The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*), *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$ 98	1-year subscription	$138
2-year subscription	$165	2-year subscription	$245
3-year subscription	$225	3-year subscription	$345
1-year student subscription	$ 75	1-year student subscription	$ 99
1-year institutional subscription	$245	1-year institutional subscription	$285
2-year institutional subscription	$445	2-year institutional subscription	$525
3-year institutional subscription	$645	3-year institutional subscription	$765
Electronic only		**Electronic only**	
1-year subscription	$ 75	1-year subscription	$ 75
2-year subscription	$125	2-year subscription	$125
3-year subscription	$165	3-year subscription	$165
1-year student subscription	$ 45	1-year student subscription	$ 45

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

Users of any of the software, ideas, data, or other materials published in the *Stata Journal* or the supporting files understand that such use is made without warranty of any kind, by either the *Stata Journal*, the author, or StataCorp. In particular, there is no warranty of fitness of purpose or merchantability, nor for special, incidental, or consequential damages such as loss of profits. The purpose of the *Stata Journal* is to promote free communication among Stata users.

The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, stata, Stata Press, Mata, mata, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2014)
14, Number 2, pp. 237–258

Self-consistent density estimation

Joerg Luedicke
Yale University and University of Florida
Gainesville, FL
joerg.luedicke@ufl.edu

Alberto Bernacchia
Jacobs University Bremen
Bremen, Germany
a.bernacchia@jacobs-university.de

Abstract. Estimating a continuous density function from a finite set of data points is an important tool in many scientific disciplines. Popular nonparametric density estimators include histograms and kernel density methods. These methods require the researcher to control the degree of smoothing inherent in an estimated function. In a recent approach, a new method for nonparametric density estimation was proposed that finds the estimate self-consistently, that is without requiring the researcher to choose a smoothing parameter a priori. In this article, we outline the basic ideas of the self-consistent density estimator, and we present a Stata implementation of the method. In addition, we present results of Monte Carlo simulations that show that the self-consistent estimator performs better than other methods, especially for larger data samples.

Keywords: st0334, scdensity, density estimation, kernel density, nonparametric statistics, self-consistent density estimator

1 Introduction: Nonparametric density estimation

Estimating a continuous density function from a finite set of data points is an important tool in virtually any quantitatively oriented scientific field. The most widely used nonparametric methods for estimating probability distribution functions are histograms and kernel density estimators (Härdle et al. 2004; Silverman 1998). Let $X_1, \ldots, X_N$ be a sample of size N from a continuous random variable X with probability density function $f(x)$. The goal of a density estimator is to estimate this function from the sample, and the estimate is denoted as $\widehat{f}(x)$.

Estimating $\widehat{f}(x)$ by plotting a histogram is fairly simple. First, the domain of data points is divided into intervals of width h, or "bins", running from an initial point (origin) x_0 to a final point x_f. The jth bin is denoted by B_j and is identified by the interval

$$B_j = [x_0 + (j-1)h,\ x_0 + jh)$$

where the index j runs from 1 to its maximum value determined by the final point x_f. Then observations X_i that fall into a given interval are counted, and that count, N_j, is divided by the total number of observations N, and divided by bin width h to ensure that the area under the histogram equals 1:

$$f_j = \frac{N_j}{Nh}$$

Finally, the histogram can be plotted by using bars of height f_j and bin width h located at the center of each interval.

We can see that there are two essential ingredients for plotting a histogram that have to be chosen by the researcher a priori: the origin and the bin width. The bin width is usually more crucial because it controls the amount of smoothing inherent in the estimate. If the bin width is small, on average, only a few points will fall in each interval, and the histogram will be characterized by many scattered bars of discrete height. If the bin width is large, many points will fall in each interval; the histogram will have a smooth look but only a few bars will be available, possibly obscuring interesting details of the distribution. A reasonable goal is to estimate a neither over- nor undersmoothed function. As pointed out frequently (for example, Cox [2007]), these choices are usually made on rather arbitrary grounds, and the density estimate itself can lead to quite different conclusions depending on these parameters. See Cox (2007) for a number of illustrative examples.

As an alternative to histograms, kernel methods for density estimation are widely used because of their performance and their being easy to understand and implement. However, as in the case of histograms, similar problems arise. Here we have to determine a kernel function and a bandwidth. The classical kernel estimator can be expressed as

$$\widehat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - X_i}{h}\right) \tag{1}$$

where $K(\cdot)$ denotes the kernel, N is the number of data points, and h is the bandwidth (for example, Silverman [1998]). The choice of a kernel is usually not of great importance. Of greater importance is the bandwidth, which represents the smoothing parameter and has to be fixed beforehand. Similarly to histograms, a resulting estimate can lead to different conclusions, conditional on the amount of smoothing. A small bandwidth determines a harsh density accounting for a lot of possibly spurious detail, while a large bandwidth determines a smoother function, potentially obscuring informative detail. The adjustment of the bandwidth may be inspired by heuristic arguments or formulas or by purely illustrative purposes, and it is for the researcher to decide whether the amount of detail provides relevant information or noise instead. However, if little or nothing is known in advance about the true density, determining how the bandwidth affects the performance of the estimate is a difficult task. Therefore, a method that does not require fixing parameters a priori—and at the same time maintains a high performance—would be desirable.

In this article, we present a method for nonparametric density estimation—the self-consistent method—that does not require any a priori fixing of parameters that are subject to arbitrary choice. This method was presented in detail previously (Bernacchia and Pigolotti 2011). We briefly describe the basic ideas of the self-consistent estimator and present a Stata implementation of the method. Previous simulation results from a set of three test densities (that is, a standard normal distribution, a Cauchy distribution, and a comb distribution; see Bernacchia and Pigolotti [2011]) show that the self-consistent method is performing well in comparison with various other kernel estimators. In this article, we extend the set of test densities and perform Monte Carlo simulations for the standard normal and three different normal mixture distributions.

We conclude with a discussion of some limitations inherent in the new method and some practical implications.

2 The self-consistent method

The self-consistent estimator is motivated by the fact that a kernel can be adjusted optimally if the true density is known. This observation may appear trivial because knowing the true density makes any estimation unnecessary. However, as we explain below, the idea of an optimally adjusted kernel can be used to build a self-consistent estimator. While we start with the assumption of knowing the true density, this assumption will be released at the end of the argument.

For example, suppose we know that the true density is Gaussian. We may take advantage of this knowledge: instead of estimating the density nonparametrically, we could use maximum likelihood (ML) to estimate the parameters (mean and variance) of the Gaussian density from the data sample. If we insist on using a nonparametric approach, the knowledge of the shape of the true density can still be used: it provides a way to find an optimal bin width or bandwidth (Silverman 1998). An even stronger result, shown in Watson and Leadbetter (1963) and Bernacchia and Pigolotti (2011), is that this knowledge allows us to find not just the optimal bandwidth but the optimal profile of the kernel, namely, its complete functional form. In other words, we do not need to define a bandwidth, and we can find the entire shape of the optimal kernel, derived at all of its points. Thus, without the need of a bandwidth h, the estimate can be expressed as

$$\widehat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - X_i)$$

The Fourier transform $k_{\rm opt}(t)$ of the optimal kernel $K_{\rm opt}(x)$ equals

$$k_{\rm opt}(t) = \frac{N}{N - 1 + |\omega(t)|^{-2}}$$

where $\omega(t)$ is the Fourier transform of the true density $f(x)$. To use this formula and build a density estimator, we can use the standard Fourier transform and antitransform (note the change in notation with respect to Bernacchia and Pigolotti [2011], where the Fourier transform is denoted by ϕ). After Fourier antitransforming the kernel from $k_{\rm opt}(t)$ to $K_{\rm opt}(x)$ and plugging this into the classical kernel estimator, we can estimate a density with the optimally shaped kernel:

$$\widehat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K_{\rm opt}(x - X_i) \tag{2}$$

Note the similarity to (1), although in (2), as explained above, bandwidth h is not needed.

In most cases, however, the Fourier transform $\omega(t)$ of the true density is not known in advance, and the above equation cannot be used to obtain a density estimate. To obtain an estimate with neither the need of strong prior knowledge of the true density nor the adjustment of a smoothing parameter, Bernacchia and Pigolotti (2011) have developed a self-consistent estimator. In the following, we show the main underlying arguments and procedure of the self-consistent method and refer to Bernacchia and Pigolotti (2011) for technical details.

The goal is to find a shape of the kernel that is optimal for the density estimate produced by the kernel itself. To achieve this goal, we will construct an iterative procedure that starts with an arbitrary function, assumed to be the true density, and determine the optimal kernel for that density. Let $\widehat{\omega}(t)$ be the Fourier transform of the density estimate in (2). This can be expressed in simple form by using the convolution theorem (Pinsky 2002) and is equal to

$$\widehat{\omega}(t) = \Delta(t) k_{\rm opt}(t) = \frac{N\Delta(t)}{N-1+|\omega(t)|^{-2}} \tag{3}$$

where $\Delta(t)$ is the empirical characteristic function

$$\Delta(t) = \frac{1}{N}\sum_{i=1}^{N} \exp(itX_i) \tag{4}$$

with i being the imaginary unit.

As explained above, (3) cannot be used to obtain a density estimate because the function $\omega(t)$ in the denominator of the right-hand side is unknown. However, we substitute this term with an arbitrary function $\omega(t)$ and obtain a candidate estimate represented by $\widehat{\omega}$. We then obtain a second estimate by applying the kernel that is optimal for $\widehat{\omega}$, namely, the kernel in (3), where ω is substituted by the estimate $\widehat{\omega}$. This procedure can be iterated by obtaining at a given step j an estimate $\widehat{\omega}_{j+1}$ from a kernel that is optimal for the estimate at the previous step $\widehat{\omega}_j$. This implies that the estimate at the jth step is equal to

$$\widehat{\omega}_{j+1} = \frac{N\Delta}{N-1+|\widehat{\omega}_j|^{-2}}$$

The self-consistent estimate is defined as the estimate whose optimal kernel reproduces the estimate itself, that is, the estimate for which $\widehat{\omega}_{j+1}$ is equal to $\widehat{\omega}_j$. It is denoted by $\omega_{\rm sc}$ and satisfies

$$\widehat{\omega}_{\rm sc} = \frac{N\Delta}{N-1+|\widehat{\omega}_{\rm sc}|^{-2}} \tag{5}$$

Bernacchia and Pigolotti (2011) show that the iterative procedure is unnecessary in practice because the exact solution can be derived analytically:

$$\widehat{\omega}_{\rm sc}(t) = \frac{N\Delta(t)}{2(N-1)}\left\{1+\sqrt{1-\frac{4(N-1)}{N^2|\Delta(t)|^2}}\right\} \tag{6}$$

This result is valid only for a subset of frequencies t (Bernacchia and Pigolotti 2011). By straightforward algebra, it is possible to show that the above estimator satisfies the definition of self-consistency introduced above; namely, it satisfies (5). It has also been shown that the estimate is asymptotically consistent; namely, it converges to the true density in the limit of large datasets. The self-consistent estimate in Fourier space can be antitransformed back to real space and then expressed as

$$\widehat{f}_{\mathrm{sc}}(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp(-itx)\widehat{\omega}_{\mathrm{sc}}(t)dt \tag{7}$$

Because the Fourier transform is unitary (Pinsky 2002), the self-consistent estimate satisfies the normalization condition

$$\int_{-\infty}^{\infty} \widehat{f}_{\mathrm{sc}}(x)dx = 1$$

Note that (7) requires the computation of the Fourier transform of a simple function of the data and therefore the computation of the integral in practical applications. This is computed numerically by using a regular grid of points. Technical issues such as the stability and uniqueness of the estimator are described in Bernacchia and Pigolotti (2011).

3 Density correction

A potential drawback of the self-consistent method is that it is not guaranteed to be nonnegative. However, if it happens that an estimate contains negative values and if strict nonnegativity is desired, one can make the density strictly nonnegative without loss of accuracy by using a correction approach described in Glad, Hjort, and Ushakov (2003). The basic idea of this approach is to find the unique and well-identified value ξ that has to be subtracted from the density such that the positive part of the density integrates to 1, after which the remaining negative values can be set to 0. More formally, if the original estimate $\widehat{f}_{\mathrm{sc}}(x)$ contains negative values and if $\int \widehat{f}_{\mathrm{sc}}(x)dx = 1$, then the integral of the positive part of the density is greater than 1:

$$\int \max\left\{0, \widehat{f}_{\mathrm{sc}}(x)\right\} dx \geq 1$$

Glad, Hjort, and Ushakov (2003) show that a unique value ξ can be found such that the modified estimator

$$\widetilde{f}_{\mathrm{sc}}(x) = \max\left\{0, \widehat{f}_{\mathrm{sc}}(x) - \xi\right\}$$

satisfies $\int \widetilde{f}_{\mathrm{sc}}(x)dx = 1$ and is strictly nonnegative by definition. Glad, Hjort, and Ushakov (2003) further demonstrate that the corrected estimate $\widetilde{f}_{\mathrm{sc}}(x)$ is at least as accurate as the original estimate.

We developed two algorithms for finding the unique value ξ. The first algorithm (the default) is fast, while the second is slower but guaranteed to converge. The default search algorithm starts with an initial ballpark estimate of ξ (ξ_s) divided by 10. This initial value, ξ_s, is derived by integrating over the positive part of the density (by using a regular grid of points) to determine excess probability mass, which is then divided by the product of the number of grid points with nonnegative density values (γ) and dx, which here denotes the grid interval used in the numerical computation of the integral:

$$\xi_s = \frac{\int \max\left\{0, \widehat{f}_{\mathrm{sc}}(x)\right\} dx - 1}{\gamma dx}$$

Then ξ is found iteratively by using the search interval δ_s, where δ_s is a constant that defaults to

$$\delta_s = 10\tau\xi_s \tag{8}$$

The search is iterated by adding δ_s to ξ_i at each iteration until the point

$$1 \leq \int \widetilde{f}_{\mathrm{sc}}(x) dx \leq 1 + \tau \tag{9}$$

is reached, where τ is a defined tolerance limit. As can be seen in (8), δ_s is proportional to τ to ensure a sufficiently small resolution of the interval with respect to the tolerance value.

Theoretically, it is not guaranteed that the value ξ is found with this algorithm such that (9) is satisfied, although this will rarely happen in practice. However, the second algorithm could be used in case the default algorithm fails to find ξ. The difference to the default algorithm is that the search interval δ_s is not a constant but a function of ϵ_i^2 ($\delta_i = \xi_i \epsilon_i^2$), where

$$\epsilon_i = \int \widetilde{f}_{\mathrm{sc}}(x)_i dx - 1$$

that is, the excess probability mass at iteration i. This ensures that the smaller the difference is between ξ_i and ξ, the smaller the search interval will get until (9) is satisfied.

Note that the practical implementation of this method includes the case of ξ being negative because the numerically computed integral over a discrete set of grid points deviates from 1 and can be smaller than 1. This implies that the computed integral for the positive part of the density can still be smaller than 1, in which case we would add ξ to the density instead of subtracting it. As a consequence of satisfying the condition in (9) (which would be equivalent to $1 - \tau \leq \int \widetilde{f}_{\mathrm{sc}}(x) dx \leq 1$ in the case of a negative ξ), the corrected estimate will be a renormalized 1. We recommend that users apply the command `scdcor` (see below), which facilitates comparisons between original and corrected estimates.

4 Stata commands

4.1 The scdensity command

The self-consistent method is implemented in Stata as an ado-file, with its main engine written in Mata. The command requires Stata 9.2 or higher and the user-written moremata package (Jann 2005b), which is available from the Statistical Software Components archive (type ssc install moremata in Stata to install it).

The algorithm implements the ideas and formulas described in section 2. In particular, the algorithm reads a list of data points and calculates the Fourier transform of the data points (4) on a grid of t values. The width and spacing of the grid is determined by an iterative procedure that results in a meaningful sampling of the Fourier space (see below). Then the transformed self-consistent estimate is calculated using (6). Note that only real values of (6) represent meaningful solutions; therefore, only values of t for which the argument of the square root in (6) is positive should be considered. For all other values, the estimate is set to 0 (Bernacchia and Pigolotti 2011). The grid of t values at which the transformed estimate is calculated is determined by those meaningful solutions and is chosen in such a way that approximately half the grid points correspond to a value of t giving a nonzero estimate. Finally, the self-consistent estimate is calculated using (7), namely, by antitransforming the estimate previously obtained. The grid of n points at which the estimate is evaluated can be determined by the input options of the Stata command. Note that as of the current version (version 1.0.1), no method is implemented for density estimation with bounded variables.

What follows is an overview of scdensity's syntax and options.

Syntax

```
scdensity varname [if] [in] [, n(#) range(# #) expand at(varname)
    correction gtd tolerance(#) initial(#) interval(#)
    generate(newvar1 [newvar2]) nograph twoway_options]
```

Options

n(#) specifies the number of grid points to be used at which the density is evaluated. If the number of data points is greater than $N = 1000$, the default is n(1000) grid points. If the number of data points is lower than $N = 1000$, the number of grid points defaults to $n = N$. If a number larger than the actual sample size is requested, then n is set to N.

range(# #) defines the grid range at which the density is to be evaluated. By default, the endpoints of the evaluation grid are determined by the minimum and maximum values of the actual data points; the range() option can be used to change this default behavior. The input of two numbers is required, with the first one being the minimum and the second one being the maximum of the range.

`expand` expands the evaluation grid. `scdensity`'s default is to use the endpoints of the data range as grid endpoints. If the `expand` option is used, the grid range is expanded at both ends as a function of sample size. Let the width of the data range be $w = \max(x) - \min(x)$, where x are the data points; then the expanded range r_e is defined by $\min(r_e) = \min(x) - 0.5N^{-0.3}w$ and $\max(r_e) = \max(x) + 0.5N^{-0.3}w$, with N being the sample size of x.

`at(`*varname*`)` evaluates the density at *varname*. If *varname* is not a regular grid of points, then densities that contain negative values cannot be corrected.

`correction` specifies that a correction be applied so that the density will be strictly nonnegative. The unique and well-identified value ξ is found such that the positive part of the density integrates to 1 (plus tolerance) when ξ is subtracted from the density, after which the negative part is set to 0. This approach is described in Glad, Hjort, and Ushakov (2003). A search algorithm is implemented that finds ξ. The tolerance τ defaults to $1e^{-4}$. All defaults can be changed using the `tolerance()`, `initial()`, and `interval()` options. Changing the `initial()` and `interval()` defaults will rarely be needed.

`gtd` specifies an alternative algorithm for finding ξ. The default algorithm usually finds ξ fast and reliably. Theoretically, however, it might not find ξ, in which case an alternative algorithm can be used by specifying the `gtd` option. With this alternative algorithm, ξ is found, but the algorithm can take a substantial amount of time, especially for very small tolerance values.

`tolerance(`*#*`)` changes the default tolerance τ.

`initial(`*#*`)` changes the initial value of ξ at which the search is started.

`interval(`*#*`)` changes the default search interval δ_s.

`generate(`*newvar1* [*newvar2*]`)` stores the density estimate in *newvar1* and the evaluation grid in *newvar2*.

`nograph` suppresses the graph.

twoway_options are any options other than `by()` documented in [G-3] ***twoway_options***.

Stored results

`scdensity` stores the following in `r()`:

Scalars

`r(n_data)`	number of data points	`r(range_min)`	minimum grid point
`r(n_points)`	number of evaluation points	`r(range_max)`	maximum grid point

4.2 The scdcor command

The `scdcor` command is a convenience wrapper and plots the original and corrected density estimates overlaid in one graph. This can be useful to compare original and corrected estimates if a nonnegativity correction is desired. Additionally, a kernel-density estimate can be added to the graph for which the number of grid points and the grid range will be the same as for the self-consistent estimates. The kernel estimates are obtained with `kdens`, which needs to be installed to use the added kernel functionality of `scdcor` (`kdens` is available from the Statistical Software Components archive; type `ssc install kdens` in Stata to install it). The syntax and options of `scdcor` follow.

Syntax

`scdcor` *varname* [*if*] [*in*] [`,` `addkde(`*kernel*`)` `bw(`# | *string*`)` `adaptive` `n(`#`)`
`range(`# #`)` `expand` `gtd` `tolerance(`#`)` `initial(`#`)` `interval(`#`)`
`cline1opts(`*cline_options*`)` `cline2opts(`*cline_options*`)`
`cline3opts(`*cline_options*`)` *twoway_options*]

Options

`addkde(`*kernel*`)` adds a kernel estimate. *kernel* can be any type of kernel that is supported by `kdens`. The evaluation grid is the same as for the self-consistent estimate (that is, range and number of grid points).

`bw(`# | *string*`)` specifies the smoothing parameter for the kernel estimate, which can be either a positive real number or an automatic bandwidth selector of `kdens`. The default is `bw(silverman)`.

`adaptive` specifies that a variable bandwidth be used.

`n(`#`)` specifies the number of grid points to be used at which the density is evaluated. If the number of data points is greater than $N = 1000$, the default is `n(1000)`. If the number of data points is lower than $N = 1000$, the number of grid points defaults to $n = N$. If a number larger than the actual sample size is requested, then n is set to N.

`range(`# #`)` defines the grid range at which the density is to be evaluated. By default, the endpoints of the evaluation grid are determined by the minimum and maximum values of the actual data points; the `range()` option can be used to change this default behavior. The input of two numbers is required, with the first one being the minimum and the second one being the maximum of the range.

`expand` expands the evaluation grid as a function of sample size (see `scdensity` for details). The default grid range is determined by the endpoints of the data range.

`gtd` uses the alternative algorithm to find ξ. See `scdensity` for further details about this and the default algorithm.

`tolerance(`#`)` changes the default tolerance τ.

`initial(`#`)` changes the initial value of ξ at which the search is started.

`interval(#)` changes the default search interval δ_s.

`cline1opts(`*cline_options*`)` specifies any options documented in [G-3] ***cline_options*** for the original estimate.

`cline2opts(`*cline_options*`)` specifies any options documented in [G-3] ***cline_options*** for the corrected estimate.

`cline3opts(`*cline_options*`)` specifies any options documented in [G-3] ***cline_options*** for the added kernel estimate.

twoway_options are any options other than `by()` documented in [G-3] ***twoway_options***.

5 Monte Carlo simulations

5.1 Experimental setup

To evaluate the accuracy of the new method, we compare self-consistent estimates with several kernel-density estimates and ML fits by using Monte Carlo simulations. As a measure of accuracy, the mean integrated squared error (MISE) is used. MISE is a global measure and captures the error across an entire distribution. Thus more specific aspects of a distribution (for example, tails and mode) are not explicitly considered here. MISE can be expressed as (Silverman 1998)

$$\text{MISE}\left(\widehat{f}\right) = E\int\left\{\widehat{f}(x) - f(x)\right\}^2 dx \tag{10}$$

where E denotes the average over the Monte Carlo replications. Two different kernel functions and three different bandwidth rules are used for comparisons. In addition, a varying bandwidth estimator is used, and the errors of (parametric) ML estimates are used as benchmarks.

We used four different test densities, which are shown in figure 1. We first used a normal distribution with density function

$$f(x) = \phi(\mu, \sigma^2) = (2\pi)^{-\frac{1}{2}}\sigma^{-1}\exp\left\{-0.5(x-\mu)^2/\sigma^2\right\}$$

where $\mu = 0$ and $\sigma^2 = 1$, that is, the standard normal distribution. We then used three normal mixture distributions (McLachlan and Peel 2000): the two-component mixture with different means, equal variances, and equal-component probabilities,

$$f(x) = 0.5\phi(0,1) + 0.5\phi(3,1)$$

the two-component mixture with different means, different variances, and equal-component probabilities,

$$f(x) = 0.5\phi(0,1) + 0.5\phi(5,2^2)$$

and the three-component mixture,

$$f(x) = 0.5\phi(0,1.2^2) + 0.25\phi(4,1.4^2) + 0.25\phi(8,0.6^2)$$

Four different sample sizes were used in the simulations: $N = 100$, $N = 1000$, $N = 10000$, and $N = 100000$.

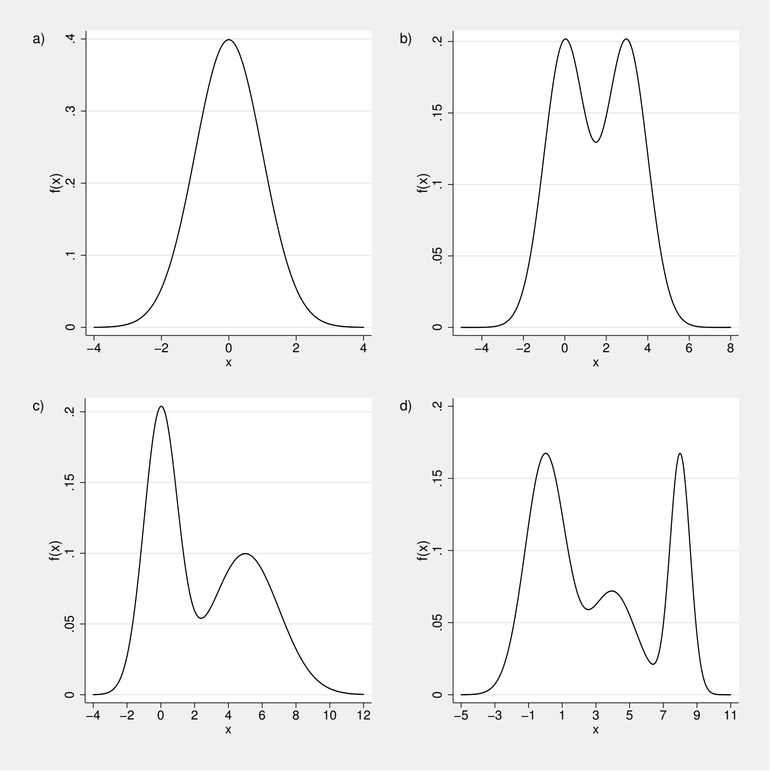

Figure 1. True density functions used as test densities in the simulations: a) $f(x) = \phi(0,1)$; b) $f(x) = 0.5\phi(0,1) + 0.5\phi(3,1)$; c) $f(x) = 0.5\phi(0,1^2) + 0.5\phi(5,2^2)$; d) $f(x) = 0.5\phi(0,1.2^2) + 0.25\phi(4,1.4^2) + 0.25\phi(8,0.6^2)$

For kernel-density estimation in the simulations, we used the user-written `kdens` package (Jann 2005a), which allows for a computationally efficient approximate estimation. The approximate estimator implemented in `kdens` is based on a linear-binning algorithm (see Jann [2007] for details). To ensure that the approximate estimation yields accurate MISEs, we ran simulations for one of our test densities to compare approximate and exact kernel-density estimates. Results indicated that these two estimation methods yielded the same MISEs as long as the number of grid points was sufficiently large relative to the sample size (see the *Appendix* and Hall and Wand [1996]).

If $X_1, \ldots, X_N$ is a sample of data points drawn from population X with density $f(x)$, the exact kernel estimator is defined by (1). We used an Epanechnikov kernel with Silverman's optimal bandwidth rule (Silverman 1998),

$$h_o = 0.9\min(\sigma, \textsc{iq}/1.349)N^{-\frac{1}{5}} \tag{11}$$

which is the default for both the kernel function and the bandwidth choice in Stata's official `kdensity` command as well as the user-written `kdens` package (Jann 2005a). We also used an Epanechnikov kernel for the varying bandwidth estimator. In addition to the Epanechnikov kernel, we used Gaussian kernels with Silverman's rule from above, Härdle's "better" rule of thumb (Härdle et al. 2004),

$$h_o = 1.06\min(\sigma, \textsc{iq}/1.349)N^{-\frac{1}{5}} \tag{12}$$

and Scott's oversmoothed bandwidth (Scott 1992),

$$h_o \geq 1.144\sigma N^{-\frac{1}{5}} \tag{13}$$

Density estimation with variable bandwidths—also known as adaptive kernel-density estimation—can be expressed as

$$\widehat{f}(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h_i}K\left(\frac{x - X_i}{h_i}\right) \tag{14}$$

where h_i is the local bandwidth determined by

$$h_i = h\lambda_i$$

for which the local bandwidth factor λ_i is estimated as

$$\widehat{\lambda_i} = \sqrt{\frac{G\left\{\widehat{f}(X)\right\}}{\widehat{f}(X_i)}}, \qquad i = 1, \ldots, N$$

where $G(\cdot)$ denotes the geometric mean of a preliminary fixed bandwidth estimate over all i (see Abramson [1982]; Van Kerm [2003]).

To provide an overall benchmark, we also calculated MISE for (parametric) ML estimates. For the standard normal distribution, the error was derived analytically, using the following formula:

$$\text{MISE}_{\text{ML}\phi(0,1)} = \frac{7}{16\sqrt{\pi}}N^{-1}$$

For the mixture distributions, parameters were estimated with ML by using the user-written package `fmm` (Deb 2007). The evaluation grid was the same for all estimates in each Monte Carlo replication. A grid of $n = 100$ points was used for sample size $N = 100$, and a grid of $n = 1000$ points was used for larger samples. The same grids were used for the computation of the integrals in (10). Because the `scdensity` command uses the endpoints of the actual data points as grid endpoints by default, the `kdens` command was slightly modified to have a comparable grid in each replication. Specifically, the `kdens_grid()` function from `kdens.mata` was modified such that the minimum and the maximum values of the data vector were taken as the endpoints of the evaluation grid instead of the bandwidth-adjusted grid range.

5.2 Results

Results are presented in figures 2–5. These graphs show MISE as a function of sample size (N) for each density estimator, with both MISE and N plotted on the logarithmic scale. All lines are sloping downward because the error of an estimator decreases with larger samples. The steeper a line is, the faster the corresponding estimator will converge to the true density.

Figure 2 shows the simulation results for the standard normal distribution. For small sample sizes, all nonparametric estimates have a similar error. However, the distance between the self-consistent and the kernel estimates increases with larger sample sizes, and the self-consistent estimates are more accurate than all kernel estimates. The worst estimate for the simple Gaussian distribution is the variable bandwidth estimator. As expected, the ML estimator performs best. However, one needs to keep in mind that this is a parametric ML estimator, which requires prior knowledge about the shape of the distribution.

Figure 3 shows results for the mixture distribution depicted in figure 1b. Again, for small sample sizes, all estimators lead to similar errors, which for this distribution is also true for the parametric ML estimator. Only when the sample size grows larger is the ML estimator more accurate than kernel-density estimates. However, the self-consistent method provides estimates that are almost as accurate as the ML estimates. Remember that the self-consistent estimate does not rely on any a priori assumptions (besides assuming a smooth function). In contrast, the ML estimate assumes that the underlying density is a distributional mixture, that it is a mixture of exactly two distributions, and that these two distributions are both Gaussians.

Similar conclusions can be drawn with respect to the third test density, depicted in figure 1c. However, for this distribution, the adaptive bandwidth estimator is performing clearly better than the other kernel estimates (figure 4). Note that the test density here is a mixture where the two components have different variances, a situation to which the varying bandwidth estimator seems to adapt well. For small sample sizes, it performs slightly better than the self-consistent method and almost as good as ML. For moderate-sized samples (that is, $N = 1000$), the errors of both the adaptive bandwidth and the self-consistent estimators are roughly the same. For larger samples (that is, $N = 10000$ and $N = 100000$), the self-consistent method performs slightly better and scales similar to the ML error.

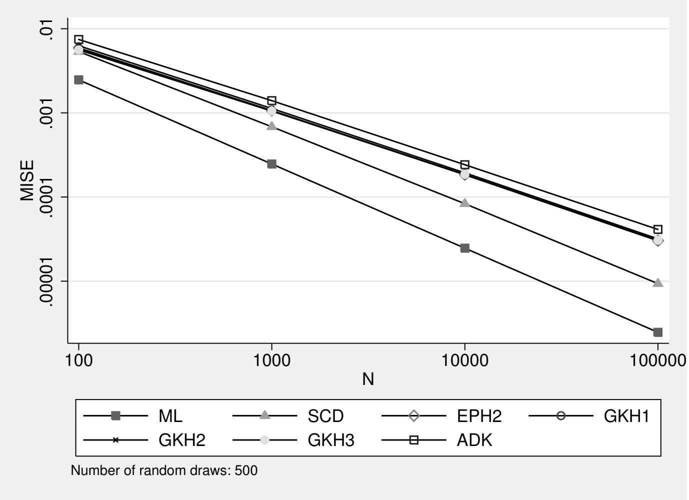

Figure 2. Accuracy of density estimates for the standard normal distribution $f(x) = \phi(0,1)$, figure 1a, measured by the mean integrated squared error (MISE) as a function of sample size N; ML = maximum likelihood; SCD = self-consistent method; EPH2 = Epanechnikov kernel with bandwidth $h_o = 0.9\min(\sigma, \text{IQ}/1.349)N^{-\frac{1}{5}}$ (Stata's default); GKH1 = Gaussian kernel with bandwidth $h_o = 1.06\min(\sigma, \text{IQ}/1.349)N^{-\frac{1}{5}}$; GKH2 = Gaussian kernel with bandwidth $h_o = 0.9\min(\sigma, \text{IQ}/1.349)N^{-\frac{1}{5}}$; GKH3 = Gaussian kernel with bandwidth $h_o \geq 1.144\sigma N^{-\frac{1}{5}}$; ADK = variable bandwidth Epanechnikov kernel

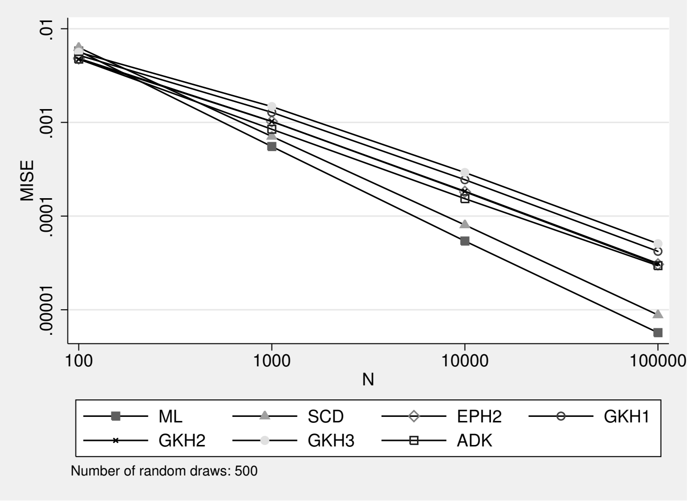

Figure 3. Accuracy of density estimates for mixture distribution $f(x) = 0.5\phi(0,1)0.5\phi(3,1)$, figure 1b, measured by MISE as a function of sample size N

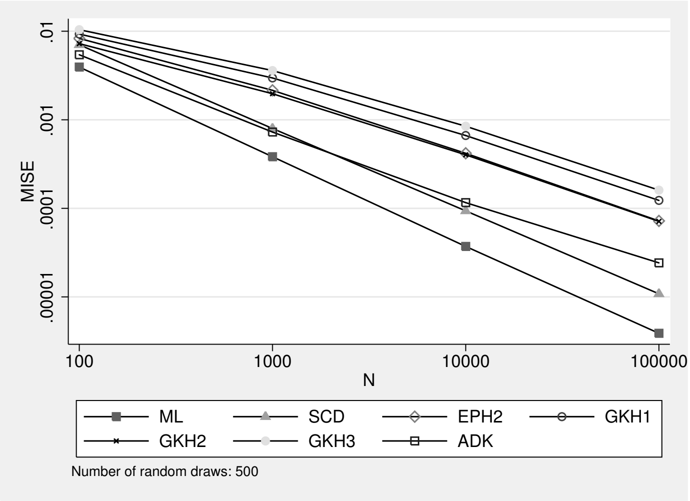

Figure 4. Accuracy of density estimates for mixture distribution $f(x) = 0.5\phi(0,1) + 0.5\phi(5,2^2)$, figure 1c, measured by MISE as a function of sample size N

Finally, figure 5 shows results for the three-component mixture for which the true density is depicted in figure 1d. While the adaptive bandwidth estimator is again the best among the kernel estimates, the self-consistent method is performing much better for this test density. Its error is equivalent to the ML error for small samples and then somewhat worse once the sample size increases. Again this is without making any prior assumptions about the nature of the density, while the ML estimate relies on the (in this case true) assumptions of a three-component mixture distribution for the single components being all Gaussians.

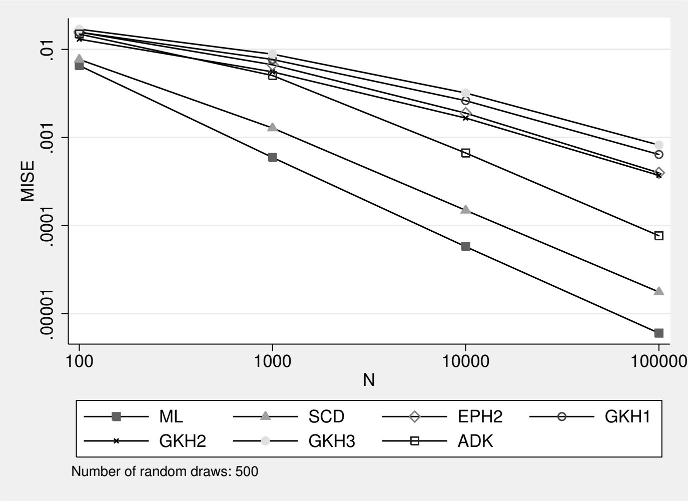

Figure 5. Accuracy of density estimates for mixture distribution $f(x) = 0.5\phi(0, 1.2^2) + 0.25\phi(4, 1.4^2) + 0.25\phi(8, 0.6^2)$, figure 1d, measured by MISE as a function of sample size N

6 Conclusions

Given the test densities and kernel-density estimators used in the simulations, the self-consistent method was the most accurate among the nonparametric estimators. For one of the test densities $[f(x) = 0.5\phi(0, 1){+}0.5\phi(3, 1)]$, the self-consistent method performed nearly as well as the (parametric) ML estimate without relying on any prior assumptions or on fixing parameters. Thus the self-consistent method is a very promising new approach in the context of nonparametric statistics. The self-consistent method may also be generalized to more than one dimension or used in the context of nonparametric regression models; however, because this is an active field of research, mathematical theory has yet to be derived for such applications. However, a self-consistent estimator for bivariate-density estimation will probably be implemented in a future version of `scdensity`.

The question remains whether the observed differences in bias have any practical implications. Using real data, we estimated the distribution function for body height from an adult population of humans ($N = 10351$ males and females; variable `height` is from the Stata example dataset `nhanes2`; type `webuse nhanes2` in Stata to load these data). Figure 6 shows results for fixed and variable bandwidth kernel estimates, a self-consistent estimate, and an ML estimate, respectively. Among the nonparametric methods, the self-consistent estimate allows for a less ambiguous interpretation of the distribution function as a mixture of two height distributions, putatively a mixture of female and male heights, and looks very similar to the ML estimate, which is assuming a mixture distribution of two Gaussians. In contrast, the fixed and the variable bandwidth estimates look noisy right around the modal area of the functions and seem difficult to interpret. Thus the self-consistent method might indeed be very useful in practice.

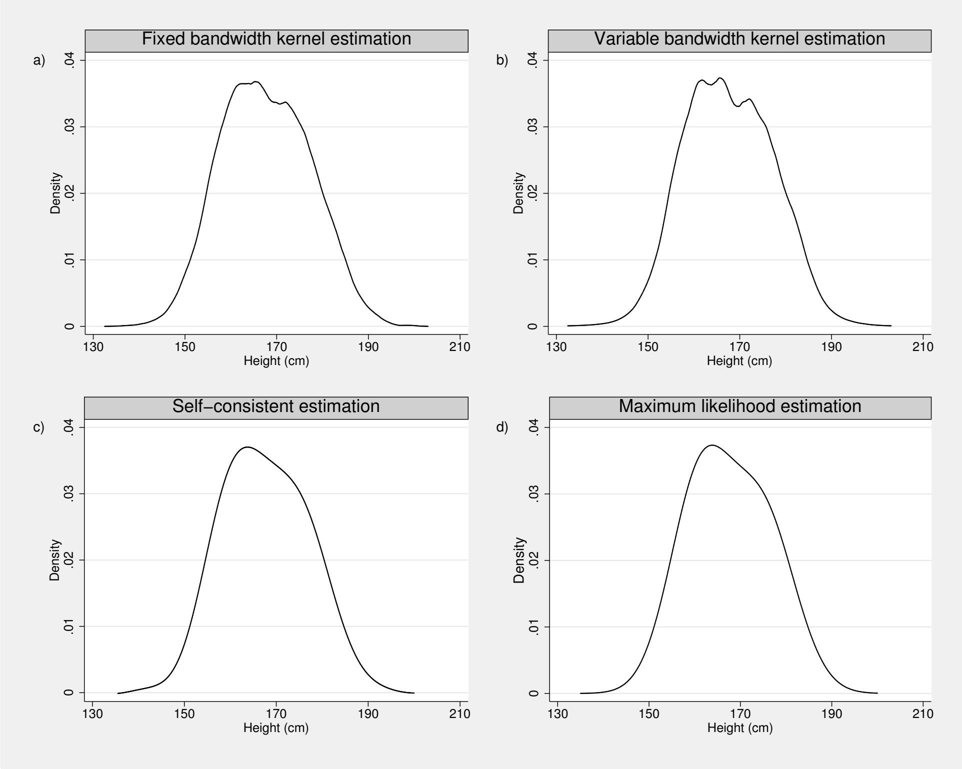

Figure 6. Comparison of density estimates using real data. Data are variable `height` from the Stata example dataset `nhanes2`, measuring body height of $N = 10351$ male and female humans. Epanechnikov kernels are used for the kernel estimates with bandwidth $h_o = 0.9\min(\sigma, \text{IQ}/1.349)N^{-\frac{1}{5}}$ for the fixed bandwidth estimate (Stata's default). The ML estimate is based on the assumption of a mixture distribution of two Gaussians.

However, the self-consistent estimate is not always expected to work well: for example, for specific families of density functions that include a) discontinuous densities (that is, densities having abrupt changes of probability) and b) diverging densities (that is, densities having an infinite value at given points) (Bernacchia and Pigolotti 2011). Densities of type (a) also include densities defined only for positive numbers and are discontinuous at 0, for example, the exponential density. Probability functions of type (b) include, for example, the χ^2 distribution with one degree of freedom, which is infinite at 0. Densities that are defined only for positive numbers but do not have a discontinuity at 0 may still be well captured by the self-consistent method. Those include, for example, the χ^2 distribution of degree 3 and higher. However, because those densities may have a discontinuous derivative, the performance of the self-consistent estimator is not expected to be excellent in these instances. Finally, discrete data are usually not suitable for analysis with the self-consistent method and can be better analyzed by simple (discrete) histograms or similar methods.

7 References

Abramson, I. S. 1982. On bandwidth variation in kernel estimates—A square root law. *Annals of Statistics* 10: 1217–1223.

Bernacchia, A., and S. Pigolotti. 2011. Self-consistent method for density estimation. *Journal of the Royal Statistical Society, Series B* 73: 407–422.

Cox, N. J. 2007. Kernel estimation as a basic tool for geomorphological data analysis. *Earth Surface Processes and Landforms* 32: 1902–1912.

Deb, P. 2007. fmm: Stata module to estimate finite mixture models. Statistical Software Components S456895, Department of Economics, Boston College.
http://ideas.repec.org/c/boc/bocode/s456895.html.

Glad, I. K., N. L. Hjort, and N. G. Ushakov. 2003. Correction of density estimators that are not densities. *Scandinavian Journal of Statistics* 30: 415–427.

Hall, P., and M. P. Wand. 1996. On the accuracy of binned kernel density estimators. *Journal of Multivariate Analysis* 56: 165–184.

Härdle, W. K., M. Müller, S. Sperlich, and A. Werwatz. 2004. *Nonparametric and Semiparametric Models*. Berlin: Springer.

Jann, B. 2005a. kdens: Stata module for univariate kernel density estimation. Statistical Software Components S456410, Department of Economics, Boston College.
http://ideas.repec.org/c/boc/bocode/s456410.html.

———. 2005b. moremata: Stata module (Mata) to provide various functions. Statistical Software Components S455001, Department of Economics, Boston College.
http://ideas.repec.org/c/boc/bocode/s455001.html.

———. 2007. Univariate kernel density estimation. Technical report, Online publication.
http://fmwww.bc.edu/RePEc/bocode/k/kdens.pdf.

McLachlan, G., and D. Peel. 2000. *Finite Mixture Models*. New York: Wiley.

Pinsky, M. A. 2002. *Introduction to Fourier Analysis and Wavelets*. Pacific Grove, CA: Brooks/Cole.

Scott, D. W. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: Wiley.

Silverman, B. W. 1998. *Density Estimation for Statistics and Data Analysis*. Boca Raton, FL: Chapman & Hall/CRC.

Van Kerm, P. 2003. Adaptive kernel density estimation. *Stata Journal* 3: 148–156.

Watson, G. S., and M. R. Leadbetter. 1963. On the estimation of the probability density, I. *Annals of Mathematical Statistics* 34: 480–491.

About the authors

Joerg Luedicke is a statistical consultant for the Rudd Center for Food Policy and Obesity at Yale University in New Haven, CT, and an adjunct research professor (by courtesy) in the Department of Psychology at the University of Florida in Gainesville, FL.

Alberto Bernacchia is a professor of neuroscience in the School of Engineering and Science at Jacobs University in Bremen, Germany.

A Appendix

A.1 Introduction: Approximate versus exact kernel-density estimation

For reasons of increased computational efficiency, Hall and Wand (1996) proposed estimating a kernel density approximately by binning the data instead of doing an exact estimation using raw data. Existing evidence from simulation studies shows that the approximated and the exact estimates are of equivalent accuracy if the number of grid points at which a density is evaluated is sufficiently large (Hall and Wand 1996). The differences between approximate and exact estimates in terms of processing time can be substantial. For example, an exact estimation for 100,000 normally distributed data points takes more than 4 seconds, while the approximate estimate takes only roughly 0.2 seconds on a modern desktop machine (using the user-written `kdens` package of Jann [2005a] for both methods). It is therefore desirable to use the approximated estimate, especially in the context of a simulation study where the density is repeatedly estimated and where the sample sizes become large. However, one needs to check whether the two estimators are indeed equivalent with respect to the specific simulation study setup. Demonstrating this is the purpose of this appendix.

The formulas for exact kernel-density estimation are shown in (1) (for the fixed bandwidth estimator) and (14) (for the variable bandwidth estimator). The linear-binning approach implemented in `kdens` is described in Hall and Wand (1996), and methods and formulas of the `kdens` implementation are concisely documented in Jann (2007). In a nutshell, the data are preprocessed by assigning data points to grid points, after which bin counts at M equally spaced grid points can be calculated. The density function is then evaluated at those grid points, and the grid counts are used instead of the actual data points. In the case of linear binning as implemented in `kdens`, linear interpolation is used for the assignment of data points to grid points to weigh the contribution of each data point to a given grid count.

A.2 Monte Carlo setup

To compare approximate and exact estimates, we used one of the test densities from the main Monte Carlo study [$f(x) = 0.5\phi(0,1) + 0.5\phi(5,2^2)$] and a number of different kernel functions and bandwidth rules. Specifically, two different kernel functions are used, an Epanechnikov and a Gaussian kernel. For the Gaussian kernel, the same bandwidth rules as in the main study were chosen. The Epanechnikov kernel was used with varying bandwidths as well as Silverman's optimal bandwidth, consistent with the main experiment [see equations (11), (12), and (13) for the different bandwidth rules]. MISE was again used as the measure of accuracy.

Two Monte Carlo experiments were carried out. First, the number of data points N was varied ($N = 10$, $N = 50$, $N = 100$, $N = 1000$, and $N = 10000$), while the number of grid points n was set to the number of data points for $N \leq 1000$ and to $n = 1000$ for $N = 10000$. Although the accuracy of a kernel-density estimate relies on large-sample asymptotics (that is, accuracy increases with increasing sample size), the difference between the binned and exact estimators does not rely on asymptotics. That means that in theory, the difference between the two estimators is not supposed to vary across different sample sizes. Consequently, the sample size is fixed in the second experiment ($N = 1000$), and now the number of grid points at which the density is evaluated varies ($n = 10$, $n = 50$, $n = 100$, $n = 1000$). Because the relevant results were the same across kernels and bandwidths in terms of differences between approximate and exact estimates, we only present a subset of the results.

A.3 Results

In figure 7, MISEs for both the exact and the approximate estimators are plotted against sample size. We can indeed see that there are no differences between the approximate and the exact estimates when the number of grid points equals the number of data points or is sufficiently large ($n = 1000$), even for very small N (the lines appear exactly on top of each other, and the crosses, which represent MISEs of the approximate estimates, appear within the hollow marker shapes, which represent the errors of the exact estimates).

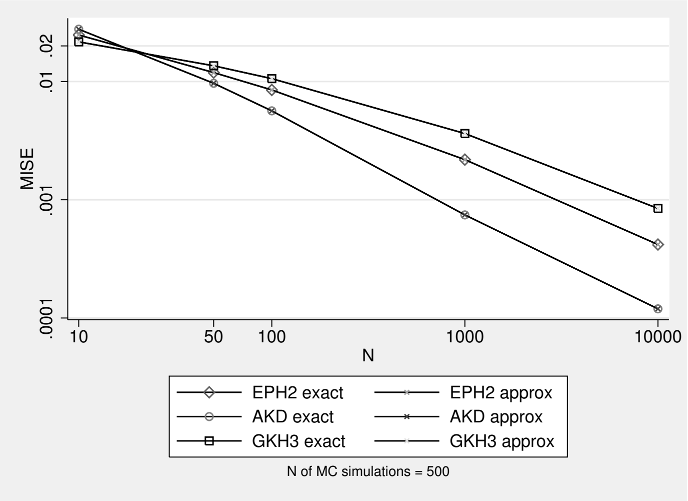

Figure 7. Accuracy of the approximate and exact kernel-density estimates with grid size equaling sample size ($n = N$) up to $N = 1000$, and $n = 1000$ for $N > 1000$; note that the lines representing approximate and exact estimates appear exactly on top of each other, and the crosses appear within the hollow marker shapes. EPH2 is an Epanechnikov kernel with Silverman's optimal bandwidth; AKD is an adaptive bandwidth kernel (Epanechnikov); GKH3 is a Gaussian kernel with Scott's oversmoothed bandwidth. "Exact" means exact estimation, and "approx" means approximate estimation using a linear-binning approach as implemented in `kdens`.

Figure 8 shows the results from the second experiment. Here we see a considerable difference between the two estimation methods when the number of evaluation points is very small ($n = 10$). However, this difference decreases at $n = 50$, and again no differences between the two methods can be observed for $n = 100$ or higher.

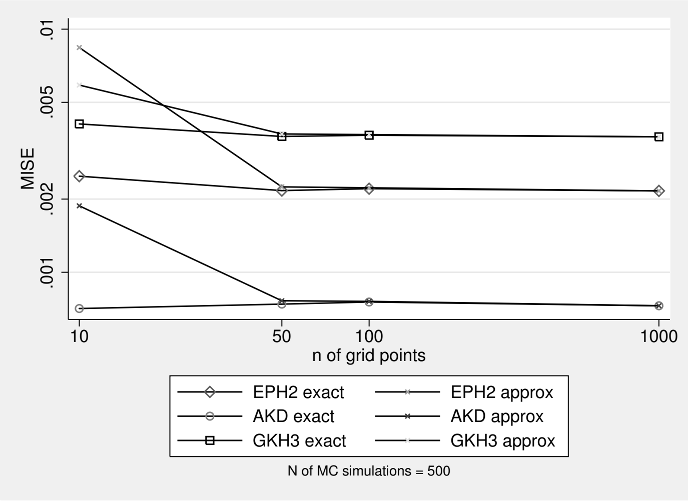

Figure 8. Accuracy of the approximate and exact kernel-density estimates using a varying number of evaluation points and a fixed sample size ($N = 1000$). EPH2 is an Epanechnikov kernel with Silverman's optimal bandwidth; AKD is an adaptive bandwidth kernel (Epanechnikov); GKH3 is a Gaussian kernel with Scott's oversmoothed bandwidth. "Exact" means exact estimation, and "approx" means approximate estimation using a linear-binning approach as implemented in `kdens`.

A.4 Conclusion

On the basis of the evidence presented here, one does not have to do an exact kernel-density estimation, and the binned approximation can be used instead—as long as the number of evaluation points equals the sample size or is sufficiently large. These results rely on a certain test density and assume reasonable bandwidth choices and are thus not necessarily generalizable. However, given these data and bandwidth choices, the only time it makes sense to use the exact estimator is when the sample size is larger than the number of grid points and when the number of grid points is lower than $n = 100$. Fortunately, this is avoidable: the grid size has no impact on computation time when using the approximation. A binned kernel-density estimation with one million data points takes around 2 seconds on a modern machine regardless of whether the number of grid points is set to 100, 500, or 1,000. Hence, erring on the side of a larger grid comes at no additional cost. In any case, because the number of grid points in our main experiment is never smaller than N if $N < 1000$ and is set to $n = 1000$ for $N \geq 1000$, an exact estimation would not yield any results that would be different from the ones obtained by using the approximation.