Generative Clustering with missing values using the rtkpp package

Serge Iovleff
University Lille 1

Abstract

The Clustering project is a part of the **stkpp** library (Iovleff 2012) that can be accessed from R (R Development Core Team 2013) using the **rtkpp** package. It is possible to cluster gaussian, gamma, categorical and Poisson mixture models or a combination of these models in case of heterogeneous data.

Keywords: R, C++, Clustering, missing values.

1. Introduction

The clustering project implemented in STK++ allows to perform clustering on various data set using generative models. There is four kinds of generative models implemented:

1. the diagonal Gaussian mixture models (8 models),
2. the diagonal gamma mixture models (24 models),
3. the diagonal categorical mixture models (8 models),
4. the diagonal Poisson mixture models (4 models).

These models and the estimation algorithms allow to handle missing values. It is thus possible to use these models in order to cluster, but also complete data set with missing values.

The **rtkpp** package provide an access in (R Development Core Team 2013) to the **STK++** (Iovleff 2012) C++ part of the library dedicated to clustering.

In this paper we will first present the mixture models, the estimation algorithm, the initialization method and the strategy that can be used in order to find a good estimate of the parameters of the mixture model. In a second step we present shortly the different mixture models implemented in STK++ that can be used form rtkpp. Finally we give some examples on various data set.

2. Mixture Modeling

Let $\mathcal{X}$ be an arbitrary measurable space and let $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ be n independent vectors in

$\mathcal{X}^d$ such that each $\mathbf{x}_i$ arises from a probability distribution with density

$$f(\mathbf{x}_i|\theta) = \sum_{k=1}^{K} p_k h(\mathbf{x}_i|\boldsymbol{\lambda}_k, \boldsymbol{\alpha}) \tag{1}$$

where the p_k's are the mixing proportions ($0 < p_k < 1$ for all $k = 1, ..., K$ and $p_1 + ... + p_K = 1$), $h(\cdot|\boldsymbol{\lambda}_k, \boldsymbol{\alpha})$ denotes a d-dimensional distribution parameterized by $\boldsymbol{\lambda}_k$ and $\boldsymbol{\alpha}$. The parameters $\boldsymbol{\alpha}$ do not depend from k and are common to all the components of the mixture. The vector parameter to be estimated is $\theta = (p_1, \ldots, p_K, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_K, \boldsymbol{\alpha})$ and is chosen to maximize the observed log-likelihood

$$L(\theta|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} \ln \left(\sum_{k=1}^{K} p_k h(\mathbf{x}_i|\boldsymbol{\lambda}_k, \boldsymbol{\alpha}) \right). \tag{2}$$

In case there is missing data, that is some $\mathbf{x}_i$ are splited in observed values $\mathbf{x}_i^o$ and missing values $\mathbf{x}_i^m$, the log-likelihood to maximize should be the integrated log-likelihood

$$L(\theta|\mathbf{x}_1^o, \ldots, \mathbf{x}_n^o) = \sum_{i=1}^{n} \int \ln \left(\sum_{k=1}^{K} p_k h(\mathbf{x}_i^o, \mathbf{x}_i^m|\boldsymbol{\lambda}_k, \boldsymbol{\alpha}) \right) d\mathbf{x}_i^m. \tag{3}$$

In the package **rtkpp**, this quantity is approximated using a monte-carlo estimator by the `SEM` or the `SemiSEM` algorithms and by a biased estimator by the `EM` or the `SemiSEM` algorithms.

It is well known that for a mixture distribution, a sample of indicator vectors or *labels* $\mathbf{z} = \{\mathbf{z}_1, ..., \mathbf{z}_n\}$, with $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})$, $z_{ik} = 1$ or 0, according to the fact that $\mathbf{x}_i$ is arising from the kth mixture component or not, is associated to the observed data $\mathbf{x}$. The sample $\mathbf{z}$ is *unknown* so that the maximum likelihood estimation of mixture models is traditionally performed via the `EM` algorithm Dempster *et al.* (1997) or by a stochastic version of `EM` called SEM (see Mclachlan and Peel (2000)), or by a k-means like algorithm called CEM. In the **rtkpp** package it is also possible to use an algorithm called `SemiSEM` which is an intermediate between the `EM` and `SEM` algorithm. In case there is no missing values, `SemiSEM` and `EM` are equivalents.

2.1. Estimation algorithms

EM algorithm

Starting from an initial arbitrary parameter θ^0, the mth iteration of the `EM` algorithm consists of repeating the following E (I if there exists missing values) and M steps.

- **I step:** The missing values $\mathbf{x}_i^m$ are imputed using the current MAP value given by the current value θ^{m-1} of the parameter.
- **E step:** The current conditional probabilities that $z_{ik} = 1$ for $i = 1, \ldots, n$ and $k = 1, \ldots, K$ are computed using the current value θ^{m-1} of the parameter:

$$t_{ik}^m = t_k^m(\mathbf{x}_i|\theta^{m-1}) = \frac{p_k^{m-1} h(\mathbf{x}_i|\boldsymbol{\lambda}_k^{m-1}, \boldsymbol{\alpha}^{m-1})}{\sum_{l=1}^{K} p_l^{m-1} h(\mathbf{x}_i|\boldsymbol{\lambda}_l^{m-1}, \boldsymbol{\alpha}^{m-1})}. \tag{4}$$

- **M step:** The maximum likelihood estimate θ^m of θ is updated using the conditional probabilities t_{ik}^m as conditional mixing weights. It leads to maximize

$$L(\theta|\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{t}^m) = \sum_{i=1}^{n}\sum_{k=1}^{K} t_{ik}^m \ln\left[p_k h(\mathbf{x}_i|\boldsymbol{\lambda}_k,\boldsymbol{\alpha})\right], \tag{5}$$

  where $\mathbf{t}^m = (t_{ik}^m, i = 1,\ldots,n, k = 1,\ldots,K)$. Updated expression of mixture proportions are, for $k = 1,\ldots,K$,

$$p_k^m = \frac{\sum_{i=1}^{n} t_{ik}^m}{n}. \tag{6}$$

  Detailed formula for the updating of the $\boldsymbol{\lambda}_k$'s and $\boldsymbol{\alpha}$ are depending of the component parameterization and are detailed in the section 3.

The `EM` algorithm may converge to a local maximum of the observed data likelihood function, depending on starting values.

SEM algorithm

The `SEM` algorithm is a stochastic version of `EM` incorporating between the E and M steps a restoration of the unknown component labels $\mathbf{z}_i$, $i = 1,\ldots,n$, by drawing them at random from their current conditional distribution. Starting from an initial parameter θ^0, an iteration of `SEM` consists of three steps.

- **I step:** The missing values are simulated using the current value θ^{m-1} of the parameter.
- **E step:** The conditional probabilities t_{ik}^m $(1 \leq i \leq n, 1 \leq k \leq K)$ are computed for the current value of θ^{m-1} as in the E step of `EM` algorithm (equation 4).
- **S step:** Generate labels $\mathbf{z}^m = \{\mathbf{z}_1^m, ..., \mathbf{z}_n^m\}$ by assigning each point $\mathbf{x}_i$ at random to one of the mixture components according to the categorical distribution with parameter $(t_{ik}^m, 1 \leq k \leq K)$.
- **M step:** The maximum likelihood estimate of θ is updated using the generated labels by maximizing

$$L(\theta|\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{t}^m) = \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}^m \ln\left[p_k h(\mathbf{x}_i|\boldsymbol{\lambda}_k,\boldsymbol{\alpha})\right], \tag{7}$$

SEM does not converge point wise. It generates a Markov chain whose stationary distribution is more or less concentrated around the m.l. parameter estimator. A natural parameter estimate from a `SEM` sequence $(\theta^r)_{r=1,\ldots,R}$ is the mean $\sum_{r=b+1}^{R} \theta^r/(R-b)$ of the iterates values where the first b burn-in iterates have been discarded when computing this mean. An alternative estimate is to consider the parameter value leading to the highest likelihood in a `SEM` sequence.

SemiSEM algorithm

The `SemiSEM` algorithm is a stochastic version of `EM` incorporating a restoration of the missing values $\mathbf{x}_i^m$, $i = 1,\ldots,n$ by drawing them at random from their current conditional distribution. Starting from an initial parameter θ^0, an iteration of `SemiSEM` consists of three steps.

- **I step:** The missing values are simulated using the current value θ^{m-1} of the parameter as in the `SEM` algorithm.
- **E step:** The conditional probabilities t_{ik}^m $(1 \leq i \leq n, 1 \leq k \leq K)$ are computed for the current value of θ^{m-1}.
- **M step:** The maximum likelihood estimate of θ is updated by maximizing conditional probabilities t_{ik}^m as conditional mixing weights as in the `EM` algorithm.

CEM algorithm

This algorithm incorporates a classification step between the E and M steps of EM. Starting from an initial parameter θ^0, an iteration of `CEM` consists of three steps.

- **I step:** The missing values are imputed using the current MAP value given by the current value θ^{m-1} of the parameter as in the `EM` algorithm.
- **E step:** The conditional probabilities t_{ik}^m $(1 \leq i \leq n, 1 \leq k \leq K)$ are computed for the current value of θ as done in the E step of EM.
- **C step:** Generate labels $\mathbf{z}^m = \{\mathbf{z}_1^m, ..., \mathbf{z}_n^m\}$ by assigning each point $\mathbf{x}_i$ to the component maximizing the conditional probability $(t_{ik}^m, 1 \leq k \leq K)$.
- **M step:** The maximum likelihood estimate of θ are computed as done in the M step of SEM.

CEM is a *K-means*-like algorithm and contrary to `EM`, it converges in a finite number of iterations. `CEM` is not maximizing the observed log-likelihood L (2) but is maximizing in θ and $\mathbf{z}_1, \ldots, \mathbf{z}_n$ the complete data log-likelihood

$$CL(\theta, \mathbf{z}_1, \ldots, \mathbf{z}_n | \mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \ln[p_k h(\mathbf{x}_i | \boldsymbol{\lambda}_k)]. \tag{8}$$

where the missing component indicator vector $\mathbf{z}_i$ of each sample point is included in the data set. As a consequence, `CEM` is not expected to converge to the maximum likelihood estimate of θ and yields inconsistent estimates of the parameters especially when the mixture components are overlapping or are in disparate proportions (see Mclachlan and Peel (2000), Section 2.21).

2.2. Defining an algorithm in rtkpp

All the algorithms (EM, `SEM`, `CEM` and SemiSEM) are encoded in a `S4` class and can be created using the utility function `clusterAlgo`. This function take as input three parameters:

- `algo`: name of the algorithm to define (`"EM"`, `"SEM"`, `"CEM"` or `"SemiSEM"`),
- `nbIteration`: maximal number of iteration to perform,
- `epsilon`: threshold to use in order to stop the iterations (not used by the `SEM` algorithm).

```
> clusterAlgo(algo="EM",nbIteration=100,epsilon=1e-08)
```

```
**************************************
*** rtkpp ClusterAlgo:
* algorithm            =  EM
* number of iterations =  100
* epsilon              =  1e-08
**************************************
```

2.3. Initialization step

All the estimation algorithms need a first value of the parameter θ. There is three kinds of initialization that can be performed: by generating directly random parameters, by using random classes labels, by using random fuzzy classes. In order to prevent unlucky initialization, multiple initialization with a limited number of estimation steps are performed and the best initialization is conserved.

The initialization step is encoded in a `S4` class and can be created using the utility function `clusterInit`. This function take as input four parameters:

- `method`: name of the initialization to perform (`"random"`, `"class"` or `"fuzzy"`),
- `nbInit` number of initialization to do,
- `algo` name of the algorithm to use during the limited estimation steps,
- `nbIteration` maximal number of iteration to perform during the initialization algorithm,
- `epsilon` threshold to use in order to stop the iterations (see also 2.2).

```
> clusterInit(method="random", nbInit= 2, algo="CEM", nbIteration=10,epsilon=1e-04)

**************************************
*** Cluster init:
* method              =  random
* number of init      =  2
* algorithm            =  CEM
* number of iterations =  10
* epsilon              =  1e-04
**************************************
```

2.4. Estimation Strategy

A strategy is a way to find a good estimate of the parameters of a mixture model and to avoid local maxima of the likelihood function. A strategy is an efficient three steps Search/Run/Select way for maximizing the likelihood:

1. Build a search method for generating `nbShortRun` initial positions. This is based on the initialization method we describe previously.

2. Run a short algorithm for each initial position.
3. Select the solution providing the best likelihood and launch a long run algorithm from this solution.

A strategy is encoded in a S4 class and can be created using the utility function `clusterStrategy()`. This function have no mandatory argument but the default strategy can be tuned. In table 1 the reader will find a summary of all the input parameters of the `clusterStrategy()` function.

Input Parameter	Description
`nbTry`	Integer defining the number of tries. `nbTry` must be a positive integer. Default value is 1.
`nbInit`	Integer defining the number of initialization to do during the initialization step. Default is 3.
`initAlgo`	List of character string with the estimation algorithm to use in the initialization step. Possible values are `"EM"`, `"SEM"`, `"CEM"`, `"SemiSEM"`. Default value is `"SEM"`.
`nbInitIteration`	Integer defining the maximal number of iteration in `initAlgo` algorithm. `nbInitIteration` can be 0. Default value is 20.
`initEpsilon`	Real defining the epsilon value for the algorithm. `initEpsilon` is not used by the `SEM` algorithm. Default value is 0.01.
`nbShortRun`	Integer defining the number of short run to try (the strategy launch an initialization before each short run). Default value is 5.
`shortRunAlgo`	List of character string with the estimation algorithm to use in s short run. Possible values are `"EM"`, `"SEM"`, `"CEM"`, `"SemiSEM"`. Default value is `"EM"`.
`nbShortIteration`	Integers defining the maximal number of iterations in a short run. Default value is 100.
`shortEpsilon`	Real defining the epsilon value in a short run. Only available if `shortRunAlgo` is not `"SEM"`. Default value is 1e-04.
`longRunAlgo`	List of character string with the estimation algorithm to use for the long run. Possible values are `"EM"`, `"SEM"`, `"CEM"`, `"SemiSEM"`. Default value is `"EM"`.
`nbLongIteration`	Integers defining the maximal number of iterations in the the long run. Default value is 1000.
`longEpsilon`	Real defining the epsilon value in the long run. Only available if `longRunAlgo` is not `"SEM"`. Default value is 1e-07.

Table 1: List of all the input parameters of the `clusterStrategy()` function.

```
> clusterStrategy(nbTry=2, nbInit=5, initMethod="class"
+                         , initAlgo="CEM", nbInitIteration=10, initEpsilon=1e-02
+                         , nbShortRun=5
+                         , shortRunAlgo="EM", nbShortIteration=50, shortEpsilon=1e-04
+                         , longRunAlgo="EM", nbLongIteration=100, longEpsilon=1e-08)
```

```
****************************************
*** Cluster Strategy:
* number of try         =  2
* number of short run   =  5
****************************************
*** Initialization :
* method =  random
* number of init       =  5
* algorithm            =  CEM
* number of iterations =  10
* epsilon              =  0.01
****************************************
*** short Algorithm :
* algorithm            =  EM
* number of iterations =  50
* epsilon              =  1e-04
****************************************
*** long algorithm :
* algorithm            =  EM
* number of iterations =  100
* epsilon              =  1e-08
****************************************
```

Users have to take care that there will be `nbInit` x `nbShortRun` starting point of the algorithm.

3. Implemented Mixture Models

3.1. Multivariate (diagonal) Gaussian Mixture Models

A Gaussian density on $\mathbb{R}$ is a density of the form:

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \quad \sigma > 0. \tag{9}$$

A joint diagonal Gaussian density on $\mathbb{R}^d$ is a density of the form:

$$h(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\sigma}) = \prod_{j=1}^{d} f(x^j;\mu^j,\sigma^j) \quad \sigma^j > 0. \tag{10}$$

The parameters $\boldsymbol{\mu} = (\mu^1,\ldots,\mu^d)$ are the position parameters and the parameters $\boldsymbol{\sigma} = (\sigma^1,\ldots,\sigma^d)$ are the standard-deviation parameters. Assumptions on the position and standard-deviation parameters among the variables and the components lead to define four families of mixture model.

Let us write a multidimensional Gaussian mixture model in the from `gaussian_s*` with `s*`, the different ways to parameterize the standard-deviation parameters of a Gaussian mixture:

- `sjk` means that we have one standard-deviation parameter for each variable and for each component,
- `sk` means that the standard-deviation parameters are the same for all the variables inside a component,
- `sj` means that the standard-deviation parameters are different for each variable but are equals between the components,
- and finally `s` means that the standard-deviation parameters are all equals.

The `gaussian_sjk` model is the most general model and have a density function of the form

$$f(\mathbf{x}|\theta) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} g(x_i^j|\mu_k^j, \sigma_k^j). \tag{11}$$

It is possible to get a vector of Gaussian model names using the `clusterDiagGaussianNames` function.

```
> clusterDiagGaussianNames()

[1] "gaussian_pk_sjk" "gaussian_pk_sj"  "gaussian_pk_sk"  "gaussian_pk_s"
[5] "gaussian_p_sjk"  "gaussian_p_sj"   "gaussian_p_sk"   "gaussian_p_s"

> clusterDiagGaussianNames("all", "equal", "free")

[1] "gaussian_pk_sk" "gaussian_p_sk"

> clusterValidDiagGaussianNames(c("gaussian_pk_sjk","poisson_p_ljk"))

[1] FALSE
```

3.2. Multivariate categorical Mixture Models

A Categorical probability distribution on a finite space $\mathcal{X} = \{1, \ldots, L\}$ is a probability distribution of the form:

$$P(x = l) = p_l \quad p_l > 0,\, l \in \mathcal{X}, \tag{12}$$

with the constraint $p_1 + \ldots + p_L = 1$.
A joint Categorical probability distribution on $\mathcal{X}^d$ is a probability distribution of the form:

$$P(\mathbf{x} = (x_1, \ldots, x_d)) = \prod_{j=1}^{d} p_{x_j}^j \tag{13}$$

The parameters $\mathbf{p} = (p^1, \ldots, p^d)$ are the probabilities of the possibles outcomes. Assumptions on the probabilities among the variables and the components lead to define two families of mixture model.

```
> clusterCategoricalNames()

[1] "categorical_pk_pjk" "categorical_pk_pk"  "categorical_p_pjk"
[4] "categorical_p_pk"

> clusterCategoricalNames("all", "equal")

[1] "categorical_pk_pk" "categorical_p_pk"

> clusterValidCategoricalNames(c("categorical_pk_pjk","categorical_p_pk"))

[1] TRUE
```

3.3. Multivariate Poisson Mixture Models

A Poisson probability distribution is a probability over $\mathbb{N}$ of the form

$$p(k;\lambda) = \frac{\lambda^k}{k!} e^{-\lambda} \quad \lambda > 0. \tag{14}$$

A joint Poisson probability on $\mathbb{N}^d$ is a probability distribution of the form

$$h(\mathbf{x};\boldsymbol{\lambda}) = \prod_{j=1}^{d} p(x^j;\lambda^j) \quad \lambda^j > 0. \tag{15}$$

The parameters $\boldsymbol{\lambda} = (\lambda^1, \ldots, \lambda^d)$ are the mean parameters. Assumptions on the mean among the variables and the components lead to define three families of mixture model.

```
> clusterPoissonNames()

[1] "poisson_pk_ljk"  "poisson_pk_ljlk" "poisson_p_ljk"   "poisson_p_ljlk"

> clusterPoissonNames("all")

[1] "poisson_pk_ljk"  "poisson_pk_ljlk" "poisson_p_ljk"   "poisson_p_ljlk"

> clusterValidPoissonNames(c("poisson_pk_ljk","poisson_p_ljlk"))

[1] TRUE
```

3.4. Multivariate Gamma Mixture Models

A gamma density on $\mathbb{R}_+$ is a density of the form:

$$g(x;a,b) = \frac{(x)^{a-1}\, e^{-x/b}}{\Gamma(a)\,(b)^a} \quad a > 0, \quad b > 0. \tag{16}$$

	ajk	ak	aj	a
bjk	`gamma_ajk_bjk` (2dK)	`gamma_ak_bjk` (dK + K)	`gamma_aj_bjk` (dK+d)	`gamma_a_bjk` (dK+1)
bk	`gamma_ajk_bk` (dK+K)	`gamma_ak_bk` (2K)	`gamma_aj_bk` (K+d)	`gamma_a_bk` (K+1)
bj	`gamma_ajk_bj` (dK+d)	`gamma_ak_bj` (K+d)	NA	NA
b	`gamma_ajk_b` (dK+1)	`gamma_ak_b` (K+1)	NA	NA

Table 2: The twelve multidimensional gamma mixture models. In parenthesis the number of parameters of each model.

A joint gamma density on ${\mathbb{R}^d}_+$ is a density of the form:

$$h(\mathbf{x};\mathbf{a},\mathbf{b}) = \prod_{j=1}^{d} g(x^j;a^j,b^j) \quad a^j > 0, \quad b^j > 0. \tag{17}$$

The parameters $\mathbf{a} = (a^1,\ldots,a^d)$ are the shape parameters and the parameters $\mathbf{b} = (b^1,\ldots,b^d)$ are the scale parameters. Assumptions on the scale and shape parameters among the variables and the components lead to define twelve families of mixture model. Let us write a multidimensional gamma mixture model in the from `gamma_a*_b*` with `a*` (resp. `b*`), the different ways to parameterize the shape (resp. scale) parameters of a gamma mixture:

- `ajk` (resp. `bjk`) means that we have one shape (resp. scale) parameter for each variable and for each component,
- `ak` (resp. `bk`) means that the shape (resp. scale) parameters are the same for all the variables inside a component,
- `aj` (resp. `bj`) means that the shape (resp. scale) parameters are different for each variable but are equals between the components,
- and finally `a` (resp. `b`) means that the shape (resp. scale) parameters are the same for all the variables and all the components.

The models we can build in this way are summarized in the table 2, in parenthesis we give the number of parameters of each models.

The `gamma_ajk_bjk` model is the most general and have a density function of the form

$$f(\mathbf{x}_i|\theta) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} g(x_i^j|a_k^j,b_k^j). \tag{18}$$

All the other models can be derived from this model by dropping the indexes in j and/or k from the expression (18). For example the mixture model `gamma_aj_bk` has a density function of the form

$$f(\mathbf{x}_i|\theta) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} g(x_i^j|a^j,b^k). \tag{19}$$

```
> clusterGammaNames()

 [1] "gamma_p_ajk_bjk"  "gamma_p_ajk_bk"   "gamma_p_ajk_bj"   "gamma_p_ajk_b"
 [5] "gamma_p_ak_bjk"   "gamma_p_ak_bk"    "gamma_p_ak_bj"    "gamma_p_ak_b"
 [9] "gamma_pk_ajk_bjk" "gamma_pk_ajk_bk"  "gamma_pk_ajk_bj"  "gamma_pk_ajk_b"
[13] "gamma_pk_ak_bjk"  "gamma_pk_ak_bk"   "gamma_pk_ak_bj"   "gamma_pk_ak_b"

> clusterGammaNames("all", "equal")

[1] "gamma_p_ak_bjk"  "gamma_p_ak_bk"   "gamma_p_ak_bj"   "gamma_p_ak_b"
[5] "gamma_pk_ak_bjk" "gamma_pk_ak_bk"  "gamma_pk_ak_bj"  "gamma_pk_ak_b"

> clusterValidGammaNames(c("gamma_pk_ajbk","gamma_p_ajk_bjk"))

[1] FALSE
```

References

Dempster A, Laird N, Rubin D (1997). "Maximum Likelihood from Incomplete Data with the EM Algorithm (with discussion)." *Journal of the Royal Statistical Society, Series B*, **39**, 1.

Iovleff S (2012). "The Statitiscal ToolKit." http://www.stkpp.org/.

Mclachlan G, Peel D (2000). *Finite Mixture Models*. Wiley Series in Probability and Statistics, 1 edition. Wiley-Interscience. ISBN 9780471006268. URL http://www.worldcat.org/isbn/0471006262.

R Development Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Affiliation:

Serge Iovleff
University Lille 1
Cité Scientifique
IUT "A", Département Informatique
59655 Villeneuve d'Ascq Cédex, France
E-mail: Serge.Iovleff@stkpp.org
URL: http://www.stkpp.org