Package ‘statip’

November 17, 2019

Type Package

Title Statistical Functions for Probability Distributions and Regression

Version 0.2.3


License GPL-3

LazyData TRUE

Depends R (>= 3.1.3)

Imports clue, graphics, rpart, stats

Suggests knitr, testthat

URL https://github.com/paulponcet/statip

BugReports https://github.com/paulponcet/statip/issues

RoxygenNote 7.0.0

NeedsCompilation yes

Author Paul Poncet [aut, cre],
  The R Core Team [aut, cph] (C function 'BinDist' copied from package 'stats'),
  The R Foundation [cph] (C function 'BinDist' copied from package 'stats'),
  Adrian Baddeley [ctb] (C function 'BinDist' copied from package 'stats')
### Description

`bandwidth` computes the bandwidth to be used in the `densityfun` function.

### Usage

```r
densityfun(x, bandwidth)
```

### Arguments

- **x**  
  numeric. The data from which the estimate is to be computed.
- **rule**  
  character. A rule to choose the bandwidth. See `bw.nrd`.

### Value

A numeric value.
cv

Coefficient of variation

Description
Compute the coefficient of variation of a numeric vector x, defined as the ratio between the standard deviation and the mean.

Usage
cv(x, na.rm = FALSE, ...)

Arguments
- x: numeric. A numeric vector.
- na.rm: logical. Should missing values be removed before computing the coefficient of variation?
- ...: Additional arguments to be passed to mean().

Value
A numeric value, the coefficient of variation.

References

dbern
The Bernoulli distribution

Description
Density, distribution function, quantile function and random generation for the Bernoulli distribution.

Usage
dbern(x, prob, log = FALSE)
qbern(p, prob, lower.tail = TRUE, log.p = FALSE)
pbern(q, prob, lower.tail = TRUE, log.p = FALSE)
rbern(n, prob)
Arguments

- **x**: numeric. Vector of quantiles.
- **prob**: Probability of success on each trial.
- **log**: logical. If TRUE, probabilities p are given as \( \log(p) \).
- **p**: numeric in \([0,1]\). Vector of probabilities.
- **lower.tail**: logical. If TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
- **log.p**: logical. If TRUE, probabilities \( p \) are given as \( \log(p) \).
- **q**: numeric. Vector of quantiles.
- **n**: number of observations. If length(n) > 1, the length is taken to be the number required.

See Also

See the help page of the **Binomial** distribution.

---

**densityfun**

*Kernel density estimation*

**Description**

Return a function performing kernel density estimation. The difference between **density** and **densityfun** is similar to that between **approx** and **approxfun**.

**Usage**

```r
densityfun(
  x,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  weights = NULL,
  window = kernel,
  width,
  n = 512,
  from,
  to,
  cut = 3,
  na.rm = FALSE,
  ...
)
```
### Arguments

- **x**
  - numeric. The data from which the estimate is to be computed.

- **bw**
  - numeric. The smoothing bandwidth to be used. See the eponymous argument of `density`.

- **adjust**
  - numeric. The bandwidth used is actually \( \text{adjust} \times \text{bw} \). This makes it easy to specify values like 'half the default' bandwidth.

- **kernel, window**
  - character. A string giving the smoothing kernel to be used. Authorized kernels are listed in `.kernelsList()`. See also the eponymous argument of `density`.

- **weights**
  - numeric. A vector of non-negative observation weights, hence of same length as `x`. See the eponymous argument of `density`.

- **width**
  - this exists for compatibility with S; if given, and `bw` is not, will set `bw` to `width` if this is a character string, or to a kernel-dependent multiple of `width` if this is numeric.

- **n**
  - The number of equally spaced points at which the density is to be estimated. See the eponymous argument of `density`.

- **from, to**
  - The left and right-most points of the grid at which the density is to be estimated; the defaults are \( \text{cut} \times \text{bw} \) outside of \( \text{range}(x) \).

- **cut**
  - By default, the values of `from` and `to` are cut bandwidths beyond the extremes of the data. This allows the estimated density to drop to approximately zero at the extremes.

- **na.rm**
  - logical. If `TRUE`, missing values are removed from `x`. If `FALSE` any missing values cause an error.

- **...**
  - Additional arguments for (non-default) methods.

### Value

A function that can be called to generate a density.

### Author(s)

Adapted from the `density` function of package `stats`. The C code of `BinDist` is copied from package `stats` and authored by the R Core Team with contributions from Adrian Baddeley.

### See Also

- `density` and `approxfun` from package `stats`.

### Examples

```r
x <- rlnorm(1000, 1, 1)
f <- densityfun(x, from = 0)
curve(f(x), xlim = c(0, 20))
```
distr2name  

**Conversion between abbreviated distribution names and proper names**

**Description**

The function `distr2name()` converts abbreviated distribution names to proper distribution names (e.g. "norm" becomes "Gaussian").

The function `name2distr()` does the reciprocal operation.

**Usage**

```r
distr2name(x)
name2distr(x)
```

**Arguments**

- `x` character. A vector of abbreviated distribution names or proper distribution names.

**Value**

A character vector of the same length as `x`. Elements of `x` that are not recognized are kept unchanged (yet in lowercase).

**Examples**

```r
distr2name(c("norm", "dnorm", "rhyper", "ppois"))
name2distr(c("Cauchy", "Gaussian", "Generalized Extreme Value"))
```

erf  

**Error function**

**Description**

The function `erf()` encodes the error function, defined as $\operatorname{erf}(x) = 2 \times F(x \times \sqrt{2}) - 1$, where $F$ is the Gaussian distribution function.

**Usage**

```r
erf(x, ...)
```

**Arguments**

- `x` numeric. A vector of input values.
- `...` Additional arguments to be passed to `pnorm`. 
find_breaks

Value
A numeric vector of the same length as x.

References

See Also
pnorm from package stats.

find_breaks Breakpoints to be passed to a Histogram

Description
The function find_breaks() isolates a piece of code of the function truehist() from package MASS that is used to compute the set of breakpoints to be applied for the construction of the histogram.

Usage
find_breaks(x, nbins = "Scott", h, x0 = -h/1000)

Arguments
x numeric. A vector.
nbins integer or character. The suggested number of bins. Either a positive integer, or a character string naming a rule: "Scott" (the default) or "Freedman-Diaconis" or "FD". (Case is ignored.)
h numeric. The bin width, a strictly positive number (takes precedence over nbins).
x0 numeric. Shift for the bins - the breaks are at x0 + h * (...,-1,0,1,...).

Value
A numeric vector.

See Also
histo() in this package; truehist() from package MASS; hist() from package graphics.
**hellinger**

### Hellinger distance

**Description**

Estimate the Hellinger distance between two random samples whose underlying distributions are continuous.

**Usage**

```r
hellinger(x, y, lower = -Inf, upper = Inf, method = 1, ...)```

**Arguments**

- `x`: numeric. A vector giving the first sample.
- `y`: numeric. A vector giving the second sample.
- `lower`: numeric. Lower limit passed to `integrate`.
- `upper`: numeric. Upper limit passed to `integrate`.
- `method`: integer. If `method = 1`, the usual definition of the Hellinger distance is used; if `method = 2`, an alternative formula is used.
- `...`: Additional parameters to be passed to `densityfun`.

**Details**

Probability density functions are estimated with `densityfun`. Then numeric integration is performed with `integrate`.

**Value**

A numeric value, the Hellinger distance.

**References**


**See Also**

`HellingerDist` in package `distrEx`.

**Examples**

```r
x <- rnorm(200, 0, 2)
y <- rnorm(1000, 10, 15)
hellinger(x, y, -Inf, Inf)
hellinger(x, y, -Inf, Inf, method = 2)
```
histo

Alternative Histograms

Description

A simplified version of `hist()` from package `graphics`.

Usage

`histo(x, breaks, ...)`

Arguments

- `x` numeric. A vector.
- `breaks` numeric. A vector of breakpoints to build the histogram, possibly given by `find_breaks()`.
- `...` Additional parameters (currently not used).

Value

An object of class "histogram", which can be plotted by `plot.histogram` from package `graphics`. This object is a list with components:

- `breaks`: the n+1 cell boundaries;
- `counts`: n integers giving the number of x inside each cell;
- `xname`: a string with the actual x argument name.

See Also

`find_breaks()` in this package; `truehist()` from package `MASS`; `hist()` from package `graphics`.

kernel_properties

Smoothing kernels

Description

The generic function `kernelfun` creates a smoothing kernel function.
Usage

```
kernel_properties(name, derivative = FALSE)
```

```
kernelfun(name, ...)
```

```#
## S3 method for class `function`
## S3 method for class `character`
```

```
kernelfun(name, derivative = FALSE, ...)
```

```
.kernelsList()
```

Arguments

<table>
<thead>
<tr>
<th>name</th>
<th>character. The name of the kernel to be used. Authorized kernels are listed in <code>.kernelsList()</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>derivative</td>
<td>logical. If TRUE, the derivative of the kernel is returned.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to be passed to the kernel function.</td>
</tr>
</tbody>
</table>

Value

A function.

See Also

`density` in package stats.

Examples

```
kernel_properties("gaussian")
```

```
k <- kernelfun("epanechnikov")
curve(k(x), xlim = c(-1, 1))
```

---

**lagk**

Lag a vector

Description

This function computes a lagged vector, shifting it back or forward.

Usage

```
lagk(x, k, na = FALSE, cst = FALSE)
```
Arguments

x  A vector.

k  integer. The number of lags. If $k < 0$, la serie est avancee au lieu d’etre retardee.

na  logical. If na = TRUE and $k > 0$ (resp. $k < 0$), the $|k|$ holes created in the lagged vector are put to NA; otherwise, the imputation depends on cst.

cst  logical. If na = FALSE and cst = TRUE, the $|k|$ holes created in the lagged vector are put to $x[[1L]]$ (or to $x[[\text{length}(x)]]$ if $k < 0$). If na = FALSE and cst = FALSE, these $|k|$ holes are imputed by the k first values of x (or the k last values if $k < 0$).

Value

A vector of the same type and length as x.

Examples

```r
v <- sample(1:10)
print(v)
lagk(v, 1)
lagk(v, 1, na = TRUE)
lagk(v, -2)
lagk(v, -3, na = TRUE)
lagk(v, -3, na = FALSE, cst = TRUE)
lagk(v, -3, na = FALSE)
```

Description

The function `mfv()` returns the most frequent value(s) (or mode(s)) found in a vector. The function `mfv1` returns the first of these values, so that `mfv1(x)` is identical to `mfv(x)[[1L]]`.

Usage

```r
mfv(x, ...)

## Default S3 method:
mfv(x, na_rm = FALSE, ...)

## S3 method for class 'tableNA'
mfv(x, na_rm = FALSE, ...)

mfv1(x, na_rm = FALSE, ...)
```
Arguments

x Vector of observations (of type numeric, integer, character, factor, or logical). x is to come from a discrete distribution.

Additional arguments (currently not used).

na.rm logical. If TRUE, missing values do not interfere with the result, see 'Details'.

Details

See David Smith' blog post here to understand the philosophy followed in the code of mfv for missing values treatment.

Value

The function mfv returns a vector of the same type as x. One should be aware that this vector can be of length > 1, in case of multiple modes. mfv always returns a vector of length 1 (the first of the modes found).

Note

mfv() calls the function tabulate.

References


Examples

# Basic examples:
mfv(integer(0)) # NaN
mfv(c(3, 3, 3, 2, 4)) # 3
mfv(c(TRUE, FALSE, TRUE)) # TRUE
mfv(c("a", "a", "b", "a", "d")) # "a"

mfv(c("a", "a", "b", "b", "d")) # c("a", "b")
mfv1(c("a", "a", "b", "b", "d")) # "a"

# With missing values:
mfv(c(3, 3, 3, 2, NA)) # 3
mfv(c(3, 3, 2, NA)) # NA
mfv(c(3, 3, 2, NA), na.rm = TRUE) # 3
mfv(c(3, 3, 2, NA)) # NA
mfv(c(3, 3, 2, NA), na.rm = TRUE) # c(2, 3)
mfv1(c(3, 3, 2, NA), na.rm = TRUE) # 2

# With only missing values:
mfv(c(NA, NA)) # NA
mfv(c(NA, NA), na.rm = TRUE) # NaN

# With factors
midhinge

Description
Compute the midhinge of a numeric vector x, defined as the average of the first and third quartiles.

Usage
midhinge(x, na_rm = FALSE, ...)

Arguments
- x numeric. A numeric vector.
- na_rm logical. Should missing values be removed before computing the midhinge?
- ... Additional arguments to be passed to quantile().

Value
A numeric value, the midhinge.

References

midrange

Description
Compute the mid-range of a numeric vector x, defined as the mean of the minimum and the maximum.

Usage
midrange(x, na_rm = FALSE)

Arguments
- x numeric. A numeric vector.
- na_rm logical. Should missing values be removed before computing the mid-range?
Value

A numeric value, the mid-range.

References

https://en.wikipedia.org/wiki/Mid-range.

---

**picor**  
Piecewise-constant regression

**Description**

picor looks for a piecewise-constant function as a regression function. The regression is necessarily univariate. This is essentially a wrapper for `rpart` (regression tree) and `isoreg`.

**Usage**

```r
picor(formula, data, method, min_length = 0, ...)
```

## S3 method for class 'picor'
knots(Fn, ...)

## S3 method for class 'picor'
predict(object, newdata, ...)

## S3 method for class 'picor'
plot(x, ...)

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

**Arguments**

- **formula**  
  formula of the model to be fitted.
- **data**  
  optional data frame.
- **method**  
  character. If method = "isotonic", then isotonic regression is applied with the `isoreg` from package stats. Otherwise, `rpart` is used, with the corresponding method argument.
- **min_length**  
  integer. The minimal distance between two consecutive knots.
- **...**  
  Additional arguments to be passed to `rpart`.
- **object, x, Fn**  
  An object of class "picor".
- **newdata**  
  data.frame to be passed to the predict method.
Value

An object of class "picor", which is a list composed of the following elements:

- formula: the formula passed as an argument;
- x: the numeric vector of predictors;
- y: the numeric vector of responses;
- knots: a numeric vector (possibly of length 0), the knots found;
- values: a numeric vector (of length length(knots)+1), the constant values taken by the regression function between the knots.

Examples

```r
## Not run:
s <- stats::stepfun(c(-1,0,1), c(1., 2., 4., 3.))
x <- stats::rnorm(1000)
y <- s(x)
p <- picor(y ~ x, data.frame(x = x, y = y))
print(p)
plot(p)

## End(Not run)
```

plot.loess

Basic plot of a loess object

Description

Plots a loess object adjusted on one unique explanatory variable.

Usage

```r
## S3 method for class 'loess'
plot(x, ...)
```

Arguments

- x
  - An object of class "loess".
- ... Additional graphical arguments.

See Also

- `loess` from package stats.

Examples

```r
reg <- loess(dist ~ speed, cars)
plot(reg)
```
predict.default      Default model predictions

Description

Default method of the predict generic function, which can be used when the model object is empty.

Usage

## Default S3 method:
predict(object, newdata, ...)

Arguments

- **object**: A model object, possibly empty.
- **newdata**: An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
- **...**: Additional arguments.

Value

A vector of predictions.

See Also

predict from package stats.

Examples

```r
stats::predict(NULL)
stats::predict(NULL, newdata = data.frame(x = 1:2, y = 2:3))
```

---

tableNA     Alternative Table Creation

Description

Count the occurrences of each factor level or value in a vector.

Usage

tableNA(x)

Arguments

- **x**: numeric. An atomic vector or a factor.
Value

An object of class "tableNA", which is the result of `tabulate()` with three attributes:

- `type_of_x`: the result of `typeof(x)`;
- `is_factor_x`: the result of `is.factor(x)`;
- `levels`: the result of `levels(x)`.

The number of missing values is always reported.

Examples

tableNA(c(1,2,2,1,3))
tableNA(c(1,2,2,1,3, NA))

---

Tukey's trimean

Description

Compute the trimean of a numeric vector x.

Usage

`trimean(x, na.rm = FALSE, ...)`

Arguments

- `x` numeric. A numeric vector.
- `na.rm` logical. Should missing values be removed before computing the trimean?
- `...` Additional arguments to be passed to `quantile()`.

Value

A numeric value, the trimean.

References

`https://en.wikipedia.org/wiki/Trimean`
Index

.kernelsList, 5, 10
.kernelsList(kernel_properties), 9

approx, 4
approxfun, 4, 5

bandwidth, 2
Binomial, 4
bw.nrd, 2

cv, 3
dbern, 3
density, 4, 5, 10
densityfun, 2, 4, 8
distr2name, 6
erf, 6

find_breaks, 7, 9
hellinger, 8
HellingerDist, 8
hist, 7, 9
histo, 7, 9

integrate, 8
isoreg, 14

kernel_properties, 9
kernelfun (kernel_properties), 9
knots.picor (picor), 14

lagk, 10
loess, 15

mean, 3
mfv, 11
mfv1 (mfv), 11
midhinge, 13
midrange, 13

name2distr (distr2name), 6

pbern (dbern), 3
picor, 14
plot.histogram, 9
plot.loess, 15
plot.picor (picor), 14
pnorm, 6, 7
predict, 16
predict.default, 16
predict.picor (picor), 14
print.picor (picor), 14

qbern (dbern), 3
quantile, 13, 17

rbern (dbern), 3
rpart, 14

tableNA, 16
tabulate, 12, 17
trimean, 17
truehist, 7, 9