Package ‘nspmix’

July 22, 2025

Title Nonparametric and Semiparametric Mixture Estimation

Version 2.0-0

Date 2025-05-10

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) <doi:10.1111/j.1467-9868.2007.00583.x> and Wang (2010) <doi:10.1007/s11222-009-9117-z>.

Encoding UTF-8

License GPL (>= 2)

URL https://www.stat.auckland.ac.nz/~yongwang/

RoxygenNote 7.3.2

NeedsCompilation no

Author Yong Wang [aut, cre]

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

Repository CRAN

Date/Publication 2025-05-10 18:40:02 UTC

Contents

`betablockers` *Beta-blockers Data*

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

Aitkin, M. (1999). A general maximum likelihood analysis of variance components in generalized linear models. *Biometrics*, **55**, 117-128.

References

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86.

See Also

mlogit,cnmms.

Examples

```
data(betablockers)
x = mlogit(betablockers)
cnmms(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

Efron, B. (2004). Large-scale simultaneous hypothesis testing: the choice of a null hypothesis. *Journal of the American Statistical Association*, **99**, 96-104.

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. and C.-S. Chee (2012). Density estimation using nonparametric and semiparametric mixtures. *Statistical Modelling: An International Journal*, **12**, 67-92.

See Also

npnorm,cnm.

Examples

```
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.

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, theta)`

`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`.

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).

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

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86

Wang, Y. and Taylor, S. M. (2013). Efficient computation of nonparametric survival functions via a hierarchical mixture formulation. *Statistics and Computing*, **23**, 713-725.

See Also

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

Examples

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

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

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

data(brca)
plot(cnm(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.

Function `cnmms` can also be used to compute the maximum likelihood estimate of a finite or nonparametric 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 $pi_j \ge 0$ 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 β is the structural parameter.

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

Functions `cnmms`, `cnmpl` and `cnmap` 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`.

Usage

```
cnmms(x, init=NULL, maxit=1000, model=c("spmle","npmle"), tol=1e-6,
      grid=100, kmax=Inf, plot=c("null", "gradient", "probability"),
      verbose=0)
cnmpl(x, init=NULL, tol=1e-6, tol.npmle=tol*1e-4, grid=100, maxit=1000,
      plot=c("null", "gradient", "probability"), verbose=0)
cnmap(x, init=NULL, maxit=1000, tol=1e-6, grid=100, plot=c("null",
      "gradient"), verbose=0)
```

Arguments

`x`	a data object of some class that can be defined fully by the user
`init`	list of user-provided initial values for the mixing distribution `mix` and the structural parameter `beta`
`maxit`	maximum number of iterations
`model`	the type of model that is to estimated: non-parametric MLE (`npmle`) or semi-parametric MLE (`spmle`).
`tol`	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 `-log10(tol)` is roughly the number of accurate digits in log-likelihood.
`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, an individual `gridpoints` function may return a different number of grid points than specified by `grid`.
`kmax`	upper bound on the number of support points. This is particularly useful for fitting a finite mixture model.
`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 of 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).

tol.npmle — a tolerance value that is used to terminate the computing of the NPMLE internally.

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

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86

See Also

`nnls`, `cnm`, `cvp`, `cvps`, `mlogit`.

Examples

```
## 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
cnmpl(x)              # CNM-PL algorithm
```

```
cnmap(x)             # CNM-AP algorithm

# Logistic regression with a random intercept
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)

data(toxo)           # k = 136
cnmms(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)
rcvp(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 `cvps`, or an object of class `cvps` in `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 cvps is used to store the CVP data in a summarized form.

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

Function rcvp 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

Neyman, J. and Scott, E. L. (1948). Consistent estimates based on partially consistent observations. *Econometrica*, **16**, 1-32.

Kiefer, J. and Wolfowitz, J. (1956). Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters. *Ann. Math. Stat.*, **27**, 886-906.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86.

See Also

nnls, cnmms.

Examples

```
x = rcvps(k=50, ni=5:10, mu=c(0,4), pr=c(0.7,0.3), sd=3)
cnmms(x)              # CNM-MS algorithm
cnmpl(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, sort=TRUE, collapse=FALSE)
```

Arguments

pt	a numeric vector for support points.
pr	a numeric vector for probability values at the support points.
sort	=TRUE, by default. If TRUE, support points are sorted (in increasing order).
collapse	=TRUE, by default. If TRUE, identical support points are collapsed, with their masses aggregated.

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)))
```

dmix *Density function of a mixture distribution*

Description

Computes the density or their logarithmic values of a mixture distribution, where the component family depends on the class of x.

x must belong to a mixture family, as specified by its class.

Usage

```
dmix(x, mix, beta = NULL, log = FALSE)
```

Arguments

x	a data object of a mixture model class.
mix	a discrete distribution, as defined by class disc.
beta	the structural parameter, if any.
log	if TRUE, computes the log-values, or else just the density values.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86

See Also

cnm, cnmms, npnorm, nppois, disc,

Examples

```
## Poisson mixture
mix0 = disc(c(1,4), c(0.7,0.3))
x = rnppois(10, mix0)
dmix(x, mix0)
dmix(x, mix0, log=TRUE)

## Normal mixture
x = rnpnorm(10, mix0, sd=1)
dmix(x, mix0, 1)
dmix(x, mix0, 1, log=TRUE)
```

```
dmix(x, mix0, 0.5, log=TRUE)
```

`gridpoints` *Grid points*

Description

A generic method used to return a vector of grid points used for searching local maxima of the gradient funcion.

Usage

```
gridpoints(x, beta, grid)
```

Arguments

`x`	an object of a class for data.
`beta`	instrumental parameter in a semiparametric mixture.
`grid`	number of grid points to be generated.

Value

A numeric vector containing grid points.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

`hcnm` *Hierarchical Constrained Newton method*

Description

Function `hcnm` can be used to compute the MLE of a finite discrete mixing distribution, given the component density values of each observation. It implements the hierarchical CNM algorithm of Wang and Taylor (2013).

Usage

```
hcnm(
  D,
  p0 = NULL,
  w = 1,
  maxit = 1000,
  tol = 1e-06,
  blockpar = NULL,
  recurs.maxit = 2,
  compact = TRUE,
  depth = 1,
  verbose = 0
)
```

Arguments

`D`	A numeric matrix, each row of which stores the component density values of an observation.
`p0`	Initial mixture component proportions.
`w`	Duplicity of each row in matrix `D` (i.e., that of a corresponding observation).
`maxit`	Maximum number of iterations.
`tol`	A tolerance value to terminate the algorithm. Specifically, the algorithm is terminated, if the increase of the log-likelihood value after an iteration is less than `tol`.
`blockpar`	Block partitioning parameter. If > 1, the number of blocks is roughly `nrol(D)/blockpar`. If < 1, the number of blocks is roughly `nrol(D)^blockpar`.
`recurs.maxit`	Maximum number of iterations in recursions.
`compact`	Whether iteratively select and use a compact subset (which guarantees convergence), or not (if already done so before calling the function).
`depth`	Depth of recursion/hierarchy.
`verbose`	Verbosity level for printing intermediate results.

Value

`p`	Computed probability vector.
`convergence`	convergence code. `=0` means a success, and `=1` reaching the maximum number of iterations
`ll`	log-likelihood value at convergence
`maxgrad`	Maximum gradient value.
`numiter`	number of iterations required by the algorithm

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. and Taylor, S. M. (2013). Efficient computation of nonparametric survival functions via a hierarchical mixture formulation. *Statistics and Computing*, **23**, 713-725.

See Also

cnm, nppois, disc.

Examples

```
x = rnppois(1000, disc(0:50))    # Poisson mixture
D = outer(x$v, 0:1000/10, dpois)
(r = hcnm(D, w=x$w))
disc(0:1000/10, r$p, collapse=TRUE)

cnm(x, init=list(mix=disc(0:1000/10)), model="p")
```

---

initial — *Initialization for a nonparametric/semiparametric mixture*

Description

A generic method used to return an initialization for a nonparametric/semiparametric mixture.

Usage

```
initial(x, beta, mix, kmax)
```

Arguments

x	an object of a class for data.
beta	instrumental parameter in a semiparametric mixture.
mix	an object of class disc for the mixing distribution.
kmax	the maximum allowed number of support points used.

Value

beta	an initialized value of beta
mix	an initialised or updated object of class disc for the mixing distribution.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

initial0 *Initialisation*

Description

The functions examines whether the initial values are proper. If not, proper ones are provided, by employing the function "initial" provided by the class.

Usage

```
initial0(x, init = NULL, kmax = NULL)
```

Arguments

`x`	an object of a class for data.
`init`	a list with initial values for beta and mix (as in the output of `initial`)
`kmax`	the maximum allowed number of support points used.

Value

`beta`	an initialized value of beta
`mix`	an initialised or updated object of class `disc` for the mixing distribution.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

llex *Log-likleihood Extra Term.*

Description

Value of possibly an extra term in the log-likleihood function for the instrumental parameter beta

Usage

```
llex(x, beta, mix)
```

Arguments

`x`	an object of a class for data.
`beta`	instrumental parameter in a semiparametric mixture.
`mix`	an object of class `disc` for the mixing distribution.

Value

a scalar value

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

`llexdb` *Derivative of the log-likleihood Extra Term*

Description

Derivative of the log-likleihood extra term wrt beta

Usage

```
llexdb(x, beta, mix)
```

Arguments

`x`	an object of a class for data.
`beta`	instrumental parameter in a semiparametric mixture.
`mix`	an object of class `disc` for the mixing distribution.

Value

a scalar value

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

`logd` *Log-density and its derivative values*

Description

A generic method to compute the log-density values and possibly their first derivatives with respec to theta and beta.

Usage

```
logd(x, beta, pt, which)
```

Arguments

x	an object of a class for data.
beta	instrumental parameter in a semiparametric mixture.
pt	a vector of values for the mixing variable theta.
which	an integer vector of length 3, indicating if, respectively, the log-density values, the derivatives wrt beta and the derivatives wrt theta are to be computed and returned if being 1 (TRUE).

Value

ld	a matrix, storing the log-density values for each (x[i], beta, pt[j], or NULL if not asked for.
db	a matrix, storing the log-density derivatives wrt beta for each (x[i], beta, pt[j], or NULL if not asked for.
dt	a matrix, storing the log-density derivatives wrt theta for each (x[i], beta, pt[j], or NULL if not asked for.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

---

loglik — *Log-likelihood value of a mixture*

Description

Computes the log-likelihood value

x must belong to a mixture family, as specified by its class.

Usage

```
loglik(mix, x, beta = NULL, attr = FALSE)
```

Arguments

mix	a discrete distribution, as defined by class disc.
x	a data object of a mixture model class.
beta	the structural parameter, if any.
attr	=FALSE, by default. If TRUE, also returns attributes "dmix" and "logd"

Value

the log-likelihood value.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86

See Also

`cnm`, `cnmms`, `npnorm`, `nppois`, `disc`,

Examples

```
## Poisson mixture
mix0 = disc(c(1,4), c(0.7,0.3))
x = rnppois(10, mix0)
loglik(mix0, x)

## Normal mixture
x = rnpnorm(10, mix0, sd=2)
loglik(mix0, x, 2)
```

`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.

Source

Booth, J. G. and Hobert, J. P. (1999). Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society, Ser. B*, **61**, 265-285.

References

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86.

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

`x`	a numeric matrix with four or more columns that stores clustered data.
`k`	the number of groups or clusters.
`gi`	a numeric vector that gives the sample size in each group.
`ni`	a numeric vector for the number of Bernoulli trials for each observation.
`pt`	a numeric vector for all the support points.
`pr`	a numeric vector for all the probabilities associated with the support points.
`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 `cmmms`. 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

Kiefer, J. and Wolfowitz, J. (1956). Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters. *Ann. Math. Stat.*, **27**, 886-906.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86.

See Also

`nnls`, `cnmms`.

Examples

```
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))

data(betablockers)
cnmms(mlogit(betablockers))

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

npgeom *Class* npgeom

Description

Class npgeom is used to store data that will be processed as those of a nonparametric geometric mixture.

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

Function rnpgeom generates a random sample from a geometric mixture and saves the data as an object of class npgeom.

Function dnpgeom is the density function of a Poisson mixture.

Function pnpgeom is the distribution function of a Poisson mixture.

Usage

```
npgeom(v, w=1, grouping=FALSE)
rnpgeom(n, mix=disc(0.5))
dnpgeom(x, mix=disc(0.5), log=FALSE)
pnpgeom(x, mix=disc(0.5), lower.tail=TRUE, log.p=FALSE)
```

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.
grouping	logical, whether or not use frequencies (w) for identical values.
n	the sample size.
x	an object of class npgeom.
mix	an object of class disc.
log	=FALSE, if log-values are to be returned.
lower.tail	=FALSE, if lower.tail values are to be returned.
log.p	=FALSE, if log probability values are to be returned.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

See Also

nnls, cnm, cnmms, plot.nspmix.

Examples

```
mix = disc(pt=c(0.2,0.5), pr=c(0.3,0.7))
(x = rnpgeom(200, mix))
dnpgeom(x, mix)
pnpgeom(x, mix)
```

---

npnbinom *Class* npnbinom

Description

Class npnbinom is used to store data that will be processed as those of a nonparametric negative binomial mixture.

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

Function rnpnbinom generates a random sample from a negative binomial mixture and saves the data as an object of class npnbinom.

Usage

```
npnbinom(v, w=1, size, grouping=TRUE)
rnpnbinom(n, size, mix=disc(0.5))
dnpnbinom(x, mix=disc(0.5), size=NULL, log=FALSE)
pnpnbinom(x, mix=disc(0.5), size=NULL, lower.tail=TRUE, log.p=FALSE)
```

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.
size	number of successful trials (ignored if x is an object of class npnbinom).
grouping	logical, to use frequencies (w) for identical values
n	the sample size.
x	an object of class npnbinom, or a numeric vector (then value of size must be provided).
mix	an object of class disc.
log	=FALSE, if log-values are to be returned.
lower.tail	=FALSE, if lower.tail values are to be returned.
log.p	=FALSE, if log probability values are to be returned.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

See Also

nnls, cnm, cnmms, plot.nspmix.

Examples

```
mix = disc(pt=c(0.2,0.5), pr=c(0.3,0.7))
(x = rnpnbinom(200, size=10, mix))
dnpnbinom(x, mix, size=10)
pnpnbinom(x, mix, size=10)
```

---

npnorm *Class* npnorm

Description

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.

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 dnpnorm is the density function of a normal mixture.

Function pnpnorm is the distribution function of a normal mixture.

Usage

```
npnorm(v, w = 1)
rnpnorm(n, mix=disc(0), sd=1)
dnpnorm(x, mix=disc(0), sd=1, log=FALSE)
pnpnorm(x, mix=disc(0), sd=1, lower.tail=TRUE, log.p=FALSE)
```

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.
sd	a scalar for the component standard deviation that is common to all components.
x	a numeric vector or an object of class npnorm.
log, log.p	logical, for computing the log-values or not.
lower.tail	logical, for computing the lower tail value or not.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

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)
dnpnorm(-2:6, mix, sd=1)
pnpnorm(-2:6, mix, sd=1)
dnpnorm(npnorm(-2:6), mix, sd=1)
pnpnorm(npnorm(-2:6), mix, sd=1)
```

`nppois` *Class* `nppois`

Description

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

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 `dnppois` is the density function of a Poisson mixture.

Function `pnppois` is the distribution function of a Poisson mixture.

Usage

```
nppois(v, w=1, grouping=TRUE)
rnppois(n, mix=disc(1), ...)
dnppois(x, mix=disc(1), log=FALSE)
pnppois(x, mix=disc(1), lower.tail=TRUE, log.p=FALSE)
```

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.
`grouping`	logical, to use frequencies (w) for identical values
`n`	the sample size.
`x`	an object of class `nppois`.
`mix`	an object of class `disc`.
`log`	logical, to compute the log-values or not.
`lower.tail`	=FALSE, if lower.tail values are to be returned.
`log.p`	=FALSE, if log probability values are to be returned.
`...`	arguments passed on to function `plot`.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

See Also

`nnls`, `cnm`, `cnmms`, `plot.nspmix`.

Examples

```
mix = disc(pt=c(0,4), pr=c(0.3,0.7))  # a discrete distribution
x = rnppois(200, mix)
dnppois(0:10, mix)
pnppois(0:10, mix)
dnppois(nppois(0:10), mix)
pnppois(nppois(0:10), mix)
```

`plot.disc` *Plot a discrete distribution function*

Description

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

Function `disc` creates an object of class `disc`, given the support points and probability values at these points.

Function `plot.disc` plots the discrete distribution.

Usage

```
## S3 method for class 'disc'
plot(
  x,
  type = c("pdf", "cdf"),
  add = FALSE,
  col = 4,
  lwd = 1,
  ylim,
  xlab = "",
  ylab = "Probability",
  ...
)
```

Arguments

`x`	an object of class `disc`.
`type`	plot its pdf or cdf.
`add`	add the plot or not.
`col`	colour to be used.
`lwd`, `ylim`, `xlab`, `ylab`	graphical parameters.
`...`	arguments passed on to function `plot`.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

See Also

`disc`, `cnm`, `cnmms`.

Examples

```
plot(disc(pt=c(0,4), pr=c(0.3,0.7)))
plot(disc(rnorm(5), 1:5))
for(i in 1:5)
   plot(disc(rnorm(5), 1:5), type="cdf", add=(i>1), xlim=c(-3,3))
```

plot.npgeom *Plotting a nonparametric geometric mixture*

Description

Function plot.npgeom plots a geometric mixture, along with data.

Usage

```
## S3 method for class 'npgeom'
plot(
  x,
  mix,
  beta,
  col = "red",
  add = FALSE,
  components = TRUE,
  main = "npgeom",
  lwd = 1,
  lty = 1,
  xlab = "Data",
  ylab = "Density",
  ...
)
```

Arguments

x	an object of class npgeom.
mix	an object of class disc.
beta	the structural parameter (not used for a geometric 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 (with vertical lines in proportion).
main, lwd, lty, xlab, ylab	arguments for graphical parameters (see par).
...	arguments passed on to function plot.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

See Also

nnls, cnm, cnmms, plot.nspmix.

Examples

```
mix = disc(pt=c(0.2,0.6), pr=c(0.3,0.7))  # a discrete distribution
x = rnpgeom(200, mix)
plot(x, mix)
```

---

plot.npnbinom *Plotting a nonparametric negative binomial mixture*

Description

Function plot.npnbinom plots a negative binomial mixture, along with data.

Usage

```
## S3 method for class 'npnbinom'
plot(
  x,
  mix,
  beta,
  col = "red",
  add = FALSE,
  components = TRUE,
  main = "npnbinom",
  lwd = 1,
  lty = 1,
  xlab = "Data",
  ylab = "Density",
  ...
)
```

Arguments

x	an object of class npnbinom.
mix	an object of class disc.
beta	the structural parameter (not used for a negative binomial 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 (with vertical lines in proportion).
main, lwd, lty, xlab, ylab	arguments for graphical parameters (see par).
...	arguments passed on to function plot.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

See Also

nnls, cnm, cnmms, plot.nspmix.

Examples

```
mix = disc(pt=c(0.2,0.5), pr=c(0.3,0.7))  # a discrete distribution
x = rnpnbinom(200, 10, mix)
plot(x, mix)
```

---

plot.npnorm *Plotting a Nonparametric or Semiparametric Normal Mixture*

Description

Function `plot.npnorm` plots the normal mixture.

Usage

```
## S3 method for class 'npnorm'
plot(
  x,
  mix,
  beta,
  breaks = NULL,
  col = 2,
  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

`x`	an object of class `npnorm`.
`mix`	an object of class `disc`, for a discrete distribution.
`beta`	the structural parameter.
`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.
`main`, `lwd`, `lty`, `xlab`, `ylab`	arguments for graphical parameters (see `par`).
`components`	if `proportions` (default), also show the support points and mixing proportions (in proportional vertical lines); 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.
`...`	arguments passed on to function `plot`.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

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)
```

plot.nppois *Plotting a nonparametric Poisson mixture*

Description

Function `plot.nppois` plots a Poisson mixture, along with data.

Usage

```
## S3 method for class 'nppois'
plot(
  x,
  mix,
  beta,
  col = 2,
  add = FALSE,
  components = c("proportions", "curves", "null"),
  main = "nppois",
  lwd = 1,
  lty = 1,
  xlab = "Data",
  ylab = "Density",
  xlim = NULL,
  ...
)
```

Arguments

`x`	an object of class `nppois`.
`mix`	an object of class `disc`.
`beta`	the structural parameter (not used for a 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 `proportions` (default), also show the support points and mixing proportions (in proportional vertical lines); if `curves`, also show the component density curves; if `null`, components are not shown.
`main`, `lwd`, `lty`, `xlab`, `ylab`, `xlim`	arguments for graphical parameters (see `par`).
`...`	arguments passed on to function `barplot`.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

See Also

nnls, cnm, cnmms, plot.nspmix.

Examples

```
mix = disc(pt=c(0,4), pr=c(0.3,0.7))  # a discrete distribution
x = rnppois(200, mix)
plot(x, mix)
```

---

plot.nspmix *Plots a function for an object of class* nspmix

Description

Plots a function for the object of class nspmix, currently either using the plot function of the class or plotting the gradient curve (or its first derivative)

data must belong to a mixture family, as specified by its class.

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

Usage

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

## 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.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86

See Also

plot.nspmix, nnls, cnm, cnmms, npnorm, nppois.

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

Examples

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

## 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
plot(r, x)
plot(r, x, type="g")
plot(r, x, type="g", order=1)


## 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")
```

plotgrad *Plot the Gradient Function*

Description

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

Usage

```
plotgrad(
  x,
  mix,
  beta,
  len = 500,
  order = 0,
  col = 4,
  col2 = 2,
  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.
`len`	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 downwards at the support points are proportional to the mixing proportions at these points.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86

See Also

`plot.nspmix`, `nnls`, `cnm`, `cnmms`, `npnorm`, `nppois`.

Examples

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

## 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)
```

`print.disc` *Prints a discrete distribution function*

Description

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

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.

Usage

```
## S3 method for class 'disc'
print(x, ...)
```

Arguments

`x`	an object of class `disc`.
`...`	arguments passed on to function `print`.

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)))
```

`sort.npnorm` *Sorting of an Object of Class* `npnorm`

Description

Function `sort.npnorm` sorts an object of class `npnorm` in the order of the obsersed values.

Usage

```
## S3 method for class 'npnorm'
sort(x, decreasing = FALSE, ...)
```

Arguments

`x`	an object of class `npnorm`.
`decreasing`	logical, in the decreasing (default) or increasing order.
`...`	arguments passed to function `order`.

Examples

```
mix = disc(pt=c(0,4), pr=c(0.3,0.7))  # a discrete distribution
x = rnpnorm(20, mix, sd=1)
sort(x)
```

sort.nppois — *Sorting of an Object of Class* nppois

Description

Function `sort.nppois` sorts an object of class `nppois` in the order of the obsersed values.

Usage

```
## S3 method for class 'nppois'
sort(x, decreasing = FALSE, ...)
```

Arguments

`x`	an object of class `nppois`.
`decreasing`	logical, in the decreasing (default) or increasing order.
`...`	arguments passed to function `order`.

Examples

```
mix = disc(pt=c(0,4), pr=c(0.3,0.7))  # a discrete distribution
x = rnppois(20, mix)
sort(x)
```

suppspace — *Support space*

Description

Range of the mixing variable (theta).

Usage

```
suppspace(x, beta)
```

Arguments

`x`	an object of a class for data.
`beta`	instrumental parameter in a semiparametric mixture.

Value

A vector of length 2.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

---

thai *Illness Spells and Frequencies of Thai Preschool Children*

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

Bohning, D. (2000). *Computer-assisted Analysis of Mixtures and Applications: Meta-analysis, Disease Mapping, and Others*. Boca Raton: Chapman and Hall-CRC.

References

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution. *Journal of the Royal Statistical Society, Ser. B*, **69**, 185-198.

See Also

nppois,cnm.

Examples

```
data(thai)
x = nppois(thai)
plot(cnm(x), x)
```

`toxo` *Toxoplasmosis Data*

Description

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

Efron, B. (1986). Double exponential families and their use in generalized linear regression. *Journal of the American Statistical Association*, **81**, 709-721.

Aitkin, M. (1996). A general maximum likelihood analysis of overdispersion in generalised linear models. *Statistics and Computing*, **6**, 251-262.

Wang, Y. (2010). Maximum likelihood computation for fitting semiparametric mixture models. *Statistics and Computing*, **20**, 75-86.

See Also

`mlogit`,`cnmms`.

Examples

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

`valid` *Valid parameter values*

Description

A generic method used to return `TRUE` if the values of the paramters use for a nonparametric/semiparametric mixture are valid, or `FALSE` if otherwise.

Usage

```
valid(x, beta, theta)
```

Arguments

`x`	an object of a class for data.
`beta`	instrumental parameter in a semiparametric mixture.
`theta`	values of the mixing variable.

Value

A logical value.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

`weight` *Weights*

Description

Weights or frequencis of observations.

Usage

```
weight(x, beta)
```

Arguments

`x`	an object of a class for data.
`beta`	instrumental parameter in a semiparametric mixture.

Value

a numeric vector of the weights.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

`whist` *Weighted Histograms Plots or computes the histogram with observations with multiplicities/weights. 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`*.*

Description

Weighted Histograms

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

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`.

Usage

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

Value

`breaks`	the break points.
`counts`	weighted counts over the intervals determined by `breaks`
`density`	density values over the intervals determined by `breaks`
`mids`	midpoints of the intervals determined by `breaks`

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

See Also

`hist`.

Index