Package 'kdensity'

October 13, 2022

Type Package

Title Kernel Density Estimation with Parametric Starts and Asymmetric Kernels

Version 1.1.0

Author Jonas Moss, Martin Tveten

Maintainer Jonas Moss <jonas.gjertsen@gmail.com>

Description Handles univariate non-parametric density estimation with parametric starts and asymmetric kernels in a simple and flexible way. Kernel density estimation with parametric starts involves fitting a parametric density to the data before making a correction with kernel density estimation, see Hjort & Glad (1995) <doi:10.1214/aos/1176324627>. Asymmetric kernels make kernel density estimation more efficient on bounded intervals such as (0, 1) and the positive half-line. Supported asymmetric kernels are the gamma kernel of Chen (2000) <doi:10.1023/A:1004165218295>, the beta kernel of Chen (1999) <doi:10.1016/S0167-9473(99)00010-9>, and the copula kernel of Jones & Henderson (2007) <doi:10.1093/biomet/asm068>. User-supplied kernels, parametric starts, and bandwidths are supported.

License MIT + file LICENSE

URL https://github.com/JonasMoss/kdensity

BugReports https://github.com/JonasMoss/kdensity/issues

Encoding UTF-8

LazyData true

Suggests extraDistr, SkewHyperbolic, testthat, covr, knitr, rmarkdown

Imports assertthat, univariateML, EQL

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-09-30 09:00:05 UTC

R topics documented:

`bandwidths` *Bandwidth Selectors*

Description

The available options for bandwidth selectors, passed as the `bw` argument to `kdensity`.

Arguments

`x`	The input data.
`kernel_str`	A string specifying the kernel, e.g. "gaussian."
`start_str`	A string specifying the parametric start, e.g. "normal".
`support`	The domain of definition for the kernel. (-Inf, Inf) for symmetric kernels.

Details

The bandwidth functions are not exported. They are members of the environment `bw_environments`, and can be accessed by `kdensity:::bw_environments`.

Bandwidth selectors

`"nrd0", "nrd", "bcv", "SJ"`: Bandwidth selectors from `stats`. They are documented in `[bandwidth][stats::bandwidth] stats:bandwidth`. "nrd0" is the standard bandwidth selector for symmetric kernels with constant parametric starts.

`"ucv"`: Unbiased cross validation. The standard option for asymmetric kernels.

`"RHE"`: Selector for parametric starts with a symmetric kernel, based on a reference rule with Hermite polynomials. Described in Hjort & Glad (1995). The default method in `kdensity` when a parametric start is supplied and the kernel is symmetric.

`"JH"`: Selector for the Gaussian copula kernel, based on normal reference rule. Described in Jones & Henderson. The default method when the `gcopula` kernel is used in `kdensity`.

Structure

The bandwidth selector is a function of four arguments: The data `x`, a kernel string `kernel`, a start string `start`, and a support vector `support`. To obtain the functions associated with these strings, use `get_kernel` and `get_start`. The function should return a double.

References

Jones, M. C., and D. A. Henderson. "Kernel-type density estimation on the unit interval." Biometrika 94.4 (2007): 977-984. Hjort, Nils Lid, and Ingrid K. Glad. "Nonparametric density estimation with a parametric start." The Annals of Statistics (1995): 882-904.

See Also

`kdensity()`, `stats::bandwidth.kernel()` for the bandwidth selectors of `stats::density()`. In addition, `kernels()`; `parametric_starts()`

Examples

```
## Not a serious bandwidth function.
silly_width = function(x, kernel = NULL, start = NULL, support = NULL) {
  rexp(1)
}
kdensity(mtcars$mpg, start = "gumbel", bw = silly_width)
```

`kdensity` *Parametrically guided kernel density estimation*

Description

`kdensity` computes a parametrically guided kernel density estimate for univariate data. It supports asymmetric kernels and parametric starts through the `kernel` and `start` arguments.

Usage

```
kdensity(
  x,
  bw = NULL,
  adjust = 1,
  kernel = NULL,
  start = NULL,
  support = NULL,
  na.rm = FALSE,
  normalized = TRUE,
  tolerance = 0.01
)
```

Arguments

`x`	Numeric vector containing the data.
`bw`	A bandwidth function. Can be either a string, a custom-made function, or a double. The supported bandwidth functions are documented in `bandwidths()`.
`adjust`	An adjustment constant, so that `h = adjust*bw*sd`, where `sd` varies with the chosen kernel.

`kernel`	The kernel function. Can be chosen from the list of built-in kernels or be custom-made. See `kernels()` for details.
`start`	Parametric start. Can be chosen from the list of built-in parametric starts or be custom-made. See `parametric_starts()` for details.
`support`	The support of the data. Must be compatible with the supplied `x` and the supplied `start` and `kernel`. Is used to find the normalization constant, see `normalized`.
`na.rm`	Logical; if `TRUE`, `NA`s will be removed from `x`.
`normalized`	Logical; if `TRUE`, the density is normalized.
`tolerance`	Numeric; the relative error to tolerate in normalization.

Details

The default values for `bw`, `kernel`, `start`, and `support` are interdependent, and are chosen to make sense. E.g., the default value for `support` when `start = beta` is `c(0, 1)`.

The `start` argument defaults to `uniform`, which corresponds to ordinary kernel density estimation. The typical default value for `kernel` is `gaussian`.

If `normalized` is `FALSE` and `start != "uniform"`, the resulting density will not integrate to 1 in general.

Value

`kdensity` returns an S3 function object of `base::class()` "kdensity". This is a callable function with the following elements, accessible by '$':

`x` The data supplied in `x`.

`bw_str, bw, adjust, h` The bandwidth function, the resulting bandwidth, the `adjust` argument, and the adjusted bandwidth.

`kernel_str, kernel, start, start_str, support` Name of the kernel, the kernel object, name of the parametric start, the start object, and the support of the density.

`data.name, n, range, has.na, na.rm, normalized` Name of the data, number of observations, the range of the data, whether the data `x` contained `NA` values, whether na.rm is `TRUE` or not, and whether the density is normalized.

`call` The `call` to `kdensity`.

`estimates` Named numeric vector containing the parameter estimates from the parametric start.

`logLik` The log-likelihood of the parametric starts. Is `NA` for the uniform start.

References

Hjort, Nils Lid, and Ingrid K. Glad. "Nonparametric density estimation with a parametric start." The Annals of Statistics (1995): 882-904.

Jones, M. C., and D. A. Henderson. "Miscellanea kernel-type density estimation on the unit interval." Biometrika 94.4 (2007): 977-984.

Chen, Song Xi. "Probability density function estimation using gamma kernels." Annals of the Institute of Statistical Mathematics 52.3 (2000): 471-480.

Silverman, Bernard W. Density estimation for statistics and data analysis. Vol. 26. CRC press, 1986.

See Also

The `stats` package function `stats::density()`.

Examples

```
## Use gamma kernels to model positive data, the concentration of
## theophylline

concentration = Theoph$conc + 0.001
plot(kdensity(concentration, start = "gamma", kernel = "gamma", adjust = 1/3),
     ylim = c(0, 0.15), lwd = 2, main = "Concentration of theophylline")
lines(kdensity(concentration, start = "gamma", kernel = "gaussian"),
      lty = 2, col = "grey", lwd = 2)
lines(kdensity(concentration, start = "gaussian", kernel = "gaussian"),
      lty = 3, col = "blue", lwd = 2)
lines(kdensity(concentration, start = "gaussian", kernel = "gamma", adjust = 1/3),
      lty = 4, col = "red", lwd = 2)
rug(concentration)

## Using a density and and estimator from another package.

skew_hyperbolic = list(
  density   = SkewHyperbolic::dskewhyp,
  estimator = function(x) SkewHyperbolic::skewhypFit(x, printOut = FALSE)$param,
  support   = c(-Inf, Inf)
)

kde = kdensity(diff(LakeHuron), start = skew_hyperbolic)
plot(kde, lwd = 2, col = "blue",
     main = "Annual differences in water level (ft) of Lake Huron, 1875 - 1972")
lines(kde, plot_start = TRUE, lty = 2, lwd = 2) # Plots the skew hyperbolic density.
rug(diff(LakeHuron))

kde$estimates # Also: coef(kde)
# Displays the parameter estimates:
#        mu     delta      beta        nu
# -1.140713  3.301112  2.551657 26.462469
```

---

kernels *Kernel functions*

Description

Kernel functions are an important part of `kdensity`. This document lists the available built-in functions and the structure of them. Any kernel in the list can be used in `kdensity` by using `kernel = "kernel"` for the intended kernel.

Details

Be careful combining kernels with compact support with parametric starts, as the normalizing integral typically fails to converge. Use `gaussian` instead.

Symmetric kernels

`gaussian, normal`: The Gaussian kernel. The default argument when `starts` is supported on R. `epanechnikov, rectangular (uniform), triangular, biweight, cosine, optcosine`: Standard symmetric kernels, also used in `stats::density()`. `tricube, triweight`: Standard symmetric kernels. Not supported by `stats::density()`. `laplace`: Uses the Laplace density, also known as the double exponential density.

Asymmetric kernels

`gamma, gamma_biased`: The gamma kernel of Chen (2000). For use on the positive half-line. `gamma` is the recommended biased-corrected kernel. `gcopula`: The Gaussian copula kernel of Jones & Henderson (2007). For use on the unit interval. `beta, beta_biased`: The beta kernel of Chen (1999). For use on the unit interval. `beta` is the recommended bias-corrected kernel.

Structure

A kernel is a list containing two mandatory elements and one optional element. The mandatory element '`kernel`' is the kernel function. It takes arguments `y, x, h`, where `x` is the data supplied to `kdensity` and `y` is the point of evaluation. `h` is the bandwidth. Internally, the kernel function is evaluated as `1/h*kernel(y, x, h)`. It should be vectorized in `x`, but vectorization in `y` is not needed.

The second mandatory element is `support`, stating the domain of definition for the kernel. This is used to distinguish kernels on the unit interval / positive half-line from kernels on R.

`sd` is used for symmetric kernels, and states the standard error of the kernel. This is used to make kernels comparable to the Gaussian kernel when calculating bandwidths.

References

Chen, Song Xi. "Probability density function estimation using gamma kernels." Annals of the Institute of Statistical Mathematics 52.3 (2000): 471-480. Jones, M. C., and D. A. Henderson. "Kernel-type density estimation on the unit interval." Biometrika 94.4 (2007): 977-984. Chen, Song Xi. "Beta kernel estimators for density functions." Computational Statistics & Data Analysis 31.2 (1999): 131-145.

See Also

`kdensity()`; `parametric_starts()`; `bandwidths()`.

Examples

```
gaussian = list(
  kernel  = function(y, x, h) stats::dnorm((y-x)/h),
  sd = 1,
  support = c(-Inf, Inf)
)
```

```
gcopula = list(
  kernel  = function(y, x, h) {
    rho = 1 - h^2
    inside = rho^2*(qnorm(y)^2 + qnorm(x)^2)-2*rho*qnorm(y)*qnorm(x)
    exp(-inside/(2*(1-rho^2)))
  },
  support = c(0, 1)
)
```

parametric_starts *Parametric starts*

Description

A parametric start is a density function with an associated estimator which is used as a starting point in `kdensity`. Several parametric starts are implemented, all with maximum likelihood estimation. Custom-made parametric starts are possible, see the Structure section.

Structure

The parametric start contains three elements: The density function, an estimation function, and the support of the density. The parameters of the density function must partially match the parameters of the estimator function. The estimator function takes one argument, a numeric vector, which is passed from `kdensity`.

Supported parametric starts

`kdensity` supports more than 20 built-in starts from the univariateML package, see `univariateML::univariateML_models` for a list. Densities with variable support, `power`, are not supported. The `pareto` density has its support fixed to `(1,Inf)`. The options `uniform, constant` makes `kdensity` estimate a kernel density without parametric starts.

See Also

`kdensity()`; `kernels()`; `bandwidths()`

Examples

```
start_exponential = list(
 density = stats::dexp,
 estimator = function(data) {
   c(rate = 1/mean(data))
 },
 support = c(0, Inf)
)

start_inverse_gaussian = list(
 density = extraDistr::dwald,
 estimator = function(data) {
  c(mu = mean(data),
    lambda = mean(1/data - 1/mean(data)))
  },
 support = c(0, Inf)
)
```

`plot.kdensity` *Plot, Lines and Points Methods for Kernel Density Estimation*

Description

The `plot` method for `kdensity` objects.

Usage

```
## S3 method for class 'kdensity'
plot(x, range = NULL, plot_start = FALSE, zero_line = TRUE, ...)

## S3 method for class 'kdensity'
lines(x, range = NULL, plot_start = FALSE, zero_line = TRUE, ...)

## S3 method for class 'kdensity'
points(x, range = NULL, plot_start = FALSE, zero_line = TRUE, ...)
```

Arguments

`x`	a `kdensity` object.
`range`	range of x values.
`plot_start`	logical; if `TRUE`, plots the parametric start instead of the kernel density estimate.
`zero_line`	logical; if `TRUE`, add a base line at `y = 0`.
`...`	further plotting parameters.

Value

None.

See Also

`kdensity()`

Examples

```
## Using the data set "precip" to eye-ball the similarity between
## a kernel fit, a parametric fit, and a kernel with parametric start fit.
kde_gamma = kdensity(precip, kernel = "gaussian", start = "gamma")
kde = kdensity(precip, kernel = "gaussian", start = "uniform")

plot(kde_gamma, main = "Annual Precipitation in US Cities")
lines(kde_gamma, plot_start = TRUE, lty = 2)
lines(kde, lty = 3)
rug(precip)
```

Index