

Package ‘nor1mix’

April 17, 2009

Title Normal (1-d) Mixture Models (S3 Classes and Methods)

Version 1.0-7

Date 2007-03-15

Author Martin Mächler

Maintainer Martin Maechler <maechler@stat.math.ethz.ch>

Description Onedimensional Normal Mixture Models Classes, for, e.g., density estimation or clustering algorithms research and teaching; providing the widely used Marron-Wand densities, see ?MarronWand.

License GPL-2

Depends R (>= 2.0.0)

Encoding latin1

NOTE Use the ‘mclust’ package for estimation!

Repository CRAN

Date/Publication 2007-06-12 18:20:37

R topics documented:

dnorMix	2
MarronWand	3
norMix	4
plot.norMix	6
pnorMix	7
r.norMix	8
rnorMix	9

Index	11
--------------	-----------

dnorMix

*Normal Mixture Density***Description**

Evaluate the density function of the normal mixture specified as `norMix` object.

Usage

```
dnorMix(x, obj, log = FALSE)
dnorMixL(obj, x = NULL, log = FALSE, xlim = NULL, n = 511)
```

Arguments

<code>obj</code>	an object of class <code>norMix</code> .
<code>x</code>	numeric vector with abscissa values where to evaluate the density. For <code>dnorMixL()</code> by default, when <code>NULL</code> , it is constructed from <code>n</code> (and <code>xlim</code> if that is specified).
<code>log</code>	logical indicating <i>log</i> -density values should be returned.
<code>xlim</code>	range of abscissa values, used if <code>x == NULL</code> . By default, <code>xlim</code> is taken as mean plus/minus 3 standard deviations of the normal mixture.
<code>n</code>	number of abscissa values to generate if <code>x</code> is not specified.

Value

`dnorMix(x)` returns the numeric vector of density values $f(x)$, logged if `log` is `TRUE`.

`dnorMixL()` returns a list with components

<code>x</code>	the abscissa values.
<code>y</code>	the density values $f(x)$ as for <code>dnorMix()</code> .

See Also

[rnormMix](#) for random number generation, and [normMix](#) for the construction and further methods, particularly [plot.normMix](#) which makes use `dnorMix`.

Examples

```
ff <- dnorMixL(MW.nm7)
str(ff)
plot(ff, type = "h") # rather use plot(ff, ...)
```

Description

The fifteen density examples used in Marron and Wand (1992)'s simulation study have been used in quite a few subsequent studies, can all be written as normal mixtures and are provided here for convenience and didactical examples of normal mixtures. Number 16 has been added by Jansen et al.

Usage

```
MW.nm1 # Gaussian
MW.nm2 # Skewed
MW.nm3 # Str Skew
MW.nm4 # Kurtotic
MW.nm5 # Outlier
MW.nm6 # Biomdal
MW.nm7 # Separated (bimodal)
MW.nm8 # Aymmettric Bimodal
MW.nm9 # Trimodal
MW.nm10 # Claw
MW.nm11 # Double Claw
MW.nm12 # Asymmetric Claw
MW.nm13 # Asymm. Double Claw
MW.nm14 # Smooth Comb
MW.nm15 # Discrete Comb
MW.nm16 # Distant Bimodal
```

Author(s)

Martin Maechler

Source

The first part are translated from Steve Marron's Matlab code at <http://www.stat.unc.edu/postscript/papers/marron/parameters/nmpar.m>

References

Marron, S. and Wand, M. (1992) Exact Mean Integrated Squared Error; *Annals of Statistics* **20**, 712–736.

For number 16,
Janssen, Marron, Verb..., Sarle (1995)

Examples

```

MW.nm10
plot(MW.nm14)

## These are defined as norMix() calls in ../R/zMarrWand-dens.R
ppos <- which("package:norlmix" == search())
nms <- ls(pat="^MW.nm", pos = ppos)
nms <- nms[order(as.numeric(substring(nms,6)))]
for(n in nms) {
  cat("\n",n,":\n"); print(get(n, pos = ppos))
}

## Plot all of them:
op <- par(mfrow=c(4,4), mgp = c(1.2, 0.5, 0), tcl = -0.2,
          mar = .1 + c(2,2,2,1), oma = c(0,0,3,0))
for(n in nms) { plot(get(n, pos = ppos))}
mtext("The Marron-Wand Densities", outer= TRUE, font= 2, cex= 1.6)

## and their Q-Q-plots (not really fast):
prob <- ppoints(N <- 100)
for(n in nms) {
  D <- get(n, pos = ppos)
  qqnorm(qnorMix(D, prob), main = n)
}
mtext("QQ-plots of Marron-Wand Densities", outer = TRUE,
      font = 2, cex = 1.6)

par(op)

```

norMix

Mixtures of Univariate Normal Distributions

Description

Objects of class `norMix` represent finite mixtures of (univariate) normal (aka Gaussian) distributions. Methods for construction, printing, plotting, and basic computations are provided.

Usage

```

norMix(mu, sig2 = rep(1,m), w = NULL, name = NULL, long.name = FALSE)

is.norMix(obj)
m.norMix(obj)
var.norMix(x, ...)
## S3 method for class 'norMix':
mean(x, ...)
## S3 method for class 'norMix':
print(x, ...)

```

Arguments

mu	numeric vector of length K , say, specifying the means μ of the K normal components.
sig2	numeric vector of length K , specifying the variances σ^2 of the K normal components.
w	numeric vector of length K , specifying the mixture proportions π_j of the normal components, $j = 1, \dots, K$. Defaults to equal proportions
name	optional name tag of the result (used for printing).
long.name	logical indicating if the <code>name</code> attribute should use punctuation and hence be slightly larger than by default.
obj, x	an object of class <code>norMix</code> .
...	further arguments passed to methods.

Details

The (one dimensional) normal mixtures, **R** objects of class "norMix", are constructed by `norMix` and tested for by `is.norMix`. `m.norMix()` returns the number of mixture components; the `mean()` method (for `class` "norMix" returns the `mu` vector of means and `var.norMix()` (not a method, call the function explicitly!) the `sig2` vector of variances.

For further methods see below.

Value

`norMix` returns objects of class "norMix" which are currently implemented as 3-column matrix with column names `mu`, `sig2`, and `w`, and further attributes. The user should rarely need to access the underlying structure directly.

Note

For *estimation* of the parameters of a normal mixture distribution, I recommend using other **R** packages, notably package **mclust**.

Author(s)

Martin Maechler

See Also

[dnorMix](#) for the density, [pnorMix](#) for the cumulative distribution and the quantile function ([qnorMix](#)), and [rnorMix](#) for random numbers and [plot.norMix](#), the plot method.

[MarronWand](#) has the Marron-Wand densities as normal mixtures.

Examples

```
ex <- norMix(mu = c(1,2,5))# s^2 = 1, equal proportions
ex
plot(ex)# looks like a mixture of only 2

plot(ex, log = "y")# maybe "revealing"
```

plot.norMix

Plotting Methods for 'norMix' Objects

Description

The `plot` and `lines` methods for `norMix` objects draw the normal mixture density, optionally additionally with a fitted normal density.

Usage

```
## S3 method for class 'norMix':
plot(x, type = "l", n = 511, xout = NULL, xlim = NULL,
      xlab = "x", ylab = "f(x)", main = attr(x, "name"), lwd = 1.4,
      p.norm = TRUE, p.h0 = TRUE,
      parNorm = list(col = 2, lty = 2, lwd = 0.4),
      parH0 = list(col = 3, lty = 3, lwd = 0.4), ...)

## S3 method for class 'norMix':
lines(x, type = "l", n = 511, xout = NULL,
      lwd = 1.4, p.norm = FALSE, parNorm = list(col = 2, lty = 2, lwd = 0.4),
      ...)
```

Arguments

<code>x</code>	object of class <code>norMix</code> .
<code>type</code>	character denoting type of plot, see, e.g. <code>lines</code> .
<code>n</code>	number of points to generate if <code>xout</code> is unspecified.
<code>xout</code>	numeric or <code>NULL</code> giving the abscissae at which to draw the density.
<code>xlim</code>	range of <code>x</code> values to use; particularly important if <code>xout</code> is not specified where <code>xlim</code> is passed to <code>dnorMix</code> and gets a smart default if unspecified.
<code>xlab, ylab</code>	labels for the <code>x</code> and <code>y</code> axis with defaults.
<code>main</code>	main title of plot, defaulting to the <code>norMix</code> name.
<code>lwd</code>	line width for plotting with a non-standard default.
<code>p.norm</code>	logical indicating if the normal density with the same mean and variance should be drawn as well.
<code>p.h0</code>	logical indicating if the line $y = 0$ should be drawn.
<code>parNorm</code>	graphical parameters for drawing the normal density if <code>p.norm</code> is true.
<code>parH0</code>	graphical parameters for drawing the line $y = 0$ if <code>p.h0</code> is true.
<code>...</code>	further arguments passed to and from methods.

Author(s)

Martin Maechler

See Also[norMix](#) for the construction and further methods, particularly [dnorMix](#) which is used here.**Examples**

```
plot(norMix(m=c(0,3),s=c(4,1)))
## Further examples in ?norMix and ?rnormix
```

pnorMix

*Normal Mixture Cumulative Distribution and Quantiles***Description**

Compute cumulative probabilities or quantiles (the inverse) for a normal mixture specified as [norMix](#) object.

Usage

```
pnorMix(q, obj, lower.tail = TRUE, log.p = FALSE)

qnorMix(p, obj, lower.tail = TRUE, log.p = FALSE,
        tol = .Machine$double.eps^0.25, maxiter = 1000)
```

Arguments

`obj` an object of class `norMix`.
`p` numeric vector of probabilities.
`q` numeric vector of quantiles
`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`log.p` logical; if TRUE, probabilities `p` are given as $\log(p)$.
`tol, maxiter` determine the root finding algorithm, see [uniroot](#).

Details

Whereas the distribution function `pnorMix` is the trivial sum of weighted normal probabilities ([pnorm](#)), its inverse is currently computed numerically using [uniroot](#) to find `q` such that `pnorMix(obj, q) == p`.

Value

a numeric vector of the same length as `p` or `q`, respectively.

Author(s)

First version by Erik Jørgensen (Erik.Jorgensen@agrsci.dk).

See Also

[dnorMix](#) for the density function.

Examples

```
MW.nm3 # the "strange skew" one
plot(MW.nm3)
## now the cumulative :
x <- seq(-4,4, length=1001)
plot(x, pnorMix(x, MW.nm3), type="l", col=2)
## and some of its inverse :
pp <- seq(.1, .9, by=.1)
plot(qnorMix(pp, MW.nm3), pp)

## The "true" median of a normal mixture:
median.norMix <- function(x) qnorMix(1/2, x)
median.norMix(MW.nm3) ## -2.32
```

r.norMix

Ratio of Normal Mixture to Corresponding Normal

Description

Compute $r(x) = f(x)/f_0(x)$ where $f()$ is a normal mixture density and f_0 the normal density with the same mean and variance as f .

Usage

```
r.norMix(obj, x = NULL, xlim = NULL, n = 511, xy.return = TRUE)
```

Arguments

obj	an object of class <code>norMix</code> .
x	numeric vector with abscissa values where to evaluate the density. Default is constructed from <code>n</code> (and <code>xlim</code> if specified).
xlim	range of abscissa values, used if <code>x == NULL</code> . By default, <code>xlim</code> taken as mean plus/minus 3 standard deviations of the normal mixture.
n	number of abscissa values to generate if <code>x</code> is not specified.
xy.return	logical indicating if the result should be a list or just a numeric vector, see below.

Value

It depends on `xy.return`. If it's false, a numeric vector of the same length as `x`, if true (as per default), a list that can be plotted, with components

<code>x</code>	abscissa values corresponding to argument <code>x</code> .
<code>y</code>	corresponding values $r(x)$.
<code>f0</code>	values of the moment matching normal density $f_0(x)$.

Note

The ratio function is used in certain semi-parametric density estimation methods (and theory).

Examples

```
d3 <- rnormMix(m = 5*(0:2), w = c(0.6, 0.3, 0.1))
plot(d3)
rd3 <- r.rnormMix(d3)
str(rd3)
stopifnot(rd3$y == r.rnormMix(d3, xy.ret = FALSE))
par(new = TRUE)
plot(rd3, type = "l", col = 3, axes = FALSE, xlab = "", ylab = "")
axis(4, col.axis=3)
```

rnormMix

Generate "Normal Mixture" Distributed Random Numbers

Description

Generate `n` random numbers, distributed according to a normal mixture.

Usage

```
rnormMix(n, obj)
```

Arguments

<code>n</code>	the number of random numbers desired.
<code>obj</code>	an object of class <code>rnormMix</code> .

Details

For a mixture of m , i.e., `m.rnormMix(obj)`, components, generate the number in each component as multinomial, and then use `rnorm` for each.

Value

numeric vector of length `n`.

See Also

[dnorMix](#) for the density, and [norMix](#) for the construction and further methods.

Examples

```
x <- rnorMix(5000, MW.nm10)
hist(x)# you don't see the claw
plot(density(x), ylim = c(0,0.6),
      main = "Estim. and true 'MW.nm10' density")
lines(MW.nm10, col = "orange")
```

Index

*Topic **datasets**

MarronWand, 3

*Topic **distribution**

dnorMix, 2

MarronWand, 3

norMix, 4

plot.norMix, 6

pnorMix, 7

r.norMix, 8

rnorMix, 9

*Topic **hplot**

plot.norMix, 6

class, 5

dnorMix, 2, 5–8, 10

dnorMixL (*dnorMix*), 2

is.norMix (*norMix*), 4

lines, 6

lines.norMix (*plot.norMix*), 6

m.norMix (*norMix*), 4

MarronWand, 3, 5

mean.norMix (*norMix*), 4

MW.nm1 (*MarronWand*), 3

MW.nm10 (*MarronWand*), 3

MW.nm11 (*MarronWand*), 3

MW.nm12 (*MarronWand*), 3

MW.nm13 (*MarronWand*), 3

MW.nm14 (*MarronWand*), 3

MW.nm15 (*MarronWand*), 3

MW.nm16 (*MarronWand*), 3

MW.nm2 (*MarronWand*), 3

MW.nm3 (*MarronWand*), 3

MW.nm4 (*MarronWand*), 3

MW.nm5 (*MarronWand*), 3

MW.nm6 (*MarronWand*), 3

MW.nm7 (*MarronWand*), 3

MW.nm8 (*MarronWand*), 3

MW.nm9 (*MarronWand*), 3

norMix, 2, 4, 6, 7, 10

plot.norMix, 2, 5, 6

pnorm, 7

pnorMix, 5, 7

print.norMix (*norMix*), 4

qnorMix (*pnorMix*), 7

r.norMix, 8

rnorm, 9

rnorMix, 2, 5, 9

uniroot, 7

var.norMix (*norMix*), 4