

# Package ‘kde1d’

November 18, 2019

**Type** Package

**Title** Univariate Kernel Density Estimation

**Version** 1.0.2

**Description** Provides an efficient implementation of univariate local polynomial kernel density estimators that can handle bounded and discrete data. See Geenens (2014) <arXiv:1303.4121>, Geenens and Wang (2018) <arXiv:1602.04862>, Nagler (2018a) <arXiv:1704.07457>, Nagler (2018b) <arXiv:1705.05431>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**LinkingTo** BH, Rcpp, RcppEigen

**Imports** graphics, Rcpp, randtoolbox, stats, utils

**RoxygenNote** 7.0.0

**Suggests** testthat

**URL** <https://github.com/tnagler/kde1d>

**BugReports** <https://github.com/tnagler/kde1d/issues>

**SystemRequirements** C++11

**NeedsCompilation** yes

**Author** Thomas Nagler [aut, cre],  
Thibault Vatter [aut]

**Maintainer** Thomas Nagler <mail@tnagler.com>

**Repository** CRAN

**Date/Publication** 2019-11-18 15:50:02 UTC

## R topics documented:

kde1d-package	2
dkde1d	2
equi_jitter	3
kde1d	4
plot.kde1d	6

<b>Index</b>	<b>8</b>
--------------	----------

---

kde1d-package	<i>One-Dimensional Kernel Density Estimation</i>
---------------	--

---

### Description

Provides an efficient implementation of univariate local polynomial kernel density estimators that can handle bounded and discrete data. The implementation utilizes spline interpolation to reduce memory usage and computational demand for large data sets.

### References

- Geenens, G. (2014). *Probit transformation for kernel density estimation on the unit interval*. Journal of the American Statistical Association, 109:505, 346-358, [arXiv:1303.4121](#)
- Geenens, G., Wang, C. (2018). *Local-likelihood transformation kernel density estimation for positive random variables*. Journal of Computational and Graphical Statistics, to appear, [arXiv:1602.04862](#)
- Nagler, T. (2018a). *A generic approach to nonparametric function estimation with mixed data*. Statistics & Probability Letters, 137:326–330, [arXiv:1704.07457](#)
- Nagler, T. (2018b). *Asymptotic analysis of the jittering kernel density estimator*. Mathematical Methods of Statistics, in press, [arXiv:1705.05431](#)

---

dkde1d	<i>Working with a kde1d object</i>
--------	------------------------------------

---

### Description

Density, distribution function, quantile function and random generation for a 'kde1d' kernel density estimate.

### Usage

```
dkde1d(x, obj)

pkde1d(q, obj)

qkde1d(p, obj)

rkde1d(n, obj, quasi = FALSE)
```

**Arguments**

x	vector of density evaluation points.
obj	a kde1d object.
q	vector of quantiles.
p	vector of probabilities.
n	integer; number of observations.
quasi	logical; the default (FALSE) returns pseudo-random numbers, use TRUE for quasi-random numbers (generalized Halton, see <a href="#">randtoolbox::sobol()</a> ).

**Details**

[dkde1d\(\)](#) gives the density, [pkde1d\(\)](#) gives the distribution function, [qkde1d\(\)](#) gives the quantile function, and [rkde1d\(\)](#) generates random deviates.

The length of the result is determined by n for [rkde1d\(\)](#), and is the length of the numerical argument for the other functions.

**Value**

The density, distribution function or quantile functions estimates evaluated respectively at x, q, or p, or a sample of n random deviates from the estimated kernel density.

**See Also**

[kde1d\(\)](#)

**Examples**

```
set.seed(0) # for reproducibility
x <- rnorm(100) # simulate some data
fit <- kde1d(x) # estimate density
dkde1d(0, fit) # evaluate density estimate (close to dnorm(0))
pkde1d(0, fit) # evaluate corresponding cdf (close to pnorm(0))
qkde1d(0.5, fit) # quantile function (close to qnorm(0))
hist(rkde1d(100, fit)) # simulate
```

---

equi\_jitter

*Conditionally equidistant jittering*

---

**Description**

Converts ordered variables to numeric and Adds deterministic uniform noise. See *Details*.

**Usage**

```
equi_jitter(x)
```

## Arguments

x observations; the function does nothing if x is already numeric.

## Details

Jittering makes discrete variables continuous by adding noise. This simple trick allows to consistently estimate densities with tools designed for the continuous case (see, Nagler, 2018a/b). The drawback is that estimates are random and the noise may deteriorate the estimate by chance.

Here, we add a form of deterministic noise that makes estimators well behaved. Tied occurrences of a factor level are spread out uniformly (i.e., equidistantly) on the interval  $[-0.5, 0.5]$ . This is similar to adding random noise that is uniformly distributed, conditional on the observed outcome. Integrating over the outcome, one can check that the unconditional noise distribution is also uniform on  $[-0.5, 0.5]$ .

Asymptotically, the deterministic jittering variant is equivalent to the random one.

## References

Nagler, T. (2018a). *A generic approach to nonparametric function estimation with mixed data*. *Statistics & Probability Letters*, 137:326–330, [arXiv:1704.07457](#)

Nagler, T. (2018b). *Asymptotic analysis of the jittering kernel density estimator*. *Mathematical Methods of Statistics*, in press, [arXiv:1705.05431](#)

## Examples

```
x <- as.factor(rbinom(10, 1, 0.5))
equi_jitter(x)
```

---

kde1d

*Univariate local-polynomial likelihood kernel density estimation*

---

## Description

The estimators can handle data with bounded, unbounded, and discrete support, see *Details*.

## Usage

```
kde1d(
  x,
  xmin = NaN,
  xmax = NaN,
  mult = 1,
  bw = NA,
  deg = 2,
  weights = numeric(0)
)
```

**Arguments**

<code>x</code>	vector (or one-column matrix/data frame) of observations; can be numeric or ordered.
<code>xmin</code>	lower bound for the support of the density (only for continuous data); NaN means no boundary.
<code>xmax</code>	upper bound for the support of the density (only for continuous data); NaN means no boundary.
<code>mult</code>	positive bandwidth multiplier; the actual bandwidth used is $bw * mult$ .
<code>bw</code>	bandwidth parameter; has to be a positive number or NA; the latter uses the plug-in methodology of Sheather and Jones (1991) with appropriate modifications for $deg > 0$ .
<code>deg</code>	degree of the polynomial; either 0, 1, or 2 for log-constant, log-linear, and log-quadratic fitting, respectively.
<code>weights</code>	optional vector of weights for individual observations.

**Details**

A gaussian kernel is used in all cases. If `xmin` or `xmax` are finite, the density estimate will be 0 outside of  $[xmin, xmax]$ . A log-transform is used if there is only one boundary (see, Geenens and Wang, 2018); a probit transform is used if there are two (see, Geenens, 2014).

Discrete variables are handled via jittering (see, Nagler, 2018a, 2018b). A specific form of deterministic jittering is used, see `equi_jitter()`.

**Value**

An object of class `kde1d`.

**References**

- Geenens, G. (2014). *Probit transformation for kernel density estimation on the unit interval*. Journal of the American Statistical Association, 109:505, 346-358, [arXiv:1303.4121](#)
- Geenens, G., Wang, C. (2018). *Local-likelihood transformation kernel density estimation for positive random variables*. Journal of Computational and Graphical Statistics, to appear, [arXiv:1602.04862](#)
- Nagler, T. (2018a). *A generic approach to nonparametric function estimation with mixed data*. Statistics & Probability Letters, 137:326–330, [arXiv:1704.07457](#)
- Nagler, T. (2018b). *Asymptotic analysis of the jittering kernel density estimator*. Mathematical Methods of Statistics, in press, [arXiv:1705.05431](#)
- Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. Journal of the Royal Statistical Society, Series B, 53, 683–690.

**See Also**

[dkde1d\(\)](#), [pkde1d\(\)](#), [qkde1d\(\)](#), [rkde1d\(\)](#), [plot.kde1d\(\)](#), [lines.kde1d\(\)](#)

**Examples**

```

## unbounded data
x <- rnorm(500) # simulate data
fit <- kde1d(x) # estimate density
dkde1d(0, fit) # evaluate density estimate
summary(fit) # information about the estimate
plot(fit) # plot the density estimate
curve(dnorm(x),
      add = TRUE, # add true density
      col = "red"
)

## bounded data, log-linear
x <- rgamma(500, shape = 1) # simulate data
fit <- kde1d(x, xmin = 0, deg = 1) # estimate density
dkde1d(seq(0, 5, by = 1), fit) # evaluate density estimate
summary(fit) # information about the estimate
plot(fit) # plot the density estimate
curve(dgamma(x, shape = 1), # add true density
      add = TRUE, col = "red",
      from = 1e-3
)

## discrete data
x <- rbinom(500, size = 5, prob = 0.5) # simulate data
x <- ordered(x, levels = 0:5) # declare as ordered
fit <- kde1d(x) # estimate density
dkde1d(sort(unique(x)), fit) # evaluate density estimate
summary(fit) # information about the estimate
plot(fit) # plot the density estimate
points(ordered(0:5, 0:5), # add true density
       dbinom(0:5, 5, 0.5),
       col = "red"
)

## weighted estimate
x <- rnorm(100) # simulate data
weights <- rexp(100) # weights as in Bayesian bootstrap
fit <- kde1d(x, weights = weights) # weighted fit
plot(fit) # compare with unweighted fit
lines(kde1d(x), col = 2)

```

---

plot.kde1d

*Plotting kde1d objects*


---

**Description**

Plotting kde1d objects

**Usage**

```
## S3 method for class 'kde1d'  
plot(x, ...)  
  
## S3 method for class 'kde1d'  
lines(x, ...)
```

**Arguments**

x	kde1d object.
...	further arguments passed to <a href="#">plot.default()</a>

**See Also**

[kde1d\(\)](#)

**Examples**

```
## continuous data  
x <- rbeta(100, shape1 = 0.3, shape2 = 0.4) # simulate data  
fit <- kde1d(x) # unbounded estimate  
plot(fit, ylim = c(0, 4)) # plot estimate  
curve(dbeta(x, 0.3, 0.4), # add true density  
      col = "red", add = TRUE  
      )  
fit_bounded <- kde1d(x, xmin = 0, xmax = 1) # bounded estimate  
lines(fit_bounded, col = "green")  
  
## discrete data  
x <- rpois(100, 3) # simulate data  
x <- ordered(x, levels = 0:20) # declare variable as ordered  
fit <- kde1d(x) # estimate density  
plot(fit, ylim = c(0, 0.25)) # plot density estimate  
points(ordered(0:20, 0:20), # add true density values  
      dpois(0:20, 3),  
      col = "red"  
      )
```

# Index

dkde1d, 2  
dkde1d(), 3, 5

equi\_jitter, 3  
equi\_jitter(), 5

kde1d, 4  
kde1d(), 3, 7  
kde1d-package, 2

lines.kde1d (plot.kde1d), 6  
lines.kde1d(), 5

pkde1d (dkde1d), 2  
pkde1d(), 3, 5  
pkde1d, (dkde1d), 2  
plot.default(), 7  
plot.kde1d, 6  
plot.kde1d(), 5

qkde1d (dkde1d), 2  
qkde1d(), 3, 5  
qkde1d, (dkde1d), 2

randtoolbox::sobol(), 3  
rkde1d (dkde1d), 2  
rkde1d(), 3, 5