Package 'demoKde'

May 8, 2026

Type Package

Title Kernel Density Estimation for Demonstration Purposes

Version 1.0.1

Date 2023-08-20

Imports stats

Suggests MASS, graphics

Author Bill Venables

Maintainer Bill Venables <Bill.Venables@gmail.com>

Description Demonstration code showing how (univariate) kernel density estimates are computed, at least conceptually, and allowing users to experiment with different kernels, should they so wish. The method used follows directly the definition, but gains efficiency by replacing the observations by frequencies in a very fine grid covering the sample range. A canonical reference is B. W. Silverman, (1998) <doi:10.1201/9781315140919>. NOTE: the density function in the stats package uses a more sophisticated method based on the fast Fourier transform and that function should be used if computational efficiency is a prime consideration.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-20 05:32:31 UTC

Contents

demoKde-package *Kernel density estimation demonstration and exploration*

Description

Teaching demonstration code for kernel density estimates. KDEs are computed in native R code directly from the definition. The slight innovation here is to replace the observations by their frequencies in a fine partition of the range of the sample. Kernels may be supplied as a function in a standard form, thus allowing alternative kernel functions to be devised and empirically investigated. A wide selection of kernel function is also provided with the package. The canonical reference is B. W. Silverman, (1998). See Refrences.

Author(s)

Bill Venables

Maintainer: Bill Venables, <Bill.Venables@gmail.com>

References

See https://en.wikipedia.org/wiki/Kernel_(statistics) for details of the kernel functions. See also B. W. Silverman, (1998) Density Estimation for Statistics and Data Analysis. Taylor & Franis Group, Boca Raton. doi:10.1201/9781315140919.

See Also

density

Examples

```
if(require("graphics")) {
  with(MASS::Boston, {
      Criminality <- log(crim)
      hist(Criminality, freq=FALSE, main="", border="grey", las=1)
      lines(stats::density(Criminality), col="skyblue", lwd=8)
      lines(kde(Criminality))
      lines(kde(Criminality, kernel = kernelUniform), col="red")
      rug(jitter(Criminality), col="blue")
      legend("topright", c("density histogram",
        "KDE gaussian (denstiy)", "KDE gaussian (kde)",
        "KDE rectangular (kde)"), lty = "solid", lwd=c(1,8,1,1),
        col=c("grey", "skyblue", "black", "red"), bty="n")
  })
}
```

kde *Univariate kernel density estimation directly in R code.*

Description

This function behaves similarly to the `density` function of the **stats** package, but uses only R code. It is a demonstration function intended to show how kernel density estimates are computed, at least conceptually. Unlike `density`, the kernel may be supplied as an R function in a standard form. Example kernel functions are provided. For computational efficiency, the `density` function of the **stats** package is far superior.

Usage

```
kde(x, bw = bw.nrd0, kernel = kernelGaussian, n = 512,
    from = min(x) - cut * sd, to = max(x) + cut * sd,
    adjust = 1, cut = 3, ...)
```

Arguments

`x`	Univeriate sample. Must be numeric.
`bw`	Either an explicit numeric bandwidth to be used for the kernel, or a function used to calculate it.
`kernel`	The kernel function to be used. Must have the same argument sequence as `kernelGaussian`, with the same meanings.
`n`	Then number of points covering the range at which to evaluate the KDE. More gives a smoother display of the result; fewer gives a quicker and more memory efficient computation.
`from`	Lower boundary for the computed KDE.
`to`	Upper boundary for the computed KDE.
`adjust`	Adjustment factor to be used for the bandwidth.
`cut`	Number of bandwidths by which to extend the range of the data for the range of the KDE
`...`	Additional arguments, if needed, to be supplied to the kernel function.

Details

This is a demonstration function intended to show, via R code, the way in which a kernel density estimate is computed.

For samples which are not too large the computation is reasonably efficient, but for serious computations the standard function `density`, or some alternative, should be used.

Value

An object of class "density", with essentially the same structure as objects generated by the `density` of the **stats** package. `plot` and allied methods should apply.

Note

Demonstration code only!

Author(s)

Bill Venables

See Also

kernelBiweight and aliases; density.

Examples

```
if(require("graphics")) {
  with(MASS::geyser, {
      hist(waiting, freq=FALSE, main="", border="grey", las=1)
      lines(stats::density(waiting), col="skyblue", lwd=8)
      lines(kde(waiting))
      lines(kde(waiting, kernel = kernelUniform), col="red")
      rug(jitter(waiting), col="blue")
      legend("topleft", c("density histogram",
        "KDE gaussian (denstiy)", "KDE gaussian (kde)",
        "KDE rectangular (kde)"), lty = "solid", lwd=c(1,8,1,1),
        col=c("grey", "skyblue", "black", "red"), bty="n")
  })
}
```

---

kernelBiweight *Kernel functions for use with kde*

Description

These functions, all with idenical argument lists, provide kernel functions for use with the KDE function.

Usage

```
kernelBiweight(x, mean = 0, sd = 1)
kernelCosine(x, mean = 0, sd = 1)
kernelEpanechnikov(x, mean = 0, sd = 1)
kernelGaussian(x, mean = 0, sd = 1)
kernelLogistic(x, mean = 0, sd = 1)
kernelOptCosine(x, mean = 0, sd = 1)
kernelRectangular(x, mean = 0, sd = 1)
kernelSquaredCosine(x, mean = 0, sd = 1)
kernelTriangular(x, mean = 0, sd = 1)
kernelTricube(x, mean = 0, sd = 1)
kernelTriweight(x, mean = 0, sd = 1)
kernelUniform(x, mean = 0, sd = 1)
```

Arguments

x	Values for which the kernel function is to be evaluated.
mean	Mean (or location parameter) of the kernel function.
sd	Standard deviation (or scale paramenter) of the kernel function.

Details

These are all continuous, symmetric probability density functions parametrised by a location and scale parameter, here taken to be the mean and standard deviation respectively. Most have finite support, he two exceptions here being `kernelGaussian` and `kernelLogistic`, which have unbounded support.

The functions provided cover all those listed in https://en.wikipedia.org/wiki/Kernel_(statistics), with obvious name correspondences. Of the additional ones, `kernelSquaredCosine` appears to be thus far new, and `kernelOptCosine` is explained in the help file for `stats::density`.

The functions `kernelUniform` and `kernelRectangular` are identical, and provided for convenience.

The functions are vectorized with respect to all three parameters.

Value

The evaluated kernel for each supplied `x` value.

Author(s)

Bill Venables

References

See this web site, primarily.

See Also

`kde`, `density`

Examples

```
if(require("graphics")) {
  curve(kernelGaussian, xlim = c(-4.5, 4.5), ylim = c(0, 0.45))
  curve(kernelLogistic, add = TRUE, col = "red")
  curve(kernelUniform, add = TRUE, col = "blue", lwd=2, n = 5000)
}
```

Index