


The autoMFA Package for Automatically Fitting the Mixture of Factor Analyzers Model in R.

John Davey 
The University of Adelaide

Gary Glonek
The University of Adelaide

Sharon Lee
The University of Queensland

Abstract

This article introduces the **autoMFA** package for R, which includes five methods for automatically fitting the mixture of factor analyzers (MFA) model. Some of the methods in this package have existing MATLAB implementations, but this is the first implementation of them to be available on the Comprehensive R Archive Network (CRAN). Each method infers the number of components, g and the number of factors, q with as little input from the user as possible. The **autoMFA** package also provides diagnostic and clustering information for the methods implemented, including log-likelihood values and the predicted clustering.

Keywords: clustering, mixture models, factor analysis, R.

1. Introduction: The mixture of factor analyzers model

The mixture of factor analyzers (MFA) model is a multivariate statistical model which can simultaneously perform clustering and local dimension reduction, introduced by Ghahramani, Hinton *et al.* (1997), who also proposed a closed form algorithm for fitting the model. McLachlan and Peel (2000) provide further discussion of the model as well as an alternate fitting model fitting procedure. The original model has been applied to various problems, such as the clustering of cell lines on the basis of gene expressions from microarray experiments (McLachlan, Peel, and Bean 2003) and in image processing, where it has been used for face detection (Yang, Ahuja, and Kriegman 1999).

More recently, various extensions of the MFA model have been applied to the characterisation of multivariate air pollutant exposures (Maruotti, Bulla, Lagona, Picone, and Martella 2017), the estimation of value at risk in investment portfolios (Ko and Baek 2018) and the automated gating of mass cytometry data (Lee 2019).

The MFA model is a mixture model where each component of the mixture follows the well known factor analysis (FA) model. For a p -dimensional data vector Y_j , the MFA model is

$$Z_j \sim \text{Multinomial}(1, \pi)$$
$$Y_j \mid (Z_{ij} = 1) = \mu_i + B_i U_j + e_j,$$

independently for $i = 1, \dots, g$ and $j = 1, \dots, n$, where

$$U_j \sim \mathcal{N}_q(0, I) \text{ and } e_j \sim \mathcal{N}_p(0, D_i)$$

independently.

As the MFA model is a mixture model, each p -dimensional data point Y_j is associated with a g -dimensional indicator vector Z_j which identifies the component to which Y_j belongs. The proportion of data in component i is given by the mixing proportion π_i , where $0 \leq \pi_i \leq 1$ for $i = 1, \dots, g$ and $\sum_i \pi_i = 1$. Each data point is also associated with a q_i -dimensional vector U_j called the factors, where $q_i < p$ for $i = 1, \dots, g$.

Conditioned on data point Y_j belonging to component i , Y_j follows the FA model with $p \times 1$ mean vector μ_i , $p \times q$ factor loading matrix B_i and $p \times p$ diagonal error variance matrix D_i .

Under the MFA model

$$Y_j \mid (Z_{ij} = 1) \sim \mathcal{N}_p(\mu_i, B_i B_i^\top + D_i), \quad (1)$$

which shows that the MFA model is a Gaussian mixture model (GMM) where each component has a restricted covariance matrix.

To maintain identifiability, the number of factors, q_j , are required to obey the Ledermann bound (Ledermann 1937),

$$q_i \leq p + \frac{1 - \sqrt{1 + 8p}}{2}, \quad (2)$$

for all i . This bound ensures that the number of parameters required to fit the MFA model is less than the number required to fit a full-covariance GMM.

The model is often simplified by assuming a common number of factors, q , for all of the components in the mixture. This assumption is made in all of the methods included in **autoMFA**, apart from **amofa**. Another popular simplification is to assume a common diagonal matrix D for all of the components in the mixture. Of the algorithms implemented in **autoMFA**, only the **vbmfa** method makes this assumption.

The traditional method of estimation for the MFA model is by maximum likelihood estimation via the expectation maximisation (EM) algorithm (Dempster, Laird, and Rubin 1977). To fit this model, both the number of components, g and the number of factors, q are required. However, for a given dataset, it is not always obvious what values to these hyperparameters should take. This package provides several methods which automatically choose sensible values for g and q with as little input from the user as possible, and then fit the resulting MFA model.

Even when the Ledermann bound is satisfied for each i , the MFA model still suffers from an identifiability issue since the distribution of $Y_j \mid (z_{ij} = 1)$ only depends on the factor loading matrices B_i through the term $B_i B_i^\top$. As a result, if \tilde{B}_i is a maximum likelihood estimate for the factor loadings of component i , then for any $q_i \times q_i$ orthogonal matrix V , the log-likelihoods of \tilde{B}_i and $\tilde{B}_i V$ will be the same. To achieve identifiability, $\frac{1}{2}q_i(q_i + 1)$ constraints must be imposed on the estimated loading matrix \tilde{B}_i . One approach to applying these constraints is the varimax rotation (Kaiser 1958). Each method in **autoMFA** has the optional input **varimax**. While the default option is **FALSE**, if set to **TRUE** then each of the factor loading matrices in the fitted model will have been subject to the varimax rotation via the **varimax** function from the **stats** package.

Name	Infers g	Infers q
MFA_ECM	✗	✗
amofa	✓	✓
vbmfa	✓	✗
AMFA	✗	✓
AMFA_inc	✓	✓

Table 1: The different methods available in the **autoMFA** package and the parameters of the MFA model that they can automatically estimate. Note that an exhaustive search over a specified parameter space is not considered automatic inference. In the above, g represents the number of components in the MFA model, whereas q represents the number of factors.

The **autoMFA** (Davey 2021) package provides 5 different methods for automatically fitting the MFA model using R (R Core Team 2021). Users may also be interested in the R packages **fabMix** (Papastamoulis 2018) and **IMIFA** (Murphy, Viroli, and Gormley 2021), which both provide methods for automatically fitting the MFA model using Bayesian frameworks. The **EMMIXmfa** (Rathnayake, McLachlan, Peel, Baek, and R Core Team 2019) may also be of interest, which provides a method for fitting MFA models when g and q are known.

2. Available methods

For given values of g and q , the traditional way to fit the MFA model is using an EM-type algorithm. We highlight three such schemes here. The first EM-type algorithm for the MFA model to be proposed was an expectation conditional maximisation (ECM) algorithm (Meng and Rubin 1993) derived by Ghahramani *et al.* (1997). This algorithm treats both the indicator vectors Z_j and the factor vectors U_j as latent variables. Later, McLachlan and Peel (2000) proposed an alternating expectation conditional maximisation (AECM) algorithm (Meng and van Dyk 1997) which treats the indicator vectors Z_j alone as the latent variables for one cycle, and then treats both the indicator vectors Z_j and the factor vectors U_j as latent variables in the other cycle. With less latent variables used in the first cycle, by the general rate of convergence properties for the EM algorithm discussed in Dempster *et al.* (1977), the AECM should in general converge faster than the original ECM scheme. Most recently, Zhao and Yu (2008) proposed an ECM algorithm which only treats the indicator vectors Z_j as latent variables. They showed that by avoiding treating the factors U_j as latent variables altogether, their ECM scheme generally achieves a much higher rate of convergence than the two other algorithms.

There are currently five methods included in the **autoMFA** package for automatically fitting the MFA model. Table 1 summarises whether each algorithm can automatically estimate g and q . Here, we do not consider an exhaustive search over a specified parameter space as automatic inference. A short description of the available methods follows.

The MFA_ECM method performs a naive grid search over all values of g and q in a user specified range. Two initialisation schemes are used; random starts and k -means clustering. By specifying the number of random initialisations and the number of k -means initialisations, users can specify how many MFA models are fitted for each combination of g and q . The MFA models are fitted using the ECM algorithm for the MFA model as described in Zhao and Yu (2008). Users can choose between two convergence criteria; the absolute difference

in log-likelihood between the current iteration and the previous iteration or the ratio of the absolute difference in log-likelihood over the log-likelihood at the previous iteration. Users also specify a maximum number of ECM iterations to be used in fitting each model. The ECM iterations are terminated once the maximum number of iterations is reached, even if the convergence criterion has not yet been met. The best model is chosen according to the Bayesian information criterion (BIC) (Schwarz 1978).

The **amofa** method is an implementation of the adaptive mixtures of factor analyzers (AMoFA) algorithm described in Kaya and Salah (2015). The algorithm comprises two phases: incremental and decremental. In the former, it progressively adds a new component or adds a new factor to an existing component until a criterion based on the minimum message length (MML) is met. In the latter phase, the algorithm chooses to remove a component from the mixture using a criterion based on the posterior probabilities of each point belonging to each component, until only one component remains. The final model is the model which obtained the lowest MML value over both phases. The fitting of each candidate model is performed using a slightly modified version of the ECM algorithm for the MFA model described in Ghahramani *et al.* (1997). As mentioned earlier, this method does not assume that the number of factors is the same in each component.

The **vbmfa** method is an implementation of the variational bayesian mixtures of factor analyzers (VBMFA) algorithm given by Ghahramani and Beal (2000). It is an incremental algorithm, which starts with a single component and infers the number of components, g , by splitting existing components into two sub-components. As the name suggests, this method is based on a Bayesian MFA model, making it unique among the methods in this package, as the rest are all based on the EM-type algorithms. The authors recommend centering and scaling data before applying the VBMFA algorithm to improve the quality of the fitted models. Hence, the **preprocess** method is also included with the **autoMFA** package. This method centers and scales the data as suggested by the authors, so **vbmfa** should be run on the output of **preprocess**. It should be noted that while Beal (2003) suggests that this method will infer the number of factors by producing very small factor loadings in some columns of the factor loading matrices, we have been unable to reproduce this behaviour.

The **AMFA** method is an implementation of the automated mixtures of factor analyzers (AMFA) algorithm from Wang and Lin (2020). Similar to the **ECM_MFA** method, this method is also based on the ECM algorithm for the MFA model proposed by Zhao and Yu (2008). However, the **AMFA** method automatically infers q by treating it as a parameter in the ECM framework. This is achieved by using an approximation of the BIC to choose the best value of q among the set $\{1, \dots, q'\}$, where q' is the largest value of q satisfying the Ledermann constraint. The number of components, g is inferred using a naive search over a user-specified range.

Finally, the **AMFA_inc** method chooses q in the same way as the **AMFA** method, but employs an incremental approach to determine the number of components g . It starts with a single component model and then chooses to split a component into two sub-components using the same heuristic as **amofa**. This process continues until the algorithm has attempted to split all of the components in the mixture a specified number of times and no improvement to the BIC has been made.

3. Package usage

We now describe the inputs and outputs for each of the methods described in Section 2.

3.1. Inputs

The inputs of the methods in the **autoMFA** package are summarised as follows.

Inputs common to all methods

- **Y**; An $n \times p$ data matrix containing the data set that the model will be fitted to. Each row represents one data point.
- **varimax**; A boolean indicating whether or not the output factor loading matrices should be constrained using varimax rotation. Defaults to **FALSE**.

Inputs common to the AMFA, AMFAinc and MFAECM methods

- **eta**; The smallest possible entry in any of the error variance matrices. See [Zhao and Yu \(2008\)](#) for more information. The default value is $5e - 3$
- **nkmeans**; The number of times that k -means clustering will be used to initialise models for each combination of g and q . The default value is 5.
- **nrandom**; The number of randomly initialised models that will be used for each combination of g and q . The default value is 5.
- **tol**; The ECM algorithm terminates if the measure of convergence falls below this value. The default value is $1e - 5$.
- **conv_measure**; The convergence measure of the ECM algorithm. The default, **diff**, stops the ECM iterations if $|l^{(k+1)} - l^{(k)}| < \text{tol}$ where $l^{(k)}$ is the log-likelihood at the k^{th} ECM iteration. The alternative, **ratio**, measures the convergence of the ECM iterations using $|(l^{(k+1)} - l^{(k)})/l^{(k+1)}|$.

Inputs common to the AMFA, AMFAinc, MFAECM and amofa methods

- **itmax**; The maximum number of EM or ECM iterations allowed when fitting any MFA model. For **amofa** this defaults to 100, for the other methods the default is 500.

Inputs common to the AMFA and MFAECM methods

- **gmin**; The smallest number of components for which an MFA model will be fitted. The default value is 1.

Input	MFA_ECM	amofa	vbmfa	AMFA	AMFA_inc
Y	✓	✓	✓	✓	✓
varimax	✓	✓	✓	✓	✓
eta	✓	✗	✗	✓	✓
nkmeans	✓	✗	✗	✓	✓
nrandom	✓	✗	✗	✓	✓
tol	✓	✗	✗	✓	✓
conv_measure	✓	✗	✗	✓	✓
itmax	✓	✓	✗	✓	✓
gmin	✓	✗	✗	✓	✗
gmax	✓	✗	✗	✓	✗
qmin	✓	✗	✗	✗	✗
qmax	✓	✗	✗	✗	✗
verbose	✗	✓	✓	✗	✗
numTries	✗	✗	✓	✗	✓

Table 2: Summary of inputs for each method in **autoMFA**.

- **gmax**; The largest number of components for which an MFA model will be fitted. The default value is 10.

Inputs common to amofa and vbmfa methods

- **verbose**; A boolean variable controlling whether or not detailed output should be printed to the console during the fitting process. The default value is **FALSE**.

Inputs common to the AMFAinc and vbmfa methods

- **numTries**; The number of attempts that should be made to split each component. The default value is 2.

Inputs unique to the MFAECM method

- **qmin**; The smallest number of components for which an MFA model will be fitted. The default value is 1.
- **qmax**; The largest number of components for which an MFA model will be fitted. The default value is the largest possible q satisfying the Ledermann bound.

Table 2 summarises the inputs for each method in **autoMFA**.

3.2. Outputs

The output of models fitted using any of the five methods from the **autoMFA** package share the structure summarised in Table 3. The returned object is an instance of the **MFA** class,

which is a list with several elements. One of these is an object containing the estimates of the parameters of the MFA model, which is a list containing the mixing proportion vector π , the factor loading matrices B_i , error variance matrices D_i and mean vectors μ_i .

Another element of the output object is the clustering information of the fitted model, which includes the posterior probabilities of each point belonging to each component of the mixture model, and the clustering implied by these posterior probabilities.

In addition, there will be an element in the output object which contains diagnostic information specific to the fitting process of each algorithm, including the BIC and log-likelihood of the fitted model, as well as the 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: The output information common to all **autoMFA** models.

Table 3 summarises the structure of the output. For example, if our fitted model is called `MFAfit`, then the factor loading matrices can be retrieved using `MFAfit$model$B`. Similarly, the BIC can be retrieved with `MFAfit$diagnostics$bic`. All models in the **autoMFA** package will provide the information in the table above. However, given the issues discussed above with the `vbmfa` method not being able to infer the number of factors reliably, the user should be aware that its estimates of `numFactors` and `bic` will not be reliable.

4. Illustrations

The following example demonstrates how we can fit two MFA models, one using the **AMFA** method and the other using the **amofa** method.

The dataset we are using, `testDataMFA`, is included in **autoMFA**. It contains 720 observations of three dimensional data generated from an MFA model with three components and one factor for each component. The component means are $\mu_1 = (3, 0, 0)$, $\mu_2 = (0, 3, 0)$ and $\mu_3 = (0, 0, 3)$ and the mixing proportion vector is $\pi = (0.572, 0.3, 0.094)$.

```
R> RNGversion('4.0.3'); set.seed(1)
R> library(autoMFA)
R> MFA_fit_AMFA <- AMFA(testDataMFA, gmin=1, gmax=5)
R> MFA_fit_amofa <- amofa(testDataMFA)
```

In this case, we have accepted all of the default inputs for the **amofa** method. For the **AMFA** method, we have specified a search for g over all integers between one and five.

The output object `MFAfit` contains many useful pieces of information about the model and the fitting process, as described above. For example, we can obtain a summary of the model fitted using the AMFA method as follows.

```
R> summary(MFA_fit_AMFA)
```

```
AMFA(Y = testDataMFA, gmin = 1, gmax = 5)
  No.components  log_like      BIC
1                3 -1885.964 3962.726
Component specific numbers of factors:
 1 1 1
```

We can also inspect the model parameters in more detail by using the `print` method.

```
R> print(MFA_fit_AMFA)
```

```
AMFA(Y = testDataMFA, gmin = 1, gmax = 5)
The mixing proportions are:
      [,1]      [,2]      [,3]
[1,] 0.3328801 0.09444444 0.5726755
The component means are:
      [,1]      [,2]      [,3]
[1,] -0.017768833 0.0337185150 3.07125797
[2,] 2.978728640 0.0007733501 -0.02916128
[3,] -0.001069208 2.9905210245 -0.03371897
The factor loading matrices are:
, , 1

      [,1]
[1,] 0.21756621
[2,] 0.04934834
[3,] 0.04876003

, , 2

      [,1]
[1,] -0.1467706
[2,] -0.6783866
[3,] -0.4495082

, , 3

      [,1]
[1,] 0.91269177
[2,] -0.60770996
[3,] 0.02829478
```


The error variance matrices are:

, , 1

```

          [,1]      [,2]      [,3]
[1,] 0.04894896 0.0000000 0.00000000
[2,] 0.00000000 0.1098368 0.00000000
[3,] 0.00000000 0.0000000 0.09656364

```

, , 2

```

          [,1]      [,2]      [,3]
[1,] 0.1174174 0.0000000 0.00000000
[2,] 0.0000000 0.2109435 0.00000000
[3,] 0.0000000 0.0000000 0.07912817

```

, , 3

```

          [,1]      [,2]      [,3]
[1,] 0.2225416 0.0000000 0.00000000
[2,] 0.0000000 0.06377377 0.00000000
[3,] 0.0000000 0.0000000 0.1044417

```

From this output, we see that the model fitted using the **AMFA** method has correctly chosen a three component model. We can also see that the final model has a single factor for each component. However, the **AMFA** method strictly adheres to the Ledermann bound, which was $q = 1$ in this instance, so the only possible value for q that it considered was one. We can also see that the model has accurately inferred the underlying means of each component.

If we want to know how long the models took to fit, then the following commands

```
R> (MFA_fit_AMFA$diagnostics$totalTime)
```

```
[1] 54.57857
```

```
R> (MFA_fit_amofa$diagnostics$totalTime)
```

```
[1] 1.178681
```

tell us that it took approximately 54.58 seconds to fit the **AMFA** model and approximately 1.18 seconds to fit the **amofa** model. Similarly, we can obtain the BIC of each model using the following code,

```
R> (MFA_fit_AMFA$diagnostics$bic)
```

```
[1] 3962.726
```

```
R> (MFA_fit_amofa$diagnostics$bic)
```

```
[1] 3963.495
```

which was approximately 3962.73 for the model fitted using the AMFA method and 3963.49 for the model fitted using the amofa method. So, in this example, although the model fitted using the AMFA method took longer to run, it obtained a lower BIC than the model fitted using the amofa method.

Finally, it is often of interest to consider which component each of the data points has been assigned to. Each of the methods in **autoMFA** calculates the posterior probability that each data point belongs to each component in the mixture; the so-called responsibilities. Taking the maximum responsibility for each data point allows us to perform hard classifications, which are also included in the model output. We can access the responsibilities using the following code.

```
R> (head(MFA_fit_AMFA$clustering$responsibilities))
```

```
      [,1]      [,2] [,3]
[1,] 4.824245e-21 7.633595e-30 1
[2,] 1.093022e-37 1.732419e-37 1
[3,] 2.273377e-65 2.996617e-39 1
[4,] 3.397391e-15 5.122640e-25 1
[5,] 8.164713e-35 3.584905e-28 1
[6,] 8.822717e-58 8.662674e-31 1
```

```
R> (head(MFA_fit_amofa$clustering$responsibilities))
```

```
      [,1]      [,2]      [,3]
[1,] 1 9.480926e-30 3.448481e-20
[2,] 1 2.295826e-37 3.201432e-36
[3,] 1 3.766783e-39 5.601019e-63
[4,] 1 6.086420e-25 1.203096e-14
[5,] 1 4.428735e-28 1.369333e-33
[6,] 1 1.075697e-30 4.196938e-56
```

and the corresponding allocations using

```
R> (MFA_fit_AMFA$clustering$allocations[1:6])
```

```
[1] 3 3 3 3 3 3
```

```
R> (MFA_fit_amofa$clustering$allocations[1:6])
```

```
[1] 1 1 1 1 1 1
```

For this example, we observe that both models have assigned the first six data points to the same cluster, albeit with different group labels. In fact, the allocations made by the two models are exactly the same in this example. We can visualise the full set of allocations using

```
R> plot(MFA_fit_AMFA)
```

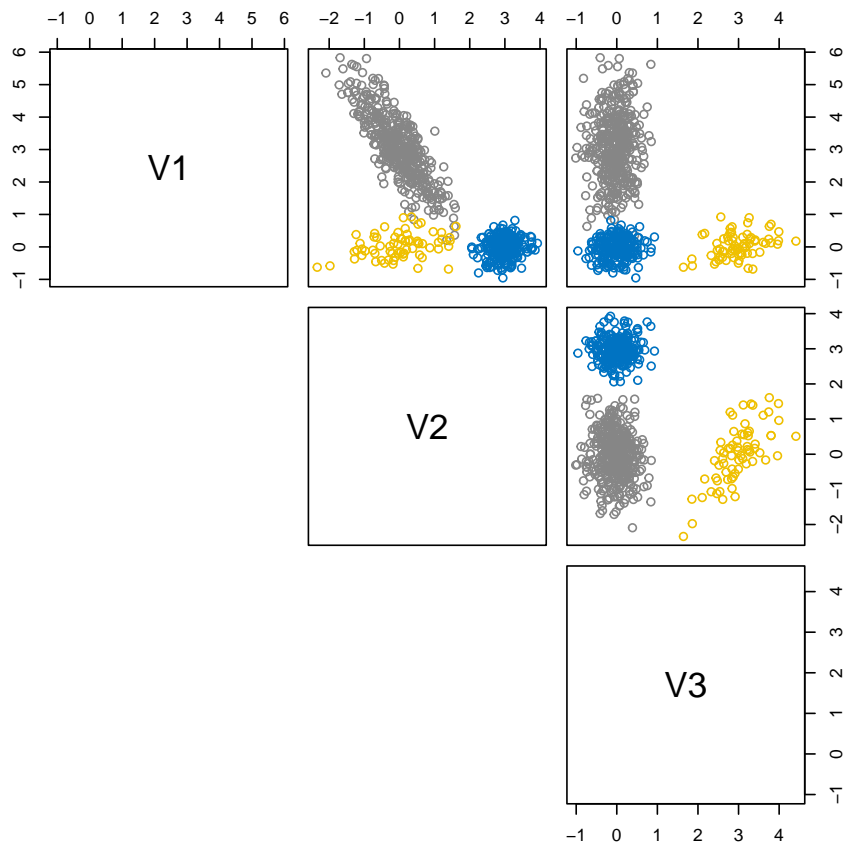


Figure 1: The clustering allocations obtained by the model fitted using the AMFA method, obtained by using the `plot` method.

the `print` method, as shown in Figure 1. We only include a plot for the model fitted using the **AMFA** method, since the model fitted using the `amofa` gives identical allocations.

5. Summary and discussion

This article introduced the **autoMFA** package for R, which includes five methods for automatically fitting MFA models. The five available methods are **AMFA**, **AMFA_inc**, **MFA_ECM**, **amofa** and **vbmfa**.

The MFA model includes two hyperparameters: g , the number of components and q , the number of factors in each component. Each of the methods in this package attempts to infer these hyperparameters with as little input from the user as possible.

In addition, the package also provides useful diagnostic and clustering information of the fitted models, such as the log-likelihood history of the final model, the responsibilities and the clustering allocations of the fitted model.

Computational details

The results in this paper were obtained using R 4.2.0 with the **autoMFA** 1.1.0 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/>.

References

- Beal MJ (2003). *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, University College London, The Gatsby Computational Neuroscience Unit, University College London, 17 Queen Square, London WC1N 3AR.
- Davey J (2021). **autoMFA**: *Algorithms for Automatically Fitting MFA Models*. R package version 1.0.0, URL <https://cran.r-project.org/package=autoMFA>.
- Dempster AP, Laird NM, Rubin DB (1977). “Maximum Likelihood from Incomplete Data via the EM Algorithm.” *Journal of the Royal Statistical Society. Series B (Methodological)*, **39**(1), 1–38. ISSN 00359246. URL <http://www.jstor.org/stable/2984875>.
- Ghahramani Z, Beal MJ (2000). “Variational Inference for Bayesian Mixtures of Factor Analysers.” In *Advances in Neural Information Processing Systems*, pp. 449–455.
- Ghahramani Z, Hinton GE, *et al.* (1997). “The EM Algorithm for Mixtures of Factor Analysers.” *Technical report*.
- Kaiser HF (1958). “The Varimax Criterion for Analytic Rotation in Factor Analysis.” *Psychometrika*, **23**, 187–200. doi:<https://doi.org/10.1007/BF02289233>.
- Kaya H, Salah AA (2015). “Adaptive Mixtures of Factor Analysers.” *arXiv Preprint arXiv:1507.02801*.

- Ko K, Baek J (2018). “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. URL <http://www.kdiss.org/journal/view.html?spage=769&volume=29&number=3#body01>.
- Ledermann W (1937). “On the Rank of the Reduced Correlational Matrix in Multiple-factor Analysis.” *Psychometrika*, **2**, 85–93. doi:<https://doi.org/10.1007/BF02288062>.
- Lee SX (2019). “CytoFA: Automated Gating of Mass Cytometry Data via Robust Skew Factor Analyzers.” In *Advances in Knowledge Discovery and Data Mining*, pp. 514–525. Springer International Publishing, Cham. ISBN 978-3-030-16148-4. doi:[10.1007/978-3-030-16148-4_40](https://doi.org/10.1007/978-3-030-16148-4_40).
- Maruotti A, Bulla J, Lagona F, Picone M, Martella F (2017). “Dynamic Mixtures of Factor Analyzers to Characterize Multivariate Air Pollutant Exposures.” *The Annals of Applied Statistics*, **11**(3), 1617 – 1648. doi:[10.1214/17-AOAS1049](https://doi.org/10.1214/17-AOAS1049).
- McLachlan G, Peel D (2000). “Mixtures of Factor Analyzers.” In *Proceedings of the Seventeenth International Conference on Machine Learning*, Langley, P (Ed.), pp. 599–606. Morgan Kaufmann, San Francisco.
- McLachlan G, Peel D, Bean R (2003). “Modelling High-Dimensional Data by Mixtures of Factor Analyzers.” *Computational Statistics & Data Analysis*, **41**, 379–388. doi:[10.1016/S0167-9473\(02\)00183-4](https://doi.org/10.1016/S0167-9473(02)00183-4).
- Meng X, Rubin D (1993). “Maximum Likelihood Estimation via the ECM Algorithm: A General Framework.” *Biometrika*, **80**(2), 267–278. ISSN 00063444. URL <http://www.jstor.org/stable/2337198>.
- Meng X, van Dyk D (1997). “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. ISSN 00359246. URL <http://www.jstor.org/stable/2346009>.
- Murphy K, Viroli C, Gormley IC (2021). *IMIFA: Infinite Mixtures of Infinite Factor Analyzers and Related Models*. R package version 2.1.8, URL <https://cran.r-project.org/package=IMIFA>.
- Papastamoulis P (2018). “Overfitting Bayesian Mixtures of Factor Analyzers with an Unknown Number of Components.” *Computational Statistics and Data Analysis*, **124**, 220–234. doi:[10.1016/j.csda.2018.03.007](https://doi.org/10.1016/j.csda.2018.03.007).
- R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Rathnayake S, McLachlan G, Peel D, Baek J, R Core Team (2019). *EMMIXmfa: Mixture Models with Component-Wise Factor Analyzers*. R package version 2.0.11, URL <https://CRAN.R-project.org/package=EMMIXmfa>.
- Schwarz G (1978). “Estimating the Dimension of a Model.” *The Annals of Statistics*, **6**(2), 461–464. ISSN 00905364. URL <http://www.jstor.org/stable/2958889>.

- Wang WL, Lin T (2020). “Automated Learning of Mixtures of Factor Analysis Models with Missing Information.” *TEST*, **29**. doi:[10.1007/s11749-020-00702-6](https://doi.org/10.1007/s11749-020-00702-6).
- Yang M, Ahuja N, Kriegman D (1999). “Face Detection using a Mixture of Factor Analyzers.” In *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, volume 3, pp. 612–616 vol.3. doi:[10.1109/ICIP.1999.817188](https://doi.org/10.1109/ICIP.1999.817188).
- Zhao JH, Yu P (2008). “Fast ML Estimation for the Mixture of Factor Analyzers via an ECM Algorithm.” *IEEE Transactions on Neural Networks*, **19**(11), 1956–1961. ISSN 1045-9227. doi:<https://doi.org/10.1109/TNN.2008.2003467>.

Affiliation:

John Davey
School of Mathematical Sciences
Faculty of Sciences, Engineering and Technology
University of Adelaide
Adelaide, Australia
E-mail: j.davey@adelaide.edu.au