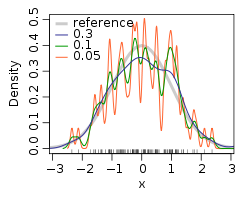
Kernel density estimation

From Wikipedia, the free encyclopedia

[](http://en.wikipedia.org/wiki/File:Kernel_density.svg)

Kernel density estimation of 100 [normally distributed](http://en.wikipedia.org/wiki/Normal_distribution) [random numbers](http://en.wikipedia.org/wiki/Random_number_generator) using different smoothing bandwidths.

In [statistics](http://en.wikipedia.org/wiki/Statistics), **kernel density estimation (KDE)** is a [non-parametric](http://en.wikipedia.org/wiki/Non-parametric_statistics) way to [estimate](http://en.wikipedia.org/wiki/Density_estimation) the [probability density function](http://en.wikipedia.org/wiki/Probability_density_function) of a[random variable](http://en.wikipedia.org/wiki/Random_variable). Kernel density estimation is a fundamental data smoothing problem where inferences about the[population](http://en.wikipedia.org/wiki/Statistical_population) are made, based on a finite data [sample](http://en.wikipedia.org/wiki/Statistical_sample). In some fields such as [signal processing](http://en.wikipedia.org/wiki/Signal_processing) and [econometrics](http://en.wikipedia.org/wiki/Econometrics) it is also termed the *Parzen–Rosenblatt window* method, after [Emanuel Parzen](http://en.wikipedia.org/wiki/Emanuel_Parzen) and [Murray Rosenblatt](http://en.wikipedia.org/wiki/Murray_Rosenblatt), who are usually credited with independently creating it in its current form.[[1]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-Ros1956-1)[[2]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-Par1962-2)

**Contents**

  [[hide](http://en.wikipedia.org/wiki/Kernel_density_estimation)]

[1 Definition](http://en.wikipedia.org/wiki/Kernel_density_estimation#Definition)

[2 Bandwidth selection](http://en.wikipedia.org/wiki/Kernel_density_estimation#Bandwidth_selection)

[2.1 Practical estimation of the bandwidth](http://en.wikipedia.org/wiki/Kernel_density_estimation#Practical_estimation_of_the_bandwidth)

[3 Relation to the characteristic function density estimator](http://en.wikipedia.org/wiki/Kernel_density_estimation#Relation_to_the_characteristic_function_density_estimator)

[4 Statistical implementation](http://en.wikipedia.org/wiki/Kernel_density_estimation#Statistical_implementation)

[5 Examples](http://en.wikipedia.org/wiki/Kernel_density_estimation#Examples)

[5.1 Example in MATLAB-Octave](http://en.wikipedia.org/wiki/Kernel_density_estimation#Example_in_MATLAB-Octave)

[5.2 Example in R](http://en.wikipedia.org/wiki/Kernel_density_estimation#Example_in_R)

[5.3 Example in Python](http://en.wikipedia.org/wiki/Kernel_density_estimation#Example_in_Python)

[6 See also](http://en.wikipedia.org/wiki/Kernel_density_estimation#See_also)

[7 References](http://en.wikipedia.org/wiki/Kernel_density_estimation#References)

[8 External links](http://en.wikipedia.org/wiki/Kernel_density_estimation#External_links)

Definition[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=1" \o "Edit section: Definition)]

Let (*x*1, *x*2, …, *xn*) be an [independent and identically distributed](http://en.wikipedia.org/wiki/Independent_and_identically_distributed_random_variables) sample drawn from some distribution with an unknown [density](http://en.wikipedia.org/wiki/Probability_density_function) *ƒ*. We are interested in estimating the shape of this function *ƒ*. Its *kernel density estimator* is


    \hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^n K_h (x - x_i) \quad = \frac{1}{nh} \sum_{i=1}^n K\Big(\frac{x-x_i}{h}\Big),
  

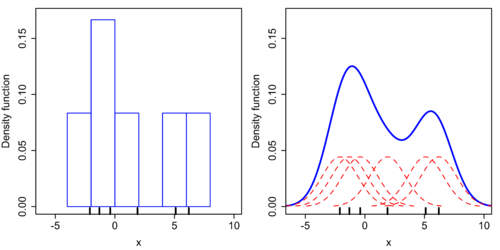
where *K*(•) is the [kernel](http://en.wikipedia.org/wiki/Kernel_(statistics)#In_non-parametric_statistics) — a non-negative function that integrates to one and has mean zero — and *h* > 0 is a [smoothing](http://en.wikipedia.org/wiki/Smoothing) parameter called the *bandwidth*. A kernel with subscript *h* is called the *scaled kernel* and defined as *Kh*(*x*) = 1/*h K*(*x/h*). Intuitively one wants to choose *h* as small as the data allow, however there is always a trade-off between the bias of the estimator and its variance; more on the choice of bandwidth below.

A range of kernel functions are commonly used: [uniform, triangular, biweight, triweight, Epanechnikov](http://en.wikipedia.org/wiki/Kernel_(statistics)#Kernel_functions_in_common_use), [normal](http://en.wikipedia.org/wiki/Normal_distribution), and others. The Epanechnikov kernel is optimal in a mean square error sense,[[3]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-3) though the loss of efficiency is small for the kernels listed previously,[[4]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-WJ1995-4) and due to its convenient mathematical properties, the normal kernel is often used *K*(*x*) = *ϕ*(*x*), where *ϕ* is the [standard normal](http://en.wikipedia.org/wiki/Standard_normal) density function.

The construction of a kernel density estimate finds interpretations in fields outside of density estimation. For example, in [thermodynamics](http://en.wikipedia.org/wiki/Thermodynamics), this is equivalent to the amount of heat generated when [heat kernels](http://en.wikipedia.org/wiki/Heat_kernel) (the fundamental solution to the [heat equation](http://en.wikipedia.org/wiki/Heat_equation)) are placed at each data point locations *xi*. Similar methods are used to construct [discrete Laplace operators](http://en.wikipedia.org/wiki/Discrete_Laplace_operator) on point clouds for [manifold learning](http://en.wikipedia.org/wiki/Manifold_learning).

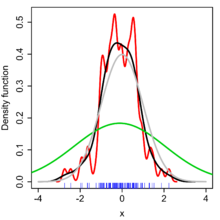
Kernel density estimates are closely related to [histograms](http://en.wikipedia.org/wiki/Histograms), but can be endowed with properties such as smoothness or continuity by using a suitable kernel. To see this, we compare the construction of histogram and kernel density estimators, using these 6 data points: *x*1 = −2.1, *x*2 = −1.3, *x*3 = −0.4, *x*4 = 1.9, *x*5 = 5.1, *x*6 = 6.2. For the histogram, first the horizontal axis is divided into sub-intervals or bins which cover the range of the data. In this case, we have 6 bins each of width 2. Whenever a data point falls inside this interval, we place a box of height 1/12. If more than one data point falls inside the same bin, we stack the boxes on top of each other.

For the kernel density estimate, we place a normal kernel with variance 2.25 (indicated by the red dashed lines) on each of the data points *xi*. The kernels are summed to make the kernel density estimate (solid blue curve). The smoothness of the kernel density estimate is evident compared to the discreteness of the histogram, as kernel density estimates converge faster to the true underlying density for continuous random variables.[[5]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-5)

[](http://en.wikipedia.org/wiki/File:Comparison_of_1D_histogram_and_KDE.png)

Comparison of the histogram (left) and kernel density estimate (right) constructed using the same data. The 6 individual kernels are the red dashed curves, the kernel density estimate the blue curves. The data points are the rug plot on the horizontal axis.

Bandwidth selection[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=2" \o "Edit section: Bandwidth selection)]

[](http://en.wikipedia.org/wiki/File:Comparison_of_1D_bandwidth_selectors.png)

Kernel density estimate (KDE) with different bandwidths of a random sample of 100 points from a standard normal distribution. Grey: true density (standard normal). Red: KDE with h=0.05. Black: KDE with h=0.337. Green: KDE with h=2.

The bandwidth of the kernel is a [free parameter](http://en.wikipedia.org/wiki/Free_parameter) which exhibits a strong influence on the resulting estimate. To illustrate its effect, we take a simulated [random sample](http://en.wikipedia.org/wiki/Random_number_generator) from the standard [normal distribution](http://en.wikipedia.org/wiki/Normal_distribution) (plotted at the blue spikes in the [rug plot](http://en.wikipedia.org/wiki/Carpet_plot) on the horizontal axis). The grey curve is the true density (a normal density with mean 0 and variance 1). In comparison, the red curve is *undersmoothed* since it contains too many spurious data artifacts arising from using a bandwidth *h* = 0.05, which is too small. The green curve is *oversmoothed* since using the bandwidth *h* = 2 obscures much of the underlying structure. The black curve with a bandwidth of *h* = 0.337 is considered to be optimally smoothed since its density estimate is close to the true density.

The most common optimality criterion used to select this parameter is the expected *L*2 [risk function](http://en.wikipedia.org/wiki/Risk_function), also termed the [mean integrated squared error](http://en.wikipedia.org/wiki/Mean_integrated_squared_error):

\operatorname{MISE} (h) = E \int (\hat{f}_h(x) - f(x))^2 \, dx.

Under weak assumptions on *ƒ* and *K*,[[1]](http://en.wikipedia.org/wiki/Kernel_density_estimation" \l "cite_note-Ros1956-1)[[2]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-Par1962-2) MISE (*h*) = AMISE(*h*) + *o(1/(nh) + h4)* where *o* is the [little o notation](http://en.wikipedia.org/wiki/Little_o_notation). The AMISE is the Asymptotic MISE which consists of the two leading terms

\operatorname{AMISE}(h) = \frac{R(K)}{nh} + \frac{1}{4} m_2(K)^2 h^4 R(f'')

where R(g) = \int g(x)^2 \, dx for a function *g*, m_2(K) = \int x^2 K(x) \, dx and *ƒ''* is the second derivative of *ƒ*. The minimum of this AMISE is the solution to this differential equation

 \frac{\partial}{\partial h} \operatorname{AMISE}(h) = -\frac{R(K)}{nh^2} +  m_2(K)^2 h^3 R(f'') = 0 

or

h_{\operatorname{AMISE}} = \frac{ R(K)^{1/5}}{m_2(K)^{2/5}R(f'')^{1/5} n^{1/5}}.

Neither the AMISE nor the *h*AMISE formulas are able to be used directly since they involve the unknown density function *ƒ* or its second derivative *ƒ''*, so a variety of automatic, data-based methods have been developed for selecting the bandwidth. Many review studies have been carried out to compare their efficacities,[[6]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-6)[[7]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-7)[[8]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-8)[[9]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-9)[[10]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-10) with the general consensus that the plug-in selectors[[11]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-SJ91-11) and [cross validation](http://en.wikipedia.org/wiki/Cross-validation_(statistics)) selectors[[12]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-12)[[13]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-13)[[14]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-14) are the most useful over a wide range of data sets.

Substituting any bandwidth *h* which has the same asymptotic order *n*−1/5 as *h*AMISE into the AMISE gives that AMISE(*h*) = *O*(*n*−4/5), where *O* is the [big o notation](http://en.wikipedia.org/wiki/Big_o_notation). It can be shown that, under weak assumptions, there cannot exist a non-parametric estimator that converges at a faster rate than the kernel estimator.[[15]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-15) Note that the *n*−4/5 rate is slower than the typical *n*−1 convergence rate of parametric methods.

If the bandwidth is not held fixed, but is varied depending upon the location of either the estimate (balloon estimator) or the samples (pointwise estimator), this produces a particularly powerful method termed [adaptive or variable bandwidth kernel density estimation](http://en.wikipedia.org/wiki/Variable_kernel_density_estimation).

**Practical estimation of the bandwidth**[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=3" \o "Edit section: Practical estimation of the bandwidth)]

If Gaussian basis functions are used to approximate [univariate](http://en.wikipedia.org/wiki/Univariate" \o "Univariate) data, and the underlying density being estimated is Gaussian then it can be shown that the optimal choice for *h* is[[16]](http://en.wikipedia.org/wiki/Kernel_density_estimation" \l "cite_note-SI1998-16)

h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \approx 1.06 \hat{\sigma} n^{-1/5},

where \hat{\sigma} is the standard deviation of the samples. This approximation is termed the *normal distribution approximation*, Gaussian approximation, or [*Silverman*](http://en.wikipedia.org/wiki/Bernard_Silverman)*'s rule of thumb*.

Relation to the characteristic function density estimator[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=4" \o "Edit section: Relation to the characteristic function density estimator)]

Given the sample (*x*1, *x*2, …, *xn*), it is natural to estimate the [characteristic function](http://en.wikipedia.org/wiki/Characteristic_function_(probability_theory)) *φ*(*t*) = E[*eitX*] as


    \hat\varphi(t) = \frac{1}{n} \sum_{j=1}^n e^{itx_j}
  

Knowing the characteristic function, it is possible to find the corresponding probability density function through the [Fourier transform](http://en.wikipedia.org/wiki/Fourier_transform) formula. One difficulty with applying this inversion formula is that it leads to a diverging integral, since the estimate \scriptstyle\hat\varphi(t) is unreliable for large *t*’s. To circumvent this problem, the estimator \scriptstyle\hat\varphi(t)is multiplied by a damping function *ψh*(*t*) = *ψ*(*ht*), which is equal to 1 at the origin and then falls to 0 at infinity. The “bandwidth parameter” *h* controls how fast we try to dampen the function \scriptstyle\hat\varphi(t). In particular when *h* is small, then *ψh*(*t*) will be approximately one for a large range of *t*’s, which means that \scriptstyle\hat\varphi(t) remains practically unaltered in the most important region of *t*’s.

The most common choice for function *ψ* is either the uniform function *ψ*(*t*) = **1**{−1 ≤ *t* ≤ 1}, which effectively means truncating the interval of integration in the inversion formula to [−1/*h*, 1/*h*], or the [gaussian function](http://en.wikipedia.org/wiki/Gaussian_function" \o "Gaussian function) *ψ*(*t*) = *e−π t*2. Once the function *ψ* has been chosen, the inversion formula may be applied, and the density estimator will be

\begin{align}
    \hat{f}(x) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat\varphi(t)\psi_h(t) e^{-itx}dt
                = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \frac{1}{n} \sum_{j=1}^n e^{it(x_j-x)} \psi(ht) dt \\
               &= \frac{1}{nh} \sum_{j=1}^n \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-i(ht)\frac{x-x_j}{h}} \psi(ht) d(ht)
                = \frac{1}{nh} \sum_{j=1}^n K\Big(\frac{x-x_j}{h}\Big),
  \end{align}

where *K* is the Fourier transform of the damping function *ψ*. Thus the kernel density estimator coincides with the characteristic function density estimator.

Statistical implementation[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=5" \o "Edit section: Statistical implementation)]

A non-exhaustive list of software implementations of kernel density estimators includes:

In [Analytica](http://en.wikipedia.org/wiki/Analytica_(software)" \o "Analytica (software)) release 4.4, the *Smoothing* option for PDF results uses KDE, and from expressions it is available via the built-in Pdf function.

In [C](http://en.wikipedia.org/wiki/C_(programming_language))/[C++](http://en.wikipedia.org/wiki/C%2B%2B), [FIGTree](http://www.umiacs.umd.edu/~morariu/figtree/) is a library that can be used to compute kernel density estimates using normal kernels. MATLAB interface available.

In [C++](http://en.wikipedia.org/wiki/C%2B%2B), [libagf](http://libagf.sf.net/) is a library for [variable kernel density estimation](http://en.wikipedia.org/wiki/Variable_kernel_density_estimation).

In [CrimeStat](http://en.wikipedia.org/wiki/CrimeStat" \o "CrimeStat), kernel density estimation is implemented using five different kernel functions – normal, uniform, quartic, negative exponential, and triangular. Both single- and dual-kernel density estimate routines are available. Kernel density estimation is also used in interpolating a Head Bang routine, in estimating a two-dimensional Journey-to-crime density function, and in estimating a three-dimensional Bayesian Journey-to-crime estimate.

In [ELKI](http://en.wikipedia.org/wiki/ELKI), kernel density functions can be found in the package de.lmu.ifi.dbs.elki.math.statistics.kernelfunctions

In [ESRI](http://en.wikipedia.org/wiki/ESRI) products, kernel density mapping is managed out of the Spatial Analyst toolbox and uses the Quartic(biweight) kernel.

In [Excel](http://en.wikipedia.org/wiki/Microsoft_Excel), the Royal Society of Chemistry has created an add-in to run kernel density estimation based on their [Analytical Methods Committee Technical Brief 4](http://www.rsc.org/Membership/Networking/InterestGroups/Analytical/AMC/Software/kerneldensities.asp).

In [gnuplot](http://en.wikipedia.org/wiki/Gnuplot" \o "Gnuplot), kernel density estimation is implemented by the smooth kdensity option, the datafile can contain a weight and bandwidth for each point, or the bandwidth can be set automatically[[17]](http://en.wikipedia.org/wiki/Kernel_density_estimation" \l "cite_note-17) according to "Silverman's rule of thumb" (see above).

In [Haskell](http://en.wikipedia.org/wiki/Haskell_(programming_language)), kernel density is implemented in the [statistics](http://hackage.haskell.org/package/statistics) package.

In [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), the [Weka (machine learning)](http://en.wikipedia.org/wiki/Weka_(machine_learning)" \o "Weka (machine learning)) package provides [weka.estimators.KernelEstimator](http://weka.sourceforge.net/doc.stable/weka/estimators/KernelEstimator.html), among others.

In [JavaScript](http://en.wikipedia.org/wiki/JavaScript), the visualization package [D3.js](http://en.wikipedia.org/wiki/D3js) offers a KDE package in its science.stats package.

In [JMP](http://en.wikipedia.org/wiki/JMP_(statistical_software)), The Fit Y by X platform can be used to estimate univariate and bivariate kernel densitities.

In [Julia](http://en.wikipedia.org/wiki/Julia_(programming_language)), kernel density estimation is implemented in the [KernelDensity.jl](https://github.com/JuliaStats/KernelDensity.jl) package.

In [MATLAB](http://en.wikipedia.org/wiki/MATLAB), kernel density estimation is implemented through the ksdensity function (Statistics Toolbox). This function does not provide an automatic data-driven bandwidth but uses a [rule of thumb](http://en.wikipedia.org/wiki/Rule_of_thumb), which is optimal only when the target density is normal. A free MATLAB software package which implements an automatic bandwidth selection method[[18]](http://en.wikipedia.org/wiki/Kernel_density_estimation" \l "cite_note-bo10-18) is available from the MATLAB Central File Exchange for [1-dimensional data](http://www.mathworks.com/matlabcentral/fileexchange/14034) and for [2-dimensional data](http://www.mathworks.com/matlabcentral/fileexchange/17204).

In [Mathematica](http://en.wikipedia.org/wiki/Mathematica), numeric kernel density estimation is implemented by the function SmoothKernelDistribution [here](http://reference.wolfram.com/mathematica/ref/SmoothKernelDistribution.html) and symbolic estimation is implemented using the function KernelMixtureDistribution [here](http://reference.wolfram.com/mathematica/ref/KernelMixtureDistribution.html) both of which provide data-driven bandwidths.

In [Minitab](http://en.wikipedia.org/wiki/Minitab), the Royal Society of Chemistry has created a macro to run kernel density estimation based on their [Analytical Methods Committee Technical Brief 4](http://www.rsc.org/Membership/Networking/InterestGroups/Analytical/AMC/Software/kerneldensities.asp).

In the [NAG Library](http://en.wikipedia.org/wiki/NAG_Numerical_Library), kernel density estimation is implemented via the g10ba routine (available in both the Fortran[[19]](http://en.wikipedia.org/wiki/Kernel_density_estimation" \l "cite_note-19) and the C[[20]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-20) versions of the Library).

In [Octave](http://en.wikipedia.org/wiki/GNU_Octave), kernel density estimation is implemented by the kernel\_density option (econometrics package).

In [Origin](http://en.wikipedia.org/wiki/Origin_(software)), 2D kernel density plot can be made from its user interface, and two functions, Ksdensity for 1D and Ks2density for 2D can be used from its [LabTalk](http://wiki.originlab.com/~originla/ltwiki/index.php?title=Category:LabTalk_Programming),[Python](http://en.wikipedia.org/wiki/Python_(programming_language)), or [C](http://en.wikipedia.org/wiki/C_(programming_language)) code.

In [Perl](http://en.wikipedia.org/wiki/Perl), an implementation can be found in the [Statistics-KernelEstimation module](http://search.cpan.org/~janert/Statistics-KernelEstimation-0.05)

In [Python](http://en.wikipedia.org/wiki/Python_(programming_language)), many implementations exist: SciPy (scipy.stats.gaussian\_kde), Statsmodels (KDEUnivariate and KDEMultivariate), and Scikit-learn (KernelDensity) (see comparison[[21]](http://en.wikipedia.org/wiki/Kernel_density_estimation" \l "cite_note-21)).

In [R](http://en.wikipedia.org/wiki/R_(programming_language)), it is implemented through the density and the bkde function in the [KernSmooth library](http://cran.r-project.org/web/packages/KernSmooth/index.html) (both included in the base distribution), the kde function in the [ks library](http://cran.r-project.org/web/packages/ks/index.html), the dkden and dbckden functions in the [evmix library](http://cran.r-project.org/web/packages/evmix/index.html) (latter for boundary corrected kernel density estimation for bounded support), thenpudens function in the [np library](http://cran.r-project.org/web/packages/np/index.html) (numeric and categorical data), the sm.density function in the [sm library](http://cran.r-project.org/web/packages/sm/index.html). For an implementation of the kde.Rfunction, which does not require installing any packages or libraries, see [kde.R](http://www-etud.iro.umontreal.ca/~botev/kde.R).

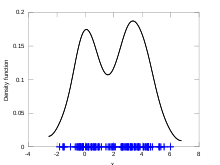
In [SAS](http://en.wikipedia.org/wiki/SAS_(software)), proc kde can be used to estimate univariate and bivariate kernel densities.

In [Stata](http://en.wikipedia.org/wiki/Stata" \o "Stata), it is implemented through kdensity; for example histogram x, kdensity. Alternatively a free Stata module KDENS is available from [here](http://ideas.repec.org/c/boc/bocode/s456410.html)allowing a user to estimate 1D or 2D density functions.

Examples[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=6" \o "Edit section: Examples)]

|  |  |
| --- | --- |
| http://upload.wikimedia.org/wikipedia/commons/thumb/b/b4/Ambox_important.svg/40px-Ambox_important.svg.png | This section **contains**[**instructions, advice, or how-to content**](http://en.wikipedia.org/wiki/Wikipedia:What_Wikipedia_is_not#NOTHOWTO). The purpose of Wikipedia is to present facts, not to train. Please help [improve this article](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit) either by rewriting the how-to content or by [moving](http://meta.wikimedia.org/wiki/Help:Transwiki) it to [Wikiversity](http://en.wikiversity.org/wiki/" \o "v:),[Wikibooks](http://en.wikibooks.org/wiki/) or [Wikivoyage](http://en.wikivoyage.org/wiki/" \o "voy:) *(May 2014)* |

**Example in MATLAB-Octave**[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=7" \o "Edit section: Example in MATLAB-Octave)]

[](http://en.wikipedia.org/wiki/File:1D_kernel_density_estimate.svg)

Kernel density estimate of synthetic data.

For this example, the data are a synthetic sample of 50 points drawn from the standard normal and 50 points from a normal distribution with mean 3.5 and variance 1. The automatic bandwidth selection and density estimation with normal kernels is carried out by [kde.m](http://www.mathworks.com/matlabcentral/fileexchange/14034). This function implements an automatic bandwidth selector that does not rely on the commonly used Gaussian plug-in [rule of thumb](http://en.wikipedia.org/wiki/Rule_of_thumb) heuristic.[[18]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-bo10-18)

randn('seed',8192);

x = [randn(50,1); randn(50,1)+3.5];

[h, fhat, xgrid] = kde(x, 401);

figure;

hold on;

plot(xgrid, fhat, 'linewidth', 2, 'color', 'black');

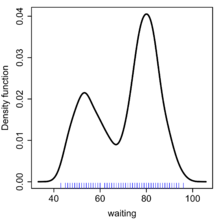
plot(x, zeros(100,1), 'b+');

xlabel('x')

ylabel('Density function')

hold off;

**Example in R**[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=8)]

[](http://en.wikipedia.org/wiki/File:Old_Faithful_Geyser_(waiting_time)_KDE_with_plugin_bandwidth.png)

Kernel density estimate of waiting times of the Old Faithful Geyser.

This example is based on the [Old Faithful Geyser](http://en.wikipedia.org/wiki/Old_Faithful_Geyser), a tourist attraction located in Yellowstone National Park. This famous dataset containing 272 records consists of two variables, eruption duration, and waiting time until next eruption, both in minutes, included in the base distribution of R. We analyse the waiting times, using the ks library since it has a wide range of visualisation options. The bandwidth function is hpi which in turn calls the dpik function in the KernSmooth library: these functions implement the plug-in selector.[[11]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-SJ91-11) The kernel density estimate using the normal kernel is computed usingkde which calls bkde from KernSmooth. The plot function allows the addition of the data points as a rug plot on the horizontal axis. The bimodal structure in the density estimate of the waiting times is clearly seen, in contrast to the rug plot where this structure is not apparent.

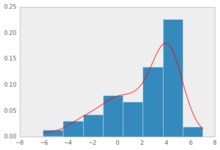
**library**(KernSmooth)

**attach**(**faithful**)

fhat <- bkde(x=waiting)

**plot** (fhat, xlab="x", ylab="Density function")

**Example in Python**[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=9" \o "Edit section: Example in Python)]

[](http://en.wikipedia.org/wiki/File:Plot_of_kernel_density_estimate_using_SciPy.png)

Kernel density estimate of a mixture of normals.

To demonstrate how kernel density estimation is performed in Python, we simulate some data from a mixture of normals, where 50 observations are generated from a normal distribution with mean zero and standard deviation 3 and another 50 from a normal with mean 4 and standard deviation 1.

**import** numpy **as** np

x1 = np.random.normal(0, 3, 50)

x2 = np.random.normal(4, 1, 50)

x = np.r\_[x1, x2]

The gaussian\_kde function from the SciPy package implements a kernel-density estimate using Gaussian kernels, and includes automatic determination of bandwidth. By default, gaussian\_kde uses Scott's rule to select the appropriate bandwidth.[[22]](http://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-22)

**from** scipy.stats **import** kde

**import** matplotlib.pyplot **as** plt

density = kde.gaussian\_kde(x)

xgrid = np.linspace(x.min(), x.max(), 100)

plt.hist(x, bins=8, normed=True)

plt.plot(xgrid, density(xgrid), 'r-')

plt.show()

The plot shows both a histogram of the simulated data, along with a red line that shows the Gaussian KDE.

See also[[edit](http://en.wikipedia.org/w/index.php?title=Kernel_density_estimation&action=edit&section=10)]

|  |  |
| --- | --- |
| http://upload.wikimedia.org/wikipedia/en/thumb/4/4a/Commons-logo.svg/30px-Commons-logo.svg.png | Wikimedia Commons has media related to [***Kernel density estimation***](http://commons.wikimedia.org/wiki/Category:Kernel_density_estimation). |

[Kernel (statistics)](http://en.wikipedia.org/wiki/Kernel_(statistics))

[Kernel smoothing](http://en.wikipedia.org/wiki/Kernel_smoothing)

[Kernel regression](http://en.wikipedia.org/wiki/Kernel_regression)

[Density estimation](http://en.wikipedia.org/wiki/Density_estimation) (with presentation of other examples)

[Mean-shift](http://en.wikipedia.org/wiki/Mean-shift)

[Scale space](http://en.wikipedia.org/wiki/Scale_space) The triplets {(*x*, *h*, KDE with bandwidth *h* evaluated at *x*: all *x*, *h* > 0} form a [scale space](http://en.wikipedia.org/wiki/Scale_space) representation of the data.

[Multivariate kernel density estimation](http://en.wikipedia.org/wiki/Multivariate_kernel_density_estimation)

[Variable kernel density estimation](http://en.wikipedia.org/wiki/Variable_kernel_density_estimation)