Package ‘logdensity’

June 1, 2020

Type Package

Title Estimate the (Logarithm) of the Density and its Derivatives

Version 0.1.0

Author Joris Pinkse and Karl Schurter

Maintainer Karl Schurter <kschurter@psu.edu>

Description This package contains functions that estimate the logarithm of an unknown density function from iid data. The estimation strategy is based on a local polynomial appoximation to the log-density. The estimates behave well near the boundary of the support of the density and can be guaranteed to be nonnegative. Details can be found in Pinske and Schurter (2020) "Estimates of derivatives of (log) densities and related objects."

Imports graphics,
parallel,
stats

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

R topics documented:

`logdensity-package` *The logdensity package*

Description

A package for estimating (log) densities and their derivatives

Details

This package contains functions that estimate the logarithm of an unknown density function from iid data. The estimation strategy is based on a local polynomial appoximation to the logarithm of the density. The estimates behave well near the boundary of the support of the density and can be guaranteed to be nonnegative. Details can be found in Pinske and Schurter (2020).

Author(s)

Joris Pinkse and Karl Schurter (maintainer, `<kschurter@psu.edu>`)

References

Pinkse, J. and Schurter, K. (2020) "Estimates of derivatives of (log) densities and related objects."

`bellpoly` *Evaluate partial Bell polynomials*

Description

Evaluates the partial Bell polynomials, also known as the incomplete Bell polynomials. The complete Bell polynomial is the sum of the partial Bell polynomials over $k = 1, \ldots, n$. The partial polynomials are evaluted using the recurrence relation

$$k\, B_{n,k}(x_1, \ldots, x_{n-k+1}) = \sum_{r=k-1}^{n-1} \binom{n}{r} x_{n-r}\, B_{r,k-1}(x_1, \ldots, x_{r-k+2})$$

where the sum is over $r = k-1, \ldots, n-1$, and $B_{0,0} = 1$, $B_{0,k} = 0$ for $k \geq 1$, and $B_{n,0} = 0$ for $n \geq 1$.

Usage

```
bellpoly(x, n = nrow(x), k = min(1, n):n)
```

Arguments

`x`	a matrix or object that can be coerced into one by `as.matrix`
`n`	nonnegative integer representing the number of elements
`k`	vector of integers no greater than `n` representing the sizes of partition

Value

`bellpoly` returns a vector or matrix depending on the dimensions of `x` that contains the value of the partial Bell polynomial(s) evaluated at columns of `as.matrix(x)`.

Examples

```
n <- 5
k <- 1:n

## Idempotent numbers
bellpoly(k)
choose(n, k) * k^(n - k) # equivalent

## Stirling numbers (unsigned) of the first kind
bellpoly(x = factorial(k-1))

## Stirling numbers of the second kind
bellpoly(x = rep(1, n))
```

`evalkernel` *Evaluate a kernel function*

Description

Evaluate a kernel function

Usage

```
evalkernel(
  u,
  kernel = c("epanechnikov", "gaussian", "triweight", "uniform", "triangle", "cc
    "quartic")
)
```

Arguments

`u`	numeric array
`kernel`	character string specifying name of kernel

Value

array with same dimensions as `u` containing kernel evaluated at `u`. Note: `NA` values of `u` evaluate to 0.

`logdensity` *A local polynomial log-density estimator*

Description

`logdensity.fit` is intended to be called from within `logdensity`, after performing basic argument verification. Use caution when calling `logdensity.fit` directly.

Usage

```
logdensity(
  data,
  x,
  h,
  g,
  dg,
  m = "epanechnikov",
  minx = -Inf,
  maxx = Inf,
  S = 1,
  logf = TRUE,
  mc.cores = 1L,
  ...
)

logdensity.fit(data, x, h, g, dg, m, minx, maxx, logf, exact, ...)
```

Arguments

`data`	numeric vector of observations
`x`	points at which to estimate (if shorter than h, recycled to length of h)
`h`	bandwidth (if shorter than x, recycled to the length of x)
`g`	function of `u`, `zl`, and `zr` that must be equal to an `S`-length vector of 0's at `zl = max((minx-x)/h,-1)` and `zr = min((maxx-x),1)`, where `S` is the order of the local polynomial approximation to the log-density. Function must be vectorized so that `g(u, zl, zr)` returns a matrix that is `length(u)` by `S`.
`dg`	function that evaluates derivative of `g` with respect to `u`. Must be vectorized and return a matrix that is `length(u)` by `S`.
`m`	kernel function used to compute the density. Can be a function, symbol, or character string that matches the name of a function or one of the kernels allowed in evalkernel. If `m == "epanechnikov"` and the polynomial order is 1, an exact solution is computed. Otherwise, the estimate of the log-density involves numerical integration.
`minx`	lower bound of support of `x`
`maxx`	upper bound of support of `x`
`S`	degree of polynomial expansion of log-density to be used with default `g`. If user supplies `g` and `dg`, this argument is ignored without warning.

`logf`	logical indicating whether the log-density should be compute, in addition to its derivative(s).
`mc.cores`	integer number of cores to use with `mcmapply`. If equal to 1 (default), `mapply` will be used to loop over `x`, instead.
`...`	Further arguments supplied to `g` and `dg`
`exact`	logical indicating whether an exact solution should be used (if available) or numerical integration. Exact solution currently only available with epanechnikov kernel and local linear approximation.

Value

an object of class `logdensity` which inherits from `matrix`. The `S+1` by `length(x)` matrix of estimated log-densities (`NA` unless `logf` is `TRUE`) and derivatives has the following additional attributes: `x` vector of points at which the estimates were computed `n` number of non-missing observations used in estimation `h` bandwidth(s) used `call` matched call

A numeric vector containing the log-density (if `logf == TRUE`) and its derivatives

References

Pinkse, J. and Schurter, K. (2020) "Estimates of derivatives of (log) densities and related objects."

See Also

`mapply`, `mcmapply`, `integrate`, `bellpoly`

Examples

```
dat <- rchisq(n = 100, df = 2)
x <- seq(from = 0, to = 2, length.out = 20)

## fixed bandwidth
ld <- logdensity(data = dat, x = x, h = 0.5, minx = 0, S = 1)
print(ld)
plot(ld)

## variable bandwidth
h <- pmax(1-x, 0.5)
ld <- logdensity(data = dat, x = x, h = h, minx = 0, S = 2)
ld
plot(ld)

## Faa di Bruno's formula for the density and its derivatives
deriv <- 0L  # integer between 0 and S (=2 for most recent estimation)
exp(ld[1, ]) * colSums(bellpoly(ld[-1, ], n = deriv))

### verify formula for deriv = 0, 1, and 2
exp(ld[1, ]) * colSums(bellpoly(ld[-1, ], n = 0L))  # density (0th derivative)
exp(ld[1, ])  # equivalent

exp(ld[1, ]) * colSums(bellpoly(ld[-1, ], n = 1L))  # 1st derivative of density
exp(ld[1, ]) * ld[2,]  # equivalent

exp(ld[1, ]) * colSums(bellpoly(ld[-1, ], n = 2L))  # 2nd derivative of density
exp(ld[1, ]) * (ld[2,]^2 + ld[3, ])  # equivalent
```

plot.logdensity *Plot density and log-density estimates*

Description

Plot density and log-density estimates

Usage

```
## S3 method for class 'logdensity'
plot(x, density = FALSE, type = "l", xlab, ylab, ...)
```

Arguments

`x`	a logdensity object
`density`	logical indicating whether to plot density or log-density
`type`	what type of plot should be drawn
`xlab`	a title for the x axis
`ylab`	a title for the y axis
`...`	further arguments passed to `plot.default`

See Also

`plot.default`

print.logdensity *Print log-density estimates*

Description

Print log-density estimates

Usage

```
## S3 method for class 'logdensity'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  max = 100,
  quote = FALSE,
  ...
)
```

Arguments

`x`	a logdensity object
`digits`	minimal number of significant digits
`max`	approximate max number of entries to be printed
`quote`	logical, indicating whether strings should be printed with surrounding quotes
`...`	further arguments passed to `print.default`

Value

invisibly returns x

See Also

`print.default`

Index