

# Package ‘QuantileGH’

May 1, 2024

**Type** Package

**Title** Quantile Least Mahalanobis Distance Estimator for Tukey g-&-h Mixture

**Version** 0.1.7

**Date** 2024-04-30

**Description** Functions for simulation, estimation, and model selection of finite mixtures of Tukey g-and-h distributions.

**License** GPL-2

**Imports** methods, mixtools, tclust, fmx, TukeyGH77

**Encoding** UTF-8

**Language** en-US

**Depends** R (>= 4.4.0)

**Suggests** knitr, rmarkdown, mixsmsn

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** Tingting Zhan [aut, cre, cph] (<<https://orcid.org/0000-0001-9971-4844>>),  
Inna Chervoneva [ctb, cph] (<<https://orcid.org/0000-0002-9104-4505>>)

**Maintainer** Tingting Zhan <Tingting.Zhan@jefferson.edu>

**Repository** CRAN

**Date/Publication** 2024-05-01 03:50:03 UTC

## R topics documented:

QuantileGH-package . . . . .	2
fmx_cluster . . . . .	2
fmx_hybrid . . . . .	3
fmx_normix . . . . .	4
QLMDe . . . . .	5
QLMDe_stepK . . . . .	6

QLMDp . . . . .	8
reAssign . . . . .	9
step_fmx . . . . .	10

<b>Index</b>	<b>12</b>
--------------	-----------

---

QuantileGH-package	<i>Quantile Least Mahalanobis Distance Estimator for Tukey <math>g</math>-&amp;-<math>h</math> Mixture</i>
--------------------	--

---

### Description

Tools for simulating and fitting finite mixtures of the 4-parameter Tukey  $g$ -&- $h$  distributions. Tukey  $g$ -&- $h$  mixture is highly flexible to model multimodal distributions with variable degree of skewness and kurtosis in the components. The Quantile Least Mahalanobis Distance estimator [QLMDe](#) is used for estimating parameters of the finite Tukey  $g$ -&- $h$  mixtures. [QLMDe](#) is an indirect estimator that minimizes the Mahalanobis distance between the sample and model-based quantiles. A backward-forward stepwise model selection algorithm is provided to find

- a parsimonious Tukey  $g$ -&- $h$  mixture model, conditional on a given number-of-components; and
- the optimal number of components within the user-specified range.

### Author(s)

**Maintainer:** Tingting Zhan <Tingting.Zhan@jefferson.edu> ([ORCID](#)) [copyright holder]

Other contributors:

- Inna Chervoneva <Inna.Chervoneva@jefferson.edu> ([ORCID](#)) [contributor, copyright holder]

### Examples

# see ?QLMDe

---

fmx_cluster	<i>Naive Estimates of Finite Mixture Distribution via Clustering</i>
-------------	--

---

### Description

Naive estimates for finite mixture distribution [fmx](#) via clustering.

**Usage**

```
fmx_cluster(
  x,
  K,
  distname = c("GH", "norm", "sn"),
  constraint = character(),
  ...
)
```

**Arguments**

x	numeric vector, observations
K	integer scalar, number of mixture components
distname	character scalar, name of parametric distribution of the mixture components
constraint	character vector, parameters ( <i>g</i> and/or <i>h</i> for Tukey <i>g</i> -&- <i>h</i> mixture) to be set at 0. See function <a href="#">fmx_constraint</a> for details.
...	additional parameters, currently not in use

**Details**

First of all, if the specified number of components  $K \geq 2$ , trimmed *k*-means clustering with re-assignment will be performed; otherwise, all observations will be considered as one single cluster. The standard *k*-means clustering is not used since the heavy tails of Tukey *g*-&-*h* distribution could be mistakenly classified as individual cluster(s).

In each of the one or more clusters,

- [letterValue](#)-based estimates of Tukey *g*-&-*h* distribution (Hoaglin, 2006) are calculated, for any  $K \geq 1$ , serving as the starting values for QLMD algorithm. These estimates are provided by function [fmx\\_cluster](#).
- the [median](#) and [mad](#) will serve as the starting values for  $\mu$  and  $\sigma$  (or *A* and *B* for Tukey *g*-&-*h* distribution, with  $g = h = 0$ ), for QLMD algorithm when  $K = 1$ .

**Value**

Function [fmx\\_cluster](#) returns an [fmx](#) object.

---

fmx\_hybrid

*Best Naive Estimates for Finite Mixture Distribution*


---

**Description**

Best estimates for finite mixture distribution [fmx](#).

**Usage**

```
fmx_hybrid(x, test = c("logLik", "CvM", "KS"), ...)
```

**Arguments**

`x`                    [numeric vector](#), observations

`test`                [character](#) scalar, criteria for selecting the optimal estimates. See **Details**.

`...`                additional parameters of functions [fmx\\_normix](#) and [fmx\\_cluster](#)

**Details**

Function [fmx\\_hybrid](#) compares Tukey *g*-&-*h* mixture estimate provided by function [fmx\\_cluster](#) and the normal mixture estimate by function [fmx\\_normix](#), and select the one either with maximum likelihood (`test = 'logLik'`, default), with minimum Cramer-von Mises distance (`test = 'CvM'`) or with minimum Kolmogorov distance ([Kolmogorov\\_fmx](#)).

**Value**

Function [fmx\\_hybrid](#) returns an [fmx](#) object.

**Examples**

```
library(fmx)
d1 = fmx('norm', mean = c(1, 2), sd = .5, w = c(.4, .6))
set.seed(100); hist(x1 <- rfmx(n = 1e3L, dist = d1))
fmx_normix(x1, distname = 'norm', K = 2L)
fmx_normix(x1, distname = 'GH', K = 2L)

(d2 = fmx('GH', A = c(1,6), B = 2, g = c(0,.3), h = c(.2,0), w = c(1,2)))
set.seed(100); hist(x2 <- rfmx(n = 1e3L, dist = d2))
fmx_cluster(x2, K = 2L)
fmx_cluster(x2, K = 2L, constraint = c('g1', 'h2'))
fmx_normix(x2, K = 2L, distname = 'GH')
fmx_hybrid(x2, distname = 'GH', K = 2L)
```

---

fmx\_normix

*Naive Parameter Estimates using Mixture of Normal*


---

**Description**

Naive parameter estimates for finite mixture distribution [fmx](#) using mixture of normal distributions.

**Usage**

```
fmx_normix(x, K, distname = c("norm", "GH", "sn"), alpha = 0.05, R = 10L, ...)
```

**Arguments**

x	numeric vector, observations
K	integer scalar, number of mixture components
distname	character scalar, name of parametric distribution of the mixture components
alpha	numeric scalar, proportion of observations to be trimmed in trimmed $k$ -means algorithm <code>tkmeans</code>
R	integer scalar, number of <code>normalmixEM</code> replicates
...	additional parameters, currently not in use

**Details**

`fmx_normix` ... the cluster centers are provided as the starting values of  $\mu$ 's for the univariate normal mixture by EM algorithm. R replicates of normal mixture estimates are obtained, and the one with maximum likelihood will be selected

**Value**

Function `fmx_normix` returns an `fmx` object.

---

QLMDe

*Quantile Least Mahalanobis Distance estimates*


---

**Description**

The quantile least Mahalanobis distance algorithm estimates the parameters of single-component or finite mixture distributions by minimizing the Mahalanobis distance between the vectors of sample and theoretical quantiles. See `QLMDp` for the default selection of probabilities at which the sample and theoretical quantiles are compared.

The default initial values are estimated based on trimmed  $k$ -means clustering with re-assignment.

**Usage**

```
QLMDe(
  x,
  distname = c("GH", "norm", "sn"),
  K,
  data.name = deparse1(substitute(x)),
  constraint = character(),
  probs = QLMDp(x = x),
  init = c("logLik", "letterValue", "normix"),
  tol = .Machine$double.eps^0.25,
  maxiter = 1000,
  ...
)
```

**Arguments**

<code>x</code>	<a href="#">numeric vector</a> , the one-dimensional observations.
<code>distname</code>	<a href="#">character</a> scalar, name of mixture distribution to be fitted. Currently supports 'norm' and 'GH'.
<code>K</code>	<a href="#">integer</a> scalar, number of components (e.g., must use 2L instead of 2).
<code>data.name</code>	<a href="#">character</a> scalar, name for the observations for user-friendly print out.
<code>constraint</code>	<a href="#">character vector</a> , parameters ( $g$ and/or $h$ for Tukey $g$ -&- $h$ mixture) to be set at 0. See function <a href="#">fmx_constraint</a> for details.
<code>probs</code>	<a href="#">numeric vector</a> , percentiles at where the sample and theoretical quantiles are to be matched. See function <a href="#">QLMDp</a> for details.
<code>init</code>	<a href="#">character</a> scalar for the method of initial values selection, or an <a href="#">fmx</a> object of the initial values. See function <a href="#">fmx_hybrid</a> for more details.
<code>tol, maxiter</code>	see function <a href="#">vuniroot2</a>
<code>...</code>	additional parameters of <a href="#">optim</a>

**Details**

Quantile Least Mahalanobis Distance estimator fits a single-component or finite mixture distribution by minimizing the Mahalanobis distance between the theoretical and observed quantiles, using the empirical quantile variance-covariance matrix [quantile\\_vcov](#).

**Value**

Function [QLMDe](#) returns an [fmx](#) object.

**See Also**

[fmx\\_hybrid](#)

**Examples**

```
data(bmi, package = 'mixsmsn')
hist(x <- bmi[[1L]])
QLMDe(x, distname = 'GH', K = 2L)
```

**Description**

To compare  $gh$ -parsimonious models of Tukey  $g$ -&- $h$  mixtures with different number of components  $K$  (up to a user-specified  $K_{\max}$ ) and select the optimal number of components.

**Usage**

```
QLMDe_stepK(
  x,
  distname = c("GH", "norm"),
  data.name = deparse1(substitute(x)),
  Kmax = 3L,
  test = c("BIC", "AIC"),
  direction = c("forward", "backward"),
  ...
)
```

**Arguments**

x	numeric vector, observations
distname, data.name	character scalars, see parameters of the same names in function <a href="#">QLMDe</a>
Kmax	integer scalar $K_{\max}$ , maximum number of components to be considered. Default 3L
test	character scalar, criterion to be used, either Akaike's information criterion <a href="#">AIC</a> , or Bayesian information criterion <a href="#">BIC</a> (default).
direction	character scalar, direct of selection in function <a href="#">step_fmx</a> , either 'forward' (default) or 'backward'
...	additional parameters

**Details**

Function [QLMDe\\_stepK](#) compares the *gh*-parsimonious models with different number of components  $K$ , and selects the optimal number of components using BIC (default) or AIC.

The forward selection starts with finding the *gh*-parsimonious model (via function [step\\_fmx](#)) at  $K = 1$ . Let the current number of component be  $K^c$ . We compare the *gh*-parsimonious models of  $K^c + 1$  and  $K^c$  component, respectively, using BIC or AIC. If  $K^c$  is preferred, then the forward selection is stopped, and  $K^c$  is considered the optimal number of components. If  $K^c + 1$  is preferred, then the forward selection is stopped if  $K^c + 1 = K_{\max}$ , otherwise update  $K^c$  with  $K^c + 1$  and repeat the previous steps.

**Value**

Function [QLMDe\\_stepK](#) returns an object of S3 class 'stepK', which is a [list](#) of selected models (in reversed order) with attribute(s) 'direction' and 'test'.

**Examples**

```
data(bmi, package = 'mixsmsn')
hist(x <- bmi[[1L]])
QLMDe_stepK(x, distname = 'GH', Kmax = 2L)
```

---

 QLMDp

*Percentages for Quantile Least Mahalanobis Distance estimation*


---

## Description

A vector of probabilities to be used in Quantile Least Mahalanobis Distance estimation ([QLMDe](#)).

## Usage

```
QLMDp(
  from = 0.05,
  to = 0.95,
  length.out = 15L,
  equidistant = c("prob", "quantile"),
  extra = c(0.005, 0.01, 0.02, 0.03, 0.97, 0.98, 0.99, 0.995),
  x
)
```

## Arguments

from, to	<a href="#">numeric</a> scalar, minimum and maximum of the equidistant (in probability or quantile) probabilities. Default .05 and .95, respectively
length.out	non-negative <a href="#">integer</a> scalar, the number of the equidistant (in probability or quantile) probabilities.
equidistant	<a href="#">character</a> scalar. If 'prob' (default), then the probabilities are equidistant. If 'quantile', then the quantiles (of the observations x) corresponding to the probabilities are equidistant.
extra	<a href="#">numeric vector</a> of <i>additional</i> probabilities, default c(.005, .01, .02, .03, .97, .98, .99, .995).
x	<a href="#">numeric vector</a> of observations, only used when equidistant = 'quantile'.

## Details

The default arguments of function [QLMDp](#) returns the probabilities of c(.005, .01, .02, .03, seq.int(.05, .95, length.out = 15L), .97, .98, .99, .995).

## Value

A [numeric vector](#) of probabilities to be supplied to parameter p of Quantile Least Mahalanobis Distance [QLMDe](#) estimation). In practice, the length of this probability [vector](#) p must be equal or larger than the number of parameters in the distribution model to be estimated.



**Examples**

```

library(fmx)
(d2 = fmx('GH', A = c(1,6), B = 2, g = c(0,.3), h = c(.2,0), w = c(1,2)))
set.seed(100); hist(x2 <- rfm(x = 1e3L, dist = d2))

# equidistant in probabilities
(p1 = QLMDp())

# equidistant in quantiles
(p2 = QLMDp(equidistant = 'quantile', x = x2))

```

---

reAssign	<i>Re-Assign Observations Trimmed Prior to Trimmed <math>k</math>-Means Clustering</i>
----------	--

---

**Description**

Re-assign the observations, which are trimmed in the trimmed  $k$ -means algorithm, back to the closest cluster as determined by the smallest Mahalanobis distance.

**Usage**

```

reAssign(x, ...)

## S3 method for class 'tkmeans'
reAssign(x, ...)

```

**Arguments**

`x` a [tkmeans](#) object  
`...` potential parameters, currently not in use.

**Details**

Given the [tkmeans](#) input, the [mahalanobis](#) distance is computed between each trimmed observation and each cluster. Each trimmed observation is assigned to the closest cluster (i.e., with the smallest Mahalanobis distance).

**Value**

Function [reAssign.tkmeans](#) returns an 'reAssign\_tkmeans' object, which inherits from [tkmeans](#) class.

**Note**

Either [kmeans](#) or [tkmeans](#) is slow for big `x`.

**Examples**

```
library(tclust)
data(geyser2)
clus = tkmeans(geyser2, k = 3L, alpha = .03)
plot(clus, main = 'Before Re-Assigning')
plot(reAssign(clus), main = 'After Re-Assigning')
```

step\_fmx

*Forward Selection of  $gh$ -parsimonious Model with Fixed Number of Components  $K$*

**Description**

To select the  $gh$ -parsimonious mixture model, i.e., with some  $g$  and/or  $h$  parameters equal to zero, conditionally on a fixed number of components  $K$ .

**Usage**

```
step_fmx(
  object,
  test = c("BIC", "AIC"),
  direction = c("forward", "backward"),
  ...
)
```

**Arguments**

object	<b>fmX</b> object
test	<b>character</b> scalar, criterion to be used, either Akaike's information criterion <b>AIC</b> -like, or Bayesian information criterion <b>BIC</b> -like (default).
direction	<b>character</b> scalar, 'forward' (default) or 'backward'
...	additional parameters, currently not in use

**Details**

The algorithm starts with quantile least Mahalanobis distance estimates of either the full mixture of Tukey  $g$ -&- $h$  distributions model, or a constrained model (i.e., some  $g$  and/or  $h$  parameters equal to zero according to the user input). Next, each of the non-zero  $g$  and/or  $h$  parameters is tested using the likelihood ratio test. If all tested  $g$  and/or  $h$  parameters are significantly different from zero at the level 0.05 the algorithm is stopped and the initial model is considered  $gh$ -parsimonious. Otherwise, the  $g$  or  $h$  parameter with the largest p-value is constrained to zero for the next iteration of the algorithm.

The algorithm iterates until only significantly-different-from-zero  $g$  and  $h$  parameters are retained, which corresponds to  $gh$ -parsimonious Tukey  $g$ -&- $h$  mixture model.

**Value**

Function [step\\_fmx](#) returns an object of S3 class 'step\_fmx', which is a [list](#) of selected models (in reversed order) with attribute(s) 'direction' and 'test'.

**See Also**

[step](#)

# Index

AIC, [7](#), [10](#)  
algorithm, [5](#)

BIC, [7](#), [10](#)

character, [3–8](#), [10](#)

fmx, [2–6](#), [10](#)  
fmx\_cluster, [2](#), [3](#), [4](#)  
fmx\_constraint, [3](#), [6](#)  
fmx\_hybrid, [3](#), [4](#), [6](#)  
fmx\_normix, [4](#), [4](#), [5](#)

integer, [3](#), [5–8](#)

kmeans, [9](#)  
Kolmogorov\_fmx, [4](#)

letterValue, [3](#)  
list, [7](#), [11](#)

mad, [3](#)  
mahalanobis, [9](#)  
median, [3](#)

normalmixEM, [5](#)  
numeric, [3–8](#)

optim, [6](#)

QLMDe, [2](#), [5](#), [6–8](#)  
QLMDe\_stepK, [6](#), [7](#)  
QLMDp, [5](#), [6](#), [8](#), [8](#)  
quantile\_vcov, [6](#)  
QuantileGH (QuantileGH-package), [2](#)  
QuantileGH-package, [2](#)

reAssign, [9](#)  
reAssign.tkmeans, [9](#)

step, [11](#)  
step\_fmx, [7](#), [10](#), [11](#)

tkmeans, [5](#), [9](#)

vector, [3–8](#)  
vuniroot2, [6](#)