Package 'spatstat.univar'

April 16, 2024

Version 2.0-2

Date 2024-04-16

Title One-Dimensional Probability Distribution Support for the 'spatstat' Family

Maintainer Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

Depends R (>= 3.5.0), stats

Imports spatstat.utils (>= 3.0-2)

Description Estimation of one-dimensional probability distributions including kernel density estimation, weighted empirical cumulative distribution functions, Kaplan-Meier and reduced-sample estimators for right-censored data, heat kernels, kernel properties, quantiles and integration.

License GPL (>= 2)

URL http://spatstat.org/

NeedsCompilation yes

ByteCompile true

BugReports https://github.com/spatstat/spatstat.univar/issues

Author Adrian Baddeley [aut, cre, cph]
 (<https://orcid.org/0000-0001-9499-8382>),
 Tilman M. Davies [aut, ctb, cph]
  (<https://orcid.org/0000-0003-0565-1825>),
 Martin L. Hazelton [aut, ctb, cph]
  (<https://orcid.org/0000-0001-7831-725X>),
 Ege Rubak [aut, cph] (<https://orcid.org/0000-0002-6675-533X>),
 Rolf Turner [aut, cph] (<https://orcid.org/0000-0001-5521-5218>),
 Greg McSwiggan [ctb, cph]

Repository CRAN

Date/Publication 2024-04-16 14:50:05 UTC

R **topics documented:**

---

spatstat.univar-package

The spatstat.univar Package

Description

The **spatstat.univar** package belongs to the **spatstat** family of packages. It provides utilities for estimating the probability distribution of one-dimensional (real-valued) data.

Details

This package is a member of the **spatstat** family of packages. It provides utilities for estimation of the probability distribution of one-dimensional (i.e. numerical, real-valued) data. The utilities include:

kernel density estimation: including unnormalised weighted densities, and cumulative distribution functions of density estimates.

weighted distributions and weighted statistics: including weighted empirical cumulative distributions, weighted median, weighted quantiles, calculating the CDF from a density estimate

estimation for right-censored data: including Kaplan-Meier, reduced-sample and other estimators of the cumulative distribution function and hazard function from right-censored data

quantiles: including calculation of quantiles from an empirical cumulative distribution or a kernel density estimate

kernels: including calculation of the probability density, cumulative distribution function, quantiles, random generation, moments and partial moments of the standard smoothing kernels

heat kernel: calculation of the one-dimensional heat kernel in an interval

integration: Numerical integration including Stieltjes integrals and indefinite integrals.

The facilities are described in more detail below.

Kernel density estimation

The package supports fixed-bandwidth and variable-bandwidth kernel estimation of probability densities from numerical data. It provides boundary corrections for kernel estimates of densities on the positive half-line (applicable when the original observations are positive numbers) for both fixed-bandwidth and variable-bandwidth estimates.

If the observations have numerical weights associated with them, these weights will not be automatically normalised, and indeed the weights may be negative or zero. This is unlike the standard R method `density.default`.

The main functions are:

unnormdensity	extension of `density.default` allowing weights to be negative or zero.
CDF.density	cumulative distribution function from kernel density estimate

Weighted distributions and weighted statistics

Weighted versions of standard operations such as the histogram and empirical distribution function are provided:

whist	weighted histogram
ewcdf	weighted empirical cumulative distribution function
mean.ewcdf	mean of weighted ecdf
quantile.ewcdf	quantiles of weighted ecdf
weighted.median	weighted median of numeric values
weighted.quantile	weighted quantile of numeric values

Estimation for right-censored data

Facilities are provided for estimating the probability distribution of right-censored lifetimes (non-negative real random variables).

kaplan.meier	Kaplan-Meier estimator of cumulative distribution function and hazard rate, from right-censored data
reduced.sample	reduced-sample estimator of cumulative distribution function, from right-censored data

Quantiles

Facilities are provided for computing the quantiles of a probability distribution, given estimates of the probability density or the cumulative distribution function and so on.

[CDF.density](#)	cumulative distribution function from kernel density estimate
[quantile.density](#)	quantiles of kernel density estimate
[quantile.ewcdf](#)	quantiles of weighted ecdf
[quantilefun](#)	quantiles as a function
[quantilefun.ewcdf](#)	quantiles as a function
[weighted.quantile](#)	weighted quantile of numeric values
[transformquantiles](#)	transform the quantiles of a dataset

Kernels

The standard R function [density.default](#) recognises a list of smoothing kernels by name: "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" and "optcosine". For these kernels, **spatstat.univar** provides various characteristics:

[dkernel](#)	probability density of the kernel
[pkernel](#)	cumulative distribution function of the kernel
[qkernel](#)	quantiles of the kernel
[rkernel](#)	generate simulated realisations from the kernel
[kernel.factor](#)	scale factor relating bandwidth to half-width of kernel
[kernel.moment](#)	partial moment of kernel
[kernel.squint](#)	integral of squared kernel

Heat kernels

The heat kernel in an interval can be calculated.

| [hotrod](#) | calculate the heat kernel in an interval |

Integration

A few facilities are provided for calculating integrals of real functions.

[indefinteg](#)	indefinite integral
[integral.density](#)	integral of a kernel density estimate
[stieltjes](#)	Stieltjes integral

Licence

This library and its documentation are usable under the terms of the "GNU General Public License", a copy of which is distributed with the package.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Tilman Davies <Tilman.Davies@otago.ac.nz>, Martin Hazelton <Martin.Hazelton@otago.ac.nz>, Ege Rubak <rubak@math.aau.dk>, Rolf Turner <rolfturner@posteo.net> and Greg McSwiggan.

CDF   *Cumulative Distribution Function From Kernel Density Estimate*

Description

Given a kernel estimate of a probability density, compute the corresponding cumulative distribution function.

Usage

```
CDF(f, ...)

## S3 method for class 'density'
CDF(f, ..., warn = TRUE)
```

Arguments

f	Density estimate (object of class "density").
...	Ignored.
warn	Logical value indicating whether to issue a warning if the density estimate f had to be renormalised because it was computed in a restricted interval.

Details

CDF is generic, with a method for class "density".

This calculates the cumulative distribution function whose probability density has been estimated and stored in the object f. The object f must belong to the class "density", and would typically have been obtained from a call to the function [density](#).

Value

A function, which can be applied to any numeric value or vector of values.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>

See Also

[density](#), [quantile.density](#)

Examples

```
b <- density(runif(10))
f <- CDF(b)
f(0.5)
plot(f)
```

---

dkernel   *Kernel distributions and random generation*

Description

Density, distribution function, quantile function and random generation for several distributions used in kernel estimation for numerical data.

Usage

```
dkernel(x, kernel = "gaussian", mean = 0, sd = 1)
pkernel(q, kernel = "gaussian", mean = 0, sd = 1, lower.tail = TRUE)
qkernel(p, kernel = "gaussian", mean = 0, sd = 1, lower.tail = TRUE)
rkernel(n, kernel = "gaussian", mean = 0, sd = 1)
```

Arguments

x, q	Vector of quantiles.
p	Vector of probabilities.
kernel	String name of the kernel. Options are "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" and "optcosine". (Partial matching is used).
n	Number of observations.
mean	Mean of distribution.
sd	Standard deviation of distribution.
lower.tail	logical; if TRUE (the default), then probabilities are $P(X \leq x)$, otherwise, $P(X > x)$.

Details

These functions give the probability density, cumulative distribution function, quantile function and random generation for several distributions used in kernel estimation for one-dimensional (numerical) data.

The available kernels are those used in density.default, namely "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" and "optcosine". For more information about these kernels, see density.default.

dkernel gives the probability density, pkernel gives the cumulative distribution function, qkernel gives the quantile function, and rkernel generates random deviates.

Value

A numeric vector. For dkernel, a vector of the same length as x containing the corresponding values of the probability density. For pkernel, a vector of the same length as x containing the corresponding values of the cumulative distribution function. For qkernel, a vector of the same length as p containing the corresponding quantiles. For rkernel, a vector of length n containing randomly generated values.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au> and Martin Hazelton <Martin.Hazelton@otago.ac.nz>.

See Also

[density.default](), [kernel.factor](), [kernel.moment](), [kernel.squint]().

Examples

```
x <- seq(-3,3,length=100)
plot(x, dkernel(x, "epa"), type="l",
          main=c("Epanechnikov kernel", "probability density"))
plot(x, pkernel(x, "opt"), type="l",
          main=c("OptCosine kernel", "cumulative distribution function"))
p <- seq(0,1, length=256)
plot(p, qkernel(p, "biw"), type="l",
          main=c("Biweight kernel", "cumulative distribution function"))
y <- rkernel(100, "tri")
hist(y, main="Random variates from triangular density")
rug(y)
```

---

ewcdf   *Weighted Empirical Cumulative Distribution Function*

Description

Compute a weighted version of the empirical cumulative distribution function.

Usage

```
ewcdf(x, weights = NULL, normalise=TRUE, adjust=1)
```

Arguments

x	Numeric vector of observations.
weights	Optional. Numeric vector of non-negative weights for x. Defaults to equal weight 1 for each entry of x.
normalise	Logical value indicating whether the weights should be rescaled so that they sum to 1.
adjust	Numeric value. Adjustment factor. The weights will be multiplied by adjust.

Details

This is a modification of the standard function `ecdf` allowing the observations x to have weights.

The weighted e.c.d.f. (empirical cumulative distribution function) Fn is defined so that, for any real number y, the value of Fn(y) is equal to the total weight of all entries of x that are less than or equal to y. That is Fn(y) = sum(weights[x <= y]).

Thus Fn is a step function which jumps at the values of x. The height of the jump at a point y is the total weight of all entries in x number of tied observations at that value. Missing values are ignored.

If `weights` is omitted, the default is equivalent to ecdf(x) except for the class membership.

The result of ewcdf is a function, of class "ewcdf", inheriting from the classes "ecdf" (only if normalise=TRUE) and "stepfun".

The class ewcdf has methods for `print`, `quantile` and `mean`.

The inherited classes ecdf and stepfun have methods for `plot` and `summary`.

Value

A function, of class "ewcdf", inheriting from "ecdf" (if normalise=TRUE) and "stepfun".

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

See Also

`ecdf`.

`quantile.ewcdf`, `mean.ewcdf`.

Integrals with respect to the weighted cumulative distribution function can be computed using `stieltjes`.

Examples

```
x <- rnorm(100)
w <- runif(100)
plot(e <- ewcdf(x,w))
e
```

---

hotrod   *Heat Kernel for a One-Dimensional Rod*

Description

Calculate values of the heat kernel on a one-dimensional rod. The ends of the rod may be assumed to be insulated, or absorbing.

Usage

```
hotrod(len, xsource, xquery, sigma, ends=c("insulated", "absorbing"), nmax=20)
```

Arguments

len	Length of the rod. A single number or numeric vector.
xsource	Positions of the source points, from the left end of the rod (in the same distance units as len). A single number or numeric vector.
xquery	Positions of the query points, from the left end of the rod (in the same distance units as len). A single number or numeric vector.
sigma	Bandwidth for kernel. A single number or a numeric vector.
ends	Character string (partially matched) specifying whether the ends of the rod are assumed to be insulated or absorbing.
nmax	Number of terms in the infinite sum to use. A single integer or an integer vector.

Details

Computes the heat kernel as an infinite sum.

Value

Number or numeric vector.

Author(s)

Greg McSwiggan and Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

Examples

```
curve(hotrod(1, 0.1, x, 0.7))

# check it's a probability density
f <- function(x) hotrod(1, 0.1, x, 0.7)
integrate(f, 0, 1)

## absorbing ends
curve(hotrod(1, 0.1, x, 0.7, ends="a"))
```

---

indefinteg   *Indefinite Integral*

Description

Computes the indefinite integral of the given function.

Usage

```
indefinteg(f, x, ...,
          method=c("trapezoid", "quadrature"),
          lower=min(x), nfine=8192)
```

Arguments

f	an R function taking a numeric first argument and returning a numeric vector of the same length.
x	Vector of values of the argument for which the indefinite integral should be evaluated.
...	additional arguments to be passed to f.
method	String (partially matched) specifying how to compute the integrals.
lower	Lower limit of integration. A single number.
nfine	Number of sub-intervals to use for computation if method='trapezoid'.

Details

The indefinite integral of the given function f is computed numerically at each of the desired values x. The lower limit of integration is taken to be min(x).

The result is a numeric vector y of the same length as x, with entries

$$y_i = \int_{\text{lower}}^{x_i} f(t)dt$$

If method='trapezoid' (the default), the integrals are computed rapidly using the trapezoid rule. If method='quadrature' the integrals are computed accurately but much more slowly, using the numerical quadrature routine [integrate](#).

If method='trapezoid' the function f is first evaluated on a finer grid of values of the function argument. The fine grid contains nfine sample points. The values of the indefinite integral on the fine grid are computed using the trapezoidal approximation. Finally the values of the indefinite integral are extracted at the desired argument values x.

Value

Numeric vector of the same length as x.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

[integrate](#)

Examples

```
curve(indefinteg(sin, x), to=pi)
```

---

integral *Integral of a Function or Spatial Object*

Description

Computes the integral of a function or spatial object.

Usage

```
integral(f, domain=NULL, ...)
```

Arguments

f	A function, or a spatial object that can be treated as a function.
domain	Optional. Data specifying the domain of integration.
...	Arguments passed to methods.

Details

The function `integral` is generic. It calculates the integral of a function, or the integral of a spatial object that can be treated as a function. It has methods for one-dimensional functions (`"density"`, `"fv"`) and for spatial objects (`"im"`, `"msr"`, `"linim"`, `"linfun"`).

Value

A single numeric or complex value, or a vector of such values.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

See Also

[integral.density](#).

integral.im in package spatstat.geom.

---

integral.density   *Compute Integral of One-Dimensional Kernel Density Estimate.*

Description

Compute the integral of a kernel density estimate over a specified range.

Usage

```
## S3 method for class 'density'
integral(f, domain = NULL, weight=NULL, ...)
```

Arguments

f   A one-dimensional probability density estimate (object of class "density") ob-
                   tained from the function density.default or from unnormdensity.

domain   Optional. Range of values of the argument x over which the density $f(x)$ should
                   be integrated. A numeric vector of length 2 giving the minimum and maximum
                   values of x. Infinite limits are permitted.

weight   Optional. A function(x) specifying a weight integrand.

...   Ignored.

Details

This is a method for the generic function integral. It computes the numerical integral

$$I = \int f(x)dx$$

of the density estimate f. If weight is specified, then the weighted integral

$$I = \int w(x)f(x)dx$$

is computed, where w is the function specified by weight. This function must return finite numeri-
cal values.

If domain is specified, the integral is restricted to the interval of x values given by the domain.

Integrals are calculated numerically using the trapezoidal rule restricted to the domain given.

Value

A single numerical value.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

density.default

quantile.density, CDF.density in package **spatstat.explore**

Examples

```
x <- runif(10)
d <- density(x, bw=0.1)
integral(d) # should be approximately 1
integral(d, domain=c(-Inf, 0)) # mass on negative half-line
## mean of density
integral(d, weight=function(x) x)
```

---

kaplan.meier *Kaplan-Meier Estimator using Histogram Data*

Description

Compute the Kaplan-Meier estimator of a survival time distribution function, from histogram data

Usage

```
kaplan.meier(obs, nco, breaks, upperobs=0)
```

Arguments

obs	vector of n integers giving the histogram of all observations (censored or uncensored survival times)
nco	vector of n integers giving the histogram of uncensored observations (those survival times that are less than or equal to the censoring time)
breaks	Vector of $n + 1$ breakpoints which were used to form both histograms.
upperobs	Number of observations beyond the rightmost breakpoint, if any.

Details

This function is needed mainly for internal use in **spatstat**, but may be useful in other applications where you want to form the Kaplan-Meier estimator from a huge dataset.

Suppose T_i are the survival times of individuals $i = 1, \ldots, M$ with unknown distribution function $F(t)$ which we wish to estimate. Suppose these times are right-censored by random censoring times C_i. Thus the observations consist of right-censored survival times $\tilde{T}_i = \min(T_i, C_i)$ and non-censoring indicators $D_i = 1\{T_i \leq C_i\}$ for each i.

If the number of observations M is large, it is efficient to use histograms. Form the histogram obs of all observed times $\tilde{T}_i$. That is, obs[k] counts the number of values $\tilde{T}_i$ in the interval (breaks[k],breaks[k+1]] for $k > 1$ and [breaks[1],breaks[2]] for $k = 1$. Also form the histogram nco of all uncensored times, i.e. those $\tilde{T}_i$ such that $D_i = 1$. These two histograms are the arguments passed to kaplan.meier.

The vectors km and lambda returned by kaplan.meier are (histogram approximations to) the Kaplan-Meier estimator of $F(t)$ and its hazard rate $\lambda(t)$. Specifically, km[k] is an estimate of F(breaks[k+1]), and lambda[k] is an estimate of the average of $\lambda(t)$ over the interval (breaks[k],breaks[k+1]).

The histogram breaks must include 0. If the histogram breaks do not span the range of the observations, it is important to count how many survival times $\tilde{T}_i$ exceed the rightmost breakpoint, and give this as the value upperobs.

Value

A list with two elements:

km   Kaplan-Meier estimate of the survival time c.d.f. $F(t)$

lambda   corresponding Nelson-Aalen estimate of the hazard rate $\lambda(t)$

These are numeric vectors of length n.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

and Rolf Turner <rolfturner@posteo.net>

See Also

reduced.sample, km.rs

---

kernel.factor   *Scale factor for density kernel*

Description

Returns a scale factor for the kernels used in density estimation for numerical data.

Usage

```
kernel.factor(kernel = "gaussian")
```

Arguments

kernel   String name of the kernel. Options are "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" and "optcosine". (Partial matching is used).

Details

Kernel estimation of a probability density in one dimension is performed by `density.default` using a kernel function selected from the list above.

This function computes a scale constant for the kernel. For the Gaussian kernel, this constant is equal to 1. Otherwise, the constant c is such that the kernel with standard deviation 1 is supported on the interval $[-c, c]$.

For more information about these kernels, see `density.default`.

Value

A single number.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au> and Martin Hazelton <Martin.Hazelton@otago.ac.nz>.

See Also

`density.default`, `dkernel`, `kernel.moment`, `kernel.squint`

Examples

```
kernel.factor("rect")
# bandwidth for Epanechnikov kernel with half-width h=1
h <- 1
bw <- h/kernel.factor("epa")
```

---

kernel.moment	*Incomplete Moment of Smoothing Kernel*

Description

Computes the complete or incomplete mth moment of a smoothing kernel.

Usage

```
kernel.moment(m, r, kernel = "gaussian", mean=0, sd=1/kernel.factor(kernel))
```

Arguments

m	Exponent (order of moment). An integer.
r	Upper limit of integration for the incomplete moment. A numeric value or numeric vector. Set r=Inf to obtain the complete moment.
kernel	String name of the kernel. Options are "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" and "optcosine". (Partial matching is used).
mean,sd	Optional numerical values giving the mean and standard deviation of the kernel.

Details

Kernel estimation of a probability density in one dimension is performed by density.default using a kernel function selected from the list above. For more information about these kernels, see density.default.

The function kernel.moment computes the integral

$$\int_{-\infty}^{r} t^{m} k(t) dt$$

where $k(t)$ is the selected kernel, r is the upper limit of integration, and m is the exponent or order.

Note that, if mean and sd are not specified, the calculations assume that $k(t)$ is the **standard form** of the kernel, which has support $[-1, 1]$ and standard deviation $sigma = 1/c$ where c = kernel.factor(kernel).

The code uses the explicit analytic expressions when m = 0, 1, 2 and numerical integration otherwise.

Value

A single number, or a numeric vector of the same length as r.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au> and Martin Hazelton <Martin.Hazelton@otago.ac.nz>.

See Also

density.default, dkernel, kernel.factor, kernel.squint

Examples

```
kernel.moment(1, 0.1, "epa")
curve(kernel.moment(2, x, "epa"), from=-1, to=1)
```

---

kernel.squint   *Integral of Squared Kernel*

Description

Computes the integral of the squared kernel, for the kernels used in density estimation for numerical data.

Usage

```
kernel.squint(kernel = "gaussian", bw=1)
```

Arguments

kernel
: String name of the kernel. Options are "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" and "optcosine". (Partial matching is used).

bw
: Bandwidth (standard deviation) of the kernel.

Details

Kernel estimation of a probability density in one dimension is performed by density.default using a kernel function selected from the list above.

This function computes the integral of the squared kernel,

$$R = \int_{-\infty}^{\infty} k(x)^2 \, \mathrm{d}x$$

where $k(x)$ is the kernel with bandwidth bw.

Value

A single number.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au> and Martin Hazelton <Martin.Hazelton@otago.ac.nz>.

See Also

density.default, dkernel, kernel.moment, kernel.factor

Examples

```
kernel.squint("gaussian", 3)

# integral of squared Epanechnikov kernel with half-width h=1
h <- 1
bw <- h/kernel.factor("epa")
kernel.squint("epa", bw)
```

---

km.rs   *Kaplan-Meier and Reduced Sample Estimator using Histograms*

Description

Compute the Kaplan-Meier and Reduced Sample estimators of a survival time distribution function, using histogram techniques

Usage

```
km.rs(o, cc, d, breaks)
```

Arguments

o	vector of observed survival times
cc	vector of censoring times
d	vector of non-censoring indicators
breaks	Vector of breakpoints to be used to form histograms.

Details

This function is needed mainly for internal use in **spatstat**, but may be useful in other applications where you want to form the Kaplan-Meier estimator from a huge dataset.

Suppose T_i are the survival times of individuals $i = 1, \ldots, M$ with unknown distribution function $F(t)$ which we wish to estimate. Suppose these times are right-censored by random censoring times C_i. Thus the observations consist of right-censored survival times $\tilde{T}_i = \min(T_i, C_i)$ and non-censoring indicators $D_i = 1\{T_i \leq C_i\}$ for each i.

The arguments to this function are vectors o, cc, d of observed values of $\tilde{T}_i$, C_i and D_i respectively. The function computes histograms and forms the reduced-sample and Kaplan-Meier estimates of $F(t)$ by invoking the functions kaplan.meier and reduced.sample. This is efficient if the lengths of o, cc, d (i.e. the number of observations) is large.

The vectors km and hazard returned by kaplan.meier are (histogram approximations to) the Kaplan-Meier estimator of $F(t)$ and its hazard rate $\lambda(t)$. Specifically, km[k] is an estimate of F(breaks[k+1]), and lambda[k] is an estimate of the average of $\lambda(t)$ over the interval (breaks[k],breaks[k+1]). This approximation is exact only if the survival times are discrete and the histogram breaks are fine enough to ensure that each interval (breaks[k],breaks[k+1]) contains only one possible value of the survival time.

The vector rs is the reduced-sample estimator, rs[k] being the reduced sample estimate of F(breaks[k+1]). This value is exact, i.e. the use of histograms does not introduce any approximation error in the reduced-sample estimator.

Value

A list with five elements

rs	Reduced-sample estimate of the survival time c.d.f. $F(t)$
km	Kaplan-Meier estimate of the survival time c.d.f. $F(t)$
hazard	corresponding Nelson-Aalen estimate of the hazard rate $\lambda(t)$
r	values of t for which $F(t)$ is estimated
breaks	the breakpoints vector

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

and Rolf Turner <rolfturner@posteo.net>

See Also

reduced.sample, kaplan.meier

mean.ewcdf *Mean of Empirical Cumulative Distribution Function*

Description

Calculates the mean of a (weighted or unweighted) empirical cumulative distribution function.

Usage

```
## S3 method for class 'ecdf'
mean(x, trim=0, ...)

## S3 method for class 'ewcdf'
mean(x, trim=0, ...)
```

Arguments

x   An empirical cumulative distribution function (object of class "ecdf" created by
         ecdf) or a weighted empirical cumulative distribution function (object of class
         "ewcdf" created by ewcdf).

trim   The fraction (0 to 0.5) of data values to be trimmed from each end of their range,
         before the mean is computed.

...   Ignored.

Details

These functions are methods for the generic mean for the classes "ecdf" and "ewcdf".

They calculate the mean of the probability distribution corresponding to the cumulative distribution
function x. This is equivalent to calculating the (weighted or unweighted) mean of the original data
values.

For *weighted* empirical cumulative distribution functions (class "ewcdf") the weights will first be
normalised so that they sum to 1. The result of mean.ewcdf is always an average or weighted
average or the original data values. The argument trim is interpreted as a probability under this
normalised distribution; the corresponding quantiles are computed, and data outside these quantiles
is deleted before calculating the weighted mean.

Value

A single number.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net>
and Ege Rubak <rubak@math.aau.dk>.

See Also

Generic mean and weighted.mean.

ecdf, ewcdf to create the cumulative distribution functions.

stieltjes for integration with respect to a cumulative distribution function.

Examples

```
x <- 1:5
mean(x)
mean(ecdf(x))
w <- 1:5
mean(ewcdf(x, w))
```

---

quantile.density   *Quantiles of a Density Estimate*

Description

Given a kernel estimate of a probability density, compute quantiles.

Usage

```
## S3 method for class 'density'
quantile(x, probs = seq(0, 1, 0.25), names = TRUE,
         ..., warn = TRUE)
```

Arguments

x	Object of class "density" computed by a method for density
probs	Numeric vector of probabilities for which the quantiles are required.
names	Logical value indicating whether to attach names (based on probs) to the result.
...	Ignored.
warn	Logical value indicating whether to issue a warning if the density estimate x had to be renormalised because it was computed in a restricted interval.

Details

This function calculates quantiles of the probability distribution whose probability density has been estimated and stored in the object x. The object x must belong to the class "density", and would typically have been obtained from a call to the function density.

The probability density is first normalised so that the total probability is equal to 1. A warning is issued if the density estimate was restricted to an interval (i.e. if x was created by a call to density which included either of the arguments from and to).

Next, the density estimate is numerically integrated to obtain an estimate of the cumulative distribution function $F(x)$. Then for each desired probability p, the algorithm finds the corresponding quantile q.

The quantile q corresponding to probability p satisfies $F(q) = p$ up to the resolution of the grid of values contained in x. The quantile is computed from the right, that is, q is the smallest available value of x such that $F(x) \geq p$.

Value

A numeric vector containing the quantiles.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

See Also

quantile, quantile.ewcdf,

CDF.

Examples

```
dd <- density(runif(10))
quantile(dd)
```

---

quantile.ewcdf   *Quantiles of Weighted Empirical Cumulative Distribution Function*

Description

Compute quantiles of a weighted empirical cumulative distribution function.

Usage

```
  ## S3 method for class 'ewcdf'
quantile(x, probs = seq(0, 1, 0.25),
                 names = TRUE, ...,
                 normalise = TRUE, type=1)
```

Arguments

x	A weighted empirical cumulative distribution function (object of class "ewcdf", produced by ewcdf) for which the quantiles are desired.
probs	probabilities for which the quantiles are desired. A numeric vector of values between 0 and 1.

names	Logical. If TRUE, the resulting vector of quantiles is annotated with names corresponding to probs.
...	Ignored.
normalise	Logical value indicating whether x should first be normalised so that it ranges between 0 and 1.
type	Integer specifying the type of quantile to be calculated, as explained in [quantile.default](). Only types 1 and 2 are currently implemented.

Details

This is a method for the generic [quantile]() function for the class ewcdf of empirical weighted cumulative distribution functions.

The quantile for a probability p is computed as the right-continuous inverse of the cumulative distribution function x (assuming type=1, the default).

If normalise=TRUE (the default), the weighted cumulative function x is first normalised to have total mass 1 so that it can be interpreted as a cumulative probability distribution function.

Value

Numeric vector of quantiles, of the same length as probs.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk> and Kevin Ummel.

See Also

[ewcdf](), [quantile]()

Examples

```
z <- rnorm(50)
w <- runif(50)
Fun <- ewcdf(z, w)
quantile(Fun, c(0.95,0.99))
```

---

quantilefun	*Quantile Function*

Description

Return the inverse function of a cumulative distribution function.

Usage

```
   quantilefun(x, ...)

   ## S3 method for class 'ecdf'
quantilefun(x, ..., type=1)

   ## S3 method for class 'ewcdf'
quantilefun(x, ..., type=1)
```

Arguments

x   Data for which the quantile function should be calculated. Either an object con-
            taining data (such as a pixel image) or an object representing a cumulative dis-
            tribution function (of class "ecdf" or "ewcdf").

...   Other arguments passed to methods.

type   Integer specifying the type of quantiles, as explained in [quantile.default]().
            Only types 1 and 2 are currently implemented.

Details

Whereas the command [quantile]() calculates the quantiles of a dataset corresponding to desired
probabilities p, the command quantilefun returns a function which can be used to compute any
quantiles of the dataset.

If f <- quantilefun(x) then f is a function such that f(p) is the quantile associated with any
given probability p. For example f(0.5) is the median of the original data, and f(0.99) is the 99th
percentile of the original data.

If x is a pixel image (object of class "im") then the pixel values of x will be extracted and the
quantile function of the pixel values is constructed.

If x is an object representing a cumulative distribution function (object of class "ecdf" or "ewcdf")
then the quantile function of the original data is constructed.

Value

A function in the R language.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net>
and Ege Rubak <rubak@math.aau.dk>.

See Also

[ewcdf](), [quantile.ewcdf](), [ecdf](), [quantile]()

Examples

```
## numeric data
z <- rnorm(50)
FZ <- ecdf(z)
QZ <- quantilefun(FZ)
QZ(0.5) # median value of z
if(interactive()) plot(QZ,xlim=c(0,1),xlab="probability",ylab="quantile of z")
```

---

reduced.sample	*Reduced Sample Estimator using Histogram Data*

Description

Compute the Reduced Sample estimator of a survival time distribution function, from histogram data

Usage

```
reduced.sample(nco, cen, ncc, show=FALSE, uppercen=0)
```

Arguments

nco   vector of counts giving the histogram of uncensored observations (those survival times that are less than or equal to the censoring time)

cen   vector of counts giving the histogram of censoring times

ncc   vector of counts giving the histogram of censoring times for the uncensored observations only

uppercen   number of censoring times greater than the rightmost histogram breakpoint (if there are any)

show   Logical value controlling the amount of detail returned by the function value (see below)

Details

This function is needed mainly for internal use in **spatstat**, but may be useful in other applications where you want to form the reduced sample estimator from a huge dataset.

Suppose T_i are the survival times of individuals $i = 1, \ldots, M$ with unknown distribution function $F(t)$ which we wish to estimate. Suppose these times are right-censored by random censoring times C_i. Thus the observations consist of right-censored survival times $\tilde{T}_i = \min(T_i, C_i)$ and non-censoring indicators $D_i = 1\{T_i \leq C_i\}$ for each i.

If the number of observations M is large, it is efficient to use histograms. Form the histogram cen of all censoring times C_i. That is, obs[k] counts the number of values C_i in the interval (breaks[k],breaks[k+1]] for $k > 1$ and [breaks[1],breaks[2]] for $k = 1$. Also form the histogram nco of all uncensored times, i.e. those $\tilde{T}_i$ such that $D_i = 1$, and the histogram of all censoring times for which the survival time is uncensored, i.e. those C_i such that $D_i = 1$. These three histograms are the arguments passed to kaplan.meier.

The return value `rs` is the reduced-sample estimator of the distribution function $F(t)$. Specifically, `rs[k]` is the reduced sample estimate of `F(breaks[k+1])`. The value is exact, i.e. the use of histograms does not introduce any approximation error.

Note that, for the results to be valid, either the histogram breaks must span the censoring times, or the number of censoring times that do not fall in a histogram cell must have been counted in `uppercen`.

Value

If `show = FALSE`, a numeric vector giving the values of the reduced sample estimator. If `show=TRUE`, a list with three components which are vectors of equal length,

`rs`	Reduced sample estimate of the survival time c.d.f. $F(t)$
`numerator`	numerator of the reduced sample estimator
`denominator`	denominator of the reduced sample estimator

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

and Rolf Turner <rolfturner@posteo.net>

See Also

[kaplan.meier](), [km.rs]()

---

stieltjes	*Compute Integral of Function Against Cumulative Distribution*

Description

Computes the Stieltjes integral of a function f with respect to a function M.

Usage

```
stieltjes(f, M, ...)
```

Arguments

`f`	The integrand. A function in the R language.
`M`	The cumulative function against which `f` will be integrated. An object of class `"fv"` or `"stepfun"`.
`...`	Additional arguments passed to `f`.

Details

This command computes the Stieltjes integral

$$I = \int f(x) dM(x)$$

of a real-valued function $f(x)$ with respect to a nondecreasing function $M(x)$.

One common use of the Stieltjes integral is to find the mean value of a random variable from its cumulative distribution function $F(x)$. The mean value is the Stieltjes integral of $f(x) = x$ with respect to $F(x)$.

The argument f should be a `function` in the R language. It should accept a numeric vector argument x and should return a numeric vector of the same length.

The argument M should be either a step function (object of class "stepfun") or a function value table (object of class "fv"). Objects of class "stepfun" are returned by `ecdf`, `ewcdf`,

and other utilities.

Value

A list containing the value of the Stieltjes integral computed using each of the versions of the function M.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

Examples

```
x <- runif(100)
w <- runif(100)
H <- ewcdf(x, w)
stieltjes(function(x) { x^2 }, H)
```

---

transformquantiles   *Transform the Quantiles*

Description

Apply a transformation to the quantiles of a vector, or to the quantiles of the pixel values in a pixel image.

Usage

```
transformquantiles(X, uniform = FALSE, reverse = FALSE, ...)
```

Arguments

X	A numeric vector, matrix, array, or a pixel image (object of class "im").
uniform	Logical value specifying whether each quantile value should be replaced by the corresponding cumulative probability (called *histogram equalisation*, *transformation to uniformity* or *probability integral transformation*).
reverse	Logical value specifying whether to swap the upper and lower quantiles.
...	Ignored.

Details

The argument X may be a vector, matrix, array, or a pixel image (object of class "im").

The algorithm will first extract the entries or pixel values of X as a vector, and sort the values into ascending order.

If uniform=TRUE, the entries in this vector will be replaced by the corresponding cumulative probabilities (the kth smallest value will be replaced by the number (k-0.5)/n where n is the total number of values).

If reverse=TRUE, the resulting vector will be reversed so that it is in descending order (so that the kth smallest value will be swapped with the kth largest value).

Finally the transformed values will be replaced into the original positions in the vector, matrix, array, or pixel image.

The case uniform=TRUE, reverse=FALSE is called *transformation to uniformity*, the *probability integral transformation*, *histogram equalisation*, or *quantile transformation*. The resulting values are uniformly distributed between 0 and 1; a histogram of the values in X is flat.

Value

Another object of the same type as X.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

See Also

To apply an arbitrary function f to the pixel values in an image, use the idiom X[] <- f(X[]).

Examples

```
X <- c(3, 5, 1, 2, 4)
transformquantiles(X, reverse=TRUE)
transformquantiles(X, uniform=TRUE)
transformquantiles(X, uniform=TRUE, reverse=TRUE)
```

---

unnormdensity   *Weighted kernel smoother*

Description

An unnormalised version of kernel density estimation where the weights are not required to sum to 1. The weights may be positive, negative or zero.

Usage

```
unnormdensity(x, ..., weights = NULL, defaults)
```

Arguments

x	Numeric vector of data
...	Optional arguments passed to [density.default](). Arguments must be *named*.
'	
weights	Optional numeric vector of weights for the data. The default is equivalent to assuming a weight of 1 for each observation.
defaults	Optional, named list of arguments passed to [density.default](). These will be overridden by arguments in . . . .

Details

This is an alternative to the standard R kernel density estimation function `density.default`.

The standard `density.default` requires the `weights` to be nonnegative numbers that add up to 1, and returns a probability density (a function that integrates to 1).

This function `unnormdensity` does not impose any requirement on the `weights` except that they be finite. Individual weights may be positive, negative or zero. The result is a function that does not necessarily integrate to 1 and may be negative. The result is the convolution of the kernel k with the weighted data,

$$f(x) = \sum_i w_i k(x - x_i)$$

where x_i are the data points and w_i are the weights.

The argument `weights` should be a numeric vector of the same length as `x`, or a single numeric value. The default is to assume a weight of 1 for each observation in `x`.

The algorithm first selects the kernel bandwidth by applying `density.default` to the data `x` with normalised, positive weight vector `w = abs(weights)/sum(abs(weights))` and extracting the selected bandwidth. Then the result is computed by applying applying `density.default` to `x` twice using the normalised positive and negative parts of the weights.

Note that the arguments `...` must be passed by name, i.e. in the form (`name=value`). Arguments that do not match an argument of `density.default` will be ignored *silently*.

Value

Object of class `"density"` as described in `density.default`.

Warning

If `weights` is not specified, the default is to assign a weight $w_i = 1$ to each observation x_i.

This is not the same behaviour as in `density.default` which effectively assumes a weight of $1/n$ for each observation x_i where `n=length(x)`.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au> and Rolf Turner <rolfturner@posteo.net>

See Also

`density.default`

Examples

```
d <- unnormdensity(1:3, weights=c(-1,0,1), bw=0.3)
if(interactive()) plot(d)
```

weighted.median	*Weighted Median, Quantiles or Variance*

Description

Compute the median, quantiles or variance of a set of numbers which have weights associated with them.

Usage

```
weighted.median(x, w, na.rm = TRUE, type=2, collapse=TRUE)

weighted.quantile(x, w, probs=seq(0,1,0.25), na.rm = TRUE, type=4, collapse=TRUE)

weighted.var(x, w, na.rm = TRUE)
```

Arguments

x	Data values. A vector of numeric values, for which the median or quantiles are required.
w	Weights. A vector of nonnegative numbers, of the same length as x.
probs	Probabilities for which the quantiles should be computed. A numeric vector of values between 0 and 1.
na.rm	Logical. Whether to ignore NA values.
type	Integer specifying the rule for calculating the median or quantile, corresponding to the rules available for [quantile](). The only valid choices are type=1, 2 or 4. See Details.
collapse	Research use only.

Details

The ith observation x[i] is treated as having a weight proportional to w[i].

The weighted median is a value m such that the total weight of data less than or equal to m is equal to half the total weight. More generally, the weighted quantile with probability p is a value q such that the total weight of data less than or equal to q is equal to p times the total weight.

If there is no such value, then

- if type=1, the next largest value is returned (this is the right-continuous inverse of the left-continuous cumulative distribution function);
- if type=2, the average of the two surrounding values is returned (the average of the right-continuous and left-continuous inverses);
- if type=4, linear interpolation is performed.

Note that the default rule for weighted.median is type=2, consistent with the traditional definition of the median, while the default for weighted.quantile is type=4.

Value

A numeric value or vector.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

[quantile](), [median]().

Examples

```
x <- 1:20
w <- runif(20)
weighted.median(x, w)
weighted.quantile(x, w)
weighted.var(x, w)
```

---

whist   *Weighted Histogram*

Description

Computes the weighted histogram of a set of observations with a given set of weights.

Usage

```
whist(x, breaks, weights = NULL, method=c("C", "interpreted"))
```

Arguments

x	Numeric vector of observed values.
breaks	Vector of breakpoints for the histogram.
weights	Numeric vector of weights for the observed values.
method	Developer use only. A character string specifying whether to use internal C code (method="C", the default) or interpreted R code (method="interpreted").

Details

This low-level function computes (but does not plot) the weighted histogram of a vector of observations x using a given vector of weights.

The arguments x and weights should be numeric vectors of equal length. They may include NA or infinite values.

The argument breaks should be a numeric vector whose entries are strictly increasing. These values define the boundaries between the successive histogram cells. The breaks *do not* have to span the range of the observations.

There are N-1 histogram cells, where N = length(breaks). An observation x[i] falls in the jth cell if breaks[j] <= x[i] < breaks[j+1] (for j < N-1) or breaks[j] <= x[i] <= breaks[j+1] (for j = N-1). The weighted histogram value h[j] for the jth cell is the sum of weights[i] for all observations x[i] that fall in the cell.

Note that, in contrast to the function [hist](#), the function whist does not require the breakpoints to span the range of the observations x. Values of x that fall outside the range of breaks are handled separately; their total weight is returned as an attribute of the histogram.

Value

A numeric vector of length N-1 containing the histogram values, where N = length(breaks).

The return value also has attributes "low" and "high" giving the total weight of all observations that are less than the lowest breakpoint, or greater than the highest breakpoint, respectively.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

and Rolf Turner <rolfturner@posteo.net>

with thanks to Peter Dalgaard.

Examples

```
x <- rnorm(100)
b <- seq(-1,1,length=21)
w <- runif(100)
whist(x,b,w)
```

Index