Package: kdensity (via r-universe)

October 28, 2026

Type Package

Title Kernel Density Estimation with Parametric Starts and Asymmetric Kernels

Version 2.0.0

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

Suggests extraDistr, testthat, knitr, rmarkdown

Imports Rcpp, univariateML

LinkingTo Rcpp

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Jonas Moss [aut, cre] (ORCID: <https://orcid.org/0000-0002-6876-6964>), Martin Tveten [ctb]

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

Repository https://cran.r-universe.dev

Date/Publication 2026-10-28 16:52:53 UTC

RemoteUrl https://github.com/cran/kdensity

RemoteRef HEAD

RemoteSha 8a7992ba4946f593b4f6787f26dc96924b78ee94

Contents

bandwidths 2
kdensity 4
kdensity_grid 6
kernels 8
parametric_starts 9
plot.kdensity 10

Index **12**

`bandwidths` *Bandwidth Selectors*

Description

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

Details

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

Defaults

Every default is a closed-form rule, so no default requires cross-validation:

Kernel	Uniform start	Parametric start
symmetric (`gaussian`, ...)	`"nrd0"`	`"RHE"`
`gamma`, `gamma_biased`	`"GR"`	`"GR"`
`beta`, `beta_biased`	`"HS"`	`"HS"`
`gcopula`	`"SJ"`	`"SJ"`
custom kernel without `sd`	`"ucv"`	`"ucv"`

The rules use only the data, not the fitted parametric start. When the start fits well, the correction it leaves for the kernel is smoother than the density itself, so a larger bandwidth would be better; the rules then undersmooth slightly. `"ucv"` takes the start into account at the cost of a computation that grows with the square of the sample size.

Bandwidth selectors

`"nrd0"`, `"nrd"`, `"bcv"`: Bandwidth selectors from `stats`, documented in `stats::bw.nrd0()`. They are on the scale of the Gaussian kernel's standard deviation; `kdensity` rescales them for the other symmetric kernels.

`"SJ"`: The Sheather & Jones (1991) plug-in selector `stats::bw.SJ()`. For the `gcopula` kernel it is applied on the probit scale: there the Gaussian copula estimator is a Gaussian kernel estimate with bandwidth `s = sqrt(1 - rho^2)`, where `rho = 1 - h^2`, so the selected `s` is converted to `h = sqrt(1 - sqrt(1 - s^2))`. If the selector fails on degenerate data, `"JH"` is used with a warning.

`"ucv"`: Unbiased cross-validation. With a parametric start it refits the start without each observation, so its cost grows with the square of the sample size.

`"HS"`: Hallberg Szabadváry's closed-form reference rule for the bias-corrected beta kernel, based on a beta reference density fitted by moments. Used for both beta kernels, with a fallback heuristic when the rule's moment conditions fail.

`"GR"`: Chen's AMISE-optimal bandwidth for the bias-corrected gamma kernel, with the unknown density replaced by a gamma reference fitted by maximum likelihood to the positive observations. The fitted shape is floored at 3, keeping the fitted mean: the rule needs a shape above 3/2, and the floor was the most robust choice in simulations with exponential, lognormal, Weibull and mixture targets. Used for both gamma kernels.

`"RHE"`: A reference rule for symmetric kernels that estimates the roughness of the density with Hermite polynomials around a fitted normal density. It uses only the data, not the parametric start.

`"JH"`: Jones & Henderson's normal-reference rule for the Gaussian copula kernel. Accurate for unimodal data, but it oversmooths multimodal data badly; capped at 0.5.

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. Hallberg Szabadváry, Johan. "A Fast, Closed-Form Bandwidth Selector for the Beta Kernel Density Estimator." arXiv preprint arXiv:2601.19553 (2026). Sheather, S. J., and M. C. Jones. "A reliable data-based bandwidth selection method for kernel density estimation." Journal of the Royal Statistical Society: Series B 53.3 (1991): 683-690. Chen, Song Xi. "Probability density function estimation using gamma kernels." Annals of the Institute of Statistical Mathematics 52.3 (2000): 471-480.

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(). Every built-in kernel defaults to a closed-form rule; use bw = "ucv" for start-aware cross-validation. For symmetric kernels, bw is on the Gaussian standard-deviation scale.
adjust	An adjustment constant. For symmetric kernels, h = adjust * bw * sd, where sd is the kernel's stored scale factor (the reciprocal of its standard deviation). For asymmetric kernels, h = adjust * bw.
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, NAs 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.

Fitted objects are read-only: $<- and [[<- assignment is not supported. Use kde <- update(kde, bw = 0.5) to refit from the stored call with changed arguments and return a new object. The data must be visible where update() is called.

The logLik(), AIC(), and BIC() methods for kdensity objects describe the *fitted parametric start* alone, not the kernel density estimate: they use the start's maximum-likelihood log-likelihood, so they can be used to compare candidate parametric starts. They are undefined for the uniform and constant starts (logLik() returns NA and errors), which have no parametric model.

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 *fitted parametric start* (not of the kernel density estimate): the sum of log start(x_i) at the maximum-likelihood estimates, evaluated directly on the log scale when supported by the start density. Is NA for the uniform and constant starts.

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)
```

---

kdensity_grid *Evaluate a kdensity fit on a fast grid*

Description

kdensity_grid() evaluates a fitted kdensity() object on a grid. The "fft" method uses weighted linear binning and an FFT convolution with the fitted kernel, the algorithm of stats::density(), for every built-in symmetric kernel. It also supports the Gaussian-copula kernel through its exact Gaussian-convolution representation on the probit scale.

Usage

```
kdensity_grid(
  object,
  from = NULL,
  to = NULL,
  n = 512L,
  method = c("auto", "exact", "fft"),
```

```
  fft_n = NULL,
  truncate = 8
)
```

Arguments

object	A fitted kdensity object.
from, to	Finite endpoints of the evaluation interval. By default, get_range() supplies a plotting interval.
n	Number of returned grid points.
method	One of "auto", "exact", or "fft". "auto" uses the FFT backend for Gaussian and Gaussian-copula kernels and otherwise uses the exact evaluator. Compact kernels retain their faster exact sorted evaluator. "fft" explicitly requests FFT evaluation and reports an error when the fitted kernel is unsupported.
fft_n	Size of the internal convolution grid. The default is at least four times n and at least 2048 points.
truncate	Number of Gaussian standard deviations used to pad an FFT convolution. The Laplace kernel is padded by truncate^2 / 2 bandwidths, where its tail is as small as the Gaussian tail at truncate standard deviations. Compact kernels use their exact support instead.

Details

The FFT method approximates only the binning and interpolation steps. The fitted parametric start and the normalization constant from object are retained. Calling the kdensity object itself always uses exact evaluation.

Value

An object of class "density". Its method element records whether exact or FFT evaluation was used.

See Also

kdensity()

Examples

```
set.seed(1)
fit <- kdensity(rbeta(1000, 2, 5), kernel = "gcopula", bw = 0.12)
fast <- kdensity_grid(fit, n = 512)
plot(fast)
```

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

Ordinary symmetric kernel estimates use exact normalization constants. With a non-uniform parametric start, compact-support kernels still require numerical normalization, which can fail to converge; use gaussian if this occurs.

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 kernels of Chen (2000). For use on the positive half-line. gamma_biased is Chen's first kernel, a gamma density with shape y / h + 1 and scale h. gamma is the recommended bias-corrected (modified) kernel, with shape y / h when y >= 2h and (y / (2h))^2 + 1 otherwise. 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 is the reciprocal of the kernel's standard deviation. This scale factor converts a bandwidth on the Gaussian standard-deviation scale to h = adjust * bw * sd.

Custom objects must use names different from the built-ins (for example, my_gaussian). Registering another custom object under the same name replaces the earlier custom object.

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

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

my_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)
    h / sqrt(1 - rho^2) * 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.

Custom objects must use names different from the built-ins (for example, my_gaussian). Registering another custom object under the same name replaces the earlier custom object.

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, grid = "auto", ...)

## S3 method for class 'kdensity'
lines(
  x,
  range = NULL,
```

```
  plot_start = FALSE,
  zero_line = TRUE,
  grid = "auto",
  ...
)

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

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.
grid	Evaluation backend. "auto" uses the fastest verified backend: FFT for Gaussian and Gaussian-copula kernels, and exact evaluation otherwise. "exact" always retains direct evaluation and "fft" explicitly requests the approximate backend supported by kdensity_grid().
...	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

bandwidths, 2
bandwidths(), *4*, *9*, *10*
base::class(), *5*

get_range(), *7*

kdensity, 4
kdensity(), *3*, *6*, *7*, *9–11*
kdensity_grid, 6
kdensity_grid(), *11*
kernels, 8
kernels(), *3*, *4*, *10*

lines.kdensity *(*plot.kdensity*)*, 10

parametric_starts, 9
parametric_starts(), *3*, *4*, *9*
plot.kdensity, 10
points.kdensity *(*plot.kdensity*)*, 10

stats::bandwidth.kernel(), *3*
stats::bw.nrd0(), *3*
stats::bw.SJ(), *3*
stats::density(), *3*, *6*, *8*

univariateML, *10*