Package ‘nspmix’

September 20, 2020

Title Nonparametric and Semiparametric Mixture Estimation
Version 1.5-0
Date 2020-09-19
Author Yong Wang
Maintainer Yong Wang <yongwang@auckland.ac.nz>
Depends lsei
Imports graphics, methods, stats
Description Mainly for maximum likelihood estimation of nonparametric
and semiparametric mixture models, but can also be used
for fitting finite mixtures. The algorithms are
developed in Wang (2007)

Encoding UTF-8
License GPL (>= 2)
URL https://www.stat.auckland.ac.nz/~yongwang/
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2020-09-20 14:40:07 UTC

R topics documented:

betablockers ................................................................. 2
brca ................................................................. 3
cnm ................................................................. 3
cnmms .............................................................. 6
cvps ............................................................. 10
disc ............................................................. 11
lungcancer ............................................................. 12
mlogit ............................................................. 13
betablockers

Description

Contains the data of the 22-center clinical trial of beta-blockers for reducing mortality after myocardial infarction.

Format

A numeric matrix with four columns:
- center: center identification code.
- deaths: the number of deaths in the center.
- total: the number of patients taking beta-blockers in the center.
- treatment: 0 for control, and 1 for treatment.

Source


References


See Also

mlogit.cnms.

Examples

data(betablockers)
x = mlogit(betablockers)
cnms(x)
### brca

**Z-values of BRCA Data**

**Description**

Contains 3226 z-values computed by Efron (2004) from the data obtained in a well-known microarray experiment concerning two types of genetic mutations causing increased breast cancer risk, BRCA1 and BRCA2.

**Format**

A numeric vector containing 3226 z-values.

**References**


**See Also**

npnorm, cnm.

**Examples**

```r
data(brca)
x = npnorm(brca)
plot(cnm(x), x)
```

---

### cnm

**Maximum Likelihood Estimation of a Nonparametric Mixture Model**

**Description**

Function `cnm` can be used to compute the maximum likelihood estimate of a nonparametric mixing distribution (NPMLE) that has a one-dimensional mixing parameter, or simply the mixing proportions with support points held fixed.
Usage

cnm(
  x,
  init = NULL,
  model = c("npmle", "proportions"),
  maxit = 100,
  tol = 1e-06,
  grid = 100,
  plot = c("null", "gradient", "probability"),
  verbose = 0
)

Arguments

x a data object of some class that is fully defined by the user. The user needs to supply certain functions as described below.

init list of user-provided initial values for the mixing distribution mix and the structural parameter beta.

model the type of model that is to estimated: the non-parametric MLE (if npmle), or mixing proportions only (if proportions).

maxit maximum number of iterations.

tol a tolerance value needed to terminate an algorithm. Specifically, the algorithm is terminated, if the increase of the log-likelihood value after an iteration is less than tol.

grid number of grid points that are used by the algorithm to locate all the local maxima of the gradient function. A larger number increases the chance of locating all local maxima, at the expense of an increased computational cost. The locations of the grid points are determined by the function gridpoints provided by each individual mixture family, and they do not have to be equally spaced. If needed, a gridpoints function may choose to return a different number of grid points than specified by grid.

plot whether a plot is produced at each iteration. Useful for monitoring the convergence of the algorithm. If ="null", no plot is produced. If ="gradient", it plots the gradient curves and if ="probability", the plot function defined by the user for the class is used.

verbose verbosity level for printing intermediate results in each iteration, including none (= 0), the log-likelihood value (= 1), the maximum gradient (= 2), the support points of the mixing distribution (= 3), the mixing proportions (= 4), and if available, the value of the structural parameter beta (= 5).

Details

A finite mixture model has a density of the form

\[ f(x; \pi, \theta, \beta) = \sum_{j=1}^{k} \pi_j f(x; \theta_j, \beta). \]
where $\pi_j \geq 0$ and $\sum_{j=1}^{k} \pi_j = 1$.

A nonparametric mixture model has a density of the form

$$f(x; G) = \int f(x; \theta) dG(\theta),$$

where $G$ is a mixing distribution that is completely unspecified. The maximum likelihood estimate of the nonparametric $G$, or the NPMLE of $G$, is known to be a discrete distribution function.

Function `cnm` implements the CNM algorithm that is proposed in Wang (2007) and the hierarchical CNM algorithm of Wang and Taylor (2013). The implementation is generic using S3 object-oriented programming, in the sense that it works for an arbitrary family of mixture models defined by the user. The user, however, needs to supply the implementations of the following functions for their self-defined family of mixture models, as they are needed internally by function `cnm`:

- `initial(x, beta, mix, kmax)`
- `valid(x, beta)`
- `logd(x, beta, pt, which)`
- `gridpoints(x, beta, grid)`
- `suppspace(x, beta)`
- `length(x)`
- `print(x, ...)`
- `weight(x, ...)`

While not needed by the algorithm for finding the solution, one may also implement `plot(x, mix, beta, ...)` so that the fitted model can be shown graphically in a user-defined way. Inside `cnm`, it is used when `plot="probability"` so that the convergence of the algorithm can be graphically monitored.

For creating a new class, the user may consult the implementations of these functions for the families of mixture models included in the package, e.g., `npnorm` and `nppois`.

**Value**

- `family` the name of the mixture family that is used to fit to the data.
- `num.iterations` number of iterations required by the algorithm
- `max.gradient` maximum value of the gradient function, evaluated at the beginning of the final iteration
- `convergence` convergence code. $=0$ means a success, and $=1$ reaching the maximum number of iterations
- `ll` log-likelihood value at convergence
- `mix` MLE of the mixing distribution, being an object of the class `disc` for discrete distributions.
- `beta` value of the structural parameter, that is held fixed throughout the computation.

**Author(s)**

Yong Wang <yongwang@auckland.ac.nz>
References


See Also

`nnls`, `npnorm`, `nppois`, `cnmms`.

Examples

```r
## Simulated data
x = rnppois(1000, disc(c(1,4), c(0.7,0.3))) # Poisson mixture
(r = cmn(x))
plot(r, x)

x = rnpnorm(1000, disc(c(0,4), c(0.3,0.7)), sd=1) # Normal mixture
plot(cmn(x), x) # sd = 1
plot(cmn(x, init=list(beta=0.5)), x) # sd = 0.5
mix0 = disc(seq(min(x$v),max(x$v), len=100)) # over a finite grid
plot(cmn(x, init=list(beta=0.5, mix=mix0), model="p"),
 x, add=TRUE, col="blue") # An approximate NPMLE

## Real-world data
data(thai)
plot(cmn(x <- nppois(thai)), x) # Poisson mixture

data(brca)
plot(cmn(x <- npnorm(brca)), x) # Normal mixture
```

---

**cnmms**

*Maximum Likelihood Estimation of a Semiparametric Mixture Model*

**Description**

Functions `cnmms`, `cnmpl` and `cnmap` can be used to compute the maximum likelihood estimate of a semiparametric mixture model that has a one-dimensional mixing parameter. The types of mixture models that can be computed include finite, nonparametric and semiparametric ones.
Usage

\[ \text{cnmms}(x, \text{init=NULL, maxit=1000, model=c("spmle","npmle")}, \text{tol=1e-6, grid=100, kmax=Inf, plot=c("null","gradient","probability")), verbose=0}) \]

\[ \text{cnmpl}(x, \text{init=NULL, tol=1e-6, tol.npmle=tol*1e-4, grid=100, maxit=1000, plot=c("null","gradient","probability")), verbose=0}) \]

\[ \text{cnmap}(x, \text{init=NULL, maxit=1000, tol=1e-6, grid=100, plot=c("null","gradient")), verbose=0}) \]

Arguments

\textit{x} \quad \text{a data object of some class that can be defined fully by the user}

\textit{init} \quad \text{list of user-provided initial values for the mixing distribution mix and the structural parameter beta}

\textit{maxit} \quad \text{maximum number of iterations}

\textit{model} \quad \text{the type of model that is to estimated: non-parametric MLE (npmle) or semi-parametric MLE (spmle).}

\textit{tol} \quad \text{a tolerance value that is used to terminate an algorithm. Specifically, the algorithm is terminated, if the relative increase of the log-likelihood value after an iteration is less than tol. If an algorithm converges rapidly enough, then } \sim \log_{10}(\text{tol}) \text{ is roughly the number of accurate digits in log-likelihood.}

\textit{grid} \quad \text{number of grid points that are used by the algorithm to locate all the local maxima of the gradient function. A larger number increases the chance of locating all local maxima, at the expense of an increased computational cost. The locations of the grid points are determined by the function gridpoints provided by each individual mixture family, and they do not have to be equally spaced. If needed, an individual gridpoints function may return a different number of grid points than specified by grid.}

\textit{kmax} \quad \text{upper bound on the number of support points. This is particularly useful for fitting a finite mixture model.}

\textit{plot} \quad \text{whether a plot is produced at each iteration. Useful for monitoring the convergence of the algorithm. If null, no plot is produced. If gradient, it plots the gradient curves and if probability, the plot function defined by the user of the class is used.}

\textit{verbose} \quad \text{verbosity level for printing intermediate results in each iteration, including none (= 0), the log-likelihood value (= 1), the maximum gradient (= 2), the support points of the mixing distribution (= 3), the mixing proportions (= 4), and if available, the value of the structural parameter beta (= 5).}

\textit{tol.npmle} \quad \text{a tolerance value that is used to terminate the computing of the NPMLE internally.}

Details

Function \textit{cnmms} can also be used to compute the maximum likelihood estimate of a finite or non-parametric mixture model.
A finite mixture model has a density of the form

\[ f(x; \pi, \theta, \beta) = \sum_{j=1}^{k} \pi_j f(x; \theta_j, \beta). \]

where \( \pi_j \geq 0 \) and \( \sum_{j=1}^{k} \pi_j = 1 \).

A nonparametric mixture model has a density of the form

\[ f(x; G) = \int f(x; \theta)dG(\theta), \]

where \( G \) is a mixing distribution that is completely unspecified. The maximum likelihood estimate of the nonparametric \( G \), or the NPMLE of \( G \), is known to be a discrete distribution function.

A semiparametric mixture model has a density of the form

\[ f(x; G, \beta) = \int f(x; \theta, \beta)dG(\theta), \]

where \( G \) is a mixing distribution that is completely unspecified and \( \beta \) is the structural parameter.

Of the three functions, cnmms is recommended for most problems; see Wang (2010).

Functions cnmms, cnmpl and cmap implement the algorithms CNM-MS, CNM-PL and CNM-AP that are described in Wang (2010). Their implementations are generic using S3 object-oriented programming, in the sense that they can work for an arbitrary family of mixture models that is defined by the user. The user, however, needs to supply the implementations of the following functions for their self-defined family of mixture models, as they are needed internally by the functions above:

- initial(x,beta,mix,kmax)
- valid(x,beta)
- logd(x,beta,pt,which)
- gridpoints(x,beta,grid)
- suppspace(x,beta)
- length(x)
- print(x,...)
- weight(x,...)

While not needed by the algorithms, one may also implement

- plot(x,mix,beta,...)

so that the fitted model can be shown graphically in a way that the user desires.

For creating a new class, the user may consult the implementations of these functions for the families of mixture models included in the package, e.g., cvp and mlogit.
Value

- **family**: the class of the mixture family that is used to fit to the data.
- **num.iterations**: Number of iterations required by the algorithm.
- **grad**: For cnmms, it contains the values of the gradient function at the support points and the first derivatives of the log-likelihood with respect to theta and beta. For cnmpl, it contains only the first derivatives of the log-likelihood with respect to beta. For cnmap, it contains only the gradient of beta.
- **max.gradient**: Maximum value of the gradient function, evaluated at the beginning of the final iteration. It is only given by function cnmap.
- **convergence**: convergence code. =0 means a success, and =1 reaching the maximum number of iterations.
- **ll**: log-likelihood value at convergence.
- **mix**: MLE of the mixing distribution, being an object of the class disc for discrete distributions.
- **beta**: MLE of the structural parameter

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


See Also

- `nnls`, `cnm`, `cvp`, `cvps`, `mlogit`

Examples

```r
## Compute the MLE of a finite mixture
x = rnpnorm(100, disc(c(0,4), c(0.7,0.3)), sd=1)
for(k in 1:6) plot(cnmms(x, kmax=k), x, add=(k>1), comp="null", col=k+1,
main="Finite Normal Mixtures")
legend("topright", 0.3, leg=paste0("k = ",1:6), lty=1, lwd=2, col=2:7)

## Compute a semiparametric MLE
# Common variance problem
x = rcvps(k=50, ni=5:10, mu=c(0,4), pr=c(0.7,0.3), sd=3)
cnmms(x) # CNM-MS algorithm
cnmpl1(x) # CNM-PL algorithm
cnmap(x) # CNM-AP algorithm

# Logistic regression with a random intercept
```
\[ x = \text{rmlogit}(k=30, \, gi=3:5, \, ni=6:10, \, pt=c(0, 4), \, pr=c(0.7, 0.3), \, \beta=c(0, 3)) \]

\text{cnmms}(x)

data(toxo) \quad \# k = 136
\text{cnmms}(\text{mlogit(toxo)})

---

cvps

Class ‘cvps’

Description

These functions can be used to study a common variance problem (CVP), where univariate observations fall in known groups. Observations in each group are assumed to have the same mean, but different groups may have different means. All observations are assumed to have a common variance, despite their different means, hence giving the name of the problem. It is a random-effects problem.

Usage

cvps(x)
rcvps(k, ni=2, mu=0, pr=1, sd=1)
rcvps(k, ni=2, mu=0, pr=1, sd=1)
  ## S3 method for class 'cvps'
print(x, ...)

Arguments

- \( x \)  
  CVP data in the raw form as an argument in \text{cvps}, or an object of class \text{cvps} in \text{print.cvps}.
- \( k \)  
  the number of groups.
- \( ni \)  
  a numeric vector that gives the sample size in each group.
- \( mu \)  
  a numeric vector for all the theoretical means.
- \( pr \)  
  a numeric vector for all the probabilities associated with the theoretical means.
- \( sd \)  
  a scalar for the standard deviation that is common to all observations.
- ...  
  arguments passed on to function print.

Details

Class \text{cvps} is used to store the CVP data in a summarized form.

Function \text{cvps} creates an object of class \text{cvps}, given a matrix that stores the values (column 2) and their grouping information (column 1).

Function \text{rcvps} generates a random sample in the raw form for a common variance problem, where the means follow a discrete distribution.
Function `rcvps` generates a random sample in the summarized form for a common variance problem, where the means follow a discrete distribution.

Function `print.cvps` prints the CVP data given in the summarized form.

The raw form of the CVP data is a two-column matrix, where each row represents an observation. The two columns along each row give, respectively, the group membership (`group`) and the value (`x`) of an observation.

The summarized form of the CVP data is a four-column matrix, where each row represents the summarized data for all observations in a group. The four columns along each row give, respectively, the group number (`group`), the number of observations in the group (`ni`), the sample mean of the observations in the group (`mi`), and the residual sum of squares of the observations in the group (`ri`).

**Author(s)**

Yong Wang <yongwang@auckland.ac.nz>

**References**


**See Also**

`nnls`, `cnmms`.

**Examples**

```r
x = rcvps(k=50, ni=5:10, mu=c(0,4), pr=c(0.7,0.3), sd=3)
cnmms(x) # CNM-MS algorithm
cnmp1(x) # CNM-PL algorithm
cnmap(x) # CNM-AP algorithm
```

---

**disc**

*Class 'disc'*

**Description**

Class `disc` is used to represent an arbitrary univariate discrete distribution with a finite number of support points.
Usage

    disc(pt, pr=1)
    ## S3 method for class 'disc'
    print(x, ...)

Arguments

pt    a numeric vector for support points.
pr    a numeric vector for probability values at the support points.
x    an object of class disc.
...   arguments passed on to function print.

Details

Function disc creates an object of class disc, given the support points and probability values at these points.
Function print.disc prints the discrete distribution.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

See Also

cnm, cnmms.

Examples

    (d = disc(pt=c(0,4), pr=c(0.3,0.7)))

lungcancer

Lung Cancer Data

Description

Contains the data of 14 studies of the effect of smoking on lung cancer.

Format

A numeric matrix with four columns:
study: study identification code.
lungcancer: the number of people diagnosed with lung cancer.
size: the number of people in the study.
smoker: 0 for smoker, and 1 for non-smoker.
mlogit

Source

References

See Also
mlogit, cnmms.

Examples

data(lungcancer)
x = mlogit(lungcancer)
cnmms(x)

---

mlogit

*Class ‘mlogit’*

Description
These functions can be used to fit a binomial logistic regression model that has a random intercept to clustered observations. Observations in each cluster are assumed to have the same intercept, while different clusters may have different intercepts. This is a mixed-effects problem.

Usage
mlogit(x)
rmlogit(k, gi=2, ni=2, pt=0, pr=1, beta=1, X)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>a numeric matrix with four or more columns that stores clustered data.</td>
</tr>
<tr>
<td>k</td>
<td>the number of groups or clusters.</td>
</tr>
<tr>
<td>gi</td>
<td>a numeric vector that gives the sample size in each group.</td>
</tr>
<tr>
<td>ni</td>
<td>a numeric vector for the number of Bernoulli trials for each observation.</td>
</tr>
<tr>
<td>pt</td>
<td>a numeric vector for all the support points.</td>
</tr>
<tr>
<td>pr</td>
<td>a numeric vector for all the probabilities associated with the support points.</td>
</tr>
</tbody>
</table>
beta 

a numeric vector for the fixed coefficients of the covariates of the observation.

X 

the numeric matrix as the design matrix. If missing, a random matrix is created from a normal distribution.

Details

Class mlogit is used to store data for fitting the binomial logistic regression model with a random intercept.

Function mlogit creates an object of class mlogit, given a matrix with four or more columns that stores, respectively, the group/cluster membership (column 1), the number of ones or successes in the Bernoulli trials (column 2), the number of the Bernoulli trials (column 3), and the covariates (columns 4+).

Function rmlogit generates a random sample that is saved as an object of class mlogit.

An object of class mlogit contains a matrix with four or more columns, that stores, respectively, the group/cluster membership (column 1), the number of ones or successes in the Bernoulli trials (column 2), the number of the Bernoulli trials (column 3), and the covariates (columns 4+).

It also has two additional attributes that facilitate the computing by function cnmms. The first attribute is ui, which stores the unique values of group memberships, and the second is gi, the number of observations in each unique group.

It is convenient to use function mlogit to create an object of class mlogit.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


See Also

nnls, cnmms.

Examples

```r
x = rmlogit(k=30, gi=3:5, ni=6:10, pt=c(0,4), pr=c(0.7,0.3),
          beta=c(0,3))
cnmms(x)

### Real-world data
# Random intercept logistic model
data(toxo)
cnmms(mlogit(toxo))
```
Class `npnorm` can be used to store data that will be processed as those of a nonparametric normal mixture. There are several functions associated with the class.

### Usage

```r
data(betablockers)
cnmms(mlogit(betablockers))

data(lungcancer)
cnmms(mlogit(lungcancer))
```

```r
npnorm(v, w=1)
rnpnorm(n, mix=disc(0), sd=1)
## S3 method for class 'npnorm'
plot(x, mix, beta, breaks=NULL, col="red", len=100,
    add=FALSE, border.col=NULL, border.lwd=1,
    fill="lightgrey", main, lwd=2, lty=1, xlab="Data",
    ylab="Density", components=c("proportions","curves","null"),
    lty.components=2, lwd.components=2, ...)
```

### Arguments

- **v**: a numeric vector that stores the values of a sample.
- **w**: a numeric vector that stores the corresponding weights/frequencies of the observations.
- **n**: the sample size.
- **mix**: an object of class `disc`, for a discrete distribution.
- **beta**: the structural parameter.
- **sd**: a scalar for the component standard deviation that is common to all components.
- **x**: an object of class `npnorm`.
- **breaks**: the rough number bins used for plotting the histogram.
- **col**: the color of the density curve to be plotted.
- **len**: the number of points roughly used to plot the density curve over the interval of length 8 times the component standard deviation around each component mean.
- **add**: if FALSE, creates a new plot; if TRUE, adds the plot to the existing one.
- **border.col**: color for the border of histogram boxes.
- **border.lwd**: line width for the border of histogram boxes.
fill
color to fill in the histogram boxes.

components
if proportions (default), also show the support points and mixing proportions;
if curves, also show the component density curves; if null, components are not shown.

lty.components, lwd.components
line type and width for the component curves.

main, lwd, lty, xlab, ylab
arguments for graphical parameters (see par).

... arguments passed on to function plot.

Details

Function npnorm creates an object of class npnorm, given values and weights/frequencies.

Function rnpnorm generates a random sample from a normal mixture and saves the data as an object of class npnorm.

Function plot.npnorm plots the normal mixture.

When components="proportions", the component means are shown on the horizontal line of density 0. The vertical lines going upwardly at the support points are proportional to the mixing proportions at these points.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


See Also

nnls, cnm, cnmms, plot.nspmix.

Examples

mix = disc(pt=c(0,4), pr=c(0.3,0.7)) # a discrete distribution
x = rnpnorm(200, mix, sd=1)
plot(x, mix, beta=1)
**Description**

Class `nppois` is used to store data that will be processed as those of a nonparametric Poisson mixture.

**Usage**

```r
nppois(v, w=1)
rnppois(n, mix=disc(1))
## S3 method for class 'nppois'
plot(x, mix, beta, col="red", add=FALSE,
    components=TRUE, main="nppois", lwd=1, lty=1, xlab="Data",
    ylab="Density", ...)
```

**Arguments**

- `v`: a numeric vector that stores the values of a sample.
- `w`: a numeric vector that stores the corresponding weights/frequencies of the observations.
- `n`: the sample size.
- `x`: an object of class `nppois`.
- `mix`: an object of class `disc`.
- `beta`: the structural parameter, which is not really needed for the Poisson mixture.
- `col`: the color of the density curve to be plotted.
- `add`: if `FALSE`, creates a new plot; if `TRUE`, adds the plot to the existing one.
- `components`: if `TRUE`, also show the support points and mixing proportions.
- `main`, `lwd`, `lty`, `xlab`, `ylab`: arguments for graphical parameters (see `par`).
- `...`: arguments passed on to function `plot`.

**Details**

Function `nppois` creates an object of class `nppois`, given values and weights/frequencies.

Function `rnppois` generates a random sample from a Poisson mixture and saves the data as an object of class `nppois`.

Function `plot.nppois` plots the Poisson mixture.

When `components=TRUE`, the support points are shown on the horizontal line of density 0. The component density curves, weighted appropriately, are also shown.
Author(s)
Yong Wang <yongwang@auckland.ac.nz>

References

See Also
nnls, cnm, cnmms, plot.nspmix.

Examples

```r
mix = disc(pt=c(1,4), pr=c(0.3,0.7))
x = rnpois(200, mix)
plot(x, mix)
```

plot.nspmix

Class 'nspmix'

Description
Class nspmix is an object returned by function cnm, cnmms, cnmpl or cnmap.

Usage

```r
## S3 method for class 'nspmix'
plot(x, data, type=c("probability","gradient"), ...)
```

Arguments

- `x` an object of a mixture model class
- `data` a data set from the mixture model
- `type` the type of function to be plotted: the probability model of the mixture family (probability), or the gradient function (gradient).
- `...` arguments passed on to the plot function called.

Details
Function plot.nspmix plots either the mixture model, if the family of the mixture provides an implementation of the generic plot function, or the gradient function. data must belong to a mixture family, as specified by its class.
plotgrad

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


See Also

nnls, cnm, cnmms, cnmpl, cnmap, npnorm, nppois.

Examples

```r
## Poisson mixture
x = rnppois(200, disc(c(1,4), c(0.7,0.3)))
r = cnm(x)
plot(r, x, "p")
plot(r, x, "g")

## Normal mixture
x = rnpnorm(200, mix=disc(c(0,4), c(0.3,0.7)), sd=1)
r = cnm(x, init=list(beta=0.5)) # sd = 0.5
plot(r, x, "p")
plot(r, x, "g")
```

Description

Function plotgrad plots the gradient function or its first derivative of a nonparametric mixture.

Usage

```r
plotgrad(
  x,
  mix,
  beta,
  len = 500,
  order = 0,
  col = "blue",
  col2 = "red",
  add = FALSE,
)```
main = paste0("Class: ", class(x)),
xlab = expression(theta),
ylab = paste0("Gradient (order = ", order, ")"),
cex = 1,
pch = 1,
lwd = 1,
xlim,
ylim,
...)
)

Arguments

x a data object of a mixture model class.

mix an object of class 'disc', for a discrete mixing distribution.

beta the structural parameter.

ten number of points used to plot the smooth curve.

order the order of the derivative of the gradient function to be plotted. If 0, it is the
gradient function itself.

col color for the curve.

col2 color for the support points.

add if FALSE, create a new plot; if TRUE, add the curve and points to the current one.

main, xlab, ylab, cex, pch, lwd, xlim, ylim
arguments for graphical parameters (see par).

... arguments passed on to function plot.

Details

data must belong to a mixture family, as specified by its class.
The support points are shown on the horizontal line of gradient 0. The vertical lines going down-
wards at the support points are proportional to the mixing proportions at these points.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


Statistics and Computing, 20, 75-86

See Also

plot.nspmix, nnls, cnm, cnmms, npnorm, npois.
Examples

```r
## Poisson mixture
x = rnppois(200, disc(c(1,4), c(0.7,0.3)))
r = cnm(x)
plotgrad(x, r$mix)
```

```r
## Normal mixture
x = rnpnorm(200, disc(c(0,4), c(0.3,0.7)), sd=1)
r = cnm(x, init=list(beta=0.5))  # sd = 0.5
plotgrad(x, r$mix, r$beta)
```

---

### Description

Contains the results of a cohort study in north-east Thailand in which 602 preschool children participated. For each child, the number of illness spells $x$, such as fever, cough or running nose, is recorded for all 2-week periods from June 1982 to September 1985. The frequency for each value of $x$ is saved in the data set.

### Format

A data frame with 24 rows and 2 variables:

- **x**: values of $x$.
- **freq**: frequencies for each value of $x$.

### Source


### References


### See Also

- `nppois`, `cnm`
Examples

```r
data(thai)
x = nppois(thai)
plot(cnmm(x), x)
```

---

**Toxo**

**Toxoplasmosis Data**

Contains the number of subjects testing positively for toxoplasmosis in 34 cities of El Salvador, with various rainfalls.

**Format**

A numeric matrix with four columns:
- `city`: city identification code.
- `y`: the number of subjects testing positively for toxoplasmosis.
- `n`: the number of subjects tested.
- `rainfall`: the annual rainfall of the city, in meters.

**References**


**See Also**

`mlogit`, `cnmms`.

**Examples**

```r
data(toxo)
x = mlogit(toxo)
cnmms(x)
```
whist

Weighted Histograms

Description

Plots or computes the histogram with observations with multiplicities/weights.

Usage

whist(
  x,
  w = 1,
  breaks = "Sturges",
  plot = TRUE,
  freq = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "Data",
  ylab = NULL,
  main = NULL,
  add = FALSE,
  col = NULL,
  border = NULL,
  lwd = 1,
  ...
)

Arguments

x a vector of values for which the histogram is desired.
w a vector of multiplicities/weights for the values in x.
breaks, plot, freq, xlim, ylim, xlab, ylab, main, add, col, border, lwd
These arguments have similar functionalities to their namesakes in function hist.
...
arguments passed on to function plot.

Details

Just like hist, whist can either plot the histogram or compute the values that define the histogram, by setting plot to TRUE or FALSE.
The histogram can either be the one for frequencies or density, by setting freq to TRUE or FALSE.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>
See Also

hist.
Index

* class
  cvps, 10
  disc, 11
  mlogit, 13
  npnorm, 15
  nppois, 17

* datasets
  betablockers, 2
  brca, 3
  lungcancer, 12
  thai, 21
  toxo, 22

* function
  cnm, 3
  cnmms, 6
  cvps, 10
  disc, 11
  mlogit, 13
  npnorm, 15
  nppois, 17
  plot.nspmix, 18
  plotgrad, 19
  whist, 23

betablockers, 2
brca, 3

  cnm, 3, 9, 12, 16, 18–21
  cnmap, 19
  cnmap (cnmms), 6
  cnmms, 2, 6, 6, 11–14, 16, 18–20, 22
  cnmpl, 19
  cnmpl (cnmms), 6
  cvp, 9
  cvp (cvps), 10
  cvps, 9, 10

  disc, 11

hist, 24

lungcancer, 12

mlogit, 2, 9, 13, 13, 22

nnls, 6, 9, 11, 14, 16, 18–20
npnorm, 3, 6, 15, 19, 20
nppois, 6, 17, 19–21
nspmix (plot.nspmix), 18

plot.npnorm (npnorm), 15
plot.nppois (nppois), 17
plot.nspmixin, 16, 18, 18, 20
plotgrad, 19
print.cvps (cvps), 10
print.disc (disc), 11

rcvps (cvps), 10
rcvp (cvps), 10
rmlogit (mlogit), 13
rnnlogit (mlogit), 13
rnpnorm (npnorm), 15
rnppois (nppois), 17

thai, 21

 toxo, 22

whist, 23