.0000
7	0.0000	0.0000	0.0400	0.0000	0.4500	0.0100	0.1700	0.1000
8	0.2100	0.3100	0.1800	0.1900	0.2100	0.3900	0.0000	0.0000
9	0.9800	0.9900	0.0000	0.4200	0.9900	0.4200	1.0000	1.0000
10	0.7600	0.7600	0.1000	0.0200	0.7600	0.9700	0.0000	0.0000
11	0.9100	0.0900	0.0300	0.0000	1.0000	0.8700	1.0000	1.0000
12	1.0000	0.5400	0.0400	1.0000	1.0000	1.0000	1.0000	1.0000
Avg	0.8008	0.6742	0.1567	0.4658	0.8575	0.6792	0.7317	0.7167

Table 4.7: Proportion of experiments where each model inferred the number of components correctly.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust	IMIFA	OMIFA
1	1.0000	1.0000	0.2900	1.0000	1.0000	1.0000	1.0000	1.0000
2	1.0000	1.0000	0.3600	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	0.1600	1.0000	1.0000	0.9900	1.0000	1.0000
4	1.0000	1.0000	0.3600	1.0000	1.0000	1.0000	1.0000	1.0000
5	1.0000	1.0000	0.9000	1.0000	1.0000	1.0000	1.0000	1.0000
6	1.0000	1.0000	0.8600	1.0000	1.0000	1.0000	1.0000	1.0000
7	0.0000	0.0000	0.1500	0.0000	0.7500	0.1000	0.8800	0.8300
8	0.9900	0.9900	0.6500	0.6800	0.9900	0.9500	0.0300	0.0100
9	1.0000	1.0000	0.0000	1.0000	1.0000	0.9800	1.0000	1.0000
10	1.0000	1.0000	0.3800	0.0400	1.0000	1.0000	0.0000	0.0000
11	1.0000	0.3000	0.0900	0.1000	1.0000	1.0000	1.0000	1.0000
12	1.0000	0.8100	0.0500	1.0000	1.0000	1.0000	1.0000	1.0000
Avg	0.9158	0.8417	0.3542	0.7350	0.9783	0.9183	0.8258	0.8200

Table 4.8: Proportion of experiments where each model inferred the number of components to within an error of ± 2 components.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust	IMIFA	OMIFA
1	1.0000	1.0000	0.9700	-	1.0000	-	0.0000	0.0000
2	0.9100	0.3100	0.6500	-	1.0000	-	0.0000	0.0000
3	0.5500	0.6800	0.0300	-	0.8600	-	0.0100	0.0100
4	0.0000	0.0100	0.0200	-	0.1900	-	0.3100	0.3200
5	1.0000	1.0000	0.9700	-	1.0000	-	0.0000	0.0100
6	1.0000	0.0000	0.9400	-	1.0000	-	0.0000	0.0000
7	0.0000	0.0000	0.0000	-	0.0300	-	0.1200	0.1600
8	1.0000	1.0000	0.9900	-	1.0000	-	0.0000	0.0000
9	1.0000	0.0000	0.1800	-	1.0000	-	0.0000	0.0000
10	1.0000	1.0000	1.0000	-	1.0000	-	0.0000	0.0000
11	0.8900	0.0000	0.4300	-	1.0000	-	0.0300	0.0300
12	0.4100	0.1100	0.0400	-	0.9900	-	0.0200	0.0200
Avg	0.7300	0.4258	0.5183	-	0.8392	-	0.0408	0.0458

Table 4.9: Proportion of experiments where each model inferred the number of factors correctly.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust	IMIFA	OMIFA
1	1.0000	1.0000	1.0000	-	1.0000	-	1.0000	1.0000
2	1.0000	0.9400	1.0000	-	1.0000	-	1.0000	1.0000
3	1.0000	0.9900	0.4500	-	1.0000	-	1.0000	1.0000
4	0.3500	0.1500	0.2900	-	1.0000	-	1.0000	1.0000
5	1.0000	1.0000	1.0000	-	1.0000	-	1.0000	1.0000
6	1.0000	0.0500	1.0000	-	1.0000	-	1.0000	1.0000
7	0.0000	0.0000	0.0000	-	0.9600	-	1.0000	1.0000
8	1.0000	1.0000	1.0000	-	1.0000	-	1.0000	1.0000
9	1.0000	0.4900	1.0000	-	1.0000	-	1.0000	1.0000
10	1.0000	1.0000	1.0000	-	1.0000	-	1.0000	1.0000
11	1.0000	1.0000	1.0000	-	1.0000	-	1.0000	1.0000
12	1.0000	0.5600	0.4100	-	1.0000	-	1.0000	1.0000
Avg	0.8625	0.6817	0.7625	-	0.9967	-	1.0000	1.0000

Table 4.10: Proportion of experiments where each model inferred the number of factors to within an error of ± 2 factors.

ID	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust	IMIFA	OMIFA
1	1.0000	1.0000	1.0000	-	1.0000	-	0.0000	0.0000
2	0.9091	0.3118	1.0000	-	1.0000	-	0.0000	0.0000
3	0.5500	0.6869	1.0000	-	0.8600	-	0.0100	0.0100
4	0.0000	0.0149	0.0000	-	0.1900	-	0.3100	0.3200
5	1.0000	1.0000	1.0000	-	1.0000	-	0.0000	0.0102
6	1.0000	0.0000	1.0000	-	1.0000	-	0.0000	0.0000
7	-	-	0.0000	-	0.0444	-	0.0000	0.0000
8	1.0000	1.0000	1.0000	-	1.0000	-	-	-
9	1.0000	0.0000	-	-	1.0000	-	0.0000	0.0000
10	1.0000	1.0000	1.0000	-	1.0000	-	-	-
11	0.8791	0.0000	1.0000	-	1.0000	-	0.0300	0.0300
12	0.4100	0.2037	0.7500	-	0.9900	-	0.0200	0.0200
Avg	0.7794	0.4907	0.9681	-	0.8649	-	0.0421	0.0453

Table 4.11: Proportion of experiments where each model inferred the number of factors correctly, after conditioning on inferring the number of components correctly.

Test	AMFA	AMFA_inc	amofa	vbmfa	MFA_ECM	Mclust	IMIFA	OMIFA
g	3.3173	1.2902	-5.9057	-1.1139	4.7089	1.3557	2.0980	1.8741
$g \pm 2$	1.1533	0.2333	-3.5024	-0.7335	2.7120	1.1926	0.0732	0.0161
q	1.8966	-0.6396	0.0629	-	3.4864	-	-4.2412	-4.1301
$q \pm 2$	0.0717	-1.2541	-0.8121	-	2.7178	-	3.3964	3.3964

Table 4.12: Each method's ability estimates for inferring the given criteria correctly, obtained by fitting a Rasch model for each of the criteria.

4.3 Analysis

We begin our analysis with a few general conclusions. Of the methods included in the `autoMFA` package, the `MFA_ECM` and `AMFA` methods performed the best, on average, for all of the metrics except the time required to fit each model and for inferring q to within an error of ± 2. The `IMIFA` and `OMIFA` methods from the `IMIFA` package generally performed comparably to the best methods in `autoMFA`. The GMMs fitted using the `mclust` package performed slightly worse than the best `autoMFA` methods, on average, except for in the time taken to fit each model. The `AMFA_inc` method generally performed well, but not as well as the aforementioned methods. The `amofa` and `vbmfa` methods almost always performed the worst out of the methods included in the comparison.

4.3.1 Fitting Time

We first consider the mean fitting times given in Table 4.2 and Table 4.3. On average, across all 1,200 datasets, the `Mclust` package had the lowest mean fitting time. The slowest methods, on average, were `IMIFA` and `MFA_ECM`. The difference in mean fitting time between `Mclust` and the two slowest methods was pronounced: the mean fitting time of the `IMIFA` method was over 440 times greater than that of the `Mclust` method, and the `MFA_ECM` method's mean fitting time was over 490 times greater than `Mclust`'s.

The dimension of the data had a clear impact on the model fitting time for the `MFA_ECM` method. This is likely because the `MFA_ECM` method is not only impacted by the higher computational burden of performing the necessary matrix computations in higher dimensions (as are all of the other models), but it also has to evaluate six times more models, since when $p = 10$, the Ledermann bound affords $q_{max} = 6$ compared to $q_{max} = 1$ when $p = 3$.

The `vbmfa`, `amofa` and `AMFA_inc` methods generally required much less time to fit than the `AMFA` and `MFA_ECM` methods. This is also unsurprising, since the last two methods both included a naïve search over g (and the `MFA_ECM` method also utilises a naïve search over q), whereas the first three methods do not involve any naïve searches. As we might expect, the `AMFA` method was generally faster than the `MFA_ECM` method since it only has to search over a range of values for g whereas the `MFA_ECM` method also had to search over q. However, this was not always the case. In experiment groups one, five, eight and ten the `MFA_ECM` method was actually faster (on average) compared to the `AMFA` method. Each of these experiment groups had $p = 3$ which means that $q_{max} = 1$. As a result, the `MFA_ECM` method only had to search over g, like the `AMFA` method. The time difference can be explained by slight differences in the implementations of the two methods in this situation.

The `IMIFA` and `OMIFA` methods were, on average, the second and third slowest over-

all. However, their fitting times were considerably less variable, between the experiment groups, when compared to those of the MFA_ECM method. To see this, consider the ratio, r, of the experiment group with the longest mean fitting time to the experiment group with the shortest mean fitting time, for each method. We see that for MFA_ECM ($r = 52.1740$), the ratio is over ten times larger than for IMIFA ($r = 4.5553$) and OMIFA ($r = 4.3811$).

4.3.2 Mean BIC Score

Next, we compare the fit of the models based on mean BIC score, given in Table 4.4 and Table 4.5. Recall that the IMIFA and OMIFA methods provided the BICM instead of the BIC, which means that we cannot compare these two methods with the rest of the methods in the comparison using BIC. For the rest of the methods in the comparison, we will quantify the differences in mean BIC by making use of the guidelines given by Raftery [1995], who suggests that a difference in BIC of greater than six is "strong" evidence that the model with the lower BIC is superior, and that a difference of more than ten constitutes "very strong" evidence.

Table 4.4 shows that the MFA_ECM method obtains the overall lowest mean BIC across all 1,200 experiments. The difference between the method with the next lowest mean BIC, AMFA, and the mean BIC of MFA_ECM is over 50, which provides very strong evidence that the MFA_ECM method found better models than the other comparable methods, on average.

At an experiment group level, the MFA_ECM method almost always achieved the lowest mean BIC score out of all of the methods included in autoMFA. The only exception was in group ten, where the AMFA_inc method scored lower on average. This is unsurprising, as we would expect that the MFA_ECM method should explore the model space more thoroughly than the other methods in autoMFA. This is analogous to the problem of predictor selection in multiple linear regression: the naïve grid search is like attempting to find the best combination of predictors by exhaustively fitting all of the possible linear models and choosing the best one according to some criterion, whereas the other MFA methods are similar to predictor selection algorithms like stepwise selection. They attempt to choose the best model by some means other than by exhaustively fitting each possible model. As a result, they will not explore the model space as thoroughly as the MFA_ECM method, which results in it generally finding better models.

Even though the naïve grid search generally attained lower mean BIC values than the other methods, in some cases the difference was minor. For example, the AMFA method was able to achieve the same mean BIC value (up to four decimal places) as the MFA_ECM method in groups one, five, six, eight and ten. In addition, in experiment group nine the difference in mean BIC was less than six, so although there is some evidence that the MFA_ECM method found better models in this group, on average, it is not strong.

In the other groups, the differences in mean BIC between the MFA_ECM and AMFA methods

were all greater than ten, providing convincing evidence that in these groups, the former method outperformed the latter. These six groups, namely experiments two, three, four, seven, eleven and twelve all had $p = 10$ dimensional data. This suggests that the `AMFA` method's ability to find the "gold standard" models fitted by the `MFA_ECM` method decreases as the number of dimensions increases, which coincides with the Ledermann bound increasing.

We observe a somewhat similar pattern for the mean BICs obtained via the `Mclust` method. Namely, in groups one, five, ten and twelve the `Mclust` method achieved average BICs lower than those of the `MFA_ECM` method. The average difference between the mean BICs of the `MFA_ECM` and `Mclust` methods is even more than ten in group ten, providing very strong evidence that the `Mclust` method has fitted better models, on average, in that group. However, in the remaining groups the average BIC of the models fitted using the `MFA_ECM` method are much lower than those fitting using the `Mclust` method. All but one of these groups (group eight) were experiments with $p = 10$. This suggests, as we might expect, that the full-covariance GMMs fitted using `Mclust` can obtain higher log-likelihoods than the restricted-covariance MFA models. When $p = 3$, the number of parameters saved by the restricted covariance structure of the MFA models is relatively small, so the greater log-likelihoods obtained by `Mclust` translates into a lower BIC. However, when $p = 10$, the number of parameters saved by the MFA models starts to have an impact. This effect is especially prominent in groups two, six, nine and eleven where the mean difference in BIC is especially large. These are all experiments where the true number of factors is three, making the MFA models much more parsimonious than the GMMs in these groups, such that the number of parameters saved by the MFA models is more than enough to compensate for the higher log-likelihoods obtained by the GMMs.

The `amofa` method performed poorly across each of the experiment groups, with much higher mean BICs than the `AMFA` and `MFA_ECM` methods. The `AMFA_inc` method also generally performed poorly in comparison to these two methods, although there is very strong evidence to suggest that on average, it fitted better models in experiment group ten.

Finally, we compare the BICM scores for the two methods from the `IMIFA` package. Across the whole 1,200 experiments, the `IMIFA` method produced a lower average BICM value. However, at an experiment group level, the `OMIFA` method actually obtains lower BICM values in nine out of the twelve groups. In fact, the only groups where the `IMIFA` method has the lower average BICM values are groups seven, eight and ten. The main reason that the `IMIFA` method scored better on average was because in group eight, it obtained an average BICM value whose magnitude was roughly three times that of the `OMIFA` method[2].

[2]Since both methods produced negative BICM values on this dataset, the larger BICM magnitude favours the IMIFA method.

Further investigation showed that this was not due to an outlier BICM value which was biasing the average for the IMIFA. Given this, generally, the BICM criterion is suggesting that in most of the experiment groups the OMIFA method produced better models, on average.

4.3.3 Mean ARI

We used the ARI to measure how well each of the methods were able to recreate the underlying sub-population structures of the datasets. We compared the models based on their mean ARI scores, as given in Table 4.6.

The MFA_ECM method obtained the highest overall mean ARI, across all 1,200 experiments. The next highest overall mean ARI was obtained by the AMFA method. The IMIFA, Mclust, AMFA_inc and OMIFA methods all obtained very similar overall mean ARI values. The vbmfa and amofa methods performed noticeably worse, on average, compared to the preceding methods.

At the experiment group level, the mean ARI of the MFA_ECM method was equal to or greater than that of the remaining methods in autoMFA, except in experiment group eight, where the AMFA_inc method attained the highest mean ARI. The amofa method performed the worst out of any of the methods for the majority of the experiment groups. The vbmfa method performed erratically: in groups three, five, six, nine and twelve it attained mean ARI's equal or only slightly less than the MFA_ECM method. These experiment groups all had the high setting for the total number of data points, $240g$, suggesting a possible link between the vbmfa method's ability to recreate the underlying sub-population structure and the size of the dataset being used. However, in group eight, which was the only other group with the high setting for the total number of points, vbmfa's performance was worse than the aforementioned groups and was even worse (proportional to the best performing method for that group) than its performance in the some of the groups where the total number of points per cluster was only $60g$.

The IMIFA and OMIFA methods obtained mean ARI scores very close to the MFA_ECM method in the majority of the experiment groups, with the exception of groups one, seven, eight and ten. Of these, all but group ten were not well separated, implying a possible relationship between the separation of the components and the ability of the IMIFA and OMIFA methods to recreate the sub-population structure of the datasets. A similar pattern can be observed in the mean ARI scores for the Mclust method whose performance was similar to that of the MFA_ECM method, except in groups two, three, four and seven. All of these except group four were not well separated groups. More broadly, as might be expected, the well separated experiment groups generally obtained higher mean ARI scores compared to the poorly separated experiment groups.

The two experiment groups with the lowest mean ARI scores were groups seven and eight.

In both of these, the sub-populations were not well separated, of which there were ten in each case. We will see later that all of the methods (except `amofa`) tended to underestimate the number of components for both of these experiment groups. Essentially, because the components were not well separated, several of the components could be overlapping, making it very difficult for the algorithms to correctly identify all ten components. The `MFA_ECM` method outperformed all of the other methods in experiment group seven by a large margin. In group eight, however, the `AMFA_inc` method obtained the highest overall mean ARI, followed by `Mclust` and then `MFA_ECM` and `AMFA` in joint third place.

4.3.4 Inference on g

We compared how well the different methods inferred the number of components, g, using two metrics. First, we calculated the proportion of the experiments where each method correctly estimated the number of components, as well as the proportion of the experiments where each method inferred the number of components to within ± 2[3]. These results are summarized in Table 4.7 and Table 4.8.

Then, we fit a Rasch model [Rasch, 1960] to each of the two cases above. Under this model, each experiment has a unique difficulty parameter and each method has a unique ability parameter. If the difficulty parameter of a given experiment is equal to the ability parameter of a method, then the probability that the method correctly infers the number of components (or the number of components $\pm$ 2 for the second case) is exactly 1/2. We are interested in the ability parameters of each method, which are summarized in Table 4.12.

The `MFA_ECM` method achieved the highest overall proportion of experiments where it inferred g correctly, and where it inferred g to within ± 2. The `AMFA` method achieved the second highest proportion of experiments where g was inferred correctly, but `Mclust` inferred g to within an error of ± 2 slightly more often making it the second best method overall, on that metric. The `amofa` method had the lowest proportions on both metrics.

Among the MFA methods, the naïve grid search method was able to determine the correct number of components the highest (or the equal highest) proportion of the time in all of the experiment groups except group eight, where the `AMFA_inc` method outperformed it, and group nine, where the `IMIFA` and `OMIFA` methods outperformed it, although only by a single experiment. It was able to determine the correct number of components 100% of the time for experiment groups two through six, nine and ten. It also inferred the correct number of components the vast majority of the time in groups one and nine. The

[3] By also calculating the proportion of experiments where each method inferred the number of components to within an error of ± 2, we want to check if any of the methods, whilst not having a high proportion of experiments where the value of g was inferred exactly, produced a distribution of inferred values for g which was distributed about the true value with a low spread.

Mclust method generally performed quite similarly to to the MFA_ECM method, except in groups two, three, four, seven, eight and nine, where it performed noticeably worse. All of these except group four were poorly separated groups, indicating that Mclust's ability to infer g may be especially sensitive to the separation of the components. However, we also note that when we instead consider Table 4.8, only group seven remains especially low compared to the MFA_ECM method's result. Hence, while Mclust appears to be sensitive to the separation of the components, it produced distributions which had narrow spreads about the true values of g.

All of the methods performed poorly in groups seven and eight. Figure 4.1 and Figure 4.2 show that almost all of the methods tended to underestimate the number of components in both of these experiment groups. The IMIFA and OMIFA methods in particular badly underestimated the true number of components in group eight. The only exception was the amofa method, which instead overestimated the number of components in group seven.

All of the models tended to perform better, on average, when $g = 3$ compared to when $g = 10$. This effect is especially clear in Table 4.8, where the first six rows correspond to the groups with $g = 3$. All of the models except amofa achieve almost perfect scores in these rows, which is not true of the last six rows (which are the groups where $g = 10$). This shows that in general, the methods appear to be able to infer g more accurately when the true number of components is small.

The MFA_ECM and AMFA methods generally performed similarly to the Mclust method. However, in group ten, both methods performed noticeably worse than Mclust in terms of the proportion of experiments where g was inferred correctly. However, Figure 4.3 shows that in this case, when these methods did not correctly infer that $g = 10$, they almost always inferred $g = 9$ instead, which is reflected in the increased proportions shown in Table 4.8.

The AMFA method generally performed comparably to the naïve grid search, with the exception of group seven. Figure 4.1 shows that in this case, the AMFA method had incorrectly inferred that all of the datasets had less than or equal to five components. As a result, even when we consider the proportion of the datasets where the AMFA method inferred a number of components less than two away from the true value, the proportion is still zero. This poor performance may be due to the fact that group seven had $p = 10$ and $g = 10$, was not well separated and had the smaller maximum cluster size.

The amofa method generally had the poorest performance of any of the methods. Figure 4.1, Figure 4.2, Figure 4.3 and Figure 4.4 show that it routinely and dramatically overestimated the number of clusters present in the data, sometimes inferring upwards of 40 components! The vbmfa method tended to underestimate the true number of components. The AMFA_inc approach performed relatively well, except in groups seven and eleven. The poor performance in group seven is similar to the AMFA method in that it

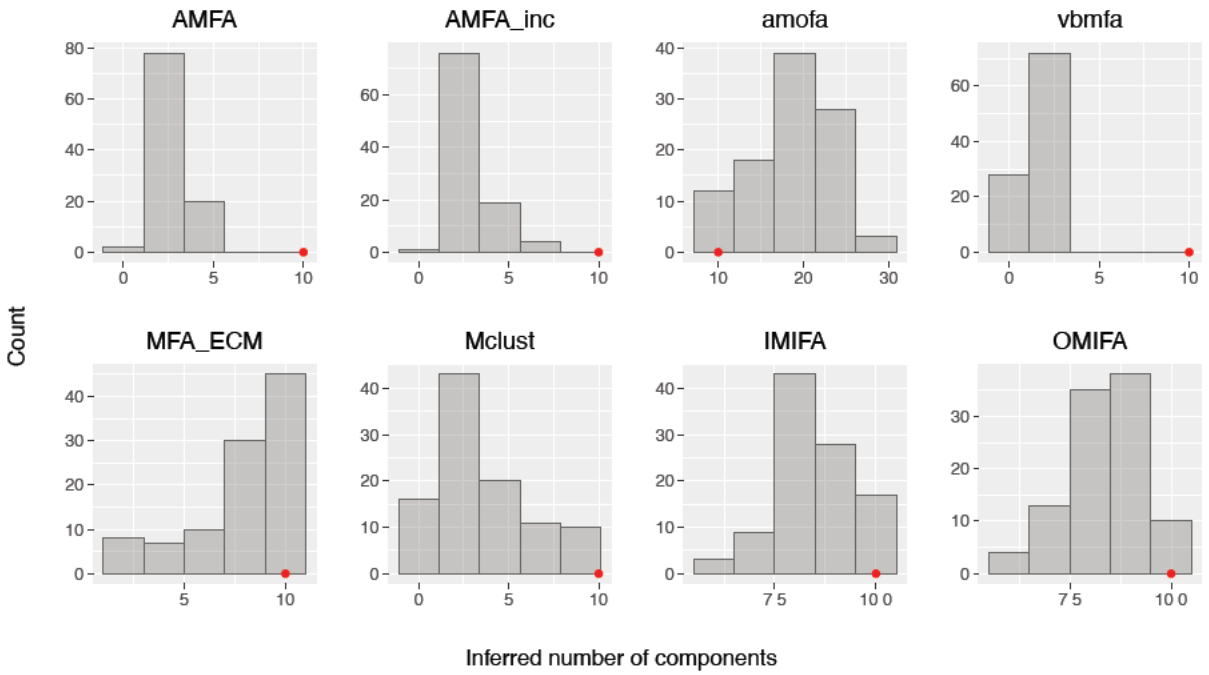


Figure 4.1: Inferred numbers of components for experiment group seven. The position of the red dot shows the true number of components.

underestimated the true number of components. Figure 4.4 shows that like the `amofa` method (from which the `AMFA_inc` method inherited its component splitting algorithm), the `AMFA_inc` method overestimated the number of components which were present in group eleven.

Finally, we refer to the Rasch model ability estimates contained in Table 4.12. We see that for both the inferring g exactly and inferring g to within an error of ± 2 criteria, the `MFA_ECM` method has the highest ability estimates. As suggested by the overall proportions in Table 4.8, while `AMFA` has the second highest ability estimate for inferring g exactly, `Mclust` has the second highest ability estimate for inferring g to within an error of ± 2. The `IMIFA` and `OMIFA` methods had the third and fourth highest ability estimates, respectively, for inferring g exactly, but slipped to fifth and sixth place, respectively, for inferring g to within an error of ± 2.

4.3.5 Inference on q

We compared how well the different methods inferred the number of factors per component, q, using the same two metrics as for the number of components, g.

Table 4.9 and Table 4.10 give the proportion of the experiments within each group where

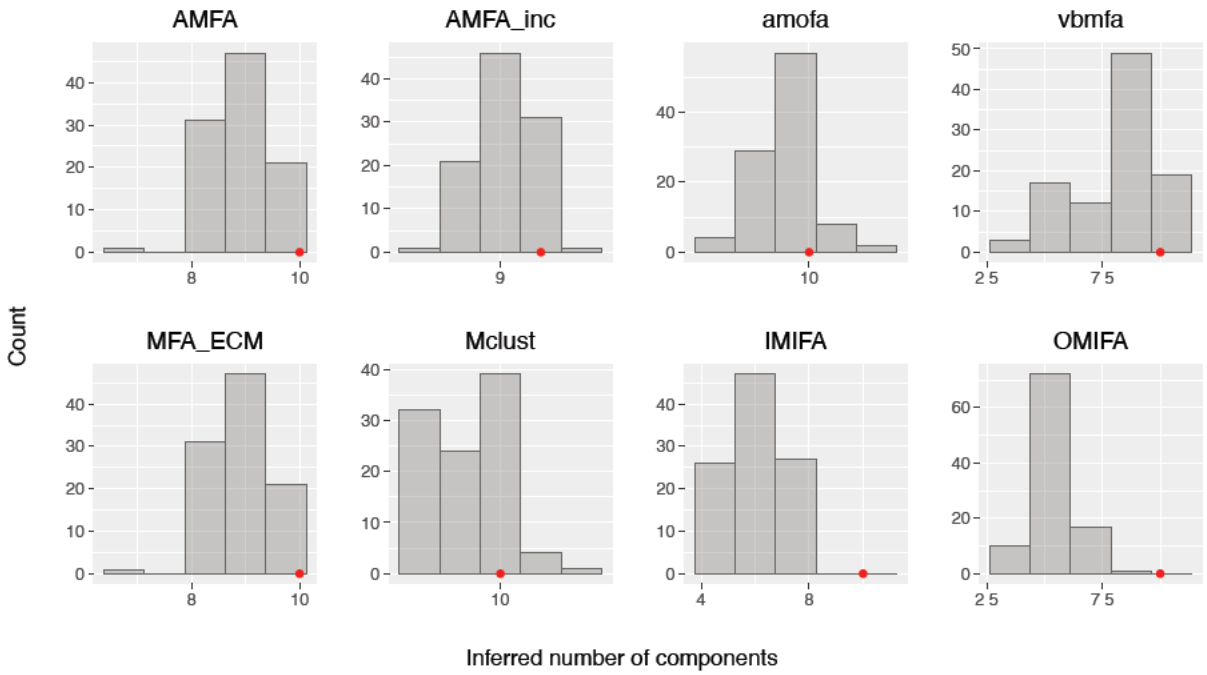


Figure 4.2: Inferred numbers of components for experiment group eight. The position of the red dot shows the true number of components.

each method inferred q correctly, and to within an error of ± 2, respectively. The ability estimates for both of these criteria under Rasch models are also given in Table 4.12.

Overall, the `MFA_ECM` method inferred q correctly the greatest proportion of the time. The next best method was `AMFA`, according to this metric. The least effective methods were `IMIFA` and `OMIFA`, which incorrectly inferred q for all 100 experiments in several groups. However, when we instead consider the overall proportion of the 1,200 experiments where the methods inferred the number of factors with an error of within ± 2, we get a very different result. On this metric, the `IMIFA` and `OMIFA` methods both achieved perfect proportions of 1.0 and were the joint best methods. The `MFA_ECM` method performed marginally worse, while the `amofa` and `AMFA_inc` methods performed poorly. This implies that while the `IMIFA` and `OMIFA` methods were not able to infer q exactly very often, they inferred values very close to the true value very consistently.

All of the methods struggled to infer q exactly in groups four and seven, which were both groups with $q = 6$. Figure 4.5 and Figure 4.6 show that in both cases, the methods included in `autoMFA` were tended to underestimate the true numbers of factors. This was also true of the `IMIFA` and `OMIFA` methods in group seven, although to a lesser extent, with the modal number of inferred factors for each method being five. However, in group four, the distribution of the inferred numbers of factors for `IMIFA` and `OMIFA` are relatively

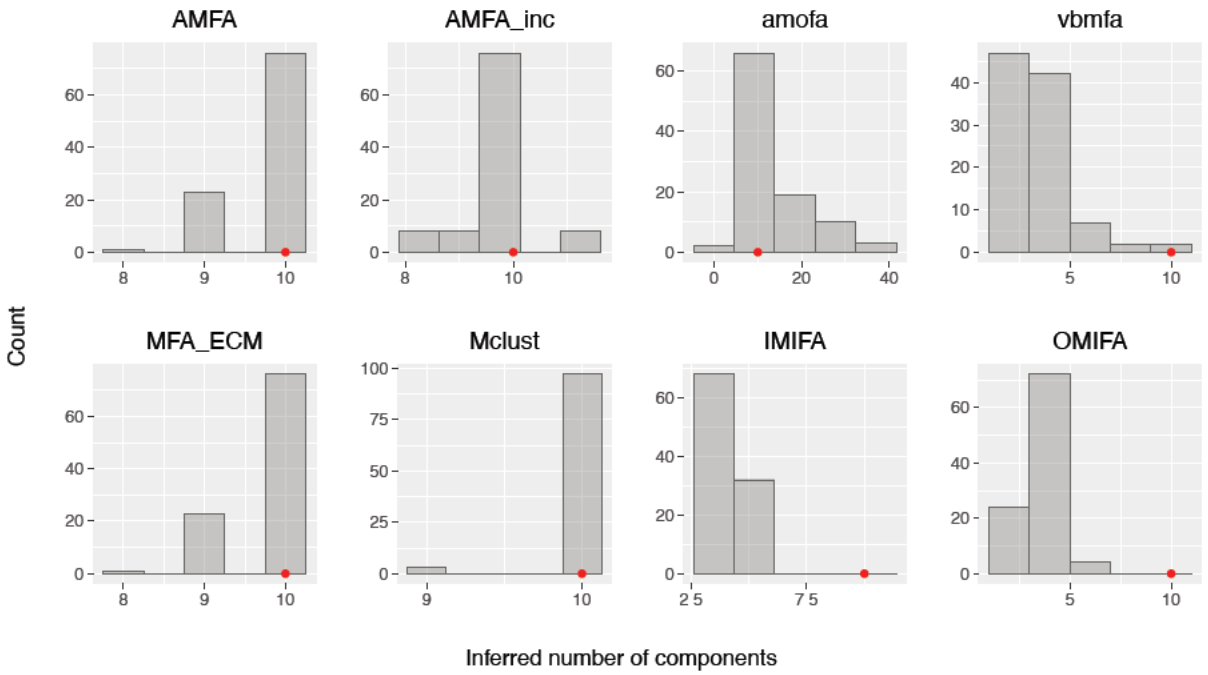


Figure 4.3: Inferred numbers of components for experiment group ten. The position of the red dot shows the true number of components.

uniform about the true value of $q = 6$.

Groups four and seven are also the only two groups where the `AMFA` method was unable to correctly infer the number of underlying factors to within an error of ± 2 100% of the time. Figure 4.5 and Figure 4.6 reveal that in both cases, the `AMFA` and `AMFA_inc` methods almost always underestimated the true number of factors. This suggests that the technique for automatically inferring q which both the `AMFA` and `AMFA_inc` methods utilise may perform less well as the true number of underlying factors increases.

The `amofa` method performed reasonably well for the experiment groups where $q = 1$ or $q = 3$, but very poorly on the groups where $q = 6$. This suggests that its performance scales poorly with the number of underlying factors.

Referring to the Rasch model ability estimates contained in Table 4.12, we see that the `MFA_ECM` method has the highest ability estimate for inferring q exactly, whereas the `IMIFA` and `OMIFA` methods have the joint highest ability estimate for inferring q to within an error of ± 2. This is in stark contrast to their ability estimates for inferring q exactly, where they have the two lowest ability estimates, by far.

To investigate whether inferring g correctly had an impact on inferring q correctly, we created Table 4.11 which shows the proportion of experiments where each method correctly

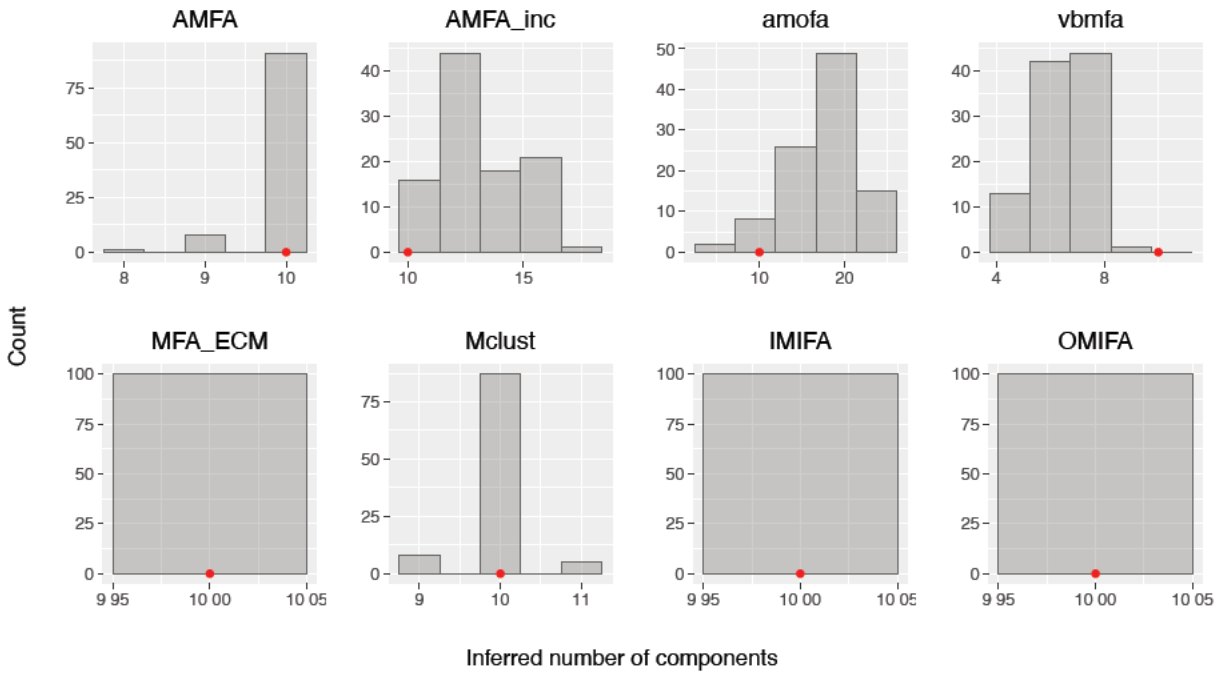


Figure 4.4: Inferred numbers of components for experiment group eleven. The position of the red dot shows the true number of components.

inferred q given that they correctly inferred g. We observe that for the `AMFA`, `AMFA_inc`, `MFA_ECM` and `IMIFA` methods, the proportion of experiments where q is inferred correctly increases slightly, however, the change is not particularly pronounced. In contrast, the proportion decreases very slightly for the `OMIFA` method. However, the most pronounced change is for the `amofa` method, where the proportion almost doubles. This is likely to be because the `amofa` method generally inferred g very poorly, often inferring many more components than were used to generate the data. This would result in a very different inferred sub-population structure than the true sub-population structure, so when q is inferred using the inferred sub-population structure, the results are poor relative to the true sub-population structure because they are so different. On the other hand, when the `amofa` method does infer g correctly its inference on q appears to be quite accurate.

4.4 Summary

The comparison shows that no one method was dominant across all of the criteria that we considered. Overall, the `MFA_ECM` method arguably performed the best as it had the highest mean ARI and lowest mean BIC across all 1,200 experiments, as well as the highest Rasch model ability estimates for inferring g correctly, inferring g to within an error of

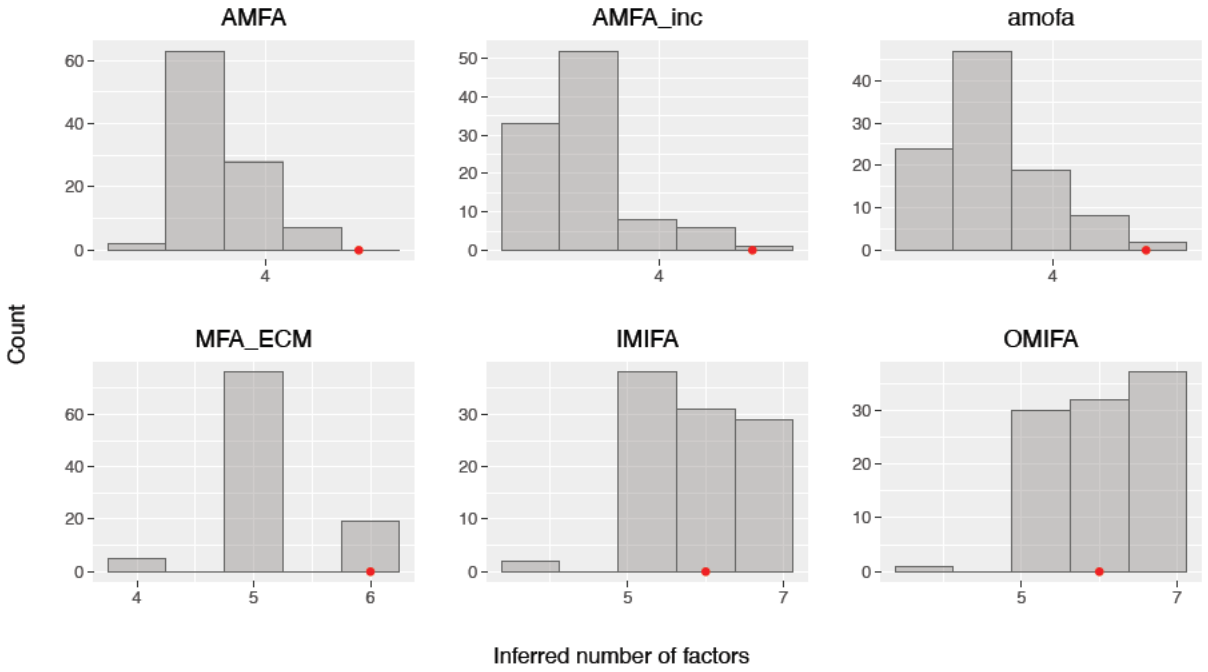


Figure 4.5: Inferred numbers of factors for experiment group four. The position of the red dot shows the true number of factors.

±2 and inferring q exactly. The `AMFA` method had the second highest overall mean ARI, second lowest overall mean BIC and the second highest ability estimates for each of the quantities that `MFA_ECM` had the highest estimates for (excluding `Mclust` which does not fit MFA models).

The `IMIFA` method achieved a lower overall mean BICM value compared to the `OMIFA` method. However, this was because the `IMIFA` method obtained drastically lower BICM values in group eight compared to the `OMIFA` method, which meant that its overall mean BICM was lower than that of the `OMIFA` method, even though in every other experiment group (apart from groups seven and ten), the `OMIFA` method obtained lower mean BICM values. Both methods generally achieved comparable results to the `MFA_ECM` and `AMFA` methods. The notable exception is the ability estimates of `IMIFA` and `OMIFA` for inferring the number of factors exactly, which were the lowest amongst any of the methods which could infer q. However, the `IMIFA` and `OMIFA` managed to infer q to within an error of ±2 for all of the 1,200 experiments. None of the other methods achieved this, so `IMIFA` and `OMIFA` had the highest ability estimates for inferring q to within an error of ±2. This suggests that they will produce reasonable estimates for q, even if they don't generally infer exactly the right value.

The `MFA_ECM` method had the highest overall mean fitting time. The overall mean fitting

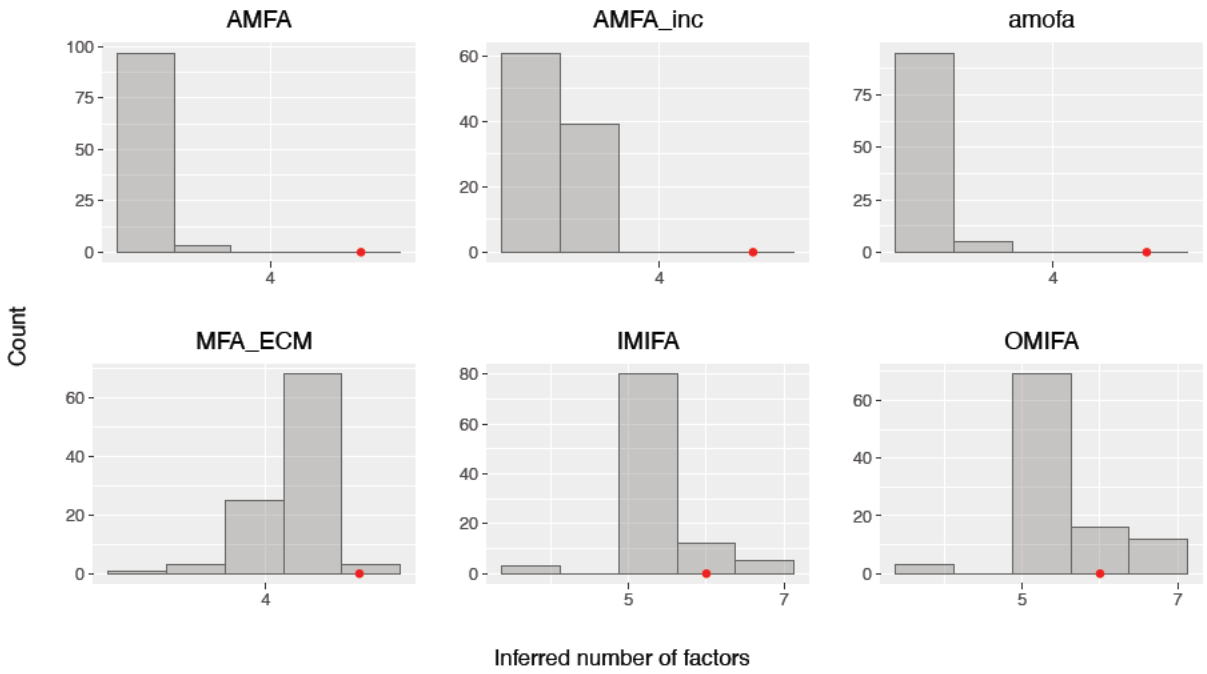


Figure 4.6: Inferred numbers of factors for experiment group seven. The position of the red dot shows the true number of factors.

time of the `AMFA` method was less than one-fifth of the overall mean fitting time of the `MFA_ECM` method. Given that it still achieved comparable performance to the `MFA_ECM` method on most metrics, it seems an appropriate choice when the `MFA_ECM` method is infeasible because of its high computational costs.

Given the favourable performance of the `MFA_ECM` and `AMFA` methods, we have produced an equivalent implementation of both in our Julia package FactorMixtures. In doing so, we hope to extend the range of datasets for which these two methods (but especially the naïve search) will be feasible by utilising Julia's computational efficiency.

Chapter 5

Extending the MFA Model

Having completed a review of several existing methods for automatically fitting the MFA model, we will now focus instead on extending the MFA model. Ideally, we would like to retain the parsimonious formulation of multivariate mixture models offered by the MFA model through its restricted covariance structure, whilst relaxing the condition that $\mathbf{Y}_j \mid Z_{ij} = 1$ must be Gaussian.

In this chapter, we will discuss two methods for achieving this, and show that the first method is a special case of the second. For each method, we also develop a novel, general EM-type parameter estimation algorithm based on ECM-MFA-2. We provide several examples of each class of model and derive complete EM-type algorithms for each example. Implementations of each of these EM-type algorithms can be found in our Julia package FactorMixtures.

5.1 Scale Mixtures of Normal Distributions

The first of the two methods which we will use to generalise the MFA model makes use of the Scale Mixtures of Normal Distributions model family, defined as follows.

Definition 5.1.1. The Scale Mixtures of Normal Distributions Model [Lange and Sinsheimer, 1993]

A p-dimensional random variable $\mathbf{Y}$ is said to follow a Scale Mixtures of Normal Distributions (SMN) model (also sometimes called a Normal Independent (NI) model) if it can be expressed as

$$\mathbf{Y} = \boldsymbol{\mu} + W^{-\frac{1}{2}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{X} \tag{5.1}$$

where $\mathbf{X} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I}_p)$ is independent of the positive, scalar random variable $W \sim h(w; \boldsymbol{\psi})$.

Equivalently, we can formulate the SMN model hierarchically as

$$\begin{aligned} W &\sim h(w; \boldsymbol{\psi}) \\ \mathbf{Y} \mid W = w &\sim \mathcal{N}_p(\boldsymbol{\mu}, w^{-1}\boldsymbol{\Sigma}). \end{aligned}$$

Here, $h(w; \boldsymbol{\psi})$ is the density function of the positive random variable W, which we will refer to as the scaling density. The vector $\boldsymbol{\psi}$ contains the parameters of W. Recall that we use the slight abuse of notation $W \sim h(w; \boldsymbol{\psi})$ to mean that W is a random variable from the distribution with density function given by $h(w; \boldsymbol{\psi})$.

The SMN model defines a whole family of distributions, which depend on the form of the scaling density $h(w; \boldsymbol{\psi})$. In comparison to a multivariate Gaussian model, the inclusion of the scaling variables in the SMN model allows for the rate of decay in the tails of its distributions to be altered. Exactly how the tails are altered, of course, depends on the form of $h(w; \boldsymbol{\psi})$. We propose combining the MFA model with the SMN model to form the Mixtures of Scale Mixtures of Normal Distributions Factor Analyzers (MSMNFA) model, defined hierarchically as

$$\begin{aligned} \mathbf{Z}_j &\sim \text{Multinomial}(1, \boldsymbol{\pi}), \\ W_j \mid Z_{ij} = 1 &\sim h_i(w_j; \boldsymbol{\psi}_i), \\ \mathbf{Y}_j \mid Z_{ij} = 1, W_j = w_j &\sim \mathcal{N}_p\left(\boldsymbol{\mu}_i, w_j^{-1}\left(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right)\right). \end{aligned} \tag{5.2}$$

The MSMNFA family of models retains the parsimonious mixture model formulation of the MFA model, whilst also inheriting the ability to have modified tails in each component of the mixture from the SMN model. This is in contrast to the regular MFA model, where the decay in the tails of each component is governed by the multivariate Gaussian density function. As a result, we expect that when compared with the MFA model, the MSMNFA model is more suitable for data with outliers or extreme values. We demonstrate this in practice in Chapter 6.

Note that in Equation (5.2), we allow each component in the mixture to have its own scaling density $h_i(w_j; \boldsymbol{\psi}_i)$. In practice, however, we generally assume that all of the scaling densities come from the same distribution. We still allow the parameters, $\boldsymbol{\psi}_i$, to differ from component to component, however.

5.2 Parameter Estimation for the MSMNFA Model

One of the attractive properties of the MSMNFA model is that it can model mixtures where the components are not multivariate Gaussian, whilst still being easy to fit using an EM-type scheme. In fact, we can generalise ECM-MFA-2 to apply to the whole MSMNFA family. To see this, first define $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ with

$$\boldsymbol{\theta}_1 = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_g, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_g)$$

and

$$\boldsymbol{\theta}_2 = (\mathbf{B}_1, \ldots, \mathbf{B}_g, \mathbf{D}_1, \ldots, \mathbf{D}_g).$$

For convenience, define $\mathrm{d}(\mathbf{a}, \mathbf{b}, \mathbf{C}, \mathbf{D}) := \mathbf{a}^\top(\mathbf{C}\mathbf{C}^\top + \mathbf{D})^{-1}\mathbf{b}$ for real $p \times 1$ vectors $\mathbf{a}$ and $\mathbf{b}$, a real $p \times q$ matrix $\mathbf{C}$ and a real $p \times p$ matrix $\mathbf{D}$ such that $(\mathbf{C}\mathbf{C}^\top + \mathbf{D})^{-1}$ exists, where $p, q \in \mathbb{N}^+$ and $p > q$. In addition, define $\mathrm{d}_{ij} = \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i, \mathbf{y}_j - \boldsymbol{\mu}_i, \mathbf{B}_i, \mathbf{D}_i)$ and $\mathrm{d}_{ij}^{(k)} = \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k)}, \mathbf{B}_i^{(k)}, \mathbf{D}_i^{(k)})$.

Let the complete data be given by $\mathbf{x}_j = (\mathbf{y}_j, \mathbf{z}_j, w_j)$. Then the complete-data log-likelihood is given by

$$\begin{aligned}\ell(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}, \mathbf{w}) \propto \sum_{j=1}^{n}\sum_{i=1}^{g}\Bigg[z_{ij}\log\pi_i + z_{ij}\log h_i(w_j;\boldsymbol{\psi}_i) - \frac{z_{ij}}{2}\log\left|\frac{\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i}{w_j}\right| \\ - \frac{1}{2}z_{ij}w_j\mathrm{d}_{ij}\Bigg].\end{aligned} \tag{5.3}$$

After noticing that

$$\log\left|\frac{\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i}{w_j}\right| = \log\left|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right| - \log w_j^p,$$

the log-likelihood simplifies slightly to

$$\begin{aligned}\ell(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}, \mathbf{w}) \propto \sum_{j=1}^{n}\sum_{i=1}^{g}\Bigg[z_{ij}\log\pi_i + z_{ij}\log h_i(w_j;\boldsymbol{\psi}_i) - \frac{z_{ij}}{2}\log\left|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right| \\ - \frac{1}{2}z_{ij}w_j\mathrm{d}_{ij}\Bigg].\end{aligned} \tag{5.4}$$

This is very similar to the complete-data log-likelihood of the regular MFA model given in Equation (2.33). The two differences being that the Mahalanobis distance term is now also scaled by a factor of w_j and the inclusion of the $\log h_i(w_j;\boldsymbol{\psi}_i)$ term.

The E-step is to calculate

$$\tau_{ij} = \mathbb{E}\left[Z_{ij} \mid \mathbf{Y}_j; \boldsymbol{\Theta}^{(k)}\right]$$

which will now be given by

$$\tau_{ij}^{(k)} = \frac{\pi_i^{(k)} f(\mathbf{y}_j \mid Z_{ij} = 1; \boldsymbol{\Theta}^{(k)})}{\sum_{l=1}^{g}\pi_l^{(k)} f(\mathbf{y}_j \mid Z_{lj} = 1; \boldsymbol{\Theta}^{(k)})} \tag{5.5}$$

and then to calculate

$$\mathbb{E}\left[Z_{ij}W_j \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)}\mathbb{E}\left[W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)}\xi_{ij}^{(k)}$$

where

$$\xi_{ij}^{(k)} := \mathbb{E}\left[W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]$$

and then any other conditional expectations of the form

$$\mathbb{E}\left[f(W_j, Z_{ij}) \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right]$$

which may arise from the $z_{ij} \log h_i(w_j; \boldsymbol{\psi}_i)$ term when Z_{ij} or W_j or both are attached to any of the parameters in $\boldsymbol{\psi}_i$.

The marginal density $f(\mathbf{y}_j \mid Z_{ij} = 1)$ in Equation (5.5) will be given by

$$f(\mathbf{y}_j \mid Z_{ij} = 1) = \int_0^\infty f(\mathbf{y}_j \mid Z_{ij} = 1, W_j) f(w_j \mid Z_{ij} = 1)\, \mathrm{d}w_j. \tag{5.6}$$

For specific choices of $h_i(w_j; \boldsymbol{\psi}_i)$ this may produce a known density function. Otherwise, it can be computed using numerical integration, which is feasible because the integral in Equation (5.6) is univariate.

The M-step for $\pi_i^{(k+1)}$ is given by Equation (2.30). The M-step for $\boldsymbol{\mu}_i^{(k+1)}$ is

$$\boldsymbol{\mu}_i^{(k+1)} = \frac{\sum_{j=1}^n \tau_{ij}^{(k)} \xi_{ij}^{(k)} \mathbf{y}_j}{\sum_{j=1}^n \tau_{ij}^{(k)} \xi_{ij}^{(k)}}, \tag{5.7}$$

which is a straightforward generalisation of Equation (2.35) that arises because the Mahalanobis distance in Equation (5.4) is scaled by w_j.

The update for $\boldsymbol{\psi}_i$ is

$$\boldsymbol{\psi}_i^{(k+1)} = \underset{\boldsymbol{\psi}_i}{\operatorname{argmax}} \left\{ \sum_{j=1}^n \mathbb{E}\left[Z_{ij} \log h_i(w_j; \boldsymbol{\psi}_i) \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right] \right\}.$$

Let $\boldsymbol{\Theta}^{(k+1/2)} := (\boldsymbol{\theta}_1^{(k+1)}, \boldsymbol{\theta}_2^{(k)})$. Given the form of Equation (5.4), it is clear that

$$\begin{aligned} Q(\boldsymbol{\theta}_2; \boldsymbol{\Theta}^{(k+1/2)}) \propto -\frac{1}{2} \sum_{j=1}^n \sum_{i=1}^g \Big[& \tau_{ij}^{(k)} \log |\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i| \\ & + \tau_{ij}^{(k)} \xi_{ij}^{(k)}\, \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i) \Big]. \end{aligned} \tag{5.8}$$

We notice that the only difference between the Q-function in Equation (5.8) and the Q-function given in Appendix A.2 (where we showed that Equation (2.36) holds under the standard MFA model) is the $\xi_{ij}^{(k)}$ attached to the Mahalanobis distance term. As a result,

we can apply the argument used to derive the M-step updates for $\mathbf{B}_i$ and $\mathbf{D}_i$ under the standard MFA model to derive the M-step updates for $\mathbf{B}_i$ and $\mathbf{D}_i$ under the MSMNFA model, with the updates given by Equation (2.46) and Equation (2.51) respectively, after redefining $\mathbf{S}_i^{(k)}$ as

$$\mathbf{S}_i^{(k)} := \frac{1}{n\pi_i^{(k+1)}} \sum_{j=1}^{n} \tau_{ij}^{(k)} \xi_{ij}^{(k)} (\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})^\top. \tag{5.9}$$

In Equation (5.3) we have assumed that $W_j \mid Z_{ij} = 1$ is a real-valued continuous random variable. However, the algorithm is almost exactly the same when $W_j \mid Z_{ij} = 1$ is a real-valued discrete random variable. To see why this is true, suppose that $W_j \mid Z_{ij} = 1$ is discrete, with a countable sample space given by $\mathcal{W} \subset \mathbb{R}$. We show in Appendix A.8 that

$$\begin{aligned} \ell(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}, \mathbf{w}) = &\sum_{j=1}^{n} \sum_{i=1}^{g} z_{ij} \log \pi_i \\ &+ \sum_{j=1}^{n} \sum_{i=1}^{g} \sum_{m \in \mathcal{W}} z_{ij} \mathbb{I}_{\{W_j = m\}} \log \left\{ \phi_p \left(\mathbf{y}_j; \boldsymbol{\mu}_i, \frac{1}{m}(\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i) \right) \right. \\ &\left. \times \Pr(W_j = m \mid Z_{ij} = 1; \boldsymbol{\Theta}) \right\}. \end{aligned} \tag{5.10}$$

Clearly, the π_i's can be updated as before. We now focus on the updates for $\mathbf{B}_i$ and $\mathbf{D}_i$. Keeping only the terms dependent on the variables in $\boldsymbol{\theta}_2$, we find that

$$\ell(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}, \mathbf{w}) \propto \sum_{j=1}^{n} \sum_{i=1}^{g} \sum_{m \in \mathcal{W}} z_{ij} \mathbb{I}_{\{W_j = m\}} \log \left\{ \phi_p \left(\mathbf{y}_j; \boldsymbol{\mu}_i, \frac{1}{m}(\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i) \right) \right\}$$

which we can further simplify to

$$\ell(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}, \mathbf{w}) \propto -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{g} \sum_{m \in \mathcal{W}} z_{ij} \mathbb{I}_{\{W_j = m\}} \left(\log |\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i| + m \mathrm{d}_{ij} \right),$$

This simplification follows because

$$\log \left| \frac{\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i}{w_j} \right| = \log |\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i| - \log w_j^p.$$

In the E-step we therefore find that

$$\begin{aligned} Q(\boldsymbol{\theta}_2; \boldsymbol{\Theta}^{(k+1/2)}) \propto -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{g} \sum_{m \in \mathcal{W}} \tau_{ij}^{(k)} \mathcal{I}_{jim}^{(k)} \Big[&\log |\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i| \\ &+ m \, \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i) \Big], \end{aligned} \tag{5.11}$$

where $\boldsymbol{\Theta}^{(k+1/2)}$ is defined as before and

$$\mathcal{I}_{jim}^{(k)} := \mathbb{E}\left[\mathbb{I}_{\{W_j=m\}} \mid \mathbf{y}_j, Z_{ij}=1; \boldsymbol{\Theta}^{(k)}\right] = \Pr(W_j = m \mid \mathbf{y}_j, Z_{ij}=1; \boldsymbol{\Theta}^{(k)}).$$

Finally, we observe that

$$\sum_{j=1}^{n}\sum_{i=1}^{g}\sum_{m\in\mathcal{W}} \tau_{ij}^{(k)}\mathcal{I}_{jim}^{(k)} \log|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i| = \sum_{j=1}^{n}\sum_{i=1}^{g} \tau_{ij}^{(k)} \log|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i|$$

because $\sum_{m\in\mathcal{W}} \mathcal{I}_{ijm}^{(k)} = 1$ and that

$$\begin{aligned}&\sum_{j=1}^{n}\sum_{i=1}^{g}\sum_{m\in\mathcal{W}} \tau_{ij}^{(k)} m\mathcal{I}_{jim}^{(k)}\, \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i)\\ &= \sum_{j=1}^{n}\sum_{i=1}^{g} \tau_{ij}^{(k)}\xi_{ij}^{(k)}\, \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i)\end{aligned}$$

because

$$\sum_{m\in\mathcal{W}} m\mathcal{I}_{jim}^{(k)} = \sum_{m\in\mathcal{W}} m \Pr(W_j = m \mid \mathbf{y}_j, Z_{ij}=1; \boldsymbol{\Theta}^{(k)}) = \xi_{ij}^{(k)}.$$

Hence,

$$\begin{aligned}Q(\boldsymbol{\theta}_2; \boldsymbol{\Theta}^{(k+1/2)}) \propto -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\Big[&\tau_{ij}^{(k)} \log|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i|\\ &+ \tau_{ij}^{(k)}\xi_{ij}^{(k)}\, \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i)\Big],\end{aligned} \tag{5.12}$$

So, we have shown that the Q-function for $\boldsymbol{\theta}_2$ when $W_j \mid Z_{ij} = 1$ is continuous, Equation (5.8), is the same as the Q-function for $\boldsymbol{\theta}_2$ when $W_j \mid Z_{ij} = 1$ is discrete, Equation (5.12). Hence, the updates for $\mathbf{B}_i$ and $\mathbf{D}_i$ will be the same as when $W_j \mid Z_{ij} = 1$ is continuous. A similar argument can be employed to show that Equation (5.10) also leads to the M-step update for $\boldsymbol{\mu}_i$ given in Equation (5.7). Therefore, the M-steps for all of the parameters except the $\boldsymbol{\psi}_i$'s are the same, whether the scaling density is discrete or continuous.

In the discrete case, the update for $\boldsymbol{\psi}_i$ is slightly different, because the support of $W_j \mid Z_{ij} = 1$ may depend on the parameters in $\boldsymbol{\psi}_i$. We can account for this by including all of the dependence on w_j in our Q-function for $\boldsymbol{\psi}$, which implies that

$$\begin{aligned}Q(\boldsymbol{\psi}; \boldsymbol{\Theta}^{(k)}) \propto \sum_{j=1}^{n}\sum_{i=1}^{g}\sum_{m\in\mathcal{W}} \tau_{ij}^{(k)}\mathcal{I}_{jim}^{(k)}\Big\{&\frac{p}{2}\log m - \frac{1}{2}m\mathrm{d}_{ij}^{(k)}\\ &+ \log\Pr(W_j = m \mid Z_{ij}=1; \boldsymbol{\Theta}^{(k)})\Big\}.\end{aligned} \tag{5.13}$$

Accordingly, our M-step for $\boldsymbol{\psi}_i$ is

$$\boldsymbol{\psi}_i^{(k+1)} = \underset{\boldsymbol{\psi}_i}{\operatorname{argmax}} \left\{ \sum_{j=1}^{n} \sum_{m \in \mathcal{W}} \tau_{ij}^{(k)} \mathcal{I}_{jim}^{(k)} \left\{ \frac{p}{2} \log m - \frac{1}{2} m \mathrm{d}_{ij}^{(k)} + \log \Pr(W_j = m \mid Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}) \right\} \right\}. \tag{5.14}$$

Depending on the form of $h_i(w_j; \boldsymbol{\psi}_i)$, Equation (5.14) may contain redundant terms. In fact, any term belonging to a realisation of $W_j \mid Z_{ij} = 1$ which is not a parameter of $h_i(w_j; \boldsymbol{\psi}_i)$ can be removed.

Since we were able to apply a generalisation of Algorithm 2.4 to the MSMNFA model, a natural question to ask is whether we can also generalise the approach used to determine q in the AMFA algorithm from Chapter 3 to work for the MSMNFA model. Given the similarity between Equation (5.4) and Equation (2.33), the M-step update for q proposed in Wang and Lin [2020] can be generalised to the MSMNFA model by performing the update

$$q^{(k+1)} = \underset{q \leq q_{\max}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{g} n \pi_i^{(k+1)} \sum_{l=1}^{q} (\log \lambda_{il} - \lambda_{il} + 1) + k^*_{\mathrm{MSMNFA}}(g, q) \log n \right\}. \tag{5.15}$$

In the context of the MSMNFA model, λ_{il} is the l^{th} largest eigenvalue of the matrix

$$\tilde{\mathbf{S}}_i := \left[\mathbf{D}_i^{(k)} \right]^{-1/2} \mathbf{S}_i^{(k)} \left[\mathbf{D}_i^{(k)} \right]^{-1/2},$$

using the updated definition of $\mathbf{S}_i^{(k)}$ from Equation (5.9). The number of parameters being estimated in the MSMNFA model is just the number of parameters used to define the MFA model plus the number of parameters required to define each component scaling density $h_i(w_j; \boldsymbol{\psi}_i)$, so

$$k^*_{\mathrm{MSMNFA}}(p, g, q) = k^*_{\mathrm{MFA}}(p, g, q) + \sum_{i=1}^{g} \mathrm{n_p}(\boldsymbol{\psi}_i),$$

where $\mathrm{n_p}$ counts the number of parameters in $\boldsymbol{\psi}_i$.

ECM-MSMNFA-1 (Algorithm 5.1) summarises the ECM algorithm obtained by generalising ECM-MFA-2 to the MSMNFA model family. Note that formulas for $\xi_{ij}^{(k)}$ and $\boldsymbol{\psi}_i^{(k+1)}$ cannot be given in general, as they depend on the distribution of the scaling variables W_j. In the following section, three particular cases of the MSMNFA model family are considered, and all necessary formulas for the E-steps and M-steps are given in each case, so that ECM-MSMNFA-1 can be implemented.

Algorithm 5.1: ECM-MSMNFA-1: General ECM algorithm for the MSMNFA model family

Input: An initial estimate of the parameters, $\boldsymbol{\Theta}^{(0)}$, a convergence criterion and a maximum number of iterations k_{max}

Result: Maximum likelihood estimates for the vector of parameters $\boldsymbol{\Theta}$

Set $k = 0$;
while *Chosen convergence criterion not satisfied and* $k < k_{max}$ **do**
  The E-Step: Compute the responsibilities $\tau_{ij}^{(k)}$ using Equation (5.5), as well as $\xi_{ij}^{(k)}$ and any other necessary E-step estimates as given in the subsection for each particular case of the MSMNFA model family, for $i \in \{1, \ldots, g\}$ and $j \in \{1, \ldots, n\}$;
  The M-Step: Compute $\boldsymbol{\Theta}^{(k+1)}$ using Equation (2.30), Equation (5.7), Equation (2.46) and Equation (2.51) (using the definition of $\mathbf{S}_i^{(k)}$ given in Equation (5.9)), respectively, as well as the specified M-step updates for the parameters in $\boldsymbol{\psi}_i$ as given in the subsection for each particular case of the MSMNFA model family;
  if *Convergence criterion satisfied* **then**
    Return $\boldsymbol{\Theta}^{(k+1)}$;
  else
    Set $k = k + 1$;
  end
end
Return $\boldsymbol{\Theta}^{(k_{\max})}$;

5.3 Examples of the MSMNFA Model

Having derived a general EM-type algorithm for the MSMNFA model, we now present three specific instances with full EM-type algorithms given for each.

5.3.1 The MtFA Model

The Mixtures of t Factor Analyzers (MtFA) model is perhaps the most well known special case of the MSMNFA model family. It was initially proposed by McLachlan et al. [2007], who wanted to develop a factor model which was less sensitive to outliers than the original MFA model. McLachlan et al. [2007] also proposed an AECM algorithm for parameter estimation of the MtFA model. More recently, Wang and Lin [2012] generalised ECM-MFA-2 to the MtFA model, and showed that it generally performed better than the AECM algorithm in terms of CPU time and numbers of iterations until convergence was achieved. In other words, Wang and Lin [2012] gave the particular case of ECM-MSMNFA-1 which applies to the MtFA model.

The MtFA model is obtained by taking

$$W_j \mid Z_{ij} = 1 \sim \text{Gamma}\left(\frac{\nu_i}{2}, \frac{\nu_i}{2}\right).$$

This model is called the MtFA model because

$$\mathbf{Y}_j \mid Z_{ij} = 1 \sim t_p\left(\boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \nu_i\right),$$

which we demonstrate in Appendix A.9. That is, $\mathbf{Y}_j \mid (Z_{ij} = 1)$ follows the p-dimensional t-distribution with location vector $\boldsymbol{\mu}_i$, scale matrix $\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i$ and ν_i degrees of freedom. Recall that the t-distribution has heavier tails than the Gaussian distribution. As a result, we expect that the MtFA model will be more robust to extreme observations compared to the regular MFA model.

Under the MtFA model, we assume that

$$h_i(w_j; \boldsymbol{\psi}_i) = \frac{\left(\frac{\nu_i}{2}\right)^{\nu_i/2}}{\Gamma\left(\frac{\nu_i}{2}\right)} w_j^{\frac{\nu_i}{2}-1} e^{-\frac{\nu_i}{2}w_j}.$$

Substituting this into Equation (5.2), our Q-function for the degrees of freedom, ν_i, is

$$Q\left(\boldsymbol{\nu}; \boldsymbol{\Theta}^{(k)}\right) \propto \sum_{j=1}^{n}\sum_{i=1}^{g} \tau_{ij}^{(k)} \left[\frac{\nu_i}{2}\log\frac{\nu_i}{2} - \log\Gamma\left(\frac{\nu_i}{2}\right) + \frac{\nu_i}{2}(\zeta_{ij}^{(k)} - \xi_{ij}^{(k)})\right] \tag{5.16}$$

where

$$\zeta_{ij}^{(k)} := \mathbb{E}\left[\log W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]. \tag{5.17}$$

This is the only other conditional expectation which needs to be calculated for the MtFA model.

We show in Appendix A.10 that

$$W_j \mid \mathbf{Y}_j, Z_{ij} = 1 \sim \text{Gamma}\left(\frac{\nu_i + p}{2}, \frac{\nu_i + \text{d}_{ij}}{2}\right).$$

Accordingly,

$$\xi_{ij}^{(k)} = \frac{\nu_i^{(k)} + p}{\nu_i^{(k)} + \text{d}_{ij}^{(k)}},$$

and

$$\zeta_{ij}^{(k)} = \text{DG}\left(\frac{\nu_i^{(k)} + p}{2}\right) - \log\left(\frac{\nu_i^{(k)} + \text{d}_{ij}^{(k)}}{2}\right), \tag{5.18}$$

where

$$\text{DG}(x) := \frac{\Gamma(x)}{\Gamma'(x)}$$

is the digamma function. Proof of the latter result is deferred to Section 5.3.2.

Given the framework above, the only M-step we need to evaluate is for ν_i. Using Equation (5.16), we find that

$$\frac{\text{d}Q}{\text{d}\nu_i} = \sum_{j=1}^{n} \tau_{ij}^{(k)} \left[\frac{1}{2}\log\frac{\nu_i}{2} + \frac{1}{2} - \frac{1}{2}\text{DG}\left(\frac{\nu_i}{2}\right) + \frac{1}{2}(\zeta_{ij}^{(k)} - \xi_{ij}^{(k)})\right],$$

which implies that $\nu_i^{(k+1)}$ will be given by the solution to

$$\log\frac{\nu_i}{2} + 1 - \text{DG}\left(\frac{\nu_i}{2}\right) + \frac{\sum_{j=1}^{n} \tau_{ij}^{(k)}(\zeta_{ij}^{(k)} - \xi_{ij}^{(k)})}{\sum_{j=1}^{n} \tau_{ij}^{(k)}} = 0,$$

for each $i \in \{1, \ldots, g\}$. This expression can be solved numerically.

For the MtFA model, we only need to estimate one parameter for each of the component density functions $h_i(w_j; \boldsymbol{\psi}_i)$, which is the degrees of freedom, ν_i. So $\text{n}_\text{s}(\boldsymbol{\psi}_i) = 1$ for each i, making

$$k^*_{\text{MtFA}} = k^*_{\text{MFA}}(p, g, q) + g.$$

This is all of the information required to implement ECM-MSMNFA-1 for the MtFA model.

5.3.2 The MSLFA Model

Another special case of the MSMNFA family is the Mixture of Slash Factor Analyzers (MSLFA) model. Under this model, we take

$$h_i(w_j;\boldsymbol{\psi}_i) = \nu_i w_j^{\nu_i - 1}\mathbb{I}_{w_j\in(0,1)} \tag{5.19}$$

for $\nu_i > 0$. It is not difficult to show that with this choice of scaling density,

$$f(\mathbf{y}_j \mid Z_{ij} = 1;\boldsymbol{\Theta}) = \nu_i \int_0^1 w_j^{\nu_i-1}\phi_p\left(\mathbf{y}_j;\boldsymbol{\mu}_i,\frac{1}{w_j}(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)\right)\mathrm{d}w_j. \tag{5.20}$$

The following definition will help us to identify the marginal distribution which results from using the scaling density in Equation (5.19).

Definition 5.3.1. The Slash Distribution [Wang and Genton, 2006]

A p-dimensional random variable $\mathbf{Y}$ is said to have the multivariate slash distribution with parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and ν if

$$\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{-\frac{1}{2}}\frac{\mathbf{X}}{U^{1/\nu}} \tag{5.21}$$

where $\mathbf{X} \sim \mathcal{N}_p(\mathbf{0},\mathbf{I}_p)$ is independent of $U \sim \mathrm{U}(0,1)$. We denote this as $\mathbf{Y} \sim \mathrm{SL}_p(\boldsymbol{\mu},\boldsymbol{\Sigma},\nu)$.

Noting the similarity between Equation (5.21) and the definition of a Scale Mixtures of Normal Distributions model given in Equation (5.1), it is clear that taking $W = U^{2/\nu}$ in Equation (5.1) (where $U \sim \mathrm{U}(0,1)$) will produce $\mathbf{Y} \sim \mathrm{SL}_p(\boldsymbol{\mu},\boldsymbol{\Sigma},\nu)$.

Taking $W_j \mid Z_{ij} = 1 = U^{1/\nu_i}$, which is equivalent to Equation (5.19), implies that $\mathbf{Y}_j \mid Z_{ij} = 1 \sim \mathrm{SL}_p(\boldsymbol{\mu},\boldsymbol{\Sigma},2\nu_i)$. Using results from Wang and Genton [2006], this leads to the density

$$f(\mathbf{y}_j \mid Z_{ij} = 1;\boldsymbol{\Theta}) = \begin{cases} \frac{2\nu_i\cdot 2^{\nu_i+p/2-1}\gamma(\nu_i+p/2;\mathrm{d}_{ij}^{(k)}/2)}{(2\pi)^{p/2}\mathrm{d}_{ij}^{(k)\,\nu_i+p/2}}, & \mathbf{y}_j \neq \mathbf{0}, \\ \frac{2\nu_i}{2\nu_i+p}(2\pi)^{-\frac{p}{2}}, & \mathbf{y}_j = \mathbf{0}, \end{cases} \tag{5.22}$$

which is equivalent to Equation (5.20). Here,

$$\gamma(a;z) := \int_0^z t^{a-1}e^{-t}\,\mathrm{d}t = \sum_{k=0}^{\infty}\frac{(-1)^k z^{a+k}}{k!(a+k)}$$

is the lower incomplete Gamma function. Equation (5.22) is a useful form for the density function of the MSLFA model because it is faster to evaluate than using numerical integration to calculate Equation (5.20).

Intuitively, in Definition 5.3.1 we see that the slash distribution can be constructed by taking a multivariate Gaussian random variable and then dividing it by a uniform random

variable on $[0, 1]$ raised to the (positive) power $1/\nu$. In particular, when ν is very small, we note that $U^{1/\nu}$ will also be very small, so the magnitude of the fraction $\mathbf{X}/U^{1/\nu}$ will be very large. As a result, we expect that the Slash Distribution will be able to model data with very extreme observations effectively.

It follows from a simple Bayesian argument that

$$f(w_j \mid \mathbf{y}_j, Z_{ij} = 1) \propto w_j^{\nu_i + \frac{p}{2} - 1} \exp\left(-\frac{1}{2} w_j \mathrm{d}_{ij}^{(k)}\right) \mathbb{I}_{w_j \in (0,1)}$$

which implies that

$$W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)} \sim \text{Truncated Gamma}\left(\nu_i^{(k)} + \frac{p}{2}, \frac{1}{2}\mathrm{d}_{ij}^{(k)}, 0, 1\right).$$

Using results from Coffey and Muller [2000], we find that

$$\xi_{ij}^{(k)} = \frac{2(\nu_i^{(k)} + \frac{p}{2})}{\mathrm{d}_{ij}^{(k)}} \frac{\mathrm{F}_\Gamma\left(1; \nu_i^{(k)} + \frac{p}{2} + 1, \frac{1}{2}\mathrm{d}_{ij}^{(k)}\right)}{\mathrm{F}_\Gamma\left(1; \nu_i^{(k)} + \frac{p}{2}, \frac{1}{2}\mathrm{d}_{ij}^{(k)}\right)},$$

where F_Γ is the cumulative distribution function of the Gamma distribution.

Our Q-function for the shape parameter, ν_i, is

$$Q\left(\boldsymbol{\nu}; \boldsymbol{\Theta}^{(k)}\right) \propto \sum_{j=1}^{n} \sum_{i=1}^{g} \tau_{ij}^{(k)} \left[\log \nu_i + (\nu_i - 1)\zeta_{ij}^{(k)} + \log \mathbb{I}_{W_j \in (0,1)}\right] \tag{5.23}$$

with

$$\zeta_{ij}^{(k)} = \mathbb{E}\left[\log W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right].$$

To evaluate ζ_{ij}, we present the following Lemma.

Lemma 5.3.2. Suppose

$$X \sim \text{Truncated Gamma}(\alpha, \beta, 0, t).$$

Then

$$\mathbb{E}\left[\log X\right] = \frac{1}{\mathrm{F}_\Gamma(t, \alpha, \beta)} \frac{\beta^\alpha}{\Gamma(\alpha)} \int_0^t x^{\alpha - 1} e^{-\beta x} \log x \, \mathrm{d}x,$$

where F_Γ is the cumulative distribution function of the Gamma distribution.

Proof. Suppose

$$X \sim \text{Truncated Gamma}(\alpha, \beta, 0, t).$$

Then

$$f_X(x) = \frac{1}{\mathrm{F}_\Gamma(t;\alpha,\beta)} \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}. \tag{5.24}$$

Defining $Y := \log X$, the usual change of variables formula shows that

$$f_Y(y) = \frac{1}{\mathrm{F}_\Gamma(t;\alpha,\beta)} \frac{\beta^\alpha}{\Gamma(\alpha)} e^{\alpha y - \beta e^y}.$$

Note that the range of Y is from $-\infty$ to $\log t$. Trivially, we must have that

$$\int_{-\infty}^{\log t} e^{\alpha y - \beta e^y} \, \mathrm{d}y = \mathrm{F}_\Gamma(t;\alpha,\beta) \frac{\Gamma(\alpha)}{\beta^\alpha}. \tag{5.25}$$

Also, notice that

$$\frac{\partial}{\partial \alpha} e^{\alpha y - \beta e^y} = y e^{\alpha y - \beta e^y} = \mathrm{F}_\Gamma(t;\alpha,\beta) \frac{\Gamma(\alpha)}{\beta^\alpha} y f_Y(y).$$

Hence,

$$\begin{aligned}
\mathbb{E}[Y] &= \int_{-\infty}^{\log t} y f_Y(y) \\
&= \frac{1}{\mathrm{F}_\Gamma(t;\alpha,\beta)} \frac{\beta^\alpha}{\Gamma(\alpha)} \int_{-\infty}^{\log t} \frac{\partial}{\partial \alpha} e^{\alpha y - \beta e^y} \, \mathrm{d}y \\
&= \frac{1}{\mathrm{F}_\Gamma(t;\alpha,\beta)} \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\partial}{\partial \alpha} \int_{-\infty}^{\log t} e^{\alpha y - \beta e^y} \, \mathrm{d}y \\
&= \frac{1}{\mathrm{F}_\Gamma(t;\alpha,\beta)} \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\partial}{\partial \alpha} \left(\mathrm{F}_\Gamma(t;\alpha,\beta) \frac{\Gamma(\alpha)}{\beta^\alpha} \right),
\end{aligned}$$

which follows from Equation (5.25). Evaluating the partial derivative and simplifying yields

$$\mathbb{E}[Y] = \frac{\frac{\partial}{\partial \alpha} \mathrm{F}_\Gamma(t;\alpha,\beta)}{\mathrm{F}_\Gamma(t;\alpha,\beta)} + \mathrm{DG}(\alpha) - \log \beta, \tag{5.26}$$

since $\Gamma'(\alpha) = \mathrm{DG}(\alpha)\Gamma(\alpha)$. By definition,

$$\mathrm{F}_\Gamma(t,\alpha,\beta) = \int_0^t \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \, \mathrm{d}x.$$

Using this fact, we evaluate $\frac{\partial}{\partial \alpha} \mathrm{F}_\Gamma(t,\alpha,\beta)$ by first applying the product rule:

$$\frac{\partial}{\partial \alpha} \mathrm{F}_\Gamma(t,\alpha,\beta) = \left(\frac{\partial}{\partial \alpha} \frac{\beta^\alpha}{\Gamma(\alpha)} \right) \int_0^t x^{\alpha-1} e^{-\beta x} \, \mathrm{d}x + \left(\frac{\partial}{\partial \alpha} \int_0^t x^{\alpha-1} e^{-\beta x} \, \mathrm{d}x \right) \frac{\beta^\alpha}{\Gamma(\alpha)}. \tag{5.27}$$

By applying the quotient rule to the first derivative in Equation (5.27) and swapping the order of integration and differentiation before evaluating the second, we see that

$$\frac{\partial}{\partial\alpha}\,\mathrm{F}_\Gamma(t,\alpha,\beta) = (\log\beta - \mathrm{DG}(\alpha))\,\mathrm{F}_\Gamma(t,\alpha,\beta) + \frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^t x^{\alpha-1}e^{-\beta x}\log x\,\mathrm{d}x. \tag{5.28}$$

Substituting Equation (5.28) into Equation (5.26) yields

$$\mathbb{E}\left[Y\right] = \frac{1}{\mathrm{F}_\Gamma(t,\alpha,\beta)}\frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^t x^{\alpha-1}e^{-\beta x}\log x\,\mathrm{d}x,$$

the desired result. □

Corollary 5.3.2.1. Suppose

$$X \sim \mathrm{Gamma}(\alpha,\beta).$$

Then

$$\mathbb{E}\left[\log X\right] = \mathrm{DG}(\alpha) - \log\beta.$$

Proof. We replace Equation (5.24) with

$$f_X(x) = \frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}.$$

Noting that the only difference is the lack of the CDF term, the result follows after applying the same argument. □

Corollary 5.3.2.1 allows us to calculate $\zeta_{ij}^{(k)}$ under the MtFA model as given in Equation (5.18).

Using Lemma 5.3.2, we can evaluate $\zeta_{ij}^{(k)}$ for the MSLFA model as

$$\zeta_{ij}^{(k)} = \frac{1}{\mathrm{F}_\Gamma\left(1,\nu_i^{(k)}+\frac{p}{2},\frac{1}{2}\mathrm{d}_{ij}^{(k)}\right)}\frac{\left[\frac{1}{2}\mathrm{d}_{ij}^{(k)}\right]^{\nu_i^{(k)}+\frac{p}{2}}}{\Gamma(\nu_i^{(k)}+\frac{p}{2})}\int_0^1 u^{\nu_i^{(k)}+\frac{p}{2}-1}e^{-\frac{1}{2}u\mathrm{d}_{ij}^{(k)}}\log u\,\mathrm{d}u. \tag{5.29}$$

The required integral can be calculated numerically, which, like Equation (5.6), is also feasible because it is univariate.

Like in the MtFA model, we only need to work out a single M-step: that of ν_i. Using Equation (5.23), we find that

$$\frac{\mathrm{d}Q}{\mathrm{d}\nu_i} = \sum_{j=1}^n \tau_{ij}^{(k)}\left[\frac{1}{\nu_i} + \zeta_{ij}^{(k)}\right],$$

which implies that

$$\nu_i^{(k+1)} = -\frac{\sum_{j=1}^{n} \tau_{ij}^{(k)}}{\sum_{j=1}^{n} \tau_{ij}^{(k)} \zeta_{ij}^{(k)}}, \tag{5.30}$$

for $i = 1, \ldots, g$. Note that the leading negative in Equation (5.23) may at first glance look like an infeasible update because $\nu_i > 0$, but recall that the support of W_j is $(0, 1)$ which implies that $\zeta_{ij}^{(k)} < 0$, while $0 \leq \tau_{ij}^{(k)} \leq 1$ and hence $\nu_i^{(k+1)} \geq 0$.

Finally, using the same argument as for the MtFA model, we find that

$$k^*_{\text{MSLFA}}(p, g, q) = k^*_{\text{MFA}}(p, g, q) + g.$$

This is all of the information required to implement ECM-MSMNFA-1 for the MSLFA model.

5.3.3 The MCNFA Model

Another example of a MSMNFA model is the Mixture of Contaminated Normal Factor Analyzers (MCNFA) model.

Under this model we assume that

$$h_i(w_j; \boldsymbol{\psi}_i) = \nu_i \mathbb{I}_{\{w_j = \gamma_i\}} + (1 - \nu_i)\mathbb{I}_{\{w_j = 1\}},$$

for $0 \leq \nu_i \leq 1$ and $0 \leq \gamma_i \leq 1$. Equivalently, we can represent this as the PMF

$$\Pr(W_j = w_j \mid Z_{ij} = 1) = \begin{cases} \nu_i & \text{for } w_j = \gamma_i \\ 1 - \nu_i & \text{for } w_j = 1. \end{cases}$$

The law of total probability dictates that

$$f(\mathbf{y}_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) = \nu_i \phi_p \left(\mathbf{y}_j; \boldsymbol{\mu}_i, \frac{1}{\gamma_i}(\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i) \right) + (1 - \nu_i)\phi_p(\mathbf{y}_j; \boldsymbol{\mu}_i, \mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i),$$

so

$$\mathbf{Y}_j \mid Z_{ij} = 1 \sim \text{CN}_p \left(\boldsymbol{\mu}_i, \mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i, \nu_i, \gamma_i \right),$$

the p-dimensional Contaminated Normal distribution from Tukey [1960].

The Contaminated Normal model is a GMM with mixing proportions $(\nu_i, 1 - \nu_i)$. The only difference in the component densities is the scaling of the covariance matrix $\boldsymbol{\Sigma}_i$ in one component. Since $0 \leq \gamma_i \leq 1$, the division of $\boldsymbol{\Sigma}_i$ by γ_i in one of the components magnifies the elements of that covariance matrix. This means that the observations belonging to this component may be more extreme. An intuitive application of the Contaminated Normal model is a dataset where the observations appear to have been generated for a

Gaussian distribution, except that some of the data points are too extreme to have been plausibly generated from exactly the same covariance matrix, instead appearing to have been generated from a magnified version of the same covariance matrix.

It is worth noting that Punzo and McNicholas [2016] have previously proposed a similar set of parsimonious mixture models which make use of the p-dimensional Contaminated Normal distribution. However, in their models, parsimony is enforced by placing constraints on the eigen-decomposition of $\boldsymbol{\Sigma}_i$ using the methodology of Celeux and Govaert [1995], instead of via the factor analytic covariance structure that we employ.

We will use the discrete formulation of the MSMNFA model from the beginning of this chapter to derive M-step estimates for ν_i and γ_i. Let the conditional expectation of the indicator variables for W_j be given by

$$\rho_{ij} := \mathbb{E}\left[\mathbb{I}_{\{W_j=1\}} \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right]$$

and

$$\kappa_{ij} := \mathbb{E}\left[\mathbb{I}_{\{W_j=\gamma_i\}} \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right].$$

Using Bayes' rule,

$$\begin{aligned}\rho_{ij}^{(k)} &= \Pr(W_j = 1 \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}) \\ &= \frac{(1-\nu_i^{(k)})\phi_{ijk,1}}{(1-\nu_i^{(k)})\phi_{ijk,1} + \nu_i^{(k)}\phi_{ijk,\gamma_i^{(k)}}},\end{aligned}$$

where

$$\phi_{ijk,1} := \phi_p\left(\mathbf{y}_j; \boldsymbol{\mu}_i^{(k)}, \mathbf{B}_i^{(k)}\mathbf{B}_i^{\top(k)} + \mathbf{D}_i^{(k)}\right)$$

and

$$\phi_{ijk,\gamma_i^{(k)}} = \phi_p\left(\mathbf{y}_j; \boldsymbol{\mu}_i^{(k)}, (\mathbf{B}_i^{(k)}\mathbf{B}_i^{\top(k)} + \mathbf{D}_i^{(k)})/\gamma_i^{(k)}\right).$$

It follows that we can calculate $\kappa_{ij}^{(k)}$ and $\xi_{ij}^{(k)}$ as

$$\begin{aligned}\kappa_{ij}^{(k)} &= \Pr(W_j = \gamma_i \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}) \\ &= 1 - \rho_{ij}^{(k)}\end{aligned}$$

and

$$\begin{aligned}\xi_{ij}^{(k)} &= \mathbb{E}\left[W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] \\ &= \rho_{ij}^{(k)} + \gamma_i^{(k)}\kappa_{ij}^{(k)},\end{aligned}$$

respectively. Beginning with Equation (5.10), by keeping only the terms which depend on $\boldsymbol{\nu}$ and $\boldsymbol{\gamma}$ and after expanding out the sum over $m \in \mathcal{W}$ we obtain

$$Q(\boldsymbol{\nu}, \boldsymbol{\gamma}; \boldsymbol{\Theta}^{(k)}) \propto \sum_{j=1}^{n} \sum_{i=1}^{g} \left\{ \frac{p}{2} \tau_{ij}^{(k)} \kappa_{ij}^{(k)} \log \gamma_i - \frac{1}{2} \tau_{ij}^{(k)} \kappa_{ij}^{(k)} \gamma_i \mathrm{d}_{ij}^{(k)} + \tau_{ij}^{(k)} \kappa_{ij}^{(k)} \log \nu_i + \tau_{ij}^{(k)} \rho_{ij}^{(k)} \log(1 - \nu_i) \right\}. \tag{5.31}$$

Next, we find that

$$\frac{\partial Q}{\partial \nu_i} = \sum_{j=1}^{n} \frac{\tau_{ij}^{(k)} \kappa_{ij}^{(k)}}{\nu_i} - \frac{\tau_{ij}^{(k)} \rho_{ij}^{(k)}}{1 - \nu_i}$$

which implies an update of the form

$$\nu_i^{(k+1)} = \frac{\sum_{j=1}^{n} \tau_{ij}^{(k)} \kappa_{ij}^{(k)}}{\sum_{j=1}^{n} \tau_{ij}^{(k)} (\kappa_{ij}^{(k)} + \rho_{ij}^{(k)})} = \frac{\sum_{j=1}^{n} \tau_{ij}^{(k)} \kappa_{ij}^{(k)}}{\sum_{j=1}^{n} \tau_{ij}^{(k)}}.$$

Finally, we find that

$$\frac{\partial Q}{\partial \gamma_i} = \sum_{j=1}^{n} \left\{ \frac{p \tau_{ij}^{(k)} \kappa_{ij}^{(k)}}{2\gamma_i} - \frac{1}{2} \tau_{ij}^{(k)} \kappa_{ij}^{(k)} \mathrm{d}_{ij}^{(k)} \right\}.$$

which implies the update

$$\gamma_i^{(k+1)} = \frac{p \sum_{j=1}^{n} \tau_{ij}^{(k)} \kappa_{ij}^{(k)}}{\sum_{j=1}^{n} \tau_{ij}^{(k)} \kappa_{ij}^{(k)} \mathrm{d}_{ij}^{(k)}}.$$

For the MCNFA model, we need to estimate two parameters for each of the component densities $h_i(w_j; \boldsymbol{\psi}_i)$: ν_i and γ_i. So $\mathrm{n_s}(\boldsymbol{\psi}_i) = 2$ for each i, making

$$k^*_{\mathrm{MCNFA}}(p, g, q) = k^*_{\mathrm{MFA}}(p, g, q) + 2g.$$

It is worth noting that the MCNFA model may, at least in theory, suffer from an identifiability issue. To see this, consider a dataset with four sub-populations that all share the same mean and covariance matrix. Under the MCNFA model, we need to assign these four sub-populations into two mixture components, which are in turn composed of two groups, one being modelled with the scaled covariance matrix and the other with a non-scaled covariance matrix. There will be $4 \times 3 \times 2 = 24$ combinations for the composition of the two components and the scaled versus non-scaled group within each component. Of the 24 combinations, only 12 are identical up to switching component labels. In other words, there would be 12 different ways to parameterise this model even after accounting

for label switching. However, in practice we would not expect this to be an issue, as the example is rather contrived. If all four sub-populations have the same mean and covariance matrix, then a simple (single component) Gaussian model for the data would likely be more appropriate than a two component Contaminated Normal mixture model.

5.4 The Mean-Variance Mixture of Normal Distribution

The second method for generalising the MFA model makes use of the Mean-Variance Mixture of Normal (MVMN) distribution family, which is defined as follows.

Definition 5.4.1. The Mean-Variance Mixture of Normal Distribution [Barndorff-Nielsen et al., 1982]

A p-dimensional random variable $\mathbf{Y}$ is said to belong to the Mean-Variance Mixture of Normal (MVMN) distribution if it can be written in the form.

$$\mathbf{Y} = \boldsymbol{\mu} + W\boldsymbol{\beta} + W^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{X}, \tag{5.32}$$

where $\mathbf{X} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I}_p)$ and W is a positive random variable with density function $h(w; \boldsymbol{\psi})$.

Equivalently, we can express a MVMN distribution hierarchically as

$$\begin{aligned} W &\sim h(w; \boldsymbol{\psi}) \\ \mathbf{Y} \mid W = w &\sim \mathcal{N}_p(\boldsymbol{\mu} + w\boldsymbol{\beta}, w\boldsymbol{\Sigma}). \end{aligned}$$

The MVMN family is clearly similar to the SMN family. The main difference is the inclusion of the stochastic mean component $W\boldsymbol{\beta}$ in the definition of the MVMN distribution. Under the SMN family, only the covariance matrix $\boldsymbol{\Sigma}$ of the multivariate Gaussian distributed vector $\mathbf{Y} \mid W = w$ is scaled by w which allows for the tails of the resulting marginal distribution of $\mathbf{Y}$ to be altered. However, under the MVMN family, the addition of the $w\boldsymbol{\beta}$ term in Equation (5.32) allows for asymmetric distributional shapes, as well as the altered tails provided by the scaled covariance matrix.

We introduce the Mixtures of Mean-Variance Mixture of Normal Distribution Factor Analyzers (MMVMNFA) model as a combination of the MFA model and the MVMN model, represented hierarchically as

$$\begin{aligned} &\mathbf{Z}_j \sim \text{Multinomial}(1, \boldsymbol{\pi}), \\ &W_j \mid Z_{ij} = 1 \sim h_i(w_j; \boldsymbol{\psi}_i), \\ &\mathbf{Y}_j \mid Z_{ij} = 1, W_j = w_j \sim \mathcal{N}_p\left(\boldsymbol{\mu}_i + w_j\boldsymbol{\beta}_i, w_j\left(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right)\right). \end{aligned} \tag{5.33}$$

Like in the MSMNFA model from Section 5.1, h_i is a density function with support over $\mathbb{R}^+$ and parameter vector $\boldsymbol{\psi}_i$, for $i = 1, \ldots, g$.

The MMVMNFA family inherits the parsimonious mixture model formulation from the MFA model, but is much more flexible, allowing for modified tails and multivariate skewness in each component. We expect it will, therefore, be a more suitable model for many real world datasets, as they often display asymmetry. We demonstrate this in practice in Chapter 6.

Example 5. The three examples of MSMNFA models from the previous section are all examples of MMVMNFA models with $\boldsymbol{\beta} = \mathbf{0}$. The corresponding scaling distributions or densities are $\mathbf{W}_j \mid Z_{ij} = 1 \sim \text{Inv-Gamma}\left(\frac{\nu_i}{2}, \frac{\nu_i}{2}\right)$ for the MtFA model,

$$h_i(w_j; \nu_i) = \frac{\nu_i}{w_j^{\nu_i+1}} \text{ for } w_j \in (1, \infty)$$

for the MSLFA model and finally

$$h_i(w_j; \nu_i, \gamma_i) = \nu_i \mathbb{I}_{\{w_j = 1/\gamma_i\}} + (1 - \nu_i)\mathbb{I}_{\{w_j = 1\}}$$

for the MCNFA model.

More generally, any MSMNFA model can by written as a MMVMNFA model setting $\boldsymbol{\beta} = \mathbf{0}$ and taking the scaling variable $W_j^* \mid (Z_{ij} = 1) = (W_j \mid Z_{ij} = 1)^{-1}$, where $W_j \mid Z_{ij} = 1$ is the scaling variable of the MSMNFA model. ■

5.5 Parameter Estimation for the MMVMNFA Model

We now show that parameter estimation for the MMVMNFA model can also be performed by using an appropriately generalised version of MFA-ECM-2. In this case, we define $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ with

$$\boldsymbol{\theta}_1 = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_g, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_g, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_g)$$

and

$$\boldsymbol{\theta}_2 = (\mathbf{B}_1, \ldots, \mathbf{B}_g, \mathbf{D}_1, \ldots, \mathbf{D}_g).$$

Let the complete data be given by $\mathbf{x}_j = (\mathbf{y}_j, \mathbf{z}_j, w_j)$. Then the complete-data log-likelihood can be written as

$$\ell(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}, \mathbf{w}) \propto \sum_{j=1}^{n}\sum_{i=1}^{g}\left[z_{ij}\log\pi_i + z_{ij}\log h_i(w_j; \boldsymbol{\psi}_i) - \frac{z_{ij}}{2}\log\left|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right|\right.$$
$$\left. - \frac{1}{2}\frac{z_{ij}}{w_j}(\mathbf{y}_j - \boldsymbol{\mu}_i - w_j\boldsymbol{\beta}_i)^\top(\mathbf{B}_i\mathbf{B}_i + \mathbf{D}_i)^{-1}(\mathbf{y}_j - \boldsymbol{\mu}_i - w_j\boldsymbol{\beta}_i)\right].$$

which we can expand to

$$\ell\left(\boldsymbol{\Theta} \mid \mathbf{y}, \mathbf{z}, \mathbf{w}\right) \propto \sum_{j=1}^{n}\sum_{i=1}^{g} z_{ij}\Bigg[\log \pi_i + \log h_i(w_j; \boldsymbol{\psi}_i) - \frac{1}{2}\log\left|\mathbf{B}_i\mathbf{B}_i^{\top} + \mathbf{D}_i\right| - \frac{1}{2}\frac{1}{w_j}\mathrm{d}_{ij} + \frac{1}{2}\,\mathrm{d}(\boldsymbol{\beta}_i, \mathbf{y}_j - \boldsymbol{\mu}_i, \mathbf{B}_i, \mathbf{D}_i) + \frac{1}{2}\,\mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i, \boldsymbol{\beta}_i, \mathbf{B}_i, \mathbf{D}_i) - \frac{1}{2}w_j\mathrm{d}^{*}_{ij}\Bigg], \tag{5.34}$$

where we define two additional special cases of $\mathrm{d}(.,.,.,.)$:

$$\mathrm{d}^{*}_{ij} := \mathrm{d}(\boldsymbol{\beta}_i, \boldsymbol{\beta}_i, \mathbf{B}_i, \mathbf{D}_i)$$

and

$$\mathrm{d}^{*\,(k)}_{ij} := \mathrm{d}\left(\boldsymbol{\beta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}, \mathbf{B}_i^{(k)}, \mathbf{D}_i^{(k)}\right).$$

From Equation (5.34), we can see that the E-step will be to calculate

$$\tau_{ij}^{(k)} = \mathbb{E}\left[Z_{ij} \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right]$$

as in Equation (5.5), where the marginal density is also calculated in the same way as for the MSMNFA model, as given in Equation (5.6). We also need to calculate

$$\mathbb{E}\left[Z_{ij}W_j \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)}\mathbb{E}\left[W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)}\xi_{ij}^{(k)}$$

and

$$\mathbb{E}\left[Z_{ij}W_j^{-1} \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)}\mathbb{E}\left[W_j^{-1} \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] = \tau_{ij}^{(k)}\varrho_{ij}^{(k)}$$

with $\xi_{ij}^{(k)}$ defined as before and

$$\varrho_{ij}^{(k)} := \mathbb{E}\left[W_j^{-1} \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right].$$

Finally, we will also need to calculate any other conditional expectations of the form

$$\mathbb{E}\left[f(W_j, Z_{ij}) \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right].$$

The M-step estimate for π_i is again given by Equation (2.30) and the update for $\boldsymbol{\psi}_i$ will be given by

$$\boldsymbol{\psi}_i^{(k+1)} = \underset{\boldsymbol{\psi}_i}{\operatorname{argmax}}\left\{\sum_{j=1}^{n}\mathbb{E}\left[Z_{ij}\log h_i(W_j; \boldsymbol{\psi}_i) \mid \mathbf{y}_j; \boldsymbol{\Theta}^{(k)}\right]\right\}.$$

The Q-function for $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ will be given by

$$\begin{aligned} Q(\boldsymbol{\mu}, \boldsymbol{\beta}; \boldsymbol{\Theta}^{(k)}) \propto -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{g} \tau_{ij}^{(k)} \Big[& \varrho_{ij}^{(k)} \, \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i, \mathbf{y}_j - \boldsymbol{\mu}_i, \mathbf{B}_i^{(k)}, \mathbf{D}_i^{(k)}) \\ & - \mathrm{d}(\boldsymbol{\beta}_i, \mathbf{y}_j - \boldsymbol{\mu}_i, \mathbf{B}_i^{(k)}, \mathbf{D}_i^{(k)}) - \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i, \boldsymbol{\beta}_i, \mathbf{B}_i^{(k)}, \mathbf{D}_i^{(k)}) \\ & + \xi_{ij}^{(k)} \, \mathrm{d}(\boldsymbol{\beta}_i, \boldsymbol{\beta}_i, \mathbf{B}_i^{(k)}, \mathbf{D}_i^{(k)}) \Big], \end{aligned} \tag{5.35}$$

which implies that

$$\frac{\partial Q}{\partial \boldsymbol{\mu}_i} = \sum_{j=1}^{n} \tau_{ij}^{(k)} \left[\varrho_{ij}^{(k)} (\mathbf{B}_i^{(k)} \mathbf{B}_i^{(k)^\top} + \mathbf{D}_i^{(k)})^{-1} (\mathbf{y}_j - \boldsymbol{\mu}_i) - (\mathbf{B}_i^{(k)} \mathbf{B}_i^{(k)^\top} + \mathbf{D}_i^{(k)})^{-1} \boldsymbol{\beta}_i \right]$$

and

$$\frac{\partial Q}{\partial \boldsymbol{\beta}_i} = \sum_{j=1}^{n} \tau_{ij}^{(k)} \left[\xi_{ij}^{(k)} (\mathbf{B}_i^{(k)} \mathbf{B}_i^{(k)^\top} + \mathbf{D}_i^{(k)})^{-1} \boldsymbol{\beta}_i - (\mathbf{B}_i^{(k)} \mathbf{B}_i^{(k)^\top} + \mathbf{D}_i^{(k)})^{-1} (\mathbf{y}_j - \boldsymbol{\mu}_i) \right].$$

Setting these jointly equal to zero yield the updates

$$\boldsymbol{\mu}_i^{(k+1)} = \frac{\sum_{j=1}^{n} \tau_{ij}^{(k)} \mathbf{y}_j (\bar{\xi}_i^{(k)} \varrho_{ij}^{(k)} - 1)}{\sum_{j=1}^{n} \tau_{ij}^{(k)} (\bar{\xi}_i^{(k)} \varrho_{ij}^{(k)} - 1)} \tag{5.36}$$

and

$$\boldsymbol{\beta}_i^{(k+1)} = \frac{\sum_{j=1}^{n} \tau_{ij}^{(k)} \mathbf{y}_j (\bar{\varrho}_i^{(k)} - \varrho_{ij}^{(k)})}{\sum_{j=1}^{n} \tau_{ij}^{(k)} (\bar{\xi}_i^{(k)} \varrho_{ij}^{(k)} - 1)}, \tag{5.37}$$

where

$$\bar{\xi}_i^{(k)} = \frac{\sum_{j=1}^{n} \tau_{ij}^{(k)} \xi_{ij}^{(k)}}{\sum_{j=1}^{n} \tau_{ij}^{(k)}} \qquad \text{and} \qquad \bar{\varrho}_i^{(k)} = \frac{\sum_{j=1}^{n} \tau_{ij}^{(k)} \varrho_{ij}^{(k)}}{\sum_{j=1}^{n} \tau_{ij}^{(k)}}.$$

Let $\boldsymbol{\Theta}^{(k+1/2)} := (\boldsymbol{\theta}_1^{(k+1)}, \boldsymbol{\theta}_2^{(k)})$. Given the form of Equation (5.34), it is clear that

$$\begin{aligned} Q\left(\boldsymbol{\theta}_2; \boldsymbol{\Theta}^{(k+1/2)}\right) \propto -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{g} \tau_{ij}^{(k)} \Big[& \log \left| \mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i \right| \\ & + \varrho_{ij}^{(k)} \, \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i) \\ & - \mathrm{d}(\boldsymbol{\beta}_i^{(k+1)}, \mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i) - \mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)}, \boldsymbol{\beta}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i) \\ & + \xi_{ij}^{(k)} \, \mathrm{d}(\boldsymbol{\beta}_i^{(k+1)}, \boldsymbol{\beta}_i^{(k+1)}, \mathbf{B}_i, \mathbf{D}_i) \Big]. \end{aligned} \tag{5.38}$$

As a result, we can use a slight variation[1] of the argument given in Appendix A.2 to show that

$$Q\left(\boldsymbol{\theta}_2; \boldsymbol{\Theta}^{(k+1/2)}\right) \propto -\frac{n}{2}\sum_{i=1}^{g}\pi_i^{(k+1)}\left[\log|\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i| + \operatorname{tr}\left\{\left(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i\right)^{-1}\mathbf{S}_i^{(k)}\right\}\right] \tag{5.39}$$

where we redefine $\mathbf{S}_i^{(k)}$ as

$$\begin{aligned}\mathbf{S}_i^{(k)} := \frac{1}{n\pi_i^{(k+1)}}\sum_{j=1}^{n}\tau_{ij}^{(k)}\Big[&\varrho_{ij}^{(k)}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})^\top - (\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})\boldsymbol{\beta}_i^{(k+1)^\top} \\ &- \boldsymbol{\beta}_i^{(k+1)}(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})^\top + \xi_{ij}^{(k)}\boldsymbol{\beta}_i{}^{(k+1)}\boldsymbol{\beta}_i^{(k+1)^\top}\Big].\end{aligned} \tag{5.40}$$

This means that, as under the MFA and MSMNFA models, under the MMVMNFA model the updates for $\mathbf{B}_i$ and $\mathbf{D}_i$ are again given by Equation (2.46) and Equation (2.51), albeit using the updated definition of $\mathbf{S}_i^{(k)}$ given in Equation (5.40).

Since the updates for $\mathbf{B}_i$ and $\mathbf{D}_i$ are still based on the method used by Zhao and Yu [2008], the M-step update for q proposed by Wang and Lin [2020] can also be generalised to

$$q^{(k+1)} = \underset{q\leq q_{\max}}{\operatorname{argmin}}\left\{\sum_{i=1}^{g}n\pi_i^{(k+1)}\sum_{l=1}^{q}(\log\lambda_{il} - \lambda_{il} + 1) + k^*_{\text{MMVMNFA}}(g,q)\log n\right\}. \tag{5.41}$$

In the context of the MMVMNFA model, λ_{il} is the l^{th} largest eigenvalue of the matrix

$$\tilde{\mathbf{S}}_i := \left[\mathbf{D}_i^{(k)}\right]^{-1/2}\mathbf{S}_i^{(k)}\left[\mathbf{D}_i^{(k)}\right]^{-1/2},$$

with $\mathbf{S}_i^{(k)}$ as defined in Equation (5.40). The number of parameters being estimated in the MMVMNFA model will be given by

$$k^*_{\text{MMVMNFA}}(p,g,q) = k^*_{\text{MFA}}(p,g,q) + pg + \sum_{i=1}^{g}\mathrm{n_p}(\boldsymbol{\psi}_i),$$

where $\mathrm{n_p}$ counts the number of parameters in $\boldsymbol{\psi}_i$. This is similar to $k^*_{\text{MSMNFA}}(p,g,q)$, except with an additional pg parameters to estimate the g $p\times 1$ stochastic mean vectors $\boldsymbol{\beta}_i$.

[1] Clearly, in this case, we have four scalar terms after the logarithm term as opposed to just the Mahalanobis distance term present in Appendix A.2. However, since the final four terms all share the $\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i$ term, we can apply the same general argument as the one given in Appendix A.2, making use of the linearity of the trace and then taking out a common factor of $\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i$ to achieve the desired result.

Algorithm 5.2: ECM-MMVMNFA-1: General ECM algorithm for the MMVMNFA model family

Input: An initial estimate of the parameters, $\boldsymbol{\Theta}^{(0)}$, a convergence criterion and a maximum number of iterations k_{max}

Result: Maximum likelihood estimates for the vector of parameters $\boldsymbol{\Theta}$

Set $k = 0$;
while *Chosen convergence criterion not satisfied and* $k < k_{max}$ **do**
  The E-Step: Compute the responsibilities $\tau_{ij}^{(k)}$ using Equation (5.5), as well as $\xi_{ij}^{(k)}$, $\varrho_{ij}^{(k)}$ and any other necessary E-step estimates as given in the subsection for each particular case of the MMVMNFA model family, for $i \in \{1, \ldots, g\}$ and $j \in \{1, \ldots, n\}$;
  The M-Step: Compute $\boldsymbol{\Theta}^{(k+1)}$ using Equation (2.30), Equation (5.36), Equation (5.37), Equation (2.46) and Equation (2.51) (using the definition of $\mathbf{S}_i^{(k)}$ given in Equation (5.40)), respectively, as well as the specified M-step updates for the parameters in $\boldsymbol{\psi}_i$ as given in the subsection for each particular case of the MMVMNFA model family;
  if *Convergence criterion satisfied* **then**
    Return $\boldsymbol{\Theta}^{(k+1)}$;
  else
    Set $k = k + 1$;
  end
end
Return $\boldsymbol{\Theta}^{(k_{\max})}$;

ECM-MMVMNFA-1 (Algorithm 5.2) summarises the ECM algorithm obtained by generalising ECM-MFA-2 to the MMVMNFA model family. Like for ECM-MSMNFA-1, formulas for $\xi_{ij}^{(k)}$, $\varrho_{ij}^{(k)}$ and $\boldsymbol{\psi}_i^{(k+1)}$ cannot be given in general, as they depend on the distribution of the scaling variables W_j, however, three particular cases of the MMVMNFA model family are given in the next section. All necessary formulas for the E-steps and M-steps are again given in each case, so that ECM-MMVMNFA-1 can be implemented.

5.6 Examples of the MMVMNFA Model

We now give three specific instances of the MMVMNFA family, along with complete EM-type algorithms for each.

5.6.1 The MGHFA Model

An example of the MMVMNFA is the Mixture of Generalised Hyperbolic Factor Analyzers (MGHFA). This special case of the MMVMNFA family was previously introduced in Tortora et al. [2015], albeit in that work the authors used an AECM algorithm for parameter estimation. As such, their algorithm required treating the factors $\mathbf{U}_j$ as latent variables to obtain updates for $\mathbf{B}_i$ and $\mathbf{D}_i$, in a similar manner to MFA-ECM-1. Our proposed algorithm, ECM-MMVMNFA-1, does not condition on the factors to obtain updates for $\mathbf{B}_i$ and $\mathbf{D}_i$.

In the MGHFA model, we take $W_j \mid Z_{ij} = 1 \sim \text{GIG}\left(\omega_i, \omega_i, \lambda_i\right)$, where GIG represents the Generalised Inverse Gaussian distribution with density

$$h_i(w_j; \omega_i, \lambda_i) = \frac{w_j^{\lambda_i - 1}}{2\,\mathrm{K}_{\lambda_i}(\omega_i)} \exp\left\{-\frac{\omega_i}{2}\left(w_j + \frac{1}{w_j}\right)\right\},$$

and where K_{λ_i} is the modified Bessel function of the third kind with index λ_i and $\omega_i > 0$.

Marginalising over w_j shows that

$$f(\mathbf{y}_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) = \left[\frac{\omega_i + \mathrm{d}_{ij}}{\omega_i + \mathrm{d}_{ij}^*}\right]^{\frac{(\lambda_i - p/2)}{2}} \times \frac{\mathrm{K}_{\lambda_i - p/2}\left(\sqrt{(\omega_i + \mathrm{d}_{ij})(\omega_i + \mathrm{d}_{ij}^*)}\right)}{(2\pi)^{p/2}|(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)|^{1/2}\,\mathrm{K}_{\lambda_i}(\omega_i)\exp\left\{-\mathrm{d}(\mathbf{y}_j - \boldsymbol{\mu}_i, \boldsymbol{\beta}_i, \mathbf{B}_i, \mathbf{D}_i)\right\}}. \tag{5.42}$$

This implies that $\mathbf{Y}_j \mid Z_{ij} = 1 \sim \text{GH}\left(\lambda_i, \omega_i, \omega_i, \boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \boldsymbol{\beta}_i\right)$, the generalised hyperbolic distribution. The repeated parameter ω_i ensures that the resulting generalised hyperbolic distribution will be identifiable.

The usual Bayesian argument shows that

$$W_j \mid Z_{ij} = 1 \sim \text{GIG}\left(\omega_i + \mathrm{d}_{ij}, \omega_i + \mathrm{d}^*_{ij}, \lambda_i - \frac{p}{2}\right).$$

Now, define

$$\mathrm{R}_{ijk}(a, b, c, d) := \frac{\mathrm{K}_{a+b}\left(\sqrt{\left(\mathrm{d}_{ij}^{(k)} + c\right)\left(\mathrm{d}_{ij}^{*\,(k)} + d\right)}\right)}{\mathrm{K}_{a}\left(\sqrt{\left(\mathrm{d}_{ij}^{(k)} + c\right)\left(\mathrm{d}_{ij}^{*\,(k)} + d\right)}\right)},$$

for $a, b \in \mathbb{R}$ and $c, d \in \mathbb{R}^+$.

Using results from Jorgensen [1982], we have

$$\xi_{ij}^{(k)} = \sqrt{\frac{\omega_i^{(k)} + \mathrm{d}_{ij}^{(k)}}{\omega_i^{(k)} + \mathrm{d}_{ij}^{*\,(k)}}}\,\mathrm{R}_{ijk}\left(\lambda_i^{(k)} - \frac{p}{2}, 1, \omega_i^{(k)}, \omega_i^{(k)}\right)$$

and

$$\varrho_{ij}^{(k)} = -\frac{2\lambda_i^{(k)} - p}{\omega_i^{(k)} + \mathrm{d}_{ij}^{(k)}} + \sqrt{\frac{\omega_i^{(k)} + \mathrm{d}_{ij}^{*\,(k)}}{\omega_i^{(k)} + \mathrm{d}_{ij}^{(k)}}}\,\mathrm{R}_{ijk}\left(\lambda_i^{(k)} - \frac{p}{2}, 1, \omega_i^{(k)}, \omega_i^{(k)}\right).$$

Similarly, we evaluate $\zeta_{ij}^{(k)}$ as

$$\begin{aligned}\zeta_{ij}^{(k)} &:= \mathbb{E}\left[\log W_j \mid \mathbf{y}_j, Z_{ij} = 1; \boldsymbol{\Theta}^{(k)}\right] \\ &= \log\sqrt{\frac{\omega_i^{(k)} + \mathrm{d}_{ij}^{(k)}}{\omega_i^{(k)} + \mathrm{d}_{ij}^{*\,(k)}}} + \frac{\partial}{\partial t}\log\left\{\mathrm{K}_t\left(\left(\mathrm{d}_{ij}^{(k)} + \omega_i^{(k)}\right)\left(\mathrm{d}_{ij}^{*\,(k)} + \omega_i^{(k)}\right)\right)\right\}\Big|_{t=\lambda_i^{(k)}-p/2}\end{aligned}$$

using results from Browne and McNicholas [2015].

The Q-function for $\boldsymbol{\omega}$ and $\boldsymbol{\lambda}$ is given by

$$Q(\boldsymbol{\omega}, \boldsymbol{\lambda}; \boldsymbol{\Theta}^{(k)}) \propto \sum_{j=1}^{n}\sum_{i=1}^{g}\tau_{ij}^{(k)}\left(\lambda_i\zeta_{ij}^{(k)} - \log\mathrm{K}_{\lambda_i}(\omega_i) - \frac{\omega_i}{2}\left(\xi_{ij}^{(k)} + \varrho_{ij}^{(k)}\right)\right),$$

or equivalently

$$Q_i(\omega_i, \lambda_i; \boldsymbol{\Theta}^{(k)}) \propto \bar{\zeta}_i^{(k)}\lambda_i - \log\mathrm{K}_{\lambda_i}(\omega_i) - \frac{\omega_i}{2}\left(\bar{\xi}_i^{(k)} + \bar{\varrho}_i^{(k)}\right),$$

where

$$\bar{\zeta}_i^{(k)} = \frac{\sum_{j=1}^{n}\tau_{ij}^{(k)}\zeta_{ij}^{(k)}}{\sum_{j=1}^{n}\tau_{ij}^{(k)}},$$

and where $\bar{\xi}_i^{(k)}$ and $\bar{\varrho}_i^{(k)}$ are defined as in Section 5.5. Previously, Browne and McNicholas [2015] and Tortora et al. [2015] proposed approximate updates for ω_i and λ_i as

$$\lambda_i^{(k+1)} = \bar{\zeta}_i^{(k)} \lambda_i^{(k)} \left[\frac{\partial}{\partial t} \log \mathrm{K}_t(\omega_i^{(k)}) \Big|_{t=\lambda_i^{(k)}} \right]$$

and

$$\omega_i^{(k+1)} = \omega_i^{(k)} - \left[\frac{\partial}{\partial t} Q_i(t, \lambda_i^{(k)}; \boldsymbol{\Theta}^{(k)}) \Big|_{t=\omega_i^{(k)}} \right] \left[\frac{\partial^2}{\partial t^2} Q_i(t, \lambda_i^{(k)}; \boldsymbol{\Theta}^{(k)}) \Big|_{t=\omega_i^{(k)}} \right]^{-1}.$$

The update for λ_i is obtained using a majorizing surrogate function [Browne and McNicholas, 2015], while the update for ω_i is just a single step of Newton's Algorithm applied to $\frac{\partial}{\partial \omega_i} Q_i$.

In our implementations of this algorithm, we have chosen to use univariate numerical optimisation instead. This avoids the possibility of taking likelihood-decreasing steps.

For the MGHFA model, we need to estimate two parameters for each of the component densities $h_i(w_j; \boldsymbol{\psi}_i)$: ν_i and γ_i. So $\mathrm{n_s}(\boldsymbol{\psi}_i) = 2$ for each i, making

$$k^*_{\mathrm{MGHFA}}(p, g, q) = k^*_{\mathrm{MFA}}(p, g, q) + pg + 2g.$$

5.6.2 The MFA-BS Model

Another special case of the MMVMNFA model is what we call the Mixture of Factor Analyzers using Birnbaum-Saunders scaling variables (MFA-BS) model. The reason for changing the naming convention for this model is to reflect that the name Birnbaum-Saunders is from the scaling variables, and not from the marginal distribution of the data which was the case in all of the other special cases thus far. This marginal density of the MFA-BS model is a mixture of the multivariate mean-variance mixtures based on the Birnbaum-Saunders distribution which was proposed in Pourmousa et al. [2015].

The univariate Birnbaum-Saunders distribution is a two parameter distribution with density function

$$f_{\mathrm{BS}}(x; \alpha, \beta) = \frac{1}{2\sqrt{2\pi}\alpha\beta} \left[\left(\frac{\beta}{x} \right)^{1/2} + \left(\frac{\beta}{x} \right)^{3/2} \right] \exp\left[-\frac{1}{2\alpha^2} \left(\frac{x}{\beta} + \frac{\beta}{x} - 2 \right) \right],$$

where $x, \alpha, \beta > 0$. Setting $\beta = 1$, we find that it can be written as the following mixture of GIG densities:

$$f_{\mathrm{BS}}(x; \alpha, 1) = \frac{1}{2} f_{\mathrm{GIG}}\left(x; \frac{1}{\alpha^2}, \frac{1}{\alpha^2}, \frac{1}{2} \right) + \frac{1}{2} f_{\mathrm{GIG}}\left(x; \frac{1}{\alpha^2}, \frac{1}{\alpha^2}, -\frac{1}{2} \right).$$

It is not hard to show that by taking

$$h_i(w_j;\alpha_i) = f_{\text{BS}}(w_j;\alpha_i,1),$$

marginalisation over w_j produces

$$\begin{aligned} f(\mathbf{y}_j \mid Z_{ij}=1;\boldsymbol{\Theta}) = \frac{1}{2} f_{\text{GH}}\left(\mathbf{y}_j;\frac{1}{2},\frac{1}{\alpha_i^2},\frac{1}{\alpha_i^2},\boldsymbol{\mu}_i,\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i,\boldsymbol{\beta}_i\right) \\ + \frac{1}{2} f_{\text{GH}}\left(\mathbf{y}_j;-\frac{1}{2},\frac{1}{\alpha_i^2},\frac{1}{\alpha_i^2},\boldsymbol{\mu}_i,\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i,\boldsymbol{\beta}_i\right). \end{aligned}$$

So the marginal distribution of the data conditioned on belonging to component i is a mixture of Generalised Hyperbolic distributions.

It is also straightforward to show that

$$\begin{aligned} f_{W_j|\mathbf{Y}_j,Z_{ij}=1}(w_j;\boldsymbol{\Theta}) = \frac{1}{2} f_{\text{GIG}}\left(w_j;\mathrm{d}_{ij}+\frac{1}{\alpha_i^2},\mathrm{d}^*_{ij}+\frac{1}{\alpha_i^2},\frac{1-p}{2}\right) \\ + \frac{1}{2} f_{\text{GIG}}\left(w_j;\mathrm{d}_{ij}+\frac{1}{\alpha_i^2},\mathrm{d}^*_{ij}+\frac{1}{\alpha_i^2},-\frac{1+p}{2}\right). \end{aligned}$$

Using results given in Naderi et al. [2018], we can therefore evaluate $\xi_{ij}^{(k)}$ and $\varrho_{ij}^{(k)}$ as

$$\begin{aligned} \xi_{ij}^{(k)} = \frac{1}{2}\sqrt{\frac{\frac{1}{\alpha_i^{(k)^2}}+\mathrm{d}_{ij}^{(k)}}{\frac{1}{\alpha_i^{(k)^2}}+\mathrm{d}^{*\,(k)}_{ij}}}\left(\mathrm{R}_{ijk}\left(\frac{1-p}{2},1,\frac{1}{\alpha_i^{(k)^2}},\frac{1}{\alpha_i^{(k)^2}}\right)\right. \\ \left. + \mathrm{R}_{ijk}\left(\frac{1+p}{2},-p,\frac{1}{\alpha_i^{(k)^2}},\frac{1}{\alpha_i^{(k)^2}}\right)\right) \end{aligned}$$

and

$$\begin{aligned} \varrho_{ij}^{(k)} = \frac{1}{2}\sqrt{\frac{\frac{1}{\alpha_i^{(k)^2}}+\mathrm{d}^{*\,(k)}_{ij}}{\frac{1}{\alpha_i^{(k)^2}}+\mathrm{d}_{ij}^{(k)}}}\left(\mathrm{R}_{ijk}\left(\frac{1-p}{2},-1,\frac{1}{\alpha_i^{(k)^2}},\frac{1}{\alpha_i^{(k)^2}}\right)\right. \\ \left. + \mathrm{R}_{ijk}\left(\frac{1+p}{2},-2-p,\frac{1}{\alpha_i^{(k)^2}},\frac{1}{\alpha_i^{(k)^2}}\right)\right). \end{aligned}$$

The Q-function for $\boldsymbol{\alpha}$ is given by

$$Q(\boldsymbol{\alpha};\boldsymbol{\Theta}^{(k)}) \propto -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\tau_{ij}^{(k)}\left(\frac{1}{\alpha_i^2}(\xi_{ij}^{(k)}+\varrho_{ij}^{(k)}-2)+2\log\alpha_i\right),$$

which implies an update of the form

$$\alpha_i^{(k+1)} = \sqrt{\bar{\xi}_i^{(k)} + \bar{\varrho}_i^{(k)} - 2}.$$

Since the scaling density has only one parameter for each component,

$$k^*_{\text{MFA-BS}}(p, g, q) = k^*_{\text{MFA}}(p, g, q) + pg + g.$$

5.6.3 The MFA-L Model

The final special case of the MMVMNFA family that we will consider is what we call the Mixture of Factor Analyzers using Lindley scaling variables (MFA-L) model. Again, the name Lindley comes from the distribution of the scaling variables and not the marginal distribution of the data, so we use the same naming convention as the MFA-BS model. The MFA-L model is based on the Mean-Variance Mixtures based on Lindley scaling variables model from Naderi et al. [2018].

The univariate Lindley distribution is a one parameter distribution with density function

$$f_{\text{L}}(x; \alpha) = \frac{\alpha^2}{1+\alpha}(1+x)e^{-\alpha x},$$

where $x, \alpha > 0$. We can also write the Lindley density as a mixture of two GIG densities, since

$$f_{\text{L}}(x; \alpha) = \frac{\alpha}{1+\alpha} f_{\text{GIG}}(x; 0, 2\alpha, 1) + \frac{1}{1+\alpha} f_{\text{GIG}}(x; 0, 2\alpha, 2).$$

Taking

$$h_i(w_j; \alpha_i) = f_{\text{L}}(w_j; \alpha_i),$$

marginalisation over w_j produces

$$\begin{aligned} f(\mathbf{y}_j \mid Z_{ij} = 1) = &\frac{\alpha_i}{1+\alpha_i} f_{\text{GH}}\left(\mathbf{y}_j; 1, 0, 2\alpha_i, \boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \boldsymbol{\beta}_i\right) \\ &+ \frac{1}{1+\alpha_i} f_{\text{GH}}\left(\mathbf{y}_j; 2, 0, 2\alpha_i, \boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \boldsymbol{\beta}_i\right). \end{aligned}$$

So, like for the MFA-BS model, under the MFA-L model the marginal distribution of $\mathbf{Y}_j \mid Z_{ij} = 1$ is also a mixture of Generalised Hyperbolic distributions, albeit with different parameters.

It is also straightforward to show that

$$\begin{aligned} f_{W_j|\mathbf{Y}_j, Z_{ij}=1}(w_j; \boldsymbol{\Theta}) = &\, \mathrm{p}_{ij}\, f_{\text{GIG}}\left(w_j; \mathrm{d}_{ij}, \mathrm{d}^*_{ij} + 2\alpha_i, 1 - \frac{p}{2}\right) \\ &+ (1 - \mathrm{p}_{ij}) f_{\text{GIG}}\left(w_j; \mathrm{d}_{ij}, \mathrm{d}^*_{ij} + 2\alpha_i, 2 - \frac{p}{2}\right), \end{aligned}$$

where

$$\mathrm{p}_{ij} = \frac{\alpha_i f_{\mathrm{GH}}(\mathbf{y}_j; 1, 0, 2\alpha_i, \boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \boldsymbol{\beta}_i)}{\alpha_i f_{\mathrm{GH}}(\mathbf{y}_j; 1, 0, 2\alpha_i, \boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \boldsymbol{\beta}_i) + f_{\mathrm{GH}}(\mathbf{y}_j; 2, 0, 2\alpha_i, \boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \boldsymbol{\beta}_i)}.$$

$$\xi_{ij}^{(k)} = \sqrt{\frac{\mathrm{d}_{ij}^{(k)}}{2\alpha_i^{(k)} + \mathrm{d}_{ij}^{*\,(k)}}}\left(\mathrm{p}_{ijk}\,\mathrm{R}_{ijk}\left(1 - \frac{p}{2}, 1, 0, 2\alpha_i^{(k)}\right) + (1 - \mathrm{p}_{ijk})\,\mathrm{R}_{ijk}\left(2 - \frac{p}{2}, 1, 0, 2\alpha_i^{(k)}\right)\right)$$

and

$$\xi_{ij}^{(k)} = \sqrt{\frac{2\alpha_i^{(k)} + \mathrm{d}_{ij}^{*\,(k)}}{\mathrm{d}_{ij}^{(k)}}}\left(\mathrm{p}_{ijk}\,\mathrm{R}_{ijk}\left(1 - \frac{p}{2}, -1, 0, 2\alpha_i^{(k)}\right) + (1 - \mathrm{p}_{ijk})\,\mathrm{R}_{ijk}\left(2 - \frac{p}{2}, -1, 0, 2\alpha_i^{(k)}\right)\right),$$

where p_{ijk} is defined the same as p_{ij}, except using the k-step estimates for each of the model parameters.

The Q-function for $\boldsymbol{\alpha}$ is given by

$$Q(\boldsymbol{\alpha}; \boldsymbol{\Theta}^{(k)}) \propto \sum_{j=1}^{n}\sum_{i=1}^{g} \tau_{ij}^{(k)}\left(2\log\alpha_i - \log(1+\alpha_i) - \alpha_i\xi_{ij}^{(k)}\right),$$

so

$$\frac{\partial Q}{\partial \alpha_i} = \sum_{j=1}^{n} \tau_{ij}^{(k)}\left(\frac{2}{\alpha_i} - \frac{1}{1+\alpha_i} - \xi_{ij}^{(k)}\right).$$

Setting $\frac{\partial Q}{\partial \alpha_i} = 0$ is therefore equivalent to

$$\bar{\xi}_{ij}^{(k)}\alpha_i^2 + (\bar{\xi}_{ij}^{(k)} - 1)\alpha_i - 2 = 0$$

which implies an update of the form

$$\alpha_i^{(k+1)} = \frac{-(\bar{\xi}_{ij}^{(k)} - 1) + \sqrt{(\bar{\xi}_{ij}^{(k)} - 1)^2 - 8\bar{\xi}_{ij}^{(k)}}}{2\bar{\xi}_{ij}^{(k)}}.$$

Since the scaling density again has only one parameter for each component,

$$k^*_{\text{MFA-L}}(p, g, q) = k^*_{\text{MFA}}(p, g, q) + pg + g.$$

5.7 Summary

In this chapter we proposed the MMVMNFA model family as a generalisation of the MFA model which retains the parsimonious mixture model formulation of the MFA model but can also model data with heavy-tails and/or multivariate skewness. We also generalised MFA-ECM-2 (Algorithm 2.4), a parameter estimation algorithm for the MFA model, to the MMVMNFA model family. In addition, we generalised the AMFA algorithm (Algorithm 3.1), to the MMVMNFA model family. We provided six specific instances of the MMVMNFA model family: the MtFA, MSLFA, MCNFA, MGHFA, MFA-BS and MFA-L models. For each of these instances, full EM-type parameter estimation algorithms were derived.

Three of these models have been proposed previously: the MtFA model by McLachlan et al. [2007] with the ECM algorithm given in Section 5.3.1 previously proposed by Wang and Lin [2012], the MCNFA model by Punzo and McNicholas [2016] and the MGHFA model by Tortora et al. [2015]. However, this is the first work, to our knowledge, which generalises the ECM algorithm given by Zhao and Yu [2008] for the MFA model and by Wang and Lin [2012] for the MtFA model to the whole MMVMNFA model family, and is also the first work to apply the general ECM algorithm to the five other special cases given. This is of practical importance, because, as demonstrated by Zhao and Yu [2008] for the MFA model and by Wang and Lin [2012] for the MtFA model, the use of an ECM routine which does not treat the underlying factors $\mathbf{U}_j$ as missing data generally leads to much faster model fits, in terms of both CPU time and the number of iterations used.

The method of generalisation which we employed in this chapter was to introduce a positive univariate scaling distribution $h_i(w_j; \boldsymbol{\psi}_i)$ for each component such that the conditional distribution $\mathbf{Y}_j \mid Z_{ij} = 1, W_j = w_j$ belongs to the MVMN family. However, this is not the only possible way to introduce asymmetry to the MFA model. For example, Lee and McLachlan [2021] discuss a number of possible ways of introducing asymmetry into factor models.

In addition, different formulations could be achieved by tightening or loosening various assumptions. For example, the number of factors, q, could be allowed to vary between components. The constraint structure for the component covariance matrices $\boldsymbol{\Sigma}_i$ from the conditional distribution of $\mathbf{Y}_j \mid Z_{ij} = 1, W_j = w_j$ could also be investigated. Our method of generalisation retained the MFA component covariance structure of $\boldsymbol{\Sigma}_i = \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i$, but this is only one of a multitude of possible constraint structures. A number of possible alternative constrained covariance structures are discussed by Celeux and Govaert [1995].

On the other hand, different models could also be achieved by restricting the factor loading matrices to be the same across all of the components, or by restricting all of the scaling distribution parameters to be the same across all of the components.

However, a more comprehensive investigation of the different possible asymmetric parsimonious multivariate finite mixture models, as well as the multitude of combinations of constraints that can be applied to the parameters of these models is beyond the scope of this thesis.

Chapter 6

Applying MMVMNFA Models

In the previous chapter, we propose the MMVMNFA model family as an extension of the MFA model which can model data with heavy tails and/or multivariate skewness. We also detailed how parameter estimates can be obtained for the MMVMNFA family of models. In this chapter, we will show how models from the MMVMNFA family can achieve better model fits and clustering results compared to the standard MFA model.

We will consider four datasets in this chapter. The first two will be synthetically generated, which will allow us to introduce heavy tails and multivariate skewness, respectively. The third is a real world dataset concerning wholesale goods orders in Portugal [Baudry et al., 2012], where the orders are either classified as retail or commerical. The fourth is the Australian Institute of Sport (AIS) dataset of Telford and Cunningham [1991], which contains measurements of various physical properties of Australian athletes, as well as their sex and the sport that they play.

The parameter estimation in this chapter will be performed using the Julia package FactorMixtures that we developed. This package includes methods for fitting each of the six instances of the MMVMNFA family discussed in Chapter 5, as well as the standard MFA model. In each case, the number of components, g and the number of factors, q, will be determined using a naïve search over the grid $1 \leq g \leq 5$ and $1 \leq q \leq q_{\max}$, where $q_{\max}$ is the Ledermann bound. Each model will use the same initialisation scheme as the `MFA_ECM` method from `autoMFA`, which was described in Section 3.1. Fifteen initializations based on the output of k-means clustering and fifteen random initializations for each combination of g and q will be used. We will select the best model of each type using the BIC. We will then compare the BICs, ARIs, fitting times and inferred values of g and q for the best models of each type.

6.1 Simulation Study 1

The first simulation study will compare the results of the MFA model with those of the MMVMNFA models on a heavy-tailed multivariate dataset. We created a synthetic two-component mixture model dataset where each component follows the multivariate t-distribution by randomly generating

$$\mathbf{Y}_1, \ldots, \mathbf{Y}_{500} \overset{i.i.d}{\sim} t_3(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \nu_1) \text{ and } \mathbf{Y}_{501}, \ldots, \mathbf{Y}_{1000} \overset{i.i.d}{\sim} t_3(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2, \nu_2)$$

where the location vectors are given by

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, \ \boldsymbol{\mu}_2 = \begin{bmatrix} 2.5 & 2.5 & 2.5 \end{bmatrix}^\top,$$

the scale matrices are given by $\boldsymbol{\Sigma}_1 = \frac{1}{2}\mathbf{I}_3$, $\boldsymbol{\Sigma}_2 = \frac{1}{4}\mathbf{I}_3$, and the degrees of freedom are given by $\nu_1 = 20$ and $\nu_2 = 3$.

Figure 6.1 shows pairwise scatter plots of the resulting dataset. We can clearly see the effects of the t-distribution's heavier tails, as we observe several extreme observations in the purple group due to its relatively low degrees of freedom. As a result, we would expect that using component densities with heavier tails than the multivariate Gaussian distribution should produce a better model fit.

To test this, we used our Julia package FactorMixtures, to fit the MFA, MtFA, MSLFA, MCNFA, MGHFA, MFA-BS and MFA-L models to this dataset. Since $p = 3$ in this example, the Ledermann bound is $q_{\max} = 1$, so no inference on q is required.

Table 6.1 and Table 6.2 show the BIC, ARI, fitting time (in seconds) and the inferred number of components, g, for the best model of each type, chosen using the BIC. We observe that as expected, the six MMVMNFA models all obtained lower BICs than the MFA model. In addition, the difference in BIC between the best MFA model and the best of each of the six other MMVMNFA models was greater than ten, providing very strong evidence that the MMVMNFA models are all able to describe the data better than the standard MFA model. All of the MMVMNFA models correctly inferred $g = 2$, except for the MFA-L model which inferred $g = 5$. The MFA model, in comparison, incorrectly inferred $g = 3$. While the MFA-L model was able to obtain the lowest BIC of any of the models, the fact that it inferred $g = 5$ shows that using the incorrect parametric form can still lead to undesirable results, even when the data is well separated.

Since the data was generated with only two components, it is unsurprising that the five-component MFA-L model achieved the lowest ARI of any of the models. The second lowest was given by the MFA model, which also overestimated the number of components. Of the remaining models, which all correctly inferred g, the MtFA model (perhaps unsurprisingly) achieved the lowest BIC.

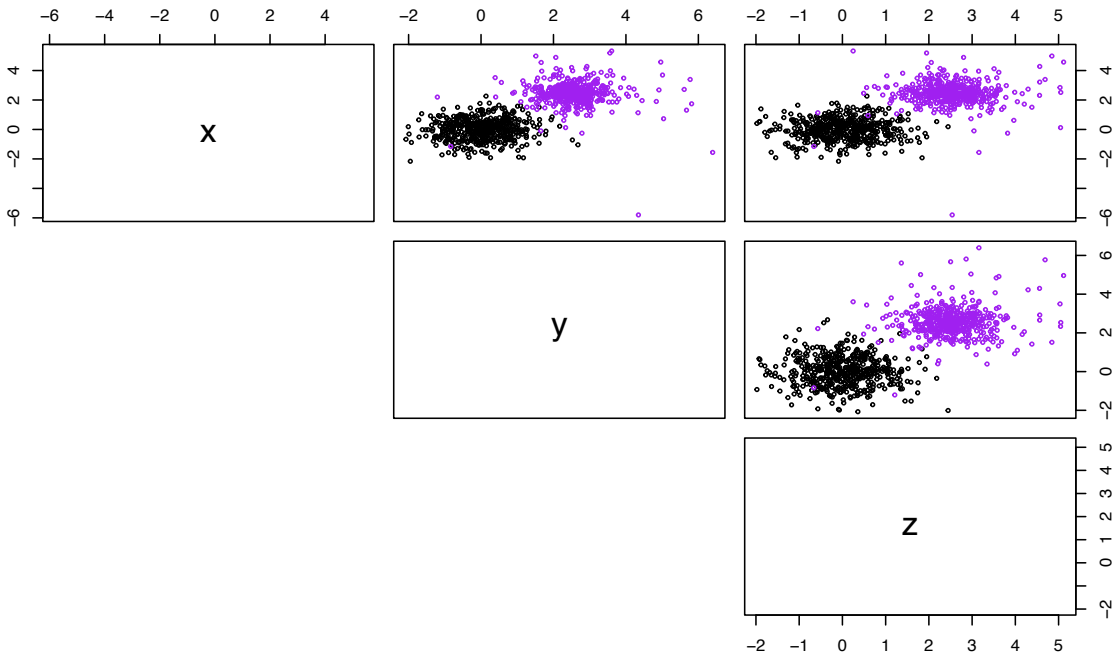


Figure 6.1: The synthetic two-component mixture of t-distributions dataset with the first 500 points coloured black and the second 500 coloured purple.

The MFA model took the least amount of time to fit. It should be noted that this is not an average time for all 30 initialisations of each type of model fitted for a particular combination of g and q. Rather, it is the time taken by the best fitting model (according to the BIC) of each type, out of all 150 models which were fitted for each type. However, it is reasonable to expect that the MFA models should take less time to fit on average, compared to the MMVMNFA models. This is because MFA-ECM-2 has less latent variables than MMVMNFA-ECM-1, since MMVMNFA-ECM-1 also treats the scaling variables W_j as latent variables. In addition, it is also clear that the MSLFA model took orders of magnitude longer to fit than the other models, mainly due to the numerical integration required to compute Equation (5.29).

This example shows that the MMVMNFA models may be worthwhile using, especially for data that appears to have heavy-tailed distributions or outliers, as they can achieve better model fits and clustering results compared to the standard MFA model. It also shows that using an incorrect parametric form can lead to undesirable answers, even when the data is well separated.

	MFA	MtFA	MSLFA	MCNFA
BIC	7839.09	7762.51	7769.87	7806.69
ARI	0.8764	0.9801	0.9801	0.984
Time (seconds)	0.1137	0.8499	100.351	0.5446
Inferred g	3	2	2	2

Table 6.1: The BIC, ARI, fitting time (in seconds) and inferred value of g for the best MFA, MtFA, MSLFA and MCNFA models (according to the BIC) on the synthetic two-component mixture of t-distributions dataset.

	MFA	MGHFA	MFA-BS	MFA-L
BIC	7839.09	7812.43	7812.26	7739.02
ARI	0.8764	0.984	0.9801	0.6419
Time (seconds)	0.1137	7.0192	0.9208	1.8228
Inferred g	3	2	2	5

Table 6.2: The BIC, ARI, fitting time (in seconds) and inferred value of g for the best MFA, MGHFA, MFA-BS and MFA-L models (according to the BIC) on the synthetic two-component mixture of t-distributions dataset

6.2 Simulation Study 2

The second simulation study will compare the MFA model to the MMVMNFA models on multivariate data exhibiting both heavy tails and multivariate skewness. To achieve this, we created a synthetic two-component mixture of generalised hyperbolic distributions dataset by randomly generating

$$\mathbf{Y}_1, \ldots, \mathbf{Y}_{500} \overset{i.i.d}{\sim} \mathrm{GH}_3\left(\lambda_1, \omega_1, \omega_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\beta}_1\right)$$

and

$$\mathbf{Y}_{501}, \ldots, \mathbf{Y}_{1000} \overset{i.i.d}{\sim} \mathrm{GH}_3\left(\lambda_2, \omega_2, \omega_2, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2, \boldsymbol{\beta}_2\right),$$

where the location vectors are given by

$$\boldsymbol{\mu}_1 = \begin{bmatrix} -10 & -5 & -15 \end{bmatrix}^\top, \ \boldsymbol{\mu}_2 = \begin{bmatrix} 15 & 0 & 10 \end{bmatrix}^\top,$$

the stochastic mean vectors are given by

$$\boldsymbol{\beta}_1 = \begin{bmatrix} 5 & 5 & 10 \end{bmatrix}^\top, \ \boldsymbol{\beta}_2 = \begin{bmatrix} -2 & 4 & -2 \end{bmatrix}^\top,$$

the scale matrices are given by $\boldsymbol{\Sigma}_1 = \sqrt{2}\mathbf{I}_3$, $\boldsymbol{\Sigma}_2 = \mathbf{I}_3$, and where $\lambda_1 = 5$, $\lambda_2 = 2$, $\omega_1 = 5$ and $\omega_2 = 10$.

Figure 6.2 shows pairwise scatter plots of the resulting dataset. This dataset clearly exhibits multivariate skewness as well as heavy tails, as, for example, the black component has heavy tails in the positive-x positive-y direction, but no corresponding heavy tail in the negative-x negative-y direction.

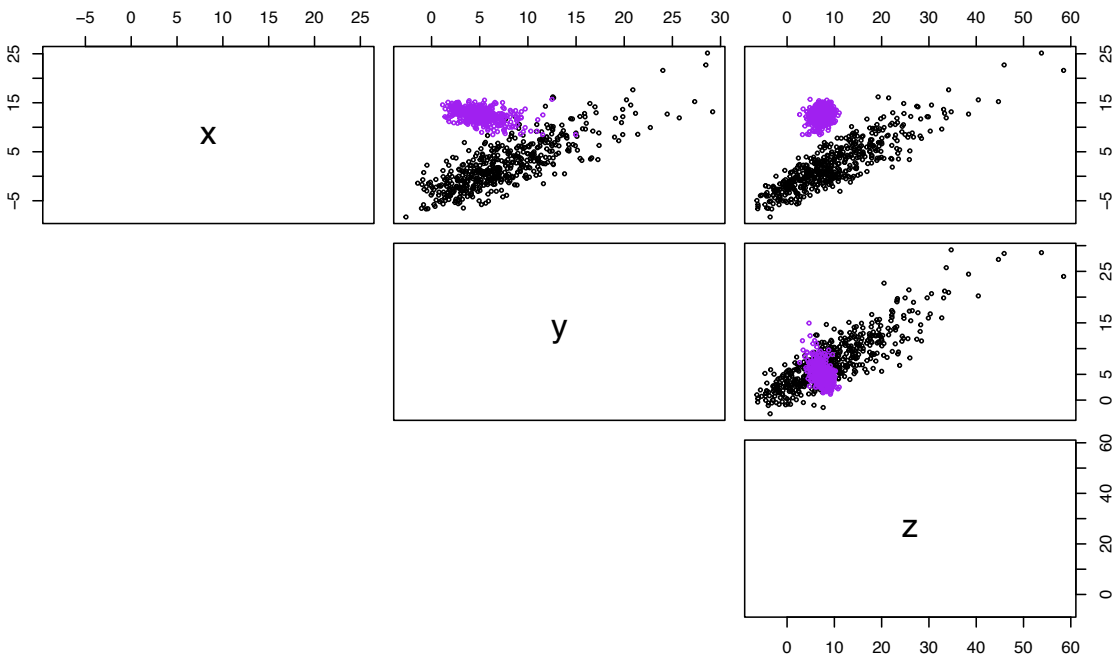


Figure 6.2: The synthetic two-component mixture of generalised hyperbolic distributions dataset with the first 500 points coloured black and the second 500 coloured purple.

	MFA	MtFA	MSLFA	MCNFA
BIC	14938.9	14939.3	14935.8	14952.7
ARI	0.839	0.7644	0.7632	1.0
Time (seconds)	0.0342	0.5188	94.4417	0.5521
Inferred g	3	3	3	2

Table 6.3: The BIC, ARI, fitting time (in seconds) and inferred value of g for the best MFA, MtFA, MSLFA and MCNFA models (according to the BIC) on the synthetic two-component mixture of generalised hyperbolic distributions dataset.

Table 6.3 and Table 6.4 show the BIC, ARI, fitting time and inferred number of components for the best model of each type, chosen according to the BIC. Since $p = 3$ again, no

	MFA	MGHFA	MFA-BS	MFA-L
BIC	14938.9	14825.4	14813.5	14855.0
ARI	0.839	1.0	1.0	0.9766
Time (seconds)	0.0342	5.9934	1.3639	1.0793
Inferred g	3	2	2	3

Table 6.4: The BIC, ARI, fitting time (in seconds) and inferred value of g for the best MFA, MGHFA, MFA-BS and MFA-L models (according to the BIC) on the synthetic two-component mixture of generalised hyperbolic distributions dataset.

inference on q is required for this example either. In this case, we observe that all but the MGHFA, MFA-BS and MCNFA models incorrectly inferred three components. The three aforementioned models all correctly inferred two components and also achieved perfect ARIs. In comparison, the MFA model inferred three components and obtained a much lower ARI of 0.839.

Interestingly, the BIC of the best MFA model was actually lower than the BIC of the best MCNFA model. However, the MCNFA model clearly describes the sub-population structure of the dataset better than the MFA model, as it correctly inferred the number of components where the MFA model did not, and also achieved a perfect ARI.

The BIC of the best MFA model was actually lower than the BIC of the best MtFA model (although by less than 0.5), but the MFA model also achieved a considerably higher ARI, which suggests that it describes the dataset better than the MtFA model. In addition, while the best MSLFA model obtained a slightly lower BIC than the best MFA model, this difference was less than four, so it does not provide strong evidence to suggest that the MSLFA model was superior. In addition, the ARI of the best MSLFA model was also much lower than the ARI of the best MFA model, which in combination suggests that the best MSLFA model is no better than the standard MFA model for this dataset.

However, we observe that the differences in BIC between the best MGHFA, MFA-BS and MFA-L models and the best MFA model were all greater than ten. The lowest BIC overall was achieved by the MFA-BS model. In addition, of these three only the MFA-L model incorrectly inferred a third component, but its ARI was still higher than that of the best MFA model which also incorrectly inferred three components. The two best models according to the BIC, the MFA-BS and MGHFA models, also both achieved perfect ARIs of 1.0. Recalling that each of these three models can all capture multivariate skewness whereas the former three cannot, these results are not altogether surprising. This example shows that, as expected, the MMVMNFA models, and in particular the most general cases where $\boldsymbol{\beta}_i \neq \mathbf{0}$, are more suitable than the standard MFA model for modelling multivariate data with skewness and heavy tails.

6.3 Seeds data

The next dataset which we will consider contains measurements of the geometrical properties of the seeds of three different wheat varieties [Charytanowicz et al., 2010]. The three varieties of wheat are Kama, Rosa and Canadian. Seventy randomly selected seeds of each variety were chosen and their geometrical properties measured. The dataset has seven predictor variables: area (A), perimeter (P), compactness ($C = 4\pi A/P^2$), the length of the kernel, the width of the kernel, the kernel's asymmetry coefficient and finally the length of the kernel's groove. All of these variables are real-valued and continuous. For this dataset, the Ledermann bound is $q_{\max} = 3$, so each model searched over the range $1 \leq q \leq 3$.

While testing the models on real data is important, it is worth noting that it introduces an additional challenge. Namely, we no longer know the "correct" clustering of the data. For example, if one of the seed varieties was a sub-species of another, it could happen that these two sub-populations are not well separated. Hence, while the clusters produced by the mixture models may be more representative of the observed sub-population structure of the data, they may also be very different to those given by the variety labels. Figure 6.3 shows pairwise scatter plots for the variables in this dataset, coloured by seed type. In this particular case, seeds of the same type appear to form clusters. Some of the variables appear to be poorly separated, but there are also several which appear to be relatively well separated. We therefore expect that a good model for the data should be able to reconstruct the seed labels via the inferred clusterings relatively effectively.

Table 6.5 and Table 6.6 show the BIC, ARI, fitting time, inferred number of components and inferred number of factors for the best model of each type, chosen according to the BIC. We see that in reality, the only model which correctly inferred the number of components was the MGHFA model, which also achieved the highest ARI. The second highest ARI was achieved by the MCNFA model, which inferred $g = 4$. The remaining MMVMNFA models all inferred $g = 2$ and achieved equal or lower ARIs than the regular MFA model, which also inferred $g = 2$. The MFA-L model achieved the lowest BIC for this dataset, followed by the MCNFA model and then the MGHFA model. The differences in BIC between each of these models and the BIC of the best MFA model were all over 1,000, which provides very strong evidence that they are better models for this data than the MFA model. Overall, we see that models from the MMVMNFA family are once again able to outperform the MFA model in terms of overall model fit and clustering ability.

6.4 Australian Institute of Sport (AIS) data

The final dataset which we will consider is the Australian Institute of Sport (AIS) dataset [Telford and Cunningham, 1991]. It contains 12 predictor variables for 102 male athletes

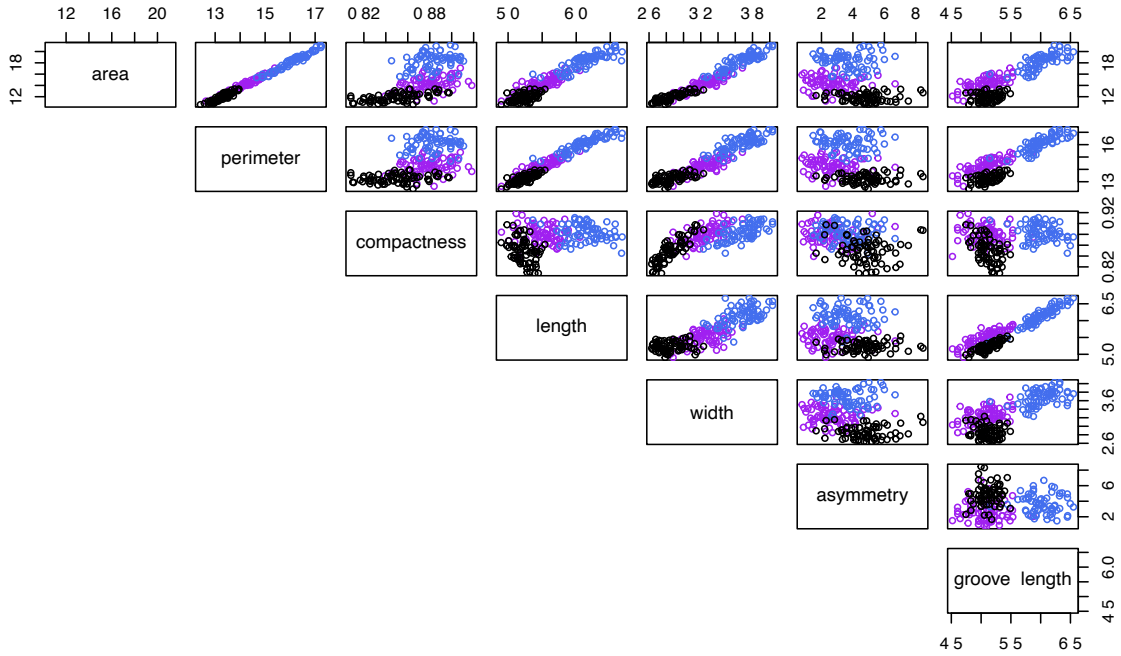


Figure 6.3: Pairwise scatter plots of the seeds dataset. The plots are coloured by seed type, purple points for Kama, blue for Rosa and black for Canadian.

	MFA	MtFA	MSLFA	MCNFA
BIC	-339.49	-332.308	-325.461	-1770.87
ARI	0.5299	0.5299	0.5299	0.6964
Time (seconds)	0.0062	0.1429	24.1094	0.2376
Inferred g	2	2	2	4
Inferred q	2	2	2	3

Table 6.5: The BIC, ARI, fitting time (in seconds), inferred value of g and inferred value of q for the best MFA, MtFA, MSLFA and MCNFA models (according to the BIC) on the seeds dataset.

	MFA	MGHFA	MFA-BS	MFA-L
BIC	-339.49	-1477.42	-468.199	-1912.25
ARI	0.5299	0.7719	0.4902	0.4462
Time (seconds)	0.0062	2.1853	0.4047	0.0591
Inferred g	2	3	2	2
Inferred q	2	3	1	2

Table 6.6: The BIC, ARI, fitting time (in seconds), inferred value of g and inferred value of q for the best MFA, MGHFA, MFA-BS and MFA-L models (according to the BIC) on the seeds dataset.

and 100 female athletes from the Australian Institute of Sport. The data contains 13 variables in total, of which two are categorical. The first of these is the sex of the athlete and the second is the sport which the athlete plays. The sport variable has ten levels: basketball, field, gym, netball, rowing, swimming, track ($>$ 400m), track (sprinting), tennis and water polo. The remaining 11 predictors are all numeric: the athletes' red cell count, white cell count, Hematocrit, Hemoglobin, plasma ferritin concentration, body mass index, sum of skin folds, body fat percentage, lean body mass, height (cm) and weight (kg). For this dataset, the Ledermann bound is $q_{\max} = 6$.

In our analysis, we will compare our model based cluster labels against the sex of the athletes. However, as noted for the seeds data, the sex variable may not be representative of the observed sub-population structure. In particular, the sport variable adds additional complexity. For example, the observed sub-population structure could include one cluster for each sex plus an additional cluster for one of the ten sports for which the body composition of its athletes is very distinct from the body composition of the rest of the athletes.

With this in mind, we will fit two sets of models on the AIS dataset. The first will have $g = 2$ fixed in advance in an attempt to "force" the models to recover the sub-population structure based on sex. The second will search over $1 \leq g \leq 5$ as in the rest of the studies

in this chapter, which will allow the models to choose the optimal number of components according to the data, even if this is not equal to two.

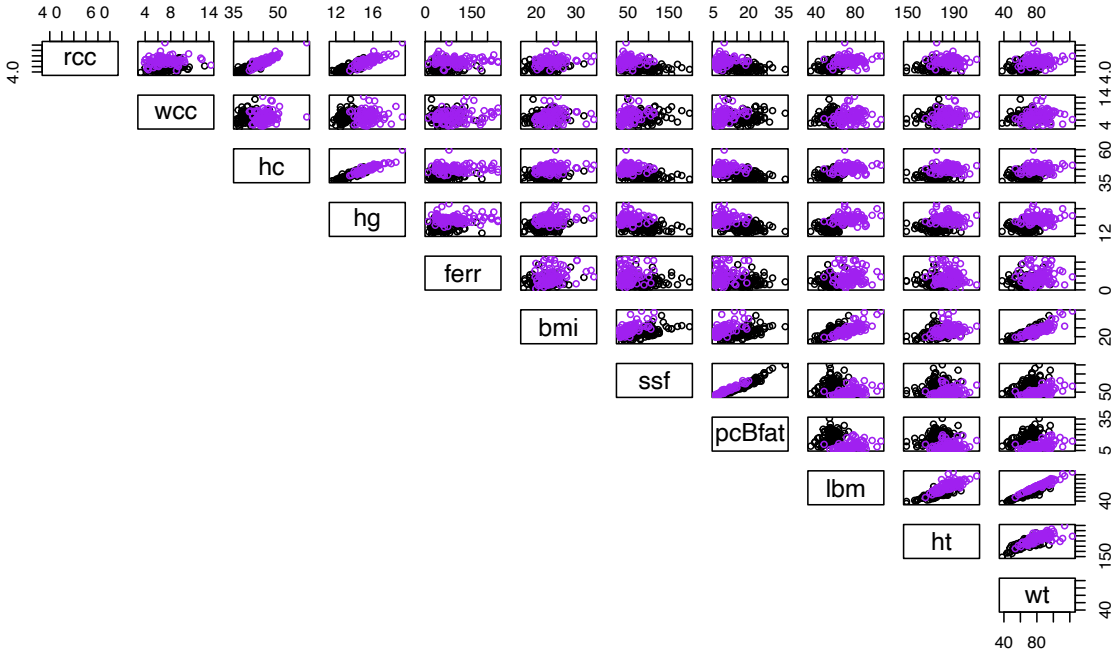


Figure 6.4: Pairwise scatter plots of the AIS dataset. The points are coloured by sex, black points for females and purple for males.

Figure 6.4 shows pairwise scatter plots of all of the numeric variables in the AIS dataset, coloured by sex. We generally observe that the points often form clusters based on sex, but that these clusters are often overlapping.

	MFA	MtFA	MSLFA	MCNFA
BIC	10080.8	9959.16	9951.67	9974.75
ARI	0.922	0.903	0.903	0.903
Time (seconds)	0.1154	0.202	20.0378	0.5204
g	2	2	2	2
Inferred q	4	4	4	4

Table 6.7: The BIC, ARI, fitting time (in seconds), value of g and inferred value of q for the best MFA, MtFA, MSLFA and MCNFA models (according to the BIC) on the AIS dataset when $g = 2$ is fixed in advance.

	MFA	MGHFA	MFA-BS	MFA-L
BIC	10080.8	9985.45	9985.19	8998.11
ARI	0.922	0.922	0.8655	0.8289
Time (seconds)	0.1154	2.4583	1.0676	0.0903
g	2	2	2	2
Inferred q	4	4	4	6

Table 6.8: The BIC, ARI, fitting time (in seconds), value of g and inferred value of q for the best MFA, MGHFA, MFA-BS and MFA-L models (according to the BIC) on the AIS dataset when $g = 2$ is fixed in advance.

Table 6.7 and Table 6.8 show the BIC, ARI, fitting time, number of components and the inferred number of factors for the best model of each type, chosen according to the BIC when $g = 2$ is fixed in advance. We note that the MFA and MGHFA models obtain the joint highest ARI, with the remaining models all obtaining lower ARIs. However, the difference in BIC between the best MFA model and the best of each MMVMNFA model is greater than ten, indicating that they can still explain the observed data than the original MFA model. In particular, the difference between the BIC of the best MFA model and the BIC of the best MGHFA model was almost 100 while their ARIs were identical, which provides strong evidence to suggest that the MGHFA model is a superior model compared to the MFA model in this case.

	MFA	MtFA	MSLFA	MCNFA
BIC	9981.9	9898.04	9892.52	9912.27
ARI	0.389	0.5326	0.543	0.4287
Time (seconds)	0.1196	0.4432	30.8393	0.3773
Inferred g	3	3	3	3
Inferred q	4	4	4	4

Table 6.9: The BIC, ARI, fitting time (in seconds), inferred value of g and inferred value of q for the best MFA, MtFA, MSLFA and MCNFA models (according to the BIC) on the AIS dataset when g is chosen via a naïve search between one and five.

Table 6.9 and Table 6.10, on the other hand, show the BIC, ARI, fitting time, inferred number of components and the inferred number of factors for the best model of each type, chosen according to the BIC when g is chosen via a naïve search between one and five. In this case, we observe that all of the models inferred $g = 3$, except for the MFA-L model which inferred $g = 4$. The ARIs of each model are all much lower than in the previous analysis, but the same is also true of the BICs, as for each model the difference between its best restricted BIC and its best unrestricted BIC is greater than ten. This suggests that while restricting the number of components to be equal to two causes the

	MFA	MGHFA	MFA-BS	MFA-L
BIC	9981.9	9960.81	9962.36	8905.62
ARI	0.389	0.4568	0.4522	0.3431
Time (seconds)	0.1196	2.1326	0.6872	0.0895
Inferred g	3	3	3	4
Inferred q	4	4	4	6

Table 6.10: The BIC, ARI, fitting time (in seconds), inferred value of g and inferred value of q for the best MFA, MGHFA, MFA-BS and MFA-L models (according to the BIC) on the AIS dataset when g is chosen via a naïve search between one and five.

inferred sub-population structure to align with the class labels based on the sex of the athletes more closely, the data actually suggests that a three component model is more appropriate. It is also worth noting that when the number of components is not fixed at two in advance, the difference in BIC between the best MFA model and the best of each MMVMNFA model is greater than ten, which is strong evidence to suggest that the MMVMNFA models are providing better model fits than the regular MFA model.

6.5 Summary

In this chapter, we used our Julia package FactorMixtures to demonstrate that the MMVMNFA models developed in Chapter 5 outperform the standard MFA model when the data exhibits heavy tails and/or skewness. We fitted the MFA model and each of the six instances of MMVMNFA models from Chapter 5 to two synthetic datasets and two real world datasets. We found that in each case, models from the MMVMNFA model family outperformed the standard MFA model in terms of model fit and clustering accuracy.

Chapter 7

Conclusion

This thesis had three main goals: to produce a systematic comparison of the currently available methods for fitting the MFA model in the case where its hyperparameters are unknown, to release a software package for the statistical software `R` which contains implementations of the automatic inference techniques for the MFA model for which no publicly available `R` implementations exist, and to examine the possibility of extending the MFA model. It achieved these aims by:

I Producing an `R` package which contains implementations of five methods for fitting the MFA model when its hyperparameters are unknown. The package, `autoMFA`, is publicly available on CRAN.

II Performing a systematic comparison of the current methods for automatically fitting the MFA model. This included the five methods from the `autoMFA` package and two additional methods from the `IMIFA` package.

III Extending the MFA model to enable parsimonious modelling of multidimensional data with heavy tails and/or multivariate skewness via the MMVMNFA model family. A new EM-type algorithm for this family was produced, and six specific instances of the family were given along with complete EM-type algorithms for each instance. Implementations of each of the EM-type algorithms are available in our Julia package FactorMixtures.

7.1 Summary

This work reviewed seven methods for fitting the MFA model when its two hyperparameters, g (the number of components) and q (the number of factors per component) are unknown. The seven methods are a naïve grid search over both hyperparameters, the

AMoFA algorithm from Kaya and Salah [2015], the VBMFA algorithm from Ghahramani and Beal [1999], two implementations of the AMFA algorithm from Wang and Lin [2020], and the OMIFA and IMIFA algorithms from Murphy et al. [2020].

To facilitate a comparison between the methods, the `autoMFA` package for the statistical programming language `R` was created. This package contains implementations of each of the methods mentioned above, except for OMIFA and IMIFA, for which `R` implementations were already available via the `IMIFA` package [Murphy et al., 2021].

The methods were compared based on their ability to infer the two hyperparameters g and q, as well as general model fit, the clustering accuracy of the fitted models and the amount of time the models take to fit. The naïve grid search (implemented as the `MFA_ECM` method in `autoMFA`) achieved the highest overall mean ARI and BIC among the MFA methods. It also had the highest Rasch model ability estimates for inferring g and q correctly, as well as for inferring g with an error of within ± 2. However, as the grid search needs to search over a range of values for both g and q, it had the longest mean fitting time for any of the methods. The IMIFA algorithm from the `IMIFA` package also performed well, generally achieving results which were comparable with the naïve search. It also obtained the joint highest ability estimate for inferring the number of factors to within an error of ± 2. However, it is based on MCMC inference and therein shares the same weakness as the naïve search, being the second slowest method, on average, in the comparison. The AMFA algorithm (implemented as the `AMFA` method in `autoMFA`) is a reasonable alternative to the aforementioned methods when computation time is a limiting factor. It also generally managed to achieve comparable results to the naïve search method, but took less than half the time to fit its models, on average.

Finally, the MFA model was extended to the MMVMNFA model family, which preserves the parsimonious mixture model formulation of the MFA model whilst allowing the component densities to belong to the more general MVMN family. A general ECM algorithm for parameter estimation of this model family was derived, and the AMFA algorithm was extended to this family as well. Six specific instances of the MMVMNFA model family were provided, with full ECM algorithms provided for each instance. The examples were demonstrated on two synthetic datasets and two real world datasets, where their superior ability to model heavy-tailed data and data exhibiting multivariate skewness was demonstrated in comparison to the standard MFA model. The Julia package FactorMixtures is provided as a computationally efficient way of fitting models from the MMVMNFA family.

7.2 Future Work

Further study in these areas could include the following topics. In the ECM algorithm for the MMVMNFA model family, we assume that $\tilde{\mathbf{S}}_i$ from Equation (2.40) has at least one eigenvalue greater than unity, so that Equation (2.39) provides a real-valued update of the factor loading matrix. In practice, this assumption doesn't always hold, which can lead to complex factor loading matrix updates. Further study should be conducted to find a condition under which at least one eigenvalue is guaranteed to be greater than unity. The introduction of the MMVMNFA family also poses a new question, namely which model to choose from the family for a particular dataset. Further study could also be conducted here to help choose an appropriate MMVMNFA model without having to fit several of each type of model and then choose a final model according to a model selection criterion. In addition, more special cases of the MMVMNFA family could be investigated. Another possible direction for future work is the investigation of automated methods for fitting the Mixtures of Common Factor Analyzers (MCFA) model [Baek et al., 2010], which is an even more parsimonious variant of the MFA model where all of the factor loading matrices are the same.

The Julia package FactorMixtures will be expanded upon in the future. As well as efficiency improvements, support for the generalisation of the AMFA algorithm (Algorithm 3.1) for the MMVMNFA family (as proposed in Chapter 5) will be added. Another potential addition is the ability for users to specify their own scaling density function with corresponding E-step and M-step estimates.

Appendix A

Mathematical details

A.1 The number of parameters required by the Gaussian model and FA model

Recall the Gaussian model for the data $\mathbf{Y}_j$ is

$$\mathbf{Y}_j \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

independently for $j = 1, \ldots, n$. Under this model, we need to estimate the $p \times 1$ mean vector $\boldsymbol{\mu}$ and the $p \times p$ covariance matrix $\boldsymbol{\Sigma}$. The estimation of $\boldsymbol{\mu}$ requires p scalar parameters. Since $\boldsymbol{\Sigma}$ is symmetrical, we only need to estimate the elements of its upper triangle (including those along the diagonal), which contains a total of $\sum_{i=1}^{p} i = p(p+1)/2$ scalar parameters. Hence,

$$k^*_{\mathrm{G}} = p + \frac{p(p-1)}{2}.$$

For the FA model, we need to estimate p scalar parameters for both the $p \times 1$ mean vector $\boldsymbol{\mu}$ and the diagonal $p \times p$ error-variance matrix $\mathbf{D}$. The estimation of the $p \times q$ loading matrix $\mathbf{B}$ requires $pq - q(q-1)/2$ scalar parameters, since we need to apply exactly $q(q-1)/2$ constraints to $\mathbf{B}$ to guarantee identifiability, as discussed in Section 2.2. Hence,

$$k^*_{\mathrm{FA}}(p, q) = pq - \frac{q(q-1)}{2} + 2p.$$

A.2 Derivation of Equation (2.36)

In Equation (2.36), we want to find the log-likelihood of the parameters in $\boldsymbol{\theta}_2$. We assume that we have the updated $k+1$st-step estimates for the parameters in $\boldsymbol{\theta}_1$, but un-updated k-step estimates for the parameters in $\boldsymbol{\theta}_2$. We define

$$\boldsymbol{\Theta}^{(k+1/2)} := (\boldsymbol{\theta}_1^{(k+1)}, \boldsymbol{\theta}_2^{(k)})$$

to be the vector of partially updated k-step estimates. For notational convenience, define

$$l_1 := \ell\left(\boldsymbol{\theta}_2 \mid \mathbf{Y}, \mathbf{Z}; \boldsymbol{\Theta}^{(k+1/2)}\right),$$

to be the log-likelihood function of the parameters in $\boldsymbol{\theta}_2$, given the $k+1$st step estimates of the parameters in $\boldsymbol{\theta}_1$. To evaluate l_1, we begin as usual, breaking up the joint density into a product of conditional densities.

$$\begin{aligned}
l_1 &= \log \prod_{j=1}^{n} f\left(\mathbf{y}_j, \mathbf{z}_j \mid \boldsymbol{\theta}_2; {\boldsymbol{\theta}_1}^{(k)}\right) \\
&= \log \prod_{j=1}^{n} f\left(\mathbf{y}_j \mid \mathbf{z}_j, \boldsymbol{\theta}_2; {\boldsymbol{\theta}_1}^{(k)}\right) f\left(\mathbf{z}_j \mid \boldsymbol{\theta}_2; {\boldsymbol{\theta}_1}^{(k)}\right) \\
&\propto -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{g} z_{ij} \left(\log |\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i| + (\mathbf{y}_j - {\boldsymbol{\mu}_i}^{(k+1)})^\top \left(\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i\right)^{-1} (\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})\right).
\end{aligned}$$

Noticing that the Mahalanobis distance term is a scalar, by definition it must be equal to its own trace. Accordingly,

$$\begin{aligned}
l_1 &= -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\Bigg(z_{ij}\log|\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i| \\
&\quad + z_{ij}\operatorname{tr}\left\{(\mathbf{y}_j-\boldsymbol{\mu}_i^{(k+1)})^\top\left(\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i\right)^{-1}(\mathbf{y}_j-\boldsymbol{\mu}_i^{(k+1)})\right\}\Bigg) \\
&= -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\Bigg(z_{ij}\log|\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i| \\
&\quad + \operatorname{tr}\left\{z_{ij}(\mathbf{y}_j-\boldsymbol{\mu}_i{}^{(k+1)})(\mathbf{y}_j-\boldsymbol{\mu}_i{}^{(k+1)})^\top\left(\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i\right)^{-1}\right\}\Bigg) \\
&= -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}z_{ij}\log|\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i| \\
&\quad - \frac{1}{2}\sum_{i=1}^{g}\operatorname{tr}\left\{\sum_{j=1}^{n}\left(\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i\right)^{-1}z_{ij}(\mathbf{y}_j-\boldsymbol{\mu}_i{}^{(k+1)})(\mathbf{y}_j-\boldsymbol{\mu}_i{}^{(k+1)})^\top\right\} \\
&= -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}z_{ij}\log|\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i| \\
&\quad - \frac{1}{2}\sum_{i=1}^{g}\operatorname{tr}\left\{\left(\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i\right)^{-1}\sum_{j=1}^{n}z_{ij}(\mathbf{y}_j-\boldsymbol{\mu}_i{}^{(k+1)})(\mathbf{y}_j-\boldsymbol{\mu}_i{}^{(k+1)})^\top\right\}.
\end{aligned}$$

Taking the conditional expectation of l_1 given the observed data and then multiplying and dividing the trace term by $n\pi_i^{(k+1)}$ shows that the Q-function is given by

$$Q\left(\boldsymbol{\theta}_2;\boldsymbol{\Theta}^{(k+1/2)}\right)\propto -\frac{n}{2}\sum_{i=1}^{g}\pi_i^{(k+1)}\left[\log|\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i|+\operatorname{tr}\left\{\left(\mathbf{B}_i\mathbf{B}_i^\top+\mathbf{D}_i\right)^{-1}\mathbf{S}_i^{(k)}\right\}\right],$$

as desired, where

$$\mathbf{S}_i^{(k)}:=\frac{1}{n\pi_i^{(k+1)}}\sum_{j=1}^{n}\tau_{ij}^{(k)}(\mathbf{y}_j-\boldsymbol{\mu}_i^{(k+1)})(\mathbf{y}_j-\boldsymbol{\mu}_i^{(k+1)})^\top.$$

A.3 The Partial Derivatives of Equation (2.36)

Consider the expression

$$Q_i\propto -\frac{n}{2}\pi_i\left(\log|\boldsymbol{\Sigma}_i|+\operatorname{tr}\left\{\boldsymbol{\Sigma}_i^{-1}\mathbf{S}_i\right\}\right),$$

where as usual $\boldsymbol{\Sigma}_i = \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i$. Then

$$\frac{\partial Q_i}{\partial \boldsymbol{\Sigma}_i} = -\frac{n}{2}\pi_i \left[\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_i^{-1}\mathbf{S}_i\boldsymbol{\Sigma}_i^{-1}\right],$$

which we obtain by applying Identities 57 and 124 from Petersen and Pedersen [2012].

Noting that when viewed as a function of $\mathbf{D}_i$, $\boldsymbol{\Sigma}_i$ is just $\mathbf{D}_i$ plus a constant, it follows immediately that

$$\frac{\partial Q_i}{\partial \mathbf{D}_i} = -\frac{n}{2}\pi_i \left[\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_i^{-1}\mathbf{S}_i\boldsymbol{\Sigma}_i^{-1}\right],$$

as desired.

To find $\frac{\partial Q_i}{\partial \mathbf{B}_i}$, we will use the matrix chain rule, which states that

$$\frac{\partial Q_i}{\partial b_{kj}} = \sum_{i'=1}^{n}\sum_{j'=1}^{n} \frac{\partial Q_i}{\partial \boldsymbol{\Sigma}_{i'j'}}\frac{\partial \boldsymbol{\Sigma}_{i'j'}}{\partial b_{kj}},$$

where we are using the shorthand $\boldsymbol{\Sigma}_{i'j'}$ to mean $[\boldsymbol{\Sigma}_i]_{i'j'}$ and $b_{kj} = [\mathbf{B}_i]_{kj}$. For convenience, define $\mathbf{F}_i = -\frac{n}{2}\pi_i(\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_i^{-1}\mathbf{S}_i\boldsymbol{\Sigma}_i^{-1})$. Then we know that

$$\frac{\partial Q_i}{\partial \boldsymbol{\Sigma}_{i'j'}} = [\mathbf{F}_i]_{i'j'} := f_{i'j'}.$$

With respect to $\mathbf{B}_i$, $\boldsymbol{\Sigma}_i$ is just $\mathbf{B}_i\mathbf{B}_i^\top$ plus a constant. If we define $\tilde{b}_{m,l} = \sum_{h=1}^{n} b_{mh}b_{lh}$, then $\boldsymbol{\Sigma}_i$ is proportional to the following matrix (here we have assumed, without loss of generality, that $k \leq j$.

$$\begin{bmatrix}
\tilde{b}_{1,1} & \cdots & \tilde{b}_{1,k-1} & \tilde{b}_{1,k} & \tilde{b}_{1,k+1} & \cdots & \tilde{b}_{1,j-1} & \tilde{b}_{1,j} & \tilde{b}_{1,j+1} & \cdots & \tilde{b}_{1,n} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\tilde{b}_{k-1,1} & \cdots & \tilde{b}_{k-1,k-1} & \tilde{b}_{k-1,k} & \tilde{b}_{k-1,k+1} & \cdots & \tilde{b}_{k-1,j-1} & \tilde{b}_{k-1,j} & \tilde{b}_{k-1,j+1} & \cdots & \tilde{b}_{k-1,n} \\
\tilde{b}_{k,1} & \cdots & \tilde{b}_{k,k-1} & \tilde{b}_{k,k} & \tilde{b}_{k,k+1} & \cdots & \tilde{b}_{k,j-1} & \tilde{b}_{k,j} & \tilde{b}_{k,j+1} & \cdots & \tilde{b}_{k,n} \\
\tilde{b}_{k+1,1} & \cdots & \tilde{b}_{k+1,k-1} & \tilde{b}_{k+1,k} & \tilde{b}_{k+1,k+1} & \cdots & \tilde{b}_{k+1,j-1} & \tilde{b}_{k+1,j} & \tilde{b}_{k+1,j+1} & \cdots & \tilde{b}_{k+1,n} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\tilde{b}_{j-1,1} & \cdots & \tilde{b}_{j-1,k-1} & \tilde{b}_{j-1,k} & \tilde{b}_{j-1,k+1} & \cdots & \tilde{b}_{j-1,j-1} & \tilde{b}_{j-1,j} & \tilde{b}_{j-1,j+1} & \cdots & \tilde{b}_{j-1,n} \\
\tilde{b}_{j,1} & \cdots & \tilde{b}_{j,k-1} & \tilde{b}_{j,k} & \tilde{b}_{j,k+1} & \cdots & \tilde{b}_{j,j-1} & \tilde{b}_{j,j} & \tilde{b}_{j,j+1} & \cdots & \tilde{b}_{j,n} \\
\tilde{b}_{j+1,1} & \cdots & \tilde{b}_{j+1,k-1} & \tilde{b}_{j+1,k} & \tilde{b}_{j+1,k+1} & \cdots & \tilde{b}_{j+1,j-1} & \tilde{b}_{j+1,j} & \tilde{b}_{j+1,j+1} & \cdots & \tilde{b}_{j+1,n} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\tilde{b}_{n,1} & \cdots & \tilde{b}_{n,k-1} & \tilde{b}_{n,k} & \tilde{b}_{n,k+1} & \cdots & \tilde{b}_{n,j-1} & \tilde{b}_{n,j} & \tilde{b}_{n,j+1} & \cdots & \tilde{b}_{n,n}
\end{bmatrix}$$

Suppose for the moment that we were interested in differentiating Q_i with respect to b_{kj}. We need to know which elements of $\boldsymbol{\Sigma}_i$ depend on b_{kj}, because these will be the only elements where $\partial \boldsymbol{\Sigma}_{i'j'}/\partial b_{kj}$ is not equal to zero.

Notice that $\tilde{b}_{m,l}$ will depend on b_{kj} only if one or both of m and l is equal to k. So we are only interested in the entries of $\boldsymbol{\Sigma}_i$ of the form $[\boldsymbol{\Sigma}_i]_{kx}$ or $[\boldsymbol{\Sigma}_i]_{xk}$, where $x \in \{1, \ldots, n\}$. In other words, we see that the only entries of $\boldsymbol{\Sigma}_i$ which depend on b_{kj} occur in the k^{th} row or column of $\boldsymbol{\Sigma}_i$. So we know that $\frac{\partial \boldsymbol{\Sigma}_{i'j'}}{\partial b_{kj}}$ will be zero except for in row or column k. In other words, we can write

$$\frac{\partial Q_i}{\partial b_{kj}} = \sum_{r=1}^{n} f_{kr} \frac{\partial \boldsymbol{\Sigma}_{kr}}{\partial b_{kj}} + \sum_{\substack{r=1 \\ r \neq k}}^{n} f_{rk} \frac{\partial \boldsymbol{\Sigma}_{rk}}{\partial b_{kj}}.$$

Also,

$$\begin{aligned} \frac{\partial \boldsymbol{\Sigma}_{kr}}{\partial b_{kj}} &= \frac{\partial}{\partial b_{kj}} \left(\sum_{h=1}^{n} b_{rh} b_{kh} \right) \\ &= b_{rj} + \mathbb{I}_{\{r=k\}} b_{rj}. \end{aligned}$$

Because of the symmetry of $\boldsymbol{\Sigma}_i$ and $\mathbf{S}_i$, we know that $f_{kr} = f_{rk}$ and $\boldsymbol{\Sigma}_{kr} = \boldsymbol{\Sigma}_{rk}$. So it follows that

$$\begin{aligned} \frac{\partial Q_i}{\partial b_{kj}} &= \sum_{\substack{r=1 \\ r \neq k}}^{n} f_{kr} \frac{\partial \boldsymbol{\Sigma}_{kr}}{\partial b_{kj}} + \sum_{\substack{r=1 \\ r \neq k}}^{n} f_{rk} \frac{\partial \boldsymbol{\Sigma}_{rk}}{\partial b_{kj}} + 2 \sum_{r=1}^{n} f_{kk} b_{kj} \\ &= 2 \sum_{r=1}^{n} f_{kr} b_{rj}, \end{aligned}$$

which is just the $(k, j)^{th}$ element of

$$-n\pi_i \left(\boldsymbol{\Sigma}_i^{-1} \mathbf{B}_i - \boldsymbol{\Sigma}_i^{-1} \mathbf{S}_i \boldsymbol{\Sigma}_i^{-1} \mathbf{B}_i \right).$$

So we can conclude that

$$\frac{\partial Q_i}{\partial \mathbf{B}_i} = -n\pi_i (\boldsymbol{\Sigma}_i^{-1} \mathbf{B}_i - \boldsymbol{\Sigma}_i^{-1} \mathbf{S}_i \boldsymbol{\Sigma}_i^{-1} \mathbf{B}_i),$$

as desired.

A.4 Derivation of Equation (2.41)

We can derive Equation (2.41) using the following argument. For notational convenience, define

$$c_1 := \left| \left[\mathbf{D}_i^{(k)} \right]^{-\frac{1}{2}} \right| \cdot \left| \hat{\mathbf{B}}_{im_i}^{(k+1)} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)} \right]^{\top} + \mathbf{D}_i^{(k)} \right| \cdot \left| \left[\mathbf{D}_i^{(k)} \right]^{-\frac{1}{2}} \right|.$$

It follows immediately that

$$\begin{aligned} c_1 &= \left| \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \left(\hat{\mathbf{B}}_{im_i}^{(k+1)} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top + \mathbf{D}_i^{(k)}\right) \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \right| \\ &= \left| \mathbf{I}_p + \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \hat{\mathbf{B}}_{im_i}^{(k+1)} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \right|. \end{aligned}$$

Next, consider the general block partitioned matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

where $\mathbf{A}$ and $\mathbf{D}$ are square invertible sub-matrices. Then

$$|\mathbf{A}||\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}| = |\mathbf{D}||\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}|.$$

For proof of this result, see Zhang [2005]. We will apply this result to the block partitioned matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{I}_{m_i} & -\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \\ \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \hat{\mathbf{B}}_{im_i}^{(k+1)} & \mathbf{I}_p \end{bmatrix}$$

which produces the identity

$$\left| \mathbf{I}_p + \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \hat{\mathbf{B}}_{im_i}^{(k+1)} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \right| = \left| \mathbf{I}_{m_i} + \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-1} \hat{\mathbf{B}}_{im_i}^{(k+1)} \right|.$$

Now recall that

$$\hat{\mathbf{B}}_{im_i}^{(k+1)} := \left[\mathbf{D}_i^{(k)}\right]^{1/2} \mathbf{U}_{m_i} (\mathbf{\Lambda}_{m_i} - \mathbf{I}_{m_i})^{1/2},$$

where $\mathbf{U}_{m_i}^\top \mathbf{U}_{m_i} = \mathbf{I}_{m_i}$, so it is easy to verify that

$$\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-1} \hat{\mathbf{B}}_{im_i}^{(k+1)} = \mathbf{\Lambda}_{m_i} - \mathbf{I}_{m_i}.$$

Hence,

$$\begin{aligned} c_1 &= \left| \mathbf{I}_{m_i} + \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-1} \hat{\mathbf{B}}_{im_i}^{(k+1)} \right| \\ &= |\mathbf{I}_{m_i} + (\mathbf{\Lambda}_{m_i} - \mathbf{I}_{m_i})| \\ &= |\mathbf{\Lambda}_{m_i}| \\ &= \prod_{i=1}^{m_i} \lambda_i \end{aligned}$$

as desired.

A.5 Derivation of Equation (2.44)

We wish to show that

$$\operatorname{tr}\left\{\left(\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top+\mathbf{D}_i^{(k)}\right)^{-1}\mathbf{S}_i^{(k)}\right\}=\sum_{i=1}^{p}\lambda_i-\sum_{i=1}^{m_i}(\lambda_i-1).$$

First, for convenience define

$$t_1=\operatorname{tr}\left\{\left(\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top+\mathbf{D}_i^{(k)}\right)^{-1}\mathbf{S}_i^{(k)}\right\}.$$

Then it follows that

$$\begin{aligned}
t_1&=\operatorname{tr}\left\{\mathbf{S}_i^{(k)}\left(\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top+\mathbf{D}_i^{(k)}\right)^{-1}\right\}\\
&=\operatorname{tr}\left\{\mathbf{S}_i^{(k)}\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\left[\mathbf{D}_i^{(k)}\right]^{\frac{1}{2}}\left(\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top+\mathbf{D}_i^{(k)}\right)^{-1}\left[\mathbf{D}_i^{(k)}\right]^{\frac{1}{2}}\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right\}\\
&=\operatorname{tr}\left\{\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\mathbf{S}_i^{(k)}\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\left[\mathbf{D}_i^{(k)}\right]^{\frac{1}{2}}\left(\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top+\mathbf{D}_i^{(k)}\right)^{-1}\left[\mathbf{D}_i^{(k)}\right]^{\frac{1}{2}}\right\}.
\end{aligned}$$

Now define

$$d_1=\left[\mathbf{D}_i^{(k)}\right]^{\frac{1}{2}}\left(\hat{\mathbf{B}}_{im_i}^{(k+1)}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top+\mathbf{D}_i^{(k)}\right)^{-1}\left[\mathbf{D}_i^{(k)}\right]^{\frac{1}{2}}.$$

Applying the Woodbury identity to d_1,

$$\begin{aligned}
d_1&=\mathbf{I}_p-\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\hat{\mathbf{B}}_{im_i}^{(k+1)}\left(\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top\left[\mathbf{D}_i^{(k)}\right]^{-1}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top+\mathbf{I}_q\right)^{-1}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\\
&=\mathbf{I}_p-\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\hat{\mathbf{B}}_{im_i}^{(k+1)}\boldsymbol{\Lambda}_{m_i}^{-1}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}.
\end{aligned}$$

So,

$$\begin{aligned}
t_1&=\operatorname{tr}\left\{\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\mathbf{S}_i^{(k)}\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\left(\mathbf{I}_p-\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\hat{\mathbf{B}}_{im_i}^{(k+1)}\boldsymbol{\Lambda}_{m_i}^{-1}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right)\right\}\\
&=\operatorname{tr}\left\{\tilde{\mathbf{S}}_i\right\}-\operatorname{tr}\left\{\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\mathbf{S}_i^{(k)}\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\hat{\mathbf{B}}_{im_i}^{(k+1)}\boldsymbol{\Lambda}_{m_i}^{-1}\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right\}
\end{aligned}$$

Equation 13 of Jöreskog [1967] implies that

$$\mathbf{S}_i^{(k)}\left[\mathbf{D}_i^{(k)}\right]^{-1}\hat{\mathbf{B}}_{im_i}^{(k+1)}=\hat{\mathbf{B}}_{im_i}^{(k+1)}\boldsymbol{\Lambda}_{m_i},$$

so

$$\begin{aligned}
t_1 &= \operatorname{tr}\left\{\tilde{\mathbf{S}}_i\right\} - \operatorname{tr}\left\{\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \hat{\mathbf{B}}_{im_i}^{(k+1)} \boldsymbol{\Lambda}_{m_i} \boldsymbol{\Lambda}_{m_i}^{-1} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right\} \\
&= \operatorname{tr}\left\{\tilde{\mathbf{S}}_i\right\} - \operatorname{tr}\left\{\left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}} \hat{\mathbf{B}}_{im_i}^{(k+1)} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-\frac{1}{2}}\right\} \\
&= \operatorname{tr}\left\{\tilde{\mathbf{S}}_i\right\} - \operatorname{tr}\left\{\left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top \left[\mathbf{D}_i^{(k)}\right]^{-1} \hat{\mathbf{B}}_{im_i}^{(k+1)}\right\} \\
&= \operatorname{tr}\left\{\tilde{\mathbf{S}}_i\right\} - \operatorname{tr}\left\{\boldsymbol{\Lambda}_{m_i} - \mathbf{I}_{m_i}\right\} \\
&= \sum_{l=1}^{p} \lambda_l - \sum_{l=1}^{m_i} (\lambda_l - 1).
\end{aligned}$$

The second to last step is justified in Appendix A.4.

A.6 Derivation of Equation (2.45)

Define

$$l_2 = \ell(\boldsymbol{\theta}_2) + \frac{1}{2} \sum_{i=1}^{g} n\pi_i^{(k+1)} \log |\tilde{\mathbf{S}}_i|.$$

Equation (2.45) follows from Equation (2.44)

$$\begin{aligned}
l_2 &= -\frac{1}{2} \sum_{i=1}^{g} n\pi_i^{(k+1)} \left[\log \left| \hat{\mathbf{B}}_{im_i}^{(k+1)} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top + \mathbf{D}_i^{(k)} \right| - \log |\tilde{\mathbf{S}}_i| \right. \\
&\qquad \left. + \operatorname{tr}\left\{\left(\hat{\mathbf{B}}_{im_i}^{(k+1)} \left[\hat{\mathbf{B}}_{im_i}^{(k+1)}\right]^\top + \mathbf{D}_i^{(k)}\right)^{-1} \mathbf{S}_i^{(k)}\right\}\right] \\
&\propto -\frac{1}{2} \sum_{i=1}^{g} n\pi_i^{(k+1)} \left[\sum_{l=1}^{m_i} \log \lambda_l - \sum_{l=1}^{p} \log \lambda_l + \sum_{l=1}^{p} \lambda_l - \sum_{l=1}^{m_i} (\lambda_l - 1)\right],
\end{aligned}$$

using Equation (2.41) and Equation (2.44). Recall that we want to find the optimal value of m_i, and the sums which run from 1 to p are clearly constant with respect to m_i, so

$$l_2 \propto -\frac{1}{2} \left[\sum_{i=1}^{g} n\pi_i^{(k+1)} \sum_{l=1}^{m_i} (\log \lambda_l - \lambda_l + 1)\right].$$

A.7 Derivation of Equation (2.50)

Recall Equation (2.48),

$$\left[\tilde{\boldsymbol{\Sigma}}_i^{-1} - \tilde{\boldsymbol{\Sigma}}_i^{-1} \hat{\mathbf{S}} \tilde{\boldsymbol{\Sigma}}_i^{-1}\right]_{ll} = \mathbf{e}_l^\top \left(\tilde{\boldsymbol{\Sigma}}_i^{-1} - \tilde{\boldsymbol{\Sigma}}_i^{-1} \hat{\mathbf{S}} \tilde{\boldsymbol{\Sigma}}_i^{-1}\right) \mathbf{e}_l = 0,$$

and Equation (2.49),

$$\tilde{\boldsymbol{\Sigma}}_{il}^{-1} = \mathbf{C}_{il}^{-1} - \frac{\omega_{il}\mathbf{C}_{il}^{-1}\mathbf{e}_l\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}}{1+\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l}.$$

First, notice that

$$\begin{aligned}
\mathbf{e}_l^\top\tilde{\boldsymbol{\Sigma}}_i^{-1}\mathbf{e}_l &= \mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l - \frac{\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l}{1+\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l} \\
&= \mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l - \frac{\omega_{il}(\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)^2}{1+\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l} \\
&= \frac{\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l}{1+\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l} \\
&= \frac{(\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)^2+\omega_{il}(\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)^2}{(1+\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)^2}.
\end{aligned}$$

Similarly, expanding and simplifying shows that

$$\begin{aligned}
(1+\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)^2\left(\tilde{\boldsymbol{\Sigma}}_i^{-1}\hat{\mathbf{S}}\tilde{\boldsymbol{\Sigma}}_i^{-1}\right) &= (1+2\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l+\omega_{il}^2(\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)^2)\mathbf{C}_{il}^{-1}\hat{\mathbf{S}}\mathbf{C}_{il}^{-1} \\
&\quad - (\omega_{il}+\omega_{il}^2\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)\mathbf{C}_{il}^{-1}\hat{\mathbf{S}}\mathbf{C}_{il}^{-1}\mathbf{e}_l\mathbf{e}_l^\top\mathbf{C}_{il}^{-1} \\
&\quad - (\omega_{il}+\omega_{il}^2\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)\mathbf{C}_{il}^{-1}\mathbf{e}_l\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\hat{\mathbf{S}}\mathbf{C}_{il}^{-1} \\
&\quad + \omega_{il}^2\mathbf{C}_{il}^{-1}\mathbf{e}_l\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\hat{\mathbf{S}}\mathbf{C}_{il}^{-1}\mathbf{e}_l\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}.
\end{aligned}$$

After premultiplying both sides by $\mathbf{e}_l^\top$ and postmultiplying both sides by $\mathbf{e}_l$, all but the very first term cancels out, leaving

$$(1+\omega_{il}\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l)^2\left(\mathbf{e}_l^\top\tilde{\boldsymbol{\Sigma}}_i^{-1}\hat{\mathbf{S}}\tilde{\boldsymbol{\Sigma}}_i^{-1}\mathbf{e}_l\right) = \mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\hat{\mathbf{S}}\mathbf{C}_{il}^{-1}\mathbf{e}_l.$$

As a result,

$$\mathbf{e}_l^\top\tilde{\boldsymbol{\Sigma}}_i^{-1}\mathbf{e}_l - \mathbf{e}_l^\top\tilde{\boldsymbol{\Sigma}}_i^{-1}\hat{\mathbf{S}}\tilde{\boldsymbol{\Sigma}}_i^{-1}\mathbf{e}_l = \frac{\mathbf{e}_i^\top\mathbf{C}_{il}^{-1}\mathbf{e}_i+\omega_i\left(\mathbf{e}_i^\top\mathbf{C}_{il}^{-1}\mathbf{e}_i\right)^2-\mathbf{e}_i^\top\mathbf{C}_{il}^{-1}\tilde{\mathbf{S}}\mathbf{C}_{il}^{-1}\mathbf{e}_i}{\left(1+\omega_i\mathbf{e}_i^\top\mathbf{C}_{il}^{-1}\mathbf{e}_i\right)^2}.$$

Setting this expression equal to zero and rearranging for ω_i, we obtain

$$\omega_{il}^{(k+1)} = \frac{\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\tilde{\mathbf{S}}_i\mathbf{C}_{il}^{-1}\mathbf{e}_l-\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l}{\left(\mathbf{e}_l^\top\mathbf{C}_{il}^{-1}\mathbf{e}_l\right)^2},$$

as desired.

A.8 Joint Density of the MSMNFA Family With Discrete Scaling Variables

The following argument derives the expression in Equation (5.10), which we use to justify that the ECM algorithm for the MSMNFA model with continuous scaling variables also applies in the case where the scaling variables are discrete. We begin by breaking up the joint density into a product of conditional densities.

Then, because we now have a PMF for the scaling variables instead of a PDF, we introduce an additional 25indicator variable for which realisation of W_j has occurred. This allows us to condition on the outcome of W_j without introducing a sum over $m \in \mathcal{W}$, which we would have obtained by applying the Law of Total Probability.

$$
\begin{aligned}
f(\mathbf{x}; \boldsymbol{\Theta}) &= \prod_{j=1}^{n} f(\mathbf{x}_j; \boldsymbol{\Theta}) \\
&= \prod_{j=1}^{n} f(\mathbf{y}_j \mid \mathbf{Z}_j, W_j; \boldsymbol{\Theta}) f(w_j \mid \mathbf{Z}_j; \boldsymbol{\Theta}) f(\mathbf{z}_j; \boldsymbol{\Theta}) \\
&= \prod_{j=1}^{n} \prod_{i=1}^{g} \left\{ \pi_i f(\mathbf{y}_j \mid Z_{ij} = 1, W_j; \boldsymbol{\Theta}) f(w_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) \right\}^{z_{ij}} \\
&= \prod_{j=1}^{n} \prod_{i=1}^{g} \left\{ \pi_i \prod_{m \in \mathcal{W}} \left\{ f(\mathbf{y}_j \mid z_{ij} = 1, w_j = m; \boldsymbol{\Theta}) \times \right. \right. \\
&\qquad \left. \left. \Pr(W_j = m \mid Z_{ij} = 1; \boldsymbol{\Theta}) \right\}^{\mathbb{I}_{\{W_j = m\}}} \right\}^{z_{ij}} \\
&= \prod_{j=1}^{n} \prod_{i=1}^{g} \left\{ \pi_i \prod_{m \in \mathcal{W}} \left\{ \phi_p \left(\mathbf{y}_j; \boldsymbol{\mu}_i, \frac{1}{m} (\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i) \right) \times \right. \right. \\
&\qquad \left. \left. \Pr(W_j = m \mid Z_{ij} = 1; \boldsymbol{\Theta}) \right\}^{\mathbb{I}_{\{W_j = m\}}} \right\}^{z_{ij}}.
\end{aligned}
$$

Taking the natural logarithm of both sides gives the desired result, that is

$$
\begin{aligned}
\ell(\boldsymbol{\Theta} \mid \mathbf{x}) = &\sum_{j=1}^{n} \sum_{i=1}^{g} z_{ij} \log \pi_i \\
&+ \sum_{j=1}^{n} \sum_{i=1}^{g} \sum_{m \in \mathcal{W}} z_{ij} \mathbb{I}_{\{w_j = m\}} \log \left\{ \phi_p \left(\mathbf{y}_j; \boldsymbol{\mu}_i, \frac{1}{m} (\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i) \right) \times \right. \\
&\qquad \left. \Pr(W_j = m \mid Z_{ij} = 1) \right\}.
\end{aligned}
$$

A.9 Marginal Density for the MtFA Model

The marginal density of the MtFA model can be obtained by applying the following marginalisation argument.

$$
\begin{aligned}
f(\mathbf{y}_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) &= \int_0^\infty f(\mathbf{y}_j \mid W_j, Z_{ij} = 1; \boldsymbol{\Theta}) f(w_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) \, \mathrm{d}w_j \\
&= \int_0^\infty (2\pi)^{-p/2} |w_j^{-1}(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)|^{-1/2} \\
&\times \exp\left\{ -\frac{1}{2} w_j (\mathbf{y}_j - \boldsymbol{\mu}_i)^\top (\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1} (\mathbf{y}_j - \boldsymbol{\mu}_i) \right\} \\
&\times \frac{\left(\frac{\nu_i}{2}\right)^{\nu_i/2}}{\Gamma(\nu_i/2)} w_j^{\nu_i/2 - 1} \exp(-\nu_i w_j/2) \, \mathrm{d}w_j \\
&= \frac{\left(\frac{\nu_i}{2}\right)^{\nu_i/2}}{\Gamma(\nu_i/2)} \frac{1}{(2\pi)^{p/2} |(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)|^{1/2}} \\
&\times \int_0^\infty w_j^{\frac{p+\nu_i}{2} - 1} \exp\left\{ -\frac{\mathrm{d}_{ij} + \nu_i}{2} w_j \right\} \mathrm{d}w_j.
\end{aligned}
$$

After noticing that this is the integral of the kernel of a Gamma$((p+\nu_i)/2, (\delta_i(\mathbf{y}_i)+\nu_i)/2)$ random variable over its support, it is clear that

$$
\begin{aligned}
f(\mathbf{y}_j \mid Z_{ij} = 1; \boldsymbol{\Theta}) &= \frac{\left(\frac{\nu_i}{2}\right)^{\nu_i/2}}{\Gamma(\nu_i/2)} \frac{1}{(2\pi)^{p/2} |(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)|^{1/2}} \frac{\Gamma\left(\frac{p+\nu_i}{2}\right)}{\left(\frac{\mathrm{d}_{ij}+\nu_i}{2}\right)^{(p+\nu_i)/2}} \\
&= \frac{\Gamma\left(\frac{p+\nu_i}{2}\right)}{\Gamma\left(\frac{\nu_i}{2}\right) (\pi\nu_i)^{p/2} |(\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)|^{1/2}} \frac{1}{\left(1 + \frac{1}{\nu_i}\mathrm{d}_{ij}\right)^{(p+\nu_i)/2}},
\end{aligned}
$$

where $\mathrm{d}_{ij} = (\mathbf{y}_j - \boldsymbol{\mu}_i)^\top (\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i)^{-1} (\mathbf{y}_j - \boldsymbol{\mu}_i)$ is the Mahalanobis distance. This is the density function of the t_p distribution with parameters $\boldsymbol{\mu}_i$, $\mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i$ and ν_i. So the marginal distribution of $\mathbf{Y}_j \mid Z_{ij} = 1$ is

$$
\mathbf{Y}_j \mid Z_{ij} = 1 \sim t_p\left(\boldsymbol{\mu}_i, \mathbf{B}_i\mathbf{B}_i^\top + \mathbf{D}_i, \nu_i\right).
$$

A.10 Posterior Distribution of MtFA Scaling Variables

To derive the posterior distribution of the scaling variables for the MtFA model, consider the following Bayesian argument.

$$\begin{aligned} p(w_j \mid \mathbf{y}_j, Z_{ij} = 1) &\propto p(\mathbf{y}_j \mid W_j, Z_{ij} = 1) p(w_j \mid Z_{ij} = 1) \\ &\propto |w_j^{-1}(\mathbf{B}_i \mathbf{B}_i^\top + \mathbf{D}_i)|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} w_j \mathrm{d}_{ij}\right\} w_j^{\frac{\nu_i}{2}-1} \exp\left\{-\frac{\nu_i}{2} w_j\right\} \\ &\propto w_j^{(p+\nu_i)/2-1} \exp\left\{-\frac{1}{2}(\nu_i + \mathrm{d}_{ij}) w_j\right\}. \end{aligned}$$

Hence,

$$W_j \mid \mathbf{Y}_j, Z_{ij} = 1 \sim \text{Gamma}\left(\frac{\nu_i + p}{2}, \frac{\nu_i + \mathrm{d}_{ij}}{2}\right).$$

Appendix B

Systematic Comparison Figures

This appendix contains histograms of the inferred number of components and the inferred number of factors for each model and for each experiment group.

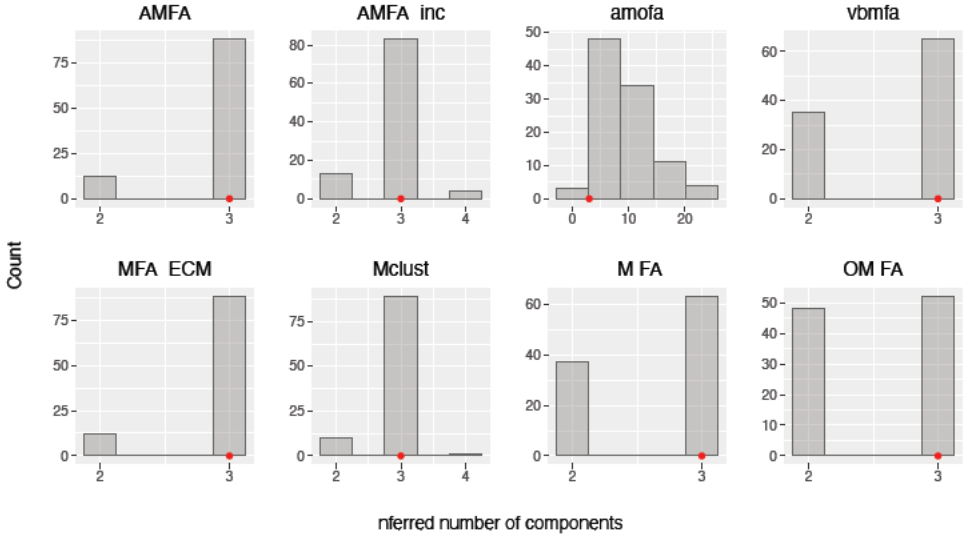


Figure B.1: Inferred numbers of components for experiment group one.

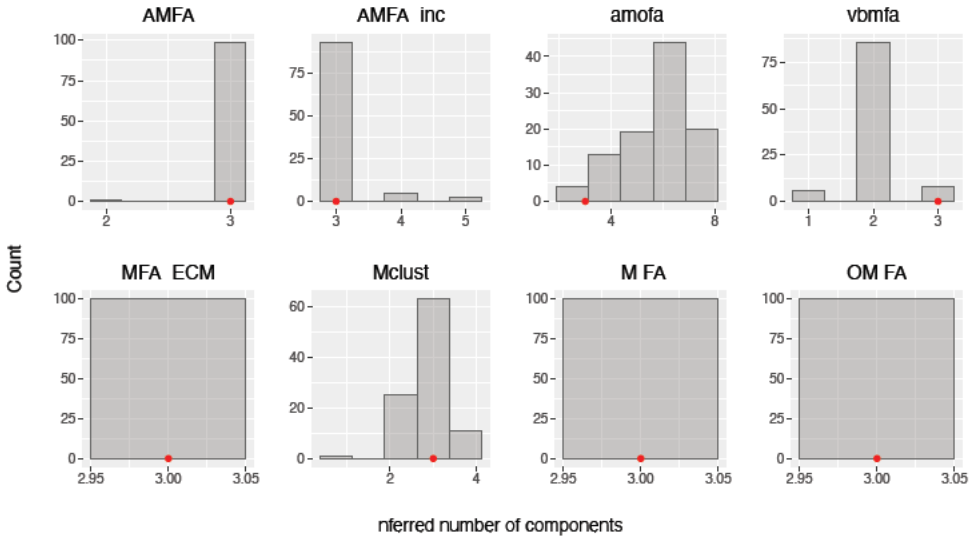


Figure B.2: Inferred numbers of components for experiment group two.

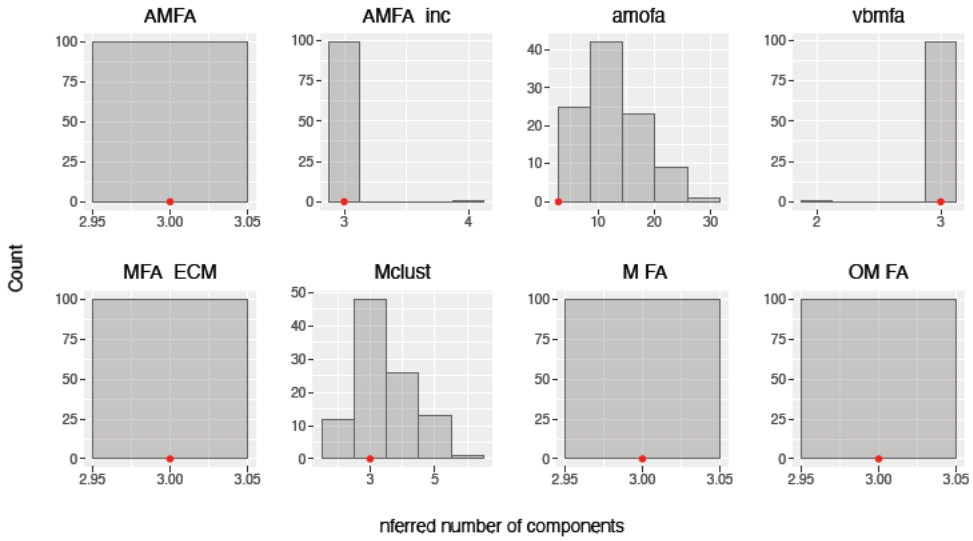


Figure B.3: Inferred numbers of components for experiment group three.

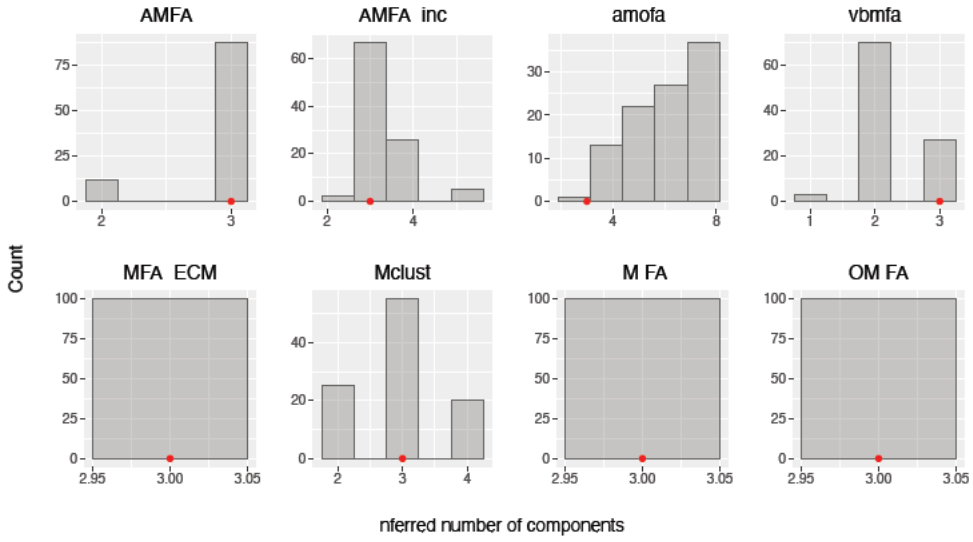


Figure B.4: Inferred numbers of components for experiment group four.

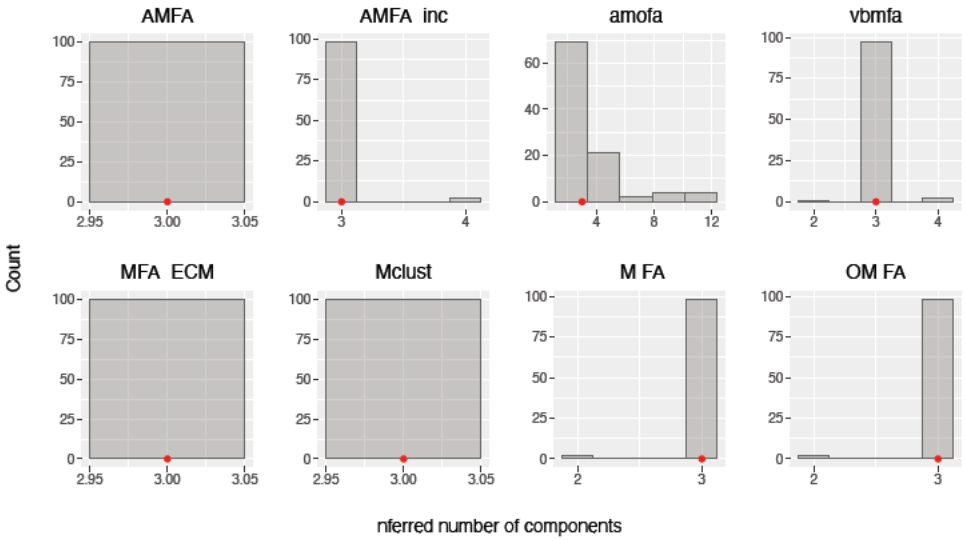


Figure B.5: Inferred numbers of components for experiment group five.

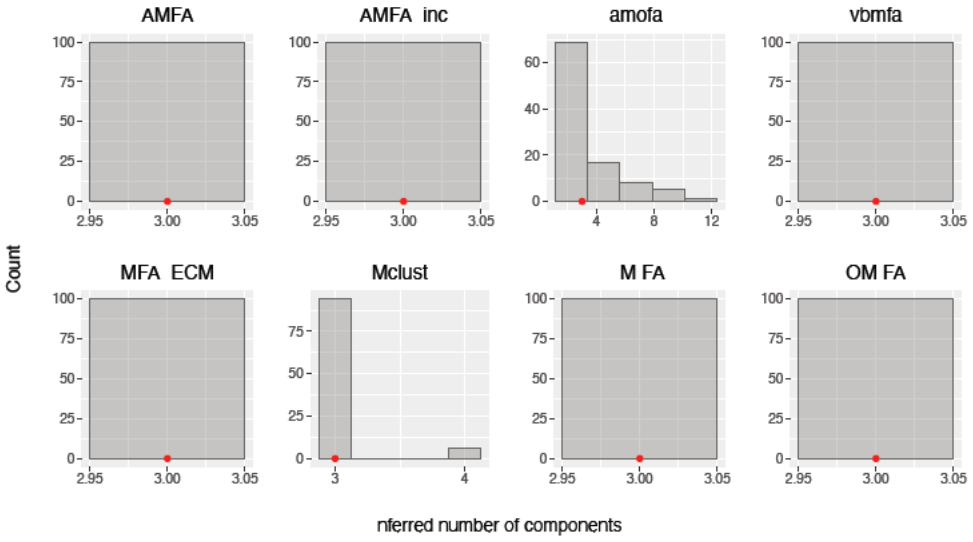


Figure B.6: Inferred numbers of components for experiment group six.

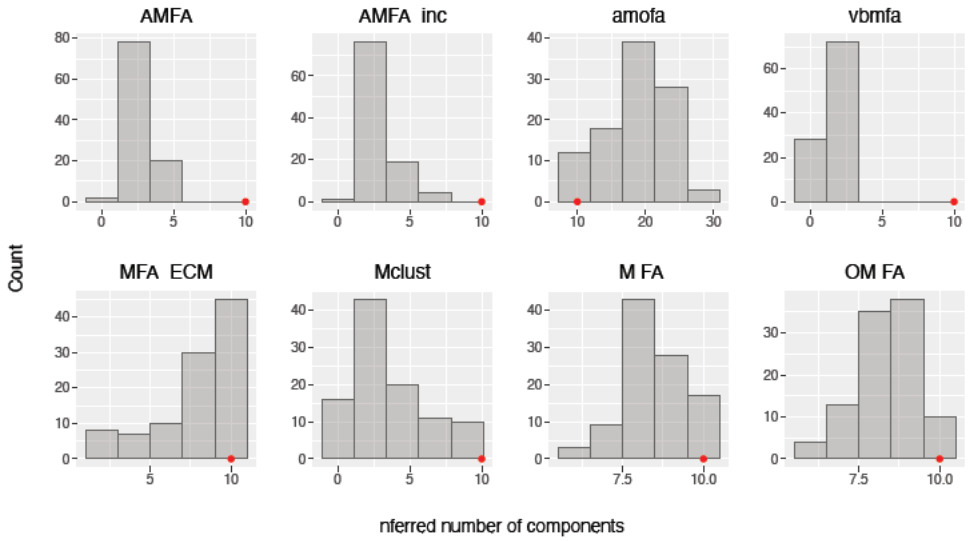


Figure B.7: Inferred numbers of components for experiment group seven.

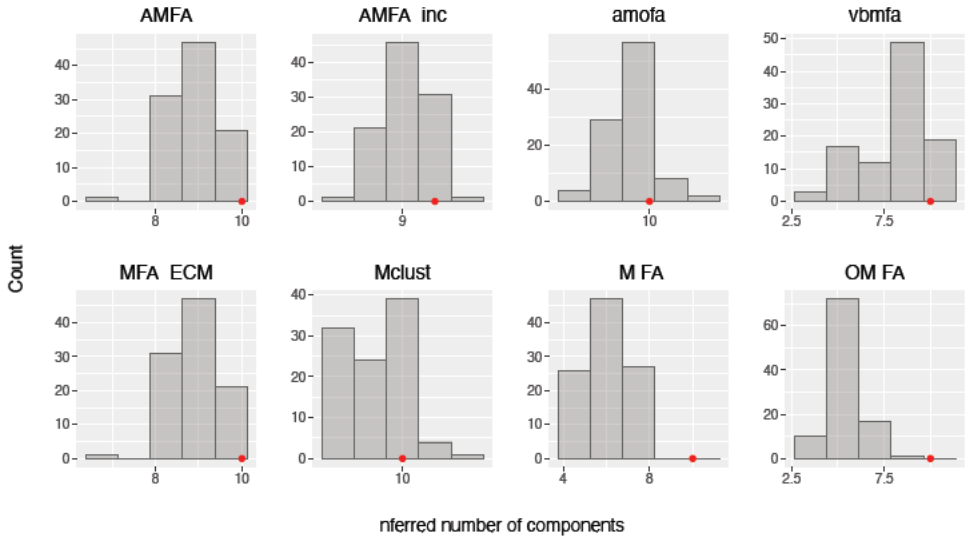


Figure B.8: Inferred numbers of components for experiment group eight.

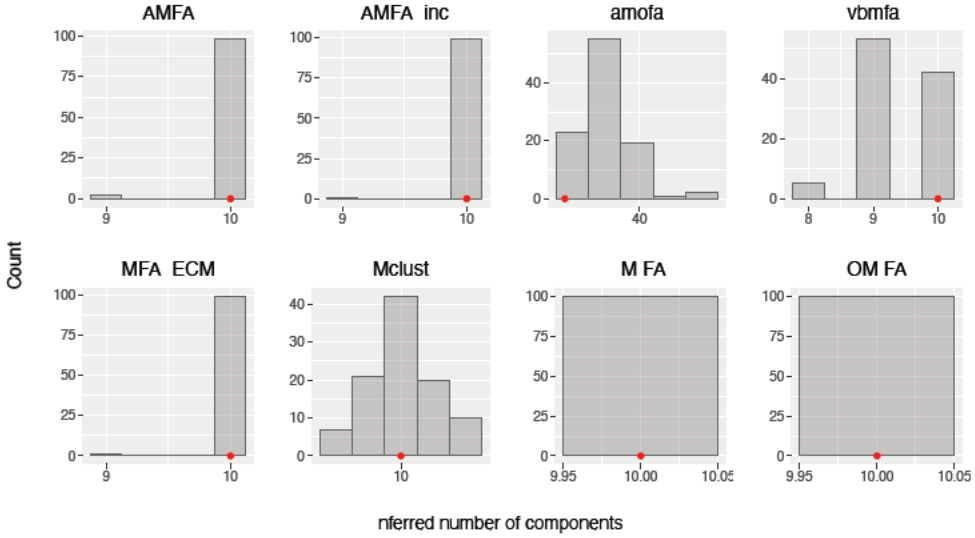


Figure B.9: Inferred numbers of components for experiment group nine.

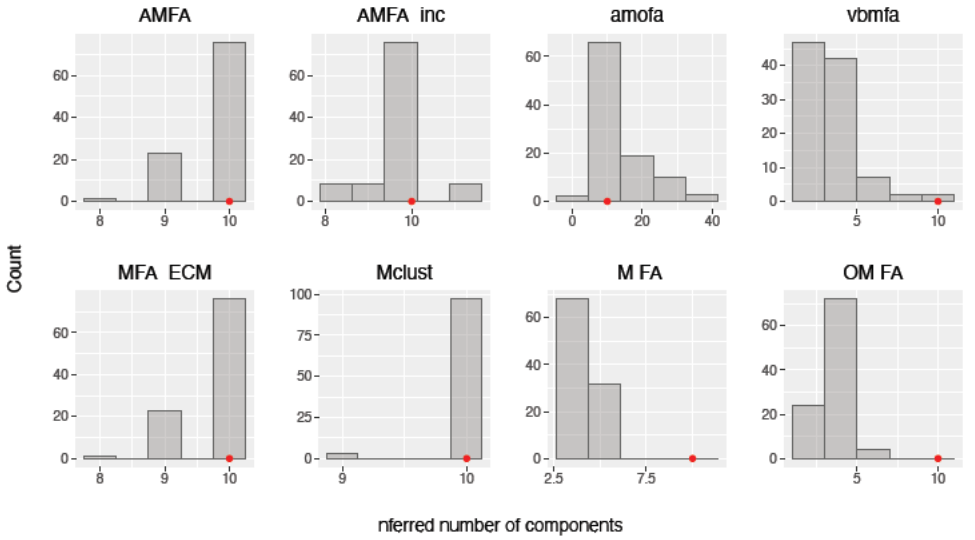


Figure B.10: Inferred numbers of components for experiment group ten.

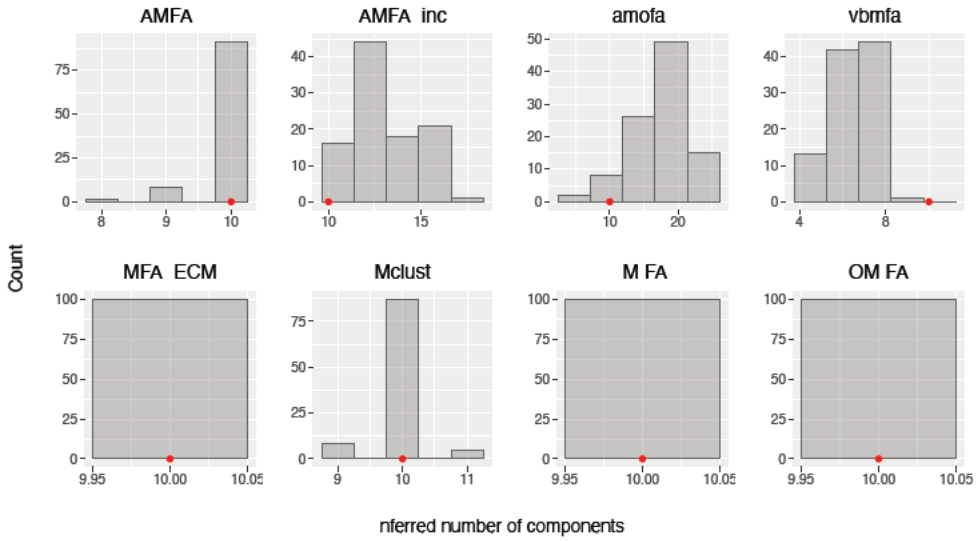


Figure B.11: Inferred numbers of components for experiment group eleven.

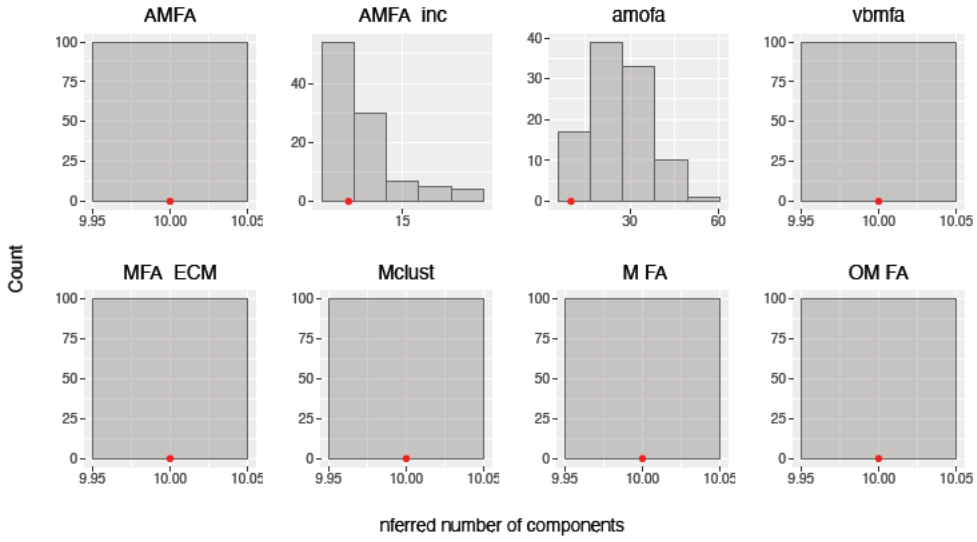


Figure B.12: Inferred numbers of components for experiment group twelve.

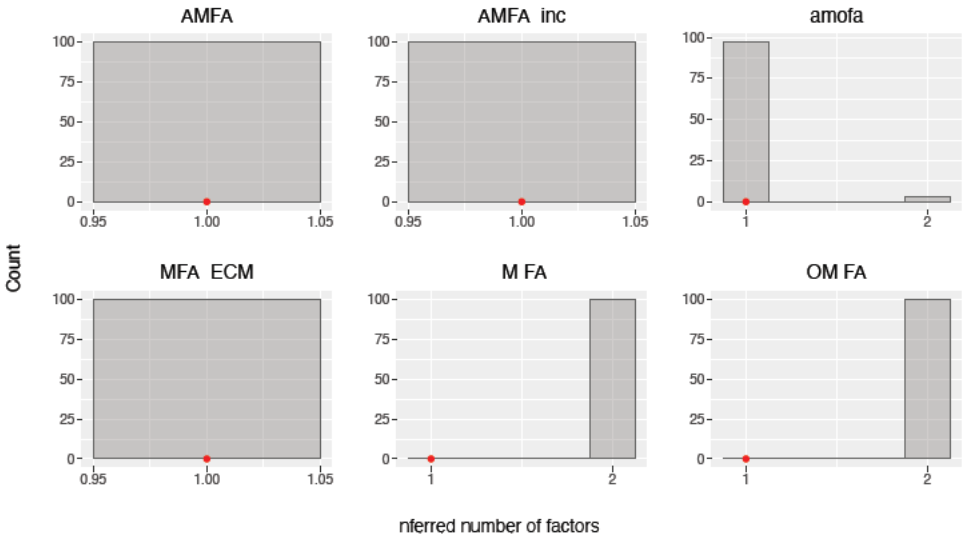


Figure B.13: Inferred numbers of factors for experiment group one.

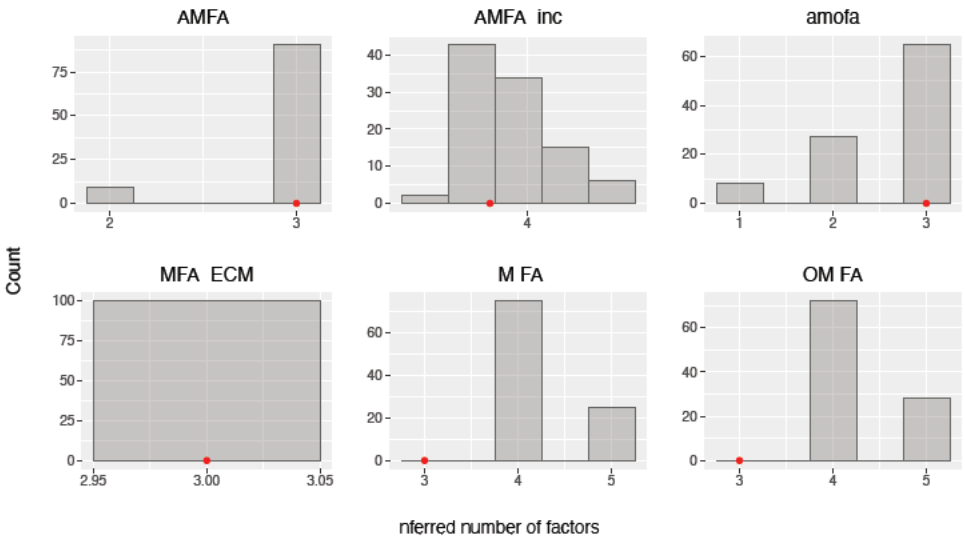


Figure B.14: Inferred numbers of factors for experiment group two.

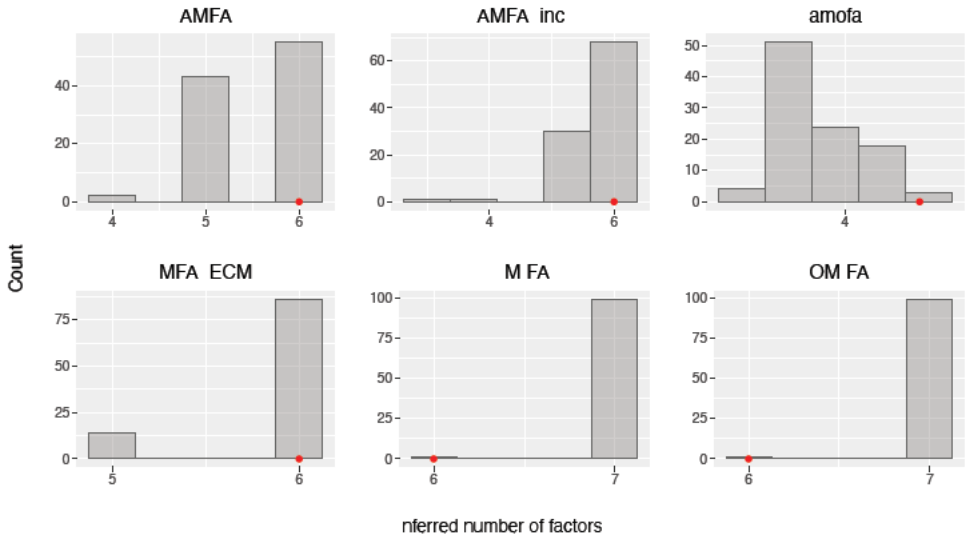


Figure B.15: Inferred numbers of factors for experiment group three.

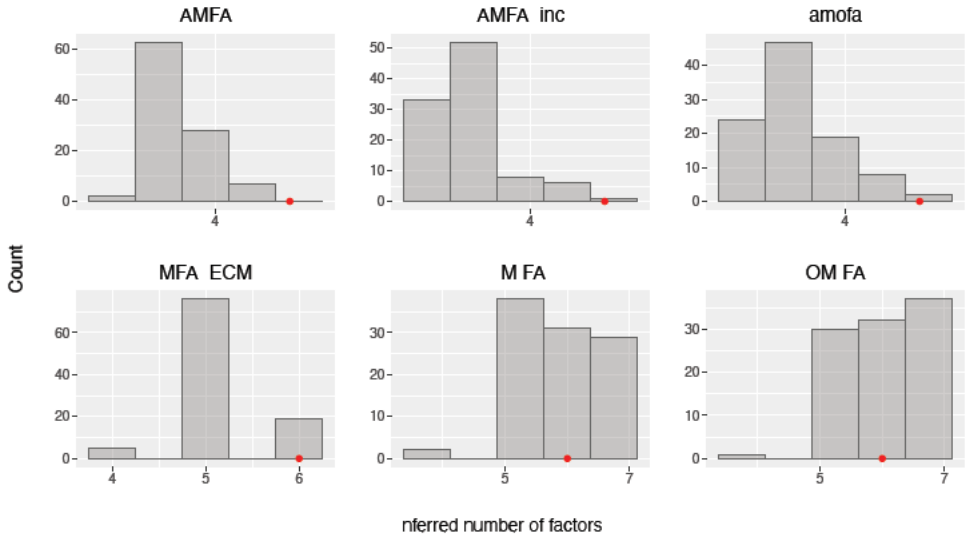


Figure B.16: Inferred numbers of factors for experiment group four.

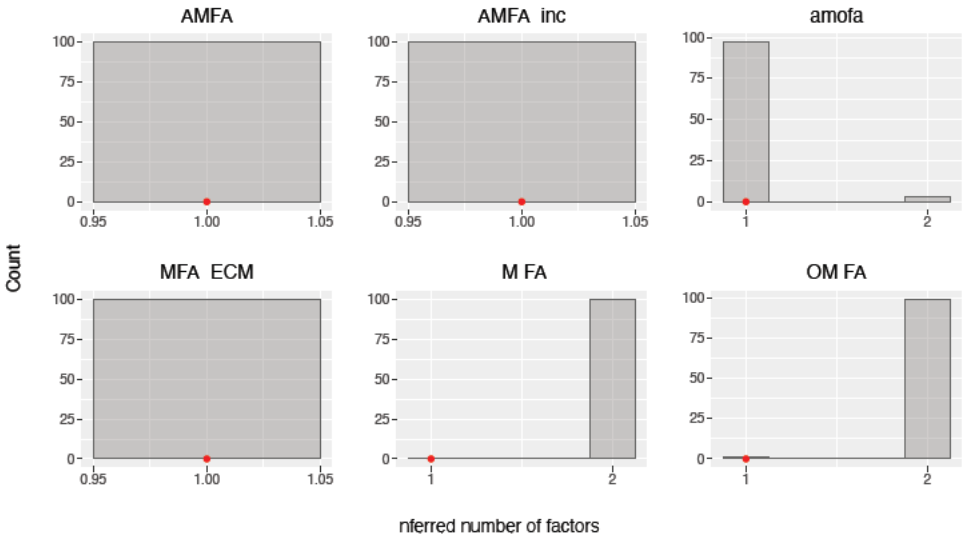


Figure B.17: Inferred numbers of factors for experiment group five.

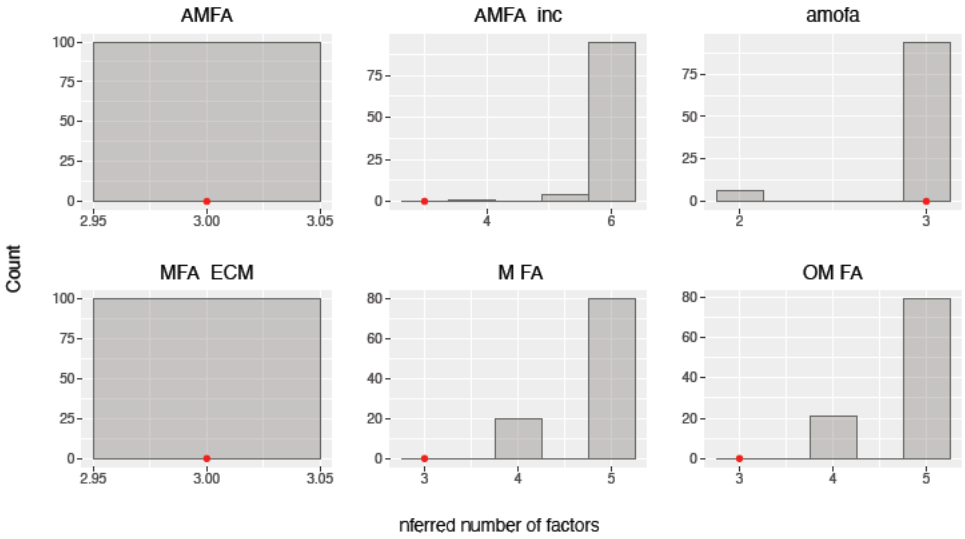


Figure B.18: Inferred numbers of factors for experiment group six.

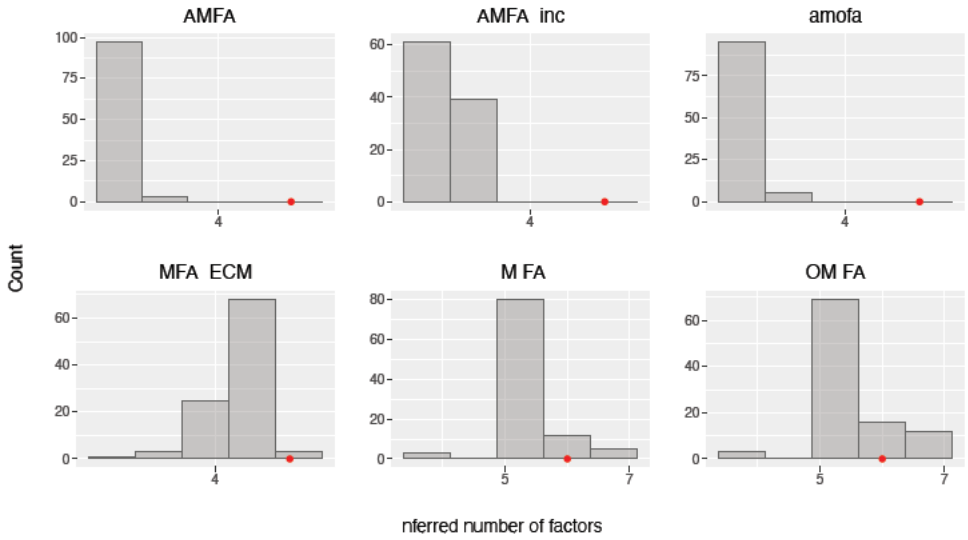


Figure B.19: Inferred numbers of factors for experiment group seven.

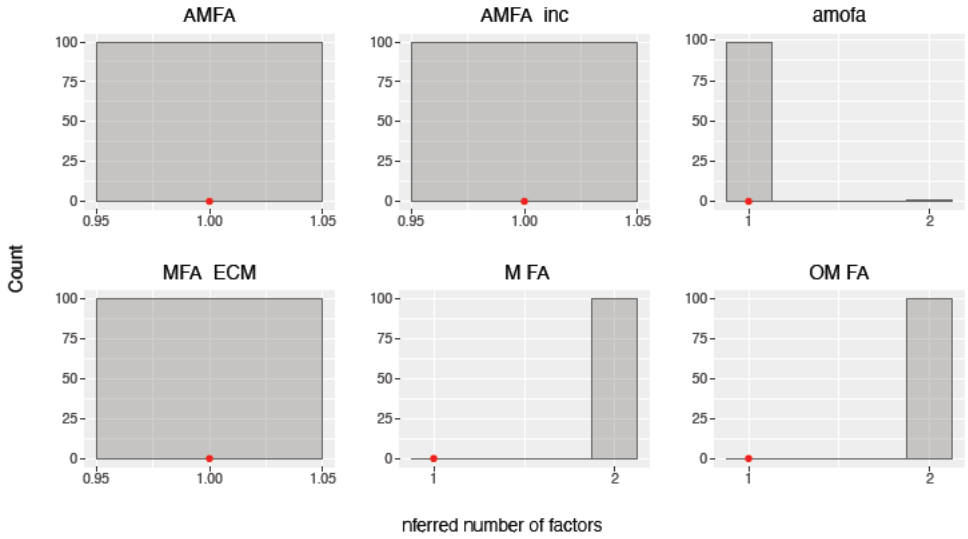


Figure B.20: Inferred numbers of factors for experiment group eight.

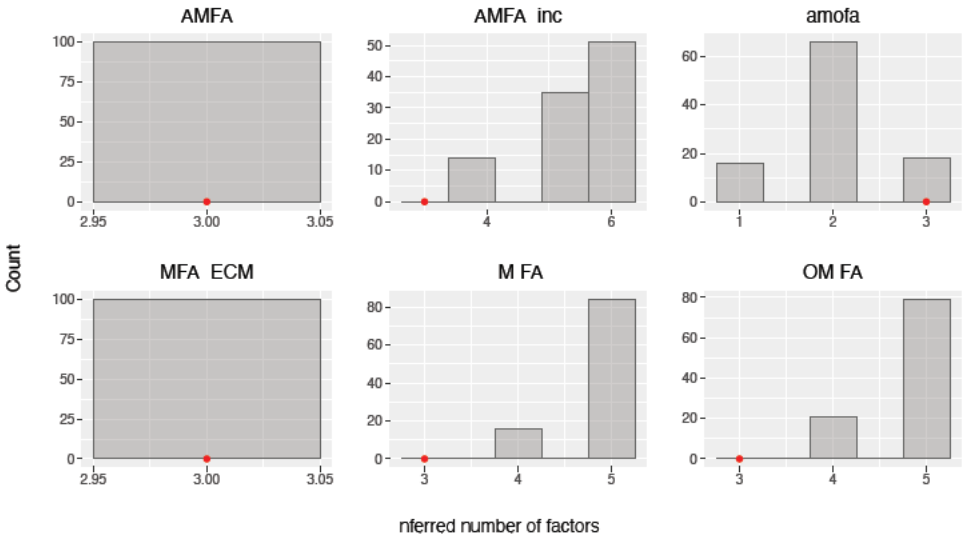


Figure B.21: Inferred numbers of factors for experiment group nine.

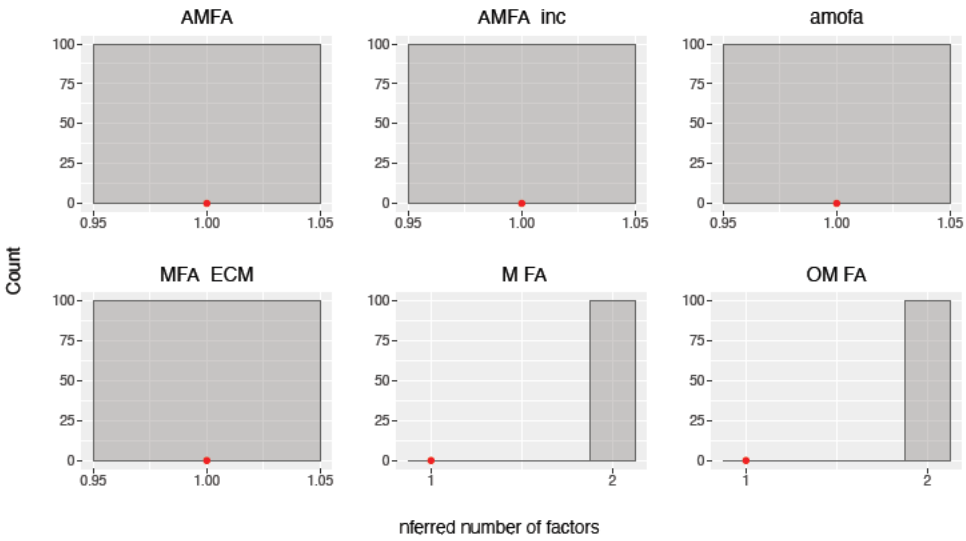


Figure B.22: Inferred numbers of factors for experiment group ten.

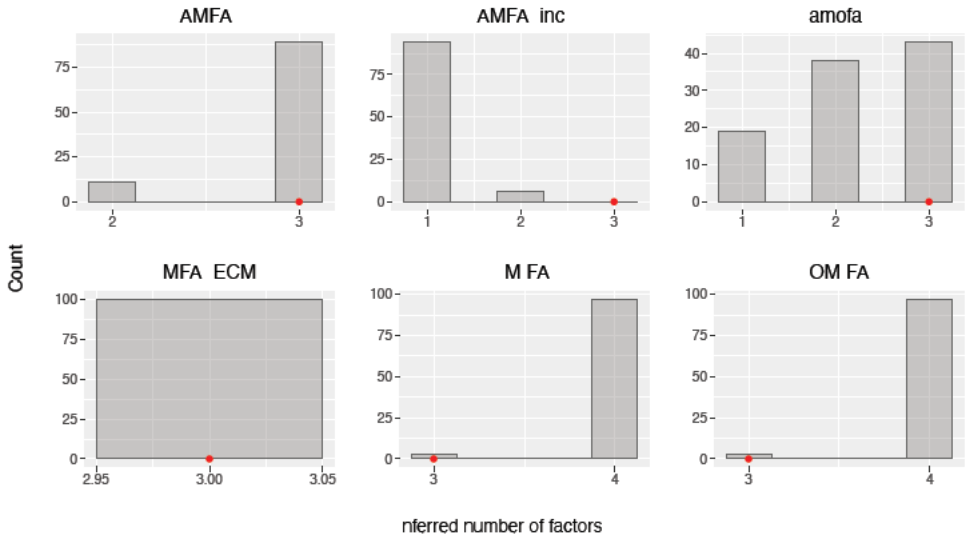


Figure B.23: Inferred numbers of factors for experiment group eleven.

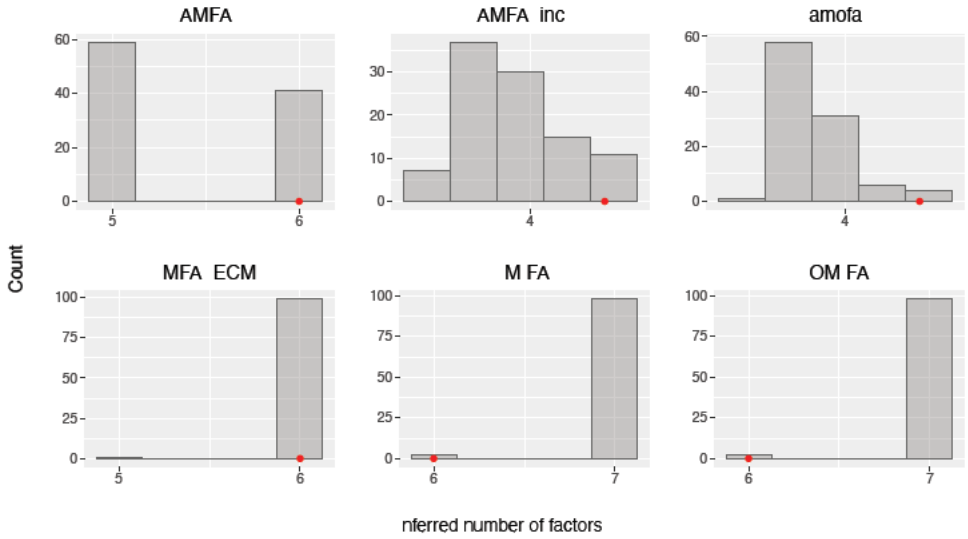


Figure B.24: Inferred numbers of factors for experiment group twelve.

Appendix C

R Code

C.1 Simple EM example

The following R code shows an implementation of the simple EM algorithm derived in Example 3.

```
#First we define a function to perform the E-steps for us
Estep <- function(Y, mu, s, pivec){
  #Simple function to calculate the E-step of the EM algorithm for
  #a univariate Gaussian Mixture model. Y is an nxp data matrix,
  #mu is a vector of component means, s is a vector of component
  #variances, pivec is a vector of mixing proportions. The length
  #of these vectors, g, is the number of components.
  g <- ncol(pivec)
  if (!is.matrix(Y))
    Y <- as.matrix(Y)
  p <- ncol(Y)
  n <- nrow(Y)
  Fji <- array(NA, c(n, g))

  for (i in 1:g) {
    Fji[, i] <- dnorm(Y,mean = mu[i], sd = sqrt(s[i]),
    log = TRUE)
  }

  Fji <- sweep(Fji, 2, log(pivec), "+")
  Fjmax <- apply(Fji, 1, max)
  Fji <- sweep(Fji, 1, Fjmax, "-")
  loglike <- sum(Fjmax, log(rowSums(exp(Fji))))
```

```
  Fji <- exp(Fji)
  tau <- sweep(Fji, 1, rowSums(Fji), "/")
  return(list(logL = loglike, tau = tau))
}
#Next we set a seed and generate some synthetic data
set.seed(123)
data <- c(rnorm(500,0,1),rnorm(1000,3,.4))
#We can plot a histogram of this data and compare it to the
#density function of the true underlying mixture model as well
#as the density of the model defined by the initial EM estimates
hist(data, breaks = 28, freq = FALSE, col="skyblue2",
  xlab = "Value", main = "")
curve(1/3*dnorm(x,0,sqrt(1))+2/3*dnorm(x,3,sqrt(.4^2)),
      add=T, lwd=3)
curve(1/2*dnorm(x,1,1)+1/2*dnorm(x,2,1),
      add=T, lwd=3, col = 'red')
#Next we initialise the containers for the EM variables
pi = matrix(c(0.5,0.5), nrow = 1)
mu = matrix(c(1,2), ncol = 1)
sigma = matrix(c(1,1),ncol = 1)
#Performing the first E-step and then the first M-step
#outside of loop
Y <- matrix(data,ncol = 1)
estep <- Estep(Y,mu,sigma,pi)
n <- nrow(Y)
pi <- matrix(1/n * colSums(estep$tau), nrow = 1)
for(i in 1:2){
  Ymu <- Y - matrix(rep(mu[i], n), nrow = n)
  sigma[i] <- 1/sum(estep$tau[,i])*
      sum(sweep(Ymu^2,1,estep$tau[,i],'*'))
  mu[i] <- 1/sum(estep$tau[,i])*
      sum(sweep(Y,1,estep$tau[,i],'*'))
}
#Plotting the densities after one EM iteration
hist(data, breaks = 28, freq = FALSE, col="skyblue2",
  xlab = "Value", main = "")
curve(1/3*dnorm(x,0,sqrt(1))+2/3*dnorm(x,3,sqrt(.4^2)),
      add=T,lwd=3)
curve(pi[1]*dnorm(x,mu[1],sqrt(sigma[1]))+
  pi[2]*dnorm(x,mu[2],sqrt(sigma[2])),
      add=T,lwd=3, col = 'red')
#Performing nine more EM steps in a loop
estep <- Estep(Y,mu,sigma,pi)
```

```
for(k in 2:10){
  pi <- matrix(1/n * colSums(estep$tau), nrow = 1)
  for(i in 1:2){
    Ymu <- Y - matrix(rep(mu[i], n), nrow = n)
    sigma[i] <- 1/sum(estep$tau[,i])*
        sum(sweep(Ymu^2,1,estep$tau[,i],'*'))
    mu[i] <- 1/sum(estep$tau[,i])*
        sum(sweep(Y,1,estep$tau[,i],'*'))
  }
  estep <- Estep(Y,mu,sigma,pi)
  #Plotting the results after each iteration
  hist(data, breaks = 28, freq = FALSE, col="skyblue2",
    xlab = "Value", main = "")
  curve(1/3*dnorm(x,0,sqrt(1))+2/3*dnorm(x,3,sqrt(.4^2)),
    add=T,lwd=3)
  curve(pi[1]*dnorm(x,mu[1],sqrt(sigma[1]))+
    pi[2]*dnorm(x,mu[2],sqrt (sigma[2])),
    add=T,lwd=3, col = 'red')
}
```

Bibliography

M. Aitkin and D. B. Rubin. Estimation and Hypothesis Testing in Finite Mixture Models. *Journal of the Royal Statistical Society. Series B, Methodological*, 47(1):67–75, 1985. ISSN 0035-9246. doi:10.1111/j.2517-6161.1985.tb01331.x.

J. Baek, G. J. McLachlan, and L. K. Flack. Mixtures of Factor Analyzers with Common Factor Loadings: Applications to the Clustering and Visualization of High-Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1298–1309, 2010. ISSN 0162-8828. doi:10.1109/TPAMI.2009.149.

O. Barndorff-Nielsen, J. Kent, and M. Sørensen. Normal Variance-Mean Mixtures and z Distributions. *International Statistical Review*, 50(2):145–159, 1982. ISSN 0306-7734. doi:10.2307/1402598.

J. Baudry, M. Cardoso, G. Celeux, M. J. Amorim, and A. S. Ferreira. Enhancing the Selection of a Model-Based Clustering with External Qualitative Variables. 2012. doi:0.1007/s11634-014-0177-3.

M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, University College London, Gower St, London, 6 2003. URL `https://cse.buffalo.edu/faculty/mbeal/thesis/`.

J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, 2017. doi:10.1137/141000671.

G. E. P. Box. *Statistics for Experimenters : Design, Innovation, and Discovery*. Wiley Series in Probability and Statistics. Wiley-Interscience, Hoboken, N.J, 2nd ed. edition, 2005. ISBN 0471718130.

R. P. Browne and P. D. McNicholas. A Mixture of Generalized Hyperbolic Distributions. *Canadian Journal of Statistics*, 43(2):176–198, Feb 2015. ISSN 0319-5724. doi:10.1002/cjs.11246.

G. Celeux and G. Govaert. Gaussian Parsimonious Clustering Models. *Pattern Recognition*, 28(5):781–793, 1995. ISSN 0031-3203.

M. Charytanowicz, J. Niewczas, P. Kulczycki, P. Kowalski, S. Lukasik, and S. Zak. A Complete Gradient Clustering Algorithm for Features Analysis of X-ray Images. In E. Pietka and J. Kawa, editors, *Information Technologies in Biomedicine*, pages 15–24. Springer-Verlag, Berlin-Heidelberg, 2010. doi:10.1007/978-3-642-13105-9_2.

C. S. Coffey and K. E. Muller. Properties of Doubly-Truncated Gamma Variables. *Communications in Statistics - Theory and Methods*, 29(4):851–857, 2000. doi:10.1080/03610920008832519.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. doi:10.1111/j.2517-6161.1977.tb01600.x.

Z. Ghahramani and M. J. Beal. Variational Inference for Bayesian Mixtures of Factor Analysers. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 449–455, Cambridge, MA, USA, 1999. MIT Press. URL http://mlg.eng.cam.ac.uk/zoubin/papers/nips99.pdf.

Z. Ghahramani, G. E. Hinton, et al. The EM Algorithm for Mixtures of Factor Analyzers. Technical report, 1997. URL http://www.cs.utoronto.ca/~hinton/absps/tr-96-1.pdf.

W. Härdle. *Smoothing Techniques With Implementation in S*. Springer Series in Statistics. Springer New York, New York, NY, 1st ed. 1991. edition, 1991. ISBN 1-4612-4432-3.

L. Hubert and P. Arabie. Comparing Partitions. *Journal of Classification*, 2:193–218, 1985. doi:10.1007/BF01908075.

K. Jöreskog. Some Contributions of Maximum Likelihood Factor Analysis. *Psychometrika*, 32:443–482, 1967. doi:10.1007/BF02289658.

B. Jorgensen. *Statistical Properties of the Generalized Inverse Gaussian Distribution*. Lecture Notes in Statistics, 9. Springer New York, New York, NY, 1st ed. 1982. edition, 1982. ISBN 1-4612-5698-4.

H. F. Kaiser. The Varimax Criterion for Analytic Rotation in Factor Analysis. *Psychometrika*, 23:187–200, 1958. doi:10.1007/BF02289233.

H. Kaya and A. Salah. Adaptive Mixtures of Factor Analyzers. 2015. URL https://arxiv.org/abs/1507.02801.

K. Ko and J. Baek. VaR Estimation Using Skewed Mixture Models and Various Mixtures of Factor Analyzers. *Journal of the Korean Data and Information Science Society*, 29 (3):769 – 778, 2018. URL http://www.kdiss.org/journal/view.html?spage=769&volume=29&number=3#body01.

K. Lange and J. S. Sinsheimer. Normal/Independent Distributions and their Applications in Robust Regression. *Journal of Computational and Graphical Statistics*, 2(2):175–198, 1993. ISSN 10618600. doi:10.1080/10618600.1993.10474606.

W. Ledermann. On the Rank of the Reduced Correlational Matrix in Multiple-Factor Analysis. *Psychometrika*, 2:85–93, 1937. doi:10.1007/BF02288062.

S. X. Lee and G. J. McLachlan. On Formulations of Skew Factor Models: Skew Factors and/or Skew Errors. *Statistics & Probability Letters*, 168:108935–, 2021. ISSN 0167-7152. doi:10.1016/j.spl.2020.108935.

C. Liu, D. B. Rubin, and Y. N. Wu. Parameter Expansion to Accelerate EM: The PX-EM Algorithm. *Biometrika*, 85(4):755–770, 1998. ISSN 00063444. doi:10.1093/biomet/85.4.755.

K. V. Mardia. Measures of Multivariate Skewness and Kurtosis with Applications. *Biometrika*, 57(3):519–530, 1970. ISSN 00063444. doi:10.1093/biomet/57.3.519.

G. McLachlan, R. Bean, and L. Ben-Tovim Jones. Extension of the Mixture of Factor Analyzers Model to Incorporate the Multivariate t-distribution. *Computational Statistics & Data Analysis*, 51(11):5327–5338, 2007. ISSN 0167-9473.

G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker Inc, New York, 1988.

G. J. McLachlan and D. Peel. Mixtures of Factor Analyzers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, Langley, P (Ed.), pages 599–606, San Francisco, 2000. Morgan Kaufmann. doi:10.1002/0471721182.ch8.

G. J. McLachlan, D. Peel, and R. Bean. Modelling High-Dimensional Data by Mixtures of Factor Analyzers. *Computational Statistics & Data Analysis*, 41:379–388, 01 2003. doi:10.1016/S0167-9473(02)00183-4.

G. J. McLachlan, S. X. Lee, and S. I. Rathnayake. Finite Mixture Models. *Annual Review of Statistics and Its Application*, 6(1):355–378, 2019. doi:10.1146/annurev-statistics-031017-100325.

X.-L. Meng and D. B. Rubin. Maximum Likelihood Estimation via the ECM Algorithm: A General Framework. *Biometrika*, 80(2):267–278, 1993. ISSN 00063444. doi:10.1093/biomet/80.2.267.

X.-L. Meng and D. van Dyk. The EM Algorithm–An Old Folk-Song Sung to a Fast New Tune. *Journal of the Royal Statistical Society. Series B (Methodological)*, 59(3): 511–567, 1997. ISSN 00359246. doi:10.1111/1467-9868.00082.

K. Murphy, C. Viroli, and I. C. Gormley. Infinite Mixtures of Infinite Factor Analysers. *Bayesian Analysis*, 15(3):937–963, 2020. doi:10.1214/19-BA1179.

K. Murphy, C. Viroli, and I. C. Gormley. `IMIFA`*: Infinite Mixtures of Infinite Factor Analysers and Related Models*, 2021. URL `https://cran.r-project.org/package=IMIFA`. R package version 2.1.8.

M. Naderi, A. Arabpour, and A. Jamalizadeh. Multivariate Normal Mean-Variance Mixture Distribution Based On Lindley Distribution. *Communications in Statistics - Simulation and Computation*, 47(4):1179–1192, 2018. doi:10.1080/03610918.2017.1307400.

P. Papastamoulis. Overfitting Bayesian Mixtures of Factor Analyzers with an Unknown Number of Components. *Computational Statistics and Data Analysis*, 124:220–234, 2018. doi:10.1016/j.csda.2018.03.007.

P. Papastamoulis. Clustering Multivariate Data Using Factor Analytic Bayesian Mixtures with an Unknown Number of Components. *Statistics and Computing*, 30:485–506, 2020. doi:10.1007/s11222-019-09891-z.

K. Pearson. III. Contributions to the Mathematical Theory of Evolution. *Philosophical Transactions of the Royal Society of London*, 185:71–110, December 1894. doi:10.1098/rsta.1894.0003.

K. B. Petersen and M. S. Pedersen. The Matrix Cookbook, November 2012. URL `http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html`. Version 20121115.

R. Pourmousa, A. Jamalizadeh, and M. Rezapour. Multivariate Normal Mean–Variance Mixture Distribution Based On Birnbaum–Saunders Distribution. *Journal of Statistical Computation and Simulation*, 85(13):2736–2749, 2015. doi:10.1080/00949655.2014.937435.

A. Punzo and P. D. McNicholas. Parsimonious Mixtures of Multivariate Contaminated Normal Distributions. *Biometrical Journal*, 58(6):1506–1537, 2016. ISSN 0323-3847.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

A. Raftery, M. Newton, J. Satagopan, and P. Krivitsky. Estimating the Integrated Likelihood via Posterior Simulation using the Harmonic Mean Identity. *Bayesian statistics*, 8:1–45, 01 2007. URL `https://pages.stat.wisc.edu/~newton/papers/publications/RafteryFinal.pdf`.

A. E. Raftery. Bayesian Model Selection in Social Research. *Sociological Methodology*, 25: 111–163, 1995. ISSN 0081-1750.

W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. doi:10.1080/01621459.1971.10482356.

G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests.* Studies in Mathematical Psychology. Danmarks Paedagogiske Institut, 1960. ISBN 9780598554512. URL https://books.google.com.au/books?id=aB9qLgEACAAJ.

D. Rizopoulos. ltm: An R Package for Latent Variable Modelling and Item Response Theory Analyses. *Journal of Statistical Software*, 17(5):1–25, 2006. doi:10.18637/jss.v017.i05.

A. Salah and E. Alpaydin. Incremental Mixtures of Factor Analysers. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 276–279 Vol.1, 2004. doi:10.1109/ICPR.2004.1334106.

G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978. ISSN 00905364. doi:10.1214/aos/1176344136.

L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1): 289–317, 2016. doi:10.32614/RJ-2016-021.

C. Spearman. "General Intelligence", Objectively Determined and Measured. *The American Journal of Psychology*, 15(2):201–292, 1904. ISSN 00029556. doi:10.2307/1412107.

R. D. Telford and R. B. Cunningham. Sex, Sport, and Body-size Dependency of Hematology in Highly Trained Athletes. *Medicine and Science in Sports and Exercise*, 23(7): 788–794, 1991. ISSN 0195-9131. doi:10.1249/00005768-199004000-00803.

MATLAB. *version 9.12.0.1884302 (R2022a).* The MathWorks Inc., Natick, Massachusetts, 2022.

L. Thurstone. Multiple Factor Analysis. *Psychological Review*, 5(38):406–427, 1931. doi:10.1037/h0069792.

C. Tortora, P. D. McNicholas, and R. P. Browne. A Mixture of Generalized Hyperbolic Factor Analyzers. *Advances in Data Analysis and Classification*, 10(4):423–440, Apr 2015. ISSN 1862-5355. doi:10.1007/s11634-015-0204-z.

J. Tukey. A survey of sampling from contaminated distributions. In I. Olkin et al., editors, *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, Stanford Studies in Mathematics and Statistics. Stanford University Press, 1960. ISBN 9780804705967.

C. S. Wallace and D. Boulton. An Information Measure for Classification. *Computer Journal*, 11:185–194, 1968. doi:10.1093/comjnl/11.2.185.

J. Wang and M. G. Genton. The Multivariate Skew-Slash Distribution. *Journal of Statistical Planning and Inference*, 136(1):209–220, 2006. ISSN 0378-3758. doi:10.1016/j.jspi.2004.06.023.

W.-L. Wang and T.-I. Lin. An Efficient ECM Algorithm for Maximum Likelihood Estimation in Mixtures of t-factor Analyzers. *Computational Statistics*, 28(2):751–769, 2012. ISSN 0943-4062.

W.-L. Wang and T.-I. Lin. Automated Learning of Mixtures of Factor Analysis Models with Missing Information. *TEST*, 29:1098–1124, 2020. doi:10.1007/s11749-020-00702-6.

M. Yang, N. Ahuja, and D. Kriegman. Face Detection using a Mixture of Factor Analyzers. In *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, volume 3, pages 612–616 vol.3, 1999. doi:10.1109/ICIP.1999.817188.

F. Zhang. *The Schur Complement and Its Applications*. Numerical Methods and Algorithms. Springer, 2005. ISBN 9780387242712. URL `https://books.google.com.au/books?id=Wjd8_AwjiIIC`.

J. Zhao and L. Shi. Automated Learning of Factor Analysis with Complete and Incomplete Data. *Computational Statistics & Data Analysis*, 72:205–218, 04 2014. doi:10.1016/j.csda.2013.11.008.

J. Zhao and P. Yu. Fast ML Estimation for the Mixture of Factor Analyzers via an ECM Algorithm. *IEEE Transactions on Neural Networks / a publication of the IEEE Neural Networks Council*, 19:1956–61, 12 2008. doi:10.1109/TNN.2008.2003467.

J. Zhao, P. Yu, and Q. Jiang. ML Estimation for Factor Analysis: EM or non-EM? *Statistics and Computing*, 18(2):109–123, 2007. ISSN 0960-3174. doi:10.1007/s11222-007-9042-y.