

# Package ‘kernelboot’

February 14, 2020

**Type** Package

**Title** Smoothed Bootstrap and Random Generation from Kernel Densities

**Version** 0.1.7

**Date** 2020-02-13

**Author** Tymoteusz Wolodzko

**Maintainer** Tymoteusz Wolodzko <twolodzko+kernelboot@gmail.com>

**Description** Smoothed bootstrap and functions for random generation from univariate and multivariate kernel densities. It does not estimate kernel densities.

**License** GPL-2

**URL** <https://github.com/twolodzko/kernelboot>

**BugReports** <https://github.com/twolodzko/kernelboot/issues>

**Depends** R (>= 3.1.0)

**LinkingTo** Rcpp

**Imports** Rcpp, future, future.apply

**Suggests** covr, testthat, ks, KernSmooth, cramer

**Encoding** UTF-8

**LazyData** TRUE

**RoxygenNote** 7.0.2

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-02-13 23:10:03 UTC

## R topics documented:

bw.silv . . . . .	2
kernelboot . . . . .	3
kernelboot-class . . . . .	8
rmvg . . . . .	8

rmvk . . . . .	10
ruvk . . . . .	12
summary.kernelboot . . . . .	14

<b>Index</b>	<b>15</b>
--------------	-----------

---

bw.silv	<i>Bandwidth selector for multivariate kernel density estimation</i>
---------	--

---

### Description

Rule of thumb bandwidth selectors for Gaussian kernels as described by Scott (1992) and Silverman (1986).

### Usage

```
bw.silv(x, na.rm = FALSE)
```

```
bw.scott(x, na.rm = FALSE)
```

### Arguments

x	numeric matrix or data.frame.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

### Details

Scott's (1992) rule is defined as

$$H = n^{-2/(m+4)} \hat{\Sigma}$$

Silverman's (1986; see Chacon, Duong and Wand, 2011) rule is defined as

$$H = \left( \frac{4}{n(m+2)} \right)^{2/(m+4)} \hat{\Sigma}$$

where  $m$  is number of variables,  $n$  is sample size,  $\hat{\Sigma}$  is the empirical covariance matrix. The bandwidth is returned as a *covariance matrix*, so to use it for a product kernel, take square root of it's diagonal: `sqrt(diag(H))`.

`bw.silv` corresponds to `Hns` method with `deriv.order=0` from the `ks` package.

## References

- Silverman, B.W. (1986). Density estimation for statistics and data analysis. Chapman and Hall/CRC.
- Wand, M.P. and Jones, M.C. (1995). Kernel smoothing. Chapman and Hall/CRC.
- Scott, D.W. (1992). Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons.
- Chacon J.E., Duong, T. and Wand, M.P. (2011). Asymptotics for general multivariate kernel density derivative estimators. *Statistica Sinica*, 21, 807-840.
- Epanechnikov, V.A. (1969). Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1): 153-158.

## See Also

[bandwidth](#)

---

kernelboot

*Smoothed bootstrap*

---

## Description

Smoothed bootstrap is an extension of standard bootstrap using kernel densities.

## Usage

```
kernelboot(
  data,
  statistic,
  R = 500L,
  bw = "default",
  kernel = c("multivariate", "gaussian", "epanechnikov", "rectangular", "triangular",
    "biweight", "cosine", "optcosine", "none"),
  weights = NULL,
  adjust = 1,
  shrunked = TRUE,
  ignore = NULL,
  parallel = FALSE,
  workers = 1L
)
```

## Arguments

- |                        |  |
|------------------------|--|
| <code>data</code>      | vector, matrix, or data.frame. For non-numeric values standard bootstrap is applied (see below).             |
| <code>statistic</code> | a function that is applied to the data. The first argument of the function will always be the original data. |
| <code>R</code>         | the number of bootstrap replicates.  |

bw	the smoothing bandwidth to be used (see <a href="#">density</a> ). The kernels are scaled such that this is the standard deviation, or covariance matrix of the smoothing kernel. By default <code>bw.nrd0</code> is used for univariate data, and <code>bw.silv</code> is used for multivariate data. When using <code>kernel = "multivariate"</code> this parameter should be a <i>covariance matrix</i> of the smoothing kernel.
kernel	a character string giving the smoothing kernel to be used. This must partially match one of "multivariate", "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine", "optcosine", or "none" with default "multivariate", and may be abbreviated. Using <code>kernel = "multivariate"</code> forces multivariate Gaussian kernel (or univariate Gaussian for univariate data). Using <code>kernel = "none"</code> forces using standard bootstrap (no kernel smoothing).
weights	vector of importance weights. It should have as many elements as there are observations in data. It defaults to uniform weights.
adjust	scalar; the bandwidth used is actually <code>adjust*bw</code> . This makes it easy to specify values like 'half the default' bandwidth.
shrunked	logical; if TRUE random generation algorithm preserves means and variances of the variables. This parameter is ignored for "multivariate" kernel.
ignore	vector of names of columns to be ignored during the smoothing phase of bootstrap procedure (their values are not altered using random noise).
parallel	if TRUE, parallel computing is used (see <a href="#">future_lapply</a> ). <i>Warning:</i> using parallel computing does not necessary have to lead to improved performance.
workers	the number of workers used for parallel computing (see <a href="#">multiprocess</a> ).

## Details

*Smoothed bootstrap* is an extension of standard bootstrap procedure, where instead of drawing samples with replacement from the empirical distribution, they are drawn from kernel density estimate of the distribution.

For smoothed bootstrap, points (in univariate case), or rows (in multivariate case), are drawn with replacement, to obtain samples of size  $n$  from the initial dataset of size  $n$ , as with standard bootstrap. Next, random noise from kernel density  $K$  is added to each of the drawn values. The procedure is repeated  $R$  times and statistic is evaluated on each of the samples.

The noise is added *only* to the numeric columns, while non-numeric columns (e.g. character, factor, logical) are not altered. What follows, to the non-numeric columns and columns listed in `ignore` parameter standard bootstrap procedure is applied.

### Univariate kernel densities

Univariate kernel density estimator is defined as

$$\hat{f}_h(x) = \sum_{i=1}^n w_i K_h(x - y_i)$$

where  $w$  is a vector of weights such that all  $w_i \geq 0$  and  $\sum_i w_i = 1$  (by default uniform  $1/n$  weights are used),  $K_h = K(x/h)/h$  is kernel  $K$  parametrized by bandwidth  $h$  and  $y$  is a vector of data points used for estimating the kernel density.

To draw samples from univariate kernel density, the following procedure can be applied (Silverman, 1986):

*Step 1* Sample  $i$  uniformly with replacement from  $1, \dots, n$ .

*Step 2* Generate  $\varepsilon$  to have probability density  $K$ .

*Step 3* Set  $x = y_i + h\varepsilon$ .

If samples are required to have the same variance as data (i.e. `shrunked = TRUE`), then *Step 3* is modified as following:

*Step 3'*  $x = \bar{y} + (y_i - \bar{y} + h\varepsilon)/(1 + h^2\sigma_K^2/\sigma_Y^2)^{1/2}$

where  $\sigma_K^2$  is variance of the kernel (fixed to 1 for kernels used in this package).

When shrinkage described in *Step 3'* is applied, the smoothed bootstrap density function changes it's form to

$$\hat{f}_{h,b}(x) = (1 + r) \hat{f}_h(x + r(x - \bar{y}))$$

where  $r = (1 + h^2\sigma_K^2/\sigma_Y^2)^{1/2} - 1$ .

This package offers the following univariate kernels:

<i>Gaussian</i>	$\frac{1}{\sqrt{2\pi}} e^{-u^2/2}$
<i>Rectangular</i>	$\frac{1}{2} \mathbf{1}_{( u  \leq 1)}$
<i>Triangular</i>	$(1 -  u ) \mathbf{1}_{( u  \leq 1)}$
<i>Epanchenikov</i>	$\frac{3}{4} (1 - u^2) \mathbf{1}_{( u  \leq 1)}$
<i>Biweight</i>	$\frac{15}{16} (1 - u^2)^2 \mathbf{1}_{( u  \leq 1)}$
<i>Cosine</i>	$\frac{1}{2} (1 + \cos(\pi u)) \mathbf{1}_{( u  \leq 1)}$
<i>Optcosine</i>	$\frac{\pi}{4} \cos\left(\frac{\pi}{2}u\right) \mathbf{1}_{( u  \leq 1)}$

All the kernels are re-scaled so that their standard deviations are equal to 1, so that bandwidth parameter controls their standard deviations.

Random generation from Epanchenikov kernel is done using algorithm described by Devroye (1986). For optcosine kernel inverse transform sampling is used. For biweight kernel random values are drawn from Beta(3, 3) distribution and Beta(3.3575, 3.3575) distribution serves as a close approximation of cosine kernel. Random generation for triangular kernel is done by taking difference of two i.i.d. uniform random variates. To sample from rectangular and Gaussian kernels standard random generation algorithms are used (see `runif` and `rnorm`).

### Product kernel densities

Univariate kernels may easily be extended to multiple dimensions by using product kernel

$$\hat{f}_H(\mathbf{x}) = \sum_{i=1}^n w_i \prod_{j=1}^m K_{h_j}(x_i - y_{ij})$$

where  $w$  is a vector of weights such that all  $w_i \geq 0$  and  $\sum_i w_i = 1$  (by default uniform  $1/n$  weights are used), and  $K_{h_j}$  are univariate kernels  $K$  parametrized by bandwidth  $h_j$ , where  $\mathbf{y}$  is a matrix of data points used for estimating the kernel density.

Random generation from product kernel is done by drawing with replacement rows of  $y$ , and then adding to the sampled values random noise from univariate kernels  $K$ , parametrized by corresponding bandwidth parameters  $h_j$ .

### Multivariate kernel densities

Multivariate kernel density estimator may also be defined in terms of multivariate kernels  $K_H$  (e.g. multivariate normal distribution, as in this package)

$$\hat{f}_H(\mathbf{x}) = \sum_{i=1}^n w_i K_H(\mathbf{x} - \mathbf{y}_i)$$

where  $w$  is a vector of weights such that all  $w_i \geq 0$  and  $\sum_i w_i = 1$  (by default uniform  $1/n$  weights are used),  $K_H$  is kernel  $K$  parametrized by bandwidth matrix  $H$  and  $\mathbf{y}$  is a matrix of data points used for estimating the kernel density.

*Notice:* When using multivariate normal (Gaussian) distribution as a kernel  $K$ , the bandwidth parameter  $H$  is a *covariance matrix* as compared to standard deviations used in univariate and product kernels.

Random generation from multivariate kernel is done by drawing with replacement rows of  $y$ , and then adding to the sampled values random noise from multivariate normal distribution centered at the data points and parametrized by corresponding bandwidth matrix  $H$ . For further details see [rmvg](#).

### References

- Silverman, B. W. (1986). Density estimation for statistics and data analysis. Chapman and Hall/CRC.
- Scott, D. W. (1992). Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons.
- Efron, B. (1981). Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods. *Biometrika*, 589-599.
- Hall, P., DiCiccio, T.J. and Romano, J.P. (1989). On smoothing and the bootstrap. *The Annals of Statistics*, 692-704.
- Silverman, B.W. and Young, G.A. (1987). The bootstrap: To smooth or not to smooth? *Biometrika*, 469-479.
- Scott, D.W. (1992). Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons.
- Wang, S. (1995). Optimizing the smoothed bootstrap. *Annals of the Institute of Statistical Mathematics*, 47(1), 65-80.
- Young, G.A. (1990). Alternative smoothed bootstraps. *Journal of the Royal Statistical Society. Series B (Methodological)*, 477-484.
- De Angelis, D. and Young, G.A. (1992). Smoothing the bootstrap. *International Statistical Review/Revue Internationale de Statistique*, 45-56.
- Polansky, A.M. and Schucany, W. (1997). Kernel smoothing to improve bootstrap confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(4), 821-838.
- Devroye, L. (1986). Non-uniform random variate generation. New York: Springer-Verlag.

Parzen, E. (1962). On estimation of a probability density function and mode. The annals of mathematical statistics, 33(3), 1065-1076.

Silverman, B.W. and Young, G.A. (1987). The bootstrap: To smooth or not to smooth? Biometrika, 469-479.

Jones, M.C. (1991). On correcting for variance inflation in kernel density estimation. Computational Statistics & Data Analysis, 11, 3-15.

### See Also

[bw.silv](#), [density](#), [bandwidth](#), [kernelboot-class](#)

### Examples

```
set.seed(1)

# smooth bootstrap of parameters of linear regression
b1 <- kernelboot(mtcars, function(data) coef(lm(mpg ~ drat + wt, data = data)) , R = 250)
b1
summary(b1)

b2 <- kernelboot(mtcars, function(data) coef(lm(mpg ~ drat + wt, data = data)) , R = 250,
                 kernel = "epanechnikov")
b2
summary(b2)

# smooth bootstrap of parameters of linear regression
# smoothing phase is not applied to "am" and "cyl" variables
b3 <- kernelboot(mtcars, function(data) coef(lm(mpg ~ drat + wt + am + cyl, data = data)) , R = 250,
                 ignore = c("am", "cyl"))
b3
summary(b3)

# standard bootstrap (without kernel smoothing)
b4 <- kernelboot(mtcars, function(data) coef(lm(mpg ~ drat + wt + am + cyl, data = data)) , R = 250,
                 ignore = colnames(mtcars))
b4
summary(b4)

# smooth bootstrap for median of univariate data
b5 <- kernelboot(mtcars$mpg, function(data) median(data) , R = 250)
b5
summary(b5)
```

---

kernelboot-class      *'kernelboot' class object*

---

### Description

'kernelboot' class object

### Details

Object of class "kernelboot", is a list with components including

orig.stat	estimates from statistic on the original data,
boot.samples	samples drawn,
call	function call,
statistic	actual statistic function that was used,
orig.data	original data used for bootstrapping,
variables	used variables: it is NULL for univariate data and for multivariate data it contains two lists of smoothed and i
type	type of kernel density that was used ("univariate", "product", "multivariate"),
param	list of parameters that were used.

param section contains:

R	number of bootstrap iterations,
bw	the bandwidth that was used,
weights	vector of the weights that were applied,
kernel	name of the kernel that was used ("multivariate", "gaussian", "epanechnikov", "rectangular", "triangular", "biv
shrunked	value of the shrunked parameter,
parallel	indicates if parallel computation was used,
random.seed	random seed used to initialize the random number generator (see <a href="#">.Random.seed</a> ).

### See Also

[kernelboot](#)

---

rmvg      *Random generation from multivariate Gaussian kernel density*

---

### Description

Random generation from multivariate Gaussian kernel density

### Usage

```
rmvg(n, y, bw = bw.silv(y), weights = NULL, adjust = 1)
```



**Arguments**

n	number of observations. If $\text{length}(n) > 1$ , the length is taken to be the number required.
y	numeric matrix or data.frame.
bw	numeric matrix with number of rows and columns equal to $\text{ncol}(y)$ ; the smoothing bandwidth to be used. This is the <i>covariance matrix</i> of the smoothing kernel. If provided as a single value, the same bandwidth is used for each variable. If provided as a single value, or as a vector, variables are considered as uncorrelated.
weights	numeric vector of length equal to $\text{nrow}(y)$ ; must be non-negative.
adjust	scalar; the bandwidth used is actually $\text{adjust} \times \text{bw}$ . This makes it easy to specify values like 'half the default' bandwidth.

**Details**

Multivariate kernel density estimator with multivariate Gaussian (normal) kernels  $K_H$  is defined as

$$\hat{f}_H(\mathbf{x}) = \sum_{i=1}^n w_i K_H(\mathbf{x} - \mathbf{y}_i)$$

where  $w$  is a vector of weights such that all  $w_i \geq 0$  and  $\sum_i w_i = 1$  (by default uniform  $1/n$  weights are used),  $K_H$  is kernel  $K$  parametrized by bandwidth matrix  $H$  and  $\mathbf{y}$  is a matrix of data points used for estimating the kernel density.

Random generation from multivariate normal distribution is possible by taking

$$x = A'z + \mu$$

where  $z$  is a vector of  $m$  i.i.d. standard normal deviates,  $\mu$  is a vector of means and  $A$  is a  $m \times m$  matrix such that  $A'A = \Sigma$  ( $A$  is a Cholesky factor of  $\Sigma$ ). In the case of multivariate Gaussian kernel density,  $\mu_i$  is the  $i$ -th row of  $\mathbf{y}$ , where  $i$  is drawn randomly with replacement with probability proportional to  $w_i$ , and  $\Sigma$  is the bandwidth matrix  $H$ .

For functions estimating kernel densities please check **KernSmooth**, **ks**, or other packages reviewed by Deng and Wickham (2011).

**References**

Deng, H. and Wickham, H. (2011). Density estimation in R. <http://vita.had.co.nz/papers/density-estimation.pdf>

**See Also**

[kernelboot](#)

**Examples**

```

set.seed(1)

dat <- mtcars[, c(1,3)]
bw <- bw.silv(dat)
X <- rmvg(5000, dat, bw = bw)

if (requireNamespace("ks", quietly = TRUE)) {

  pal <- colorRampPalette(c("chartreuse4", "yellow", "orange", "brown"))
  col <- pal(10)[cut(ks::kde(dat, H = bw, eval.points = X)$estimate, breaks = 10)]

  plot(X, col = col, pch = 19, axes = FALSE,
       main = "Multivariate Gaussian Kernel")
  points(dat, pch = 2, col = "blue")
  axis(1); axis(2)

} else {

  plot(X, pch = 16, axes = FALSE, col = "#458B004D",
       main = "Multivariate Gaussian Kernel")
  points(dat, pch = 2, col = "red", lwd = 2)
  axis(1); axis(2)

}

```

---

 rmvk

*Random generation from product kernel density*


---

**Description**

Random generation from product kernel density

**Usage**

```

rmvk(
  n,
  y,
  bw = sqrt(diag(bw.silv(y))),
  kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight",
            "cosine", "optcosine"),
  weights = NULL,
  adjust = 1,
  shrinked = FALSE
)

```

**Arguments**

n	number of observations. If <code>length(n) &gt; 1</code> , the length is taken to be the number required.
y	numeric matrix or data.frame.
bw	numeric vector of length equal to <code>ncol(y)</code> ; the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel (see <a href="#">density</a> for details). If provided as a single value, the same bandwidth is used for each variable.
kernel	a character string giving the smoothing kernel to be used. This must partially match one of "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" or "optcosine", with default "gaussian", and may be abbreviated.
weights	numeric vector of length equal to <code>nrow(y)</code> ; must be non-negative.
adjust	scalar; the bandwidth used is actually <code>adjust*bw</code> . This makes it easy to specify values like 'half the default' bandwidth.
shrunked	if TRUE random generation algorithm preserves mean and variances of the individual variables (see <a href="#">ruvk</a> ). Shrinking is applied to each of the variables individually.

**Details**

Product kernel density is defined in terms of independent univariate kernels

$$\hat{f}_H(\mathbf{x}) = \sum_{i=1}^n w_i \prod_{j=1}^m K_{h_j}(x_i - y_{ij})$$

where  $w$  is a vector of weights such that all  $w_i \geq 0$  and  $\sum_i w_i = 1$  (by default uniform  $1/n$  weights are used),  $K_{h_j}$  is univariate kernel  $K$  parametrized by bandwidth  $h_j$ , where  $\mathbf{y}$  is a matrix of data points used for estimating the kernel density.

For functions estimating kernel densities please check **KernSmooth**, **ks**, or other packages reviewed by Deng and Wickham (2011).

For random generation the algorithm described in [kernelboot](#) is used. When using `shrunked = TRUE`, random noise is drawn from independent, shrunked univariate kernels.

**References**

Deng, H. and Wickham, H. (2011). Density estimation in R. <http://vita.had.co.nz/papers/density-estimation.pdf>

**See Also**

[kernelboot](#)

**Examples**

```

dat <- mtcars[, c("mpg", "disp")]

partmp <- par(mfrow = c(1, 2), mar = c(3, 3, 3, 3))

plot(rmvk(5000, dat, shrunked = FALSE), col = "#458B004D", pch = 16,
     xlim = c(0, 45), ylim = c(-200, 800),
     main = "Product kernel", axes = FALSE)
points(dat, pch = 2, lwd = 2, col = "red")
axis(1); axis(2)

plot(rmvk(5000, dat, shrunked = TRUE), col = "#458B004D", pch = 16,
     xlim = c(0, 45), ylim = c(-200, 800),
     main = "Product kernel (shrunked)", axes = FALSE)
points(dat, pch = 2, lwd = 2, col = "red")
axis(1); axis(2)

par(partmp)

cov(dat)
cov(rmvk(5000, dat, shrunked = FALSE))
cov(rmvk(5000, dat, shrunked = TRUE))

```

---

ruvk

*Random generation from univariate kernel density*


---

**Description**

Random generation from univariate kernel density

**Usage**

```

ruvk(
  n,
  y,
  bw = bw.nrd0(y),
  kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight",
            "cosine", "optcosine"),
  weights = NULL,
  adjust = 1,
  shrunked = FALSE
)

```

**Arguments**

**n** number of observations. If  $\text{length}(n) > 1$ , the length is taken to be the number required.

y	numeric vector.
bw	the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel (see <a href="#">density</a> for details).
kernel	a character string giving the smoothing kernel to be used. This must partially match one of "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" or "optcosine", with default "gaussian", and may be abbreviated.
weights	numeric vector of length equal to length(y); must be non-negative.
adjust	scalar; the bandwidth used is actually adjust*bw. This makes it easy to specify values like 'half the default' bandwidth.
shrunked	if TRUE random generation algorithm preserves mean and variance of the original sample.

## Details

Univariate kernel density estimator is defined as

$$\hat{f}_h(x) = \sum_{i=1}^n w_i K_h(x - y_i)$$

where  $w$  is a vector of weights such that all  $w_i \geq 0$  and  $\sum_i w_i = 1$  (by default uniform  $1/n$  weights are used),  $K_h = K(x/h)/h$  is kernel  $K$  parametrized by bandwidth  $h$  and  $y$  is a vector of data points used for estimating the kernel density.

For estimating kernel densities use the [density](#) function.

The random generation algorithm is described in the documentation of [kernelboot](#) function.

## References

Deng, H. and Wickham, H. (2011). Density estimation in R. <http://vita.had.co.nz/papers/density-estimation.pdf>

## See Also

[kernelboot](#), [density](#)

## Examples

```
# ruvk() produces samples from kernel densities as estimated using
# density() function from base R

hist(ruvk(1e5, mtcars$mpg), 100, freq = FALSE, xlim = c(5, 40))
lines(density(mtcars$mpg, bw = bw.nrd0(mtcars$mpg)), col = "red")

# when using 'shrunked = TRUE', the samples differ from density() estimates
# since they are shrunked to have the same variance as the underlying data

hist(ruvk(1e5, mtcars$mpg, shrunked = TRUE), 100, freq = FALSE, xlim = c(5, 40))
lines(density(mtcars$mpg, bw = bw.nrd0(mtcars$mpg)), col = "red")
```

```
# Comparison of different univariate kernels under standard parametrization

kernels <- c("gaussian", "epanechnikov", "rectangular", "triangular",
            "biweight", "cosine", "optcosine")

partmp <- par(mfrow = c(2, 4), mar = c(3, 3, 3, 3))
for (k in kernels) {
  hist(ruvk(1e5, 0, 1, kernel = k), 25, freq = FALSE, main = k)
  lines(density(0, 1, kernel = k), col = "red")
}
par(partmp)
```

---

summary.kernelboot      *Summarize the result of kernelboot*

---

## Description

Summarize the result of kernelboot

## Usage

```
## S3 method for class 'kernelboot'
summary(object, probs = c(0.025, 0.5, 0.975), ..., na.rm = FALSE)
```

## Arguments

object	kernelboot class object.
probs	quantiles returned by summary (see <a href="#">quantile</a> ).
...	further arguments passed to or from other methods.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

# Index

.Random.seed, 8

bandwidth, 3, 7  
bw.nrd0, 4  
bw.scott (bw.silv), 2  
bw.silv, 2, 4, 7

density, 4, 7, 11, 13

future\_lapply, 4

kernelboot, 3, 8, 9, 11, 13  
kernelboot-class, 8

multiprocess, 4

quantile, 14

rmvg, 6, 8  
rmvk, 10  
rnorm, 5  
runif, 5  
ruvk, 11, 12

summary.kernelboot, 14