Introduction to the ‘kerneval’ package for density estimation

Gwen S Antell

2020-07-19

Contents

Overview

Why use kernels?

Kernel density estimation (KDE) is a foundational method in nonparametric statistics. Practical applications abound, from ecology to engineering. There are also theoretical applications of kernels to other statistical methods, such as spline smoothing and machine learning. KDE has achieved its prominence because the basic method is a straightforward, elegant, and effective way to generate a smooth probability distribution function (PDF) from a finite vector of numeric observations. Many R users will be familiar already with kernel-smoothed data visualisations, such as `ggplot2::geom_density()`.

How does it work?

One can find a brief summary of KDE in nearly any basic statistics textbook (e.g. Wasserman (2006)). Silverman (1986) gives a more thorough introduction to the topic and is written so as to be accessible to readers at a range of experience levels. Many explanations start with histograms, which are a simple variation on KDE wherein the observations are binned. Graphically, the probability function can be visualised by stacking a rectangle in the bin corresponding to each observation. The width of the rectangle along the x-axis is the 'bandwidth' in KDE terminology. The height is scaled such that the probability sums to unity across a given interval. One need not bin the data, however; one could instead place a rectangle directly at the x-value of each observation. The result is a step-wise function (the summed height of all rectangles at any given x-value) that is more jagged than a regular histogram and is called a naive estimator. One step further, one could replace rectangles with Gaussian curves (or triangles, or other kernel distributions). This is implemented in the `density()` function from the base **stats** package. There are more than 100 other packages that also involve KDE. For a comparison of the functionality and performance of 15 major KDE packages, a starting place is Deng and Wickham's (2011) review.

When should one use kerneval?

Most density estimation methods assume that the probability of observing an event (independent of the probability of the event occurring) is constant across the interval of estimation. However, many empirical datasets violate this assumption. For instance, in some cases the observation rate increases linearly as a function of x, so-called length-bias or size-bias.

kerneval implements methods to account for observation bias (also called selection bias) when estimating probability densities. One can use length-biased data or data with more complicated bias functions defined for any x in the study interval. The estimation steps will not eliminate bias, but they will reduce it, and the error of the estimate will decrease with sample size–as is the case for most statistical methods for biased data.

Mathematical framework

The point of KDE is to estimate the probability density function $f(x)$ with $\hat{f}(x)$. When there is no selection bias (equivalent to a uniform bias function), observations are drawn directly from F. However, when there exists selection bias, observations are drawn from G:

$$g(x) = \frac{w(x)f(x)}{\mu_w} \tag{1}$$

$$\text{where } u_w = \int w(u)f(u)du < \infty.$$

Note that this requires $w(u) > 0$ for all u. In the case of length-biased data, $w(x) = x$.

Many statisticans attribute the first treatment of the length-bias problem to Cox (1969), who investigated an example in textile fibre manufacturing. Cristobal and Alcala (2001) trace the ideas to earlier works by Fisher and Rao, however. Vardi (1982) proved that Cox's approach was the nonparametric maximum likelihood estimator. Jones (1991) found a kernel density estimator for this NPMLE, which has been widely adopted and extended. Borrajo and others (2017) developed a bootstrapping bandwidth selection approach for Jones' weighted KDE. Barmi and Simonoff (2000) evaluated an alternative approach: transforming the biased data to a scale where classical KDE is applied, then back-transforming and scaling the density. **kerneval** contains tools for both the weighting and transformation approaches to calculate $\hat{f}(x)$ based on a random sample of n observations drawn from G.

Transformed density estimation

Barmi and Simonoff (2000) proposed an elegant strategy to estimate density functions from selection-biased samples. The premise is that one may transform the empirical sample to an unbiased scale, perform any type of established density estimation, and then back-transform to the original scale to interpret the estimate. This approach has been described as 'adjust-first, smooth-later' because there is no need to modify the density estimation step. For instance, one could perform KDE with a Gaussian kernel and rule-of-thumb bandwidth, so long as KDE is done in the transformed space. The alternative approach involves weighting the kernels (described below) to adjust the estimation concurrent with smoothing.

Continuing with the mathematical notation of Eq. 1, let $X_1, X_2, ..., X_n$ be a random sample of n observations from G. To transform these empirical data, calculate the cumulative distribution function of the bias function, $W(x) = \int_0^x w(u)du$. Define $Y_i = W(X_i)$ for $i = 1, 2, ..., n$. To estimate the true probability distribution function f,

1. Perform density estimation on the sample $Y_1, Y_2, ..., Y_n$
2. Back-transform from y to x, using $W^{-1}(y)$
3. Scale by a constant ($\hat{\mu}$) such that the function integrates to unity

The `transdens()` function evaluates these steps, calling the `inverse()` function from **GoFKernel** (Pavia 2015) to estimate the inverse cumulative distribution function.

Depending on the shape of the true PDF and the bias function, analysts would be wise to inspect kernel density plots on the transformed scale, just as one might plot estimates (on the original scale) when selecting a bandwidth. In particular, one should consider whether the transformed distribution has a long tail or otherwise is difficult to estimate. If the density estimation problem seems more straightforward on the original than transformed scale, one could weight the KDE instead of transforming the data. Example 1 below illustrates these trade-offs. As another alternative, the original authors of the transformation method also suggest a local quadratic likelihood density estimator for distributions where the tails may be difficult to estimate. Kernels and local likelihood estimators can each vastly outperform the other in different contexts (Hall and Tao 2002). **kerneval** implements only KDE methods, but includes the option for boundary reflection (`reflect = TRUE`) to mitigate edge effects.

Weighted density estimation

The weighting operation of Jones (1991) is the KDE method for selection-biased data that has received the most attention in the statistics literature. The method is a slight modification of classical KDE and does not add onerous calculation, so it is an attractive approach. However, the choice of bandwidth is complicated and potentially time-consuming (e.g. bootstrap selection, described below).

To account for selection bias, the kernel on each datum is weighted by the inverse of the observation probability at that point. An informal, geometric way to visualise this is to imagine a one-dimensional axis with several Gaussian kernels along it, one at each datum. In regions of the axis with higher intensity of observation, pinch off some of the tops of the kernels. Then, add this area to the tops of the kernels in regions where observation is less probable. In a formal mathematical framework, weighting can be expressed by adding a scaling factor to the estimated density function. The classical definition of the KDE is:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i}^{n} K\left(\frac{x - x_i}{h}\right)$$

The weighted estimate is:

$$\hat{f}(x) = \frac{1}{n}\hat{u}\sum_{i}^{n} \frac{1}{w(x_i)} K\left(\frac{x - x_i}{h}\right)$$

$$\text{where } \hat{u} = \left[\frac{1}{n}\sum_{i}\frac{1}{w(x_i)}\right]^{-1}.$$

Weighted kernel estimation should not be confused with adaptive kenel estimation, which is most relevant when the dataset is large and the probability function is rougher in some areas than others. (The interested reader would do well to consult Wasserman (2006) 9.10, 'Do adaptive methods work'?) Both approaches modify the individual kernels that contribute to an estimate. However, in adaptive KDE the badwidth of each kernel is adjusted, whereas in weighted KDE the bandwidth is constant while the height (total probability density) of each kernel is adjusted.

Bootstrap bandwidth selection

The theory behind bandwidth selection can be summarised as a series of 3, potentially iterative steps. Luckily, in practice, there is a computational shortcut to find the bandwidth h^* that minimises bootstrapped mean integrated squared error (BMISE).

1. Pick an initial ('pilot') bandwidth λ and kernel L. Smooth the empirical distribution F_n with these parameters to get $\hat{f}_\lambda$. Bose and Dutta (2013) recommended the use of a Gaussian kernel with $\lambda = \frac{1}{8}n^{-1/9}$.
2. Resample n observation from $\hat{f}_\lambda$, for B replicates. This is equivalent to resampled with replacement from F_n, then adding ϵ to each resampled set of observations, X^*. The ϵ are i.i.d. from $\frac{K(\cdot/\lambda)}{\lambda}$. (This step is not run in practice.)
3. Calculate the MISE of the bootstrap samples. $BMISE(h, \lambda) = \frac{1}{B}\sum_{i=1}^{B}\int[f^*_{h,i}(x) - \hat{f}_\lambda(x)]^2 dx = V^*(h) + B^{*2}(h)$.

The variance and bias terms of BMISE can be expressed in terms of K_h and K_λ convolution, which do not depend on X^*. Therefore, it is possible to find the h^* that minimises BMISE without actually drawing bootstrap samples. One must still pick a pilot bandwidth, however.

The internal `selectbw()` function selects the bandwidth following the bootstrap method of Borrajo and others (2017).

Examples

This vignette gives examples that compare the performance of weighting vs. transformation approaches, and examine the effect of sample size. The data in the first example are length-biased, while the data in the second example are generated according to a custom (fourth-order polynomial) bias function.

1. Length bias and inverse length bias

The first example replications the simulation study of Barmi and Simonoff (2000). We will draw biased samples from known distributions and then estimate PDFs with transformed vs. weighted kernels. The simulations make use of the convenient result that when sampling is length biased, $w(x) = x$, samples drawn from a true distribution of χ^2_k (chi-square distribution with k degrees of freedom) will come from an observed distribution (G) of χ^2_{k+2}. If there is inverse length bias, $w(x) = 1/x$, then G is χ^2_{k-2}.

There are eight simulation scenarios in total. The first four scenarios contain length-bias, and the true PDF F is set as a chi-square distribution with either 2 or 12 degrees of freedom. (Thus, G is a chi-square distribution with 4 or 14 degrees of freedom, respectively.) For each PDF, sample size is set at either 50 or 200 observations.

```
nVals <- c(50, 200)
kValsLin <- c(2, 12)
wLin <- function(x){ x }
degLin <- function(k){ k + 2 }
```

The remaining four scenarios contain inverse-length-bias, and F is a chi-square distribution with either 3 or 16 degrees of freedom. (Thus, G is a chi-square distribution with 1 or 14 degrees of freedom, respectively.) As before, $n = 50$ or 200.

```
kValsInv <- c(3, 16)
wInv <- function(x){ 1/x }
degInv <- function(k){ k - 2 }
```

The original study replicated each scenario with 500 samples. Depending on processing speed, and without parallelising calculations, the full simulation and estimation could run for an hour or more. For a quick exploration of the example, one could run fewer iterations.

```
nreps <- 25 # 500
```

To compare the performance of weighted vs. transformed KDE in each scenario, the standard approach is to calculate the mean integrated squared error, MISE. First, draw a sample of n observations from a chi-square distribution with given degrees of freedom. Then, calculate $\hat{f}$ with each KDE method. Finally, integrate the squared difference between the estimated and true PDF over the study integral. The ultimate goal is to calculate the *expectation* of this integrated error, however, so it is helpful to wrap all of these steps into a function that can be iterated. The mean ISE will converge on the MISE with more replicates.

```
miser1 <- function(w, n, k, deg){
  df <- deg(k)
  x <- rchisq(n = n, df = df)

  # weighted KDE
  wtEst <- wdens(x, w)
  wtFun <- approxfun(wtEst$x, wtEst$y)
  wtSE <- function(x) (wtFun(x) - dchisq(x, df=k))^2
  wtISE <- integral(wtSE, min(wtEst$x), max(wtEst$x))

  # transformed KDE
  transEst <- transdens(x, w)
  transFun <- approxfun(transEst$x, transEst$y)
  transSE <- function(x) (transFun(x) - dchisq(x, df=k))^2
  transISE <- integral(transSE, min(transEst$x), max(transEst$x))

  c(wtISE, transISE)
}
```

There are many ways one could iterate through each sample and simulated distribution. Here are nested for-loops to produce the results in the same format as the MISE values from Table 1 of Barmi and Simonoff (2000).

```
# tbl1a <- tbl1b <- data.frame()
# for (n in nVals){
#  for (k in kValsLin){
```

```
#    mises <- replicate(n = nreps,
#                       miser1(w = wLin, n = n, k = k, deg = degLin)
#                       )
#    miseAvg <- rowMeans(mises)
#    iter <- data.frame(k, n, t(miseAvg))
#    tbl1a <- rbind(tbl1a, iter)
#  }
#  for (k in kValsInv){
#    mises <- replicate(n = nreps,
#                       miser1(w = wInv, n = n, k = k, deg = degInv)
#                       )
#    miseAvg <- rowMeans(mises)
#    iter <- data.frame(k, n, t(miseAvg))
#    tbl1b <- rbind(tbl1b, iter)
#  }
# }
```

The average ISE values from 500 replicates (i.e. estimates of MISE) are listed in Table 1, and agree with the original trends reported by Barmi and Simonoff (2000), reproduced in Table 2. Namely, KDE is more accurate when applied to more symmetric distributions (chi-square distributions with 12 or 16 degrees of freedom instead of 2 or 3). In addition, the transformation approach performs better for inverse-length-biased data than for length-biased data, because the transformed axis in the former case (a log-x scale) is a 'generally favourable' estimation problem.

Table 1: Replication of published results, using wdens() and transdens() functions in kerneval

k	n	Weighted	Transformed
w(x)=x			
2	50	0.0182	0.0507
2	200	0.0154	0.0521
12	50	0.0022	0.0029
12	200	0.0009	0.0012
w(x)=1/x			
3	50	0.0562	0.0130
3	200	0.0601	0.0048
16	50	0.1535	0.0022
16	200	0.0371	0.0007

Table 2: Barmi and Simonoff (2000) Table 1, using weighted KDE and transformed local likelihood estimators

k	n	Weighted	Transformed
w(x)=x			
2	50	0.0312	0.0309
2	200	0.0155	0.0317
12	50	0.002033	0.00434
12	200	0.000891	0.00184
w(x)=1/x			
3	50	0.0155	0.00568
3	200	0.00744	0.00216
16	50	0.00179	0.00105

k	n	Weighted	Transformed
16	200	0.000691	0.000293

Although the patterns in Tables 1 and 2 match, the exact values of respective entries differ. At the second and third significant digits, some variation seems to occur due to stochasticity in the replicates. Perhaps 1000 replicates would be more suitable to approximate the MISE. Additional discrepancies for weighted kernel MISE might be due to differences in bandwidth selection method. The authors chose the bandwidth 'in each replication to minimise ISE,' but this leaves some ambiguity. In the code above, bandwidth was selected with a bootstrap estimate and rule-of-thumb pilot. For the transformed estimates, the authors used a local quadratic likelihood density estimator, while here transformed kernels are used instead.

2. Polynomial bias function

Length bias is the type of selection bias that appears most frequently in the statistical literature. However, there are many situations where observation probability could behave in a more complicated way, perhaps non-monotonically. The **kerneval** user may work with any bias function that is positive and continuously defined over the study interval.

The present example is based loosely on an empirical dataset of marine species' ecological niches. The x-axis represents a resource or environmental gradient (e.g. temperature), and the PDF is the probability of a species' occurrence for a given resource density/environmental state. Species tend to achieve peak abundance at an optimum condition, and to become rarer in habitats farther from the optimum. The true PDF F in this example is a chi-square distribution with 5 degrees of freedom, representing a species with a broad tolerance.

Unfortunately for ecologists, it is impossible to measure F directly. Instead, one might have the spatial coordinates where a focal species has been observed, along with the environmental conditions at those locations. KDE on these data will provide an estimate $\hat{f}$ of the species' niche along the given habitat axes. Unless the project managers took great care with the original sampling design, however, the data will be biased by the relative availability of habitat, the way field researchers took observations, or other spatio-temporal effects. (For instance, GPS trackers tend to lose satellite connection when an animal wanders into dense vegetation, leading to a larger observation probability in open terrain (Frair et al. (2010)).) The bias function in this example is a 4th-order polynomial, defined over the interval [0,10]. This function reflects (1) the larger surface area of temperate zones relative to the equatorial or polar zones, (2) asymmetries between the Northern and Southern hemisphere, and (3) the most common routes of research vessels.

```r
w <- function(x){
  x <- 0.3*(x - 5.5)
  -0.8 * x^4 + 0.7 * x^3 + 1.5 * x^2 - 2 * x + 2
}
g <- function(x){
  g1 <- function(x) w(x) * dchisq(x, df=5)
  u <- integral(g1, 0, 10)
  w(x) * dchisq(x, df=5) / u
}
```

As in Example 1 above, the MISE is a useful way to compare the performance of different KDE methods. The weighted and transformed approaches both reduce the estimation error compared to the base **stats** method, which does not consider selection bias. This result is unsurprising; it is more interesting to examine the magnitude of the improvement, and how the performance depends on sample size.

```r
miser2 <- function(g, n){
  x <- random.function(n = n, g, lower=0, upper=10)

  # weighted KDE
  wtEst <- wdens(x, w)
  wtFun <- approxfun(wtEst$x, wtEst$y)
  wtSE <- function(x) (wtFun(x) - dchisq(x, df=5))^2
  wtISE <- integral(wtSE, min(wtEst$x), max(wtEst$x))

  # transformed KDE
  transEst <- transdens(x, w)
  transFun <- approxfun(transEst$x, transEst$y)
  transSE <- function(x) (transFun(x) - dchisq(x, df=5))^2
  transISE <- integral(transSE, min(transEst$x), max(transEst$x))

  # KDE without accounting for selection bias
  baseEst <- density(x, cut=0)
  baseFun <- approxfun(baseEst$x, baseEst$y)
  baseSE <- function(x) (baseFun(x) - dchisq(x, df=5))^2
  baseISE <- integral(baseSE, min(baseEst$x), max(baseEst$x))

  c(wtISE, transISE, baseISE)
}

# iseSmolN <- replicate(10, miser2(g = g, n = 50))
# round( rowMeans(iseSmolN), 4)

# weighted, transformed, biased
#> [1] 0.0082 0.0059 0.0169
```

If one repeats this exercise with $n = 500$, both the weighted and transformed estimates improve several times over, but the uncorrected estimate does not.

```r
# iseBigN <- replicate(10, miser2(g = g, n = 500))
# round( rowMeans(iseBigN), 4)

# weighted, transformed, biased
#> [1] 0.0029 0.0018 0.0127
```

Although MISE is a useful metric, as a single number it cannot indicate the manner in which the estimate errs along the study interval. A graphical approach provides this complementary information, as demonstrated in Figures 1 and 2. The weighted (in orange) and transformed (in red) kernel estimates approach the true density as sample size increases from 50 to 500 observations. The weighted estimate undersmoothes at both sample sizes, however. The uncorrected estimate (in blue) follows the selection-biased PDF (in grey) regardless of sample size.

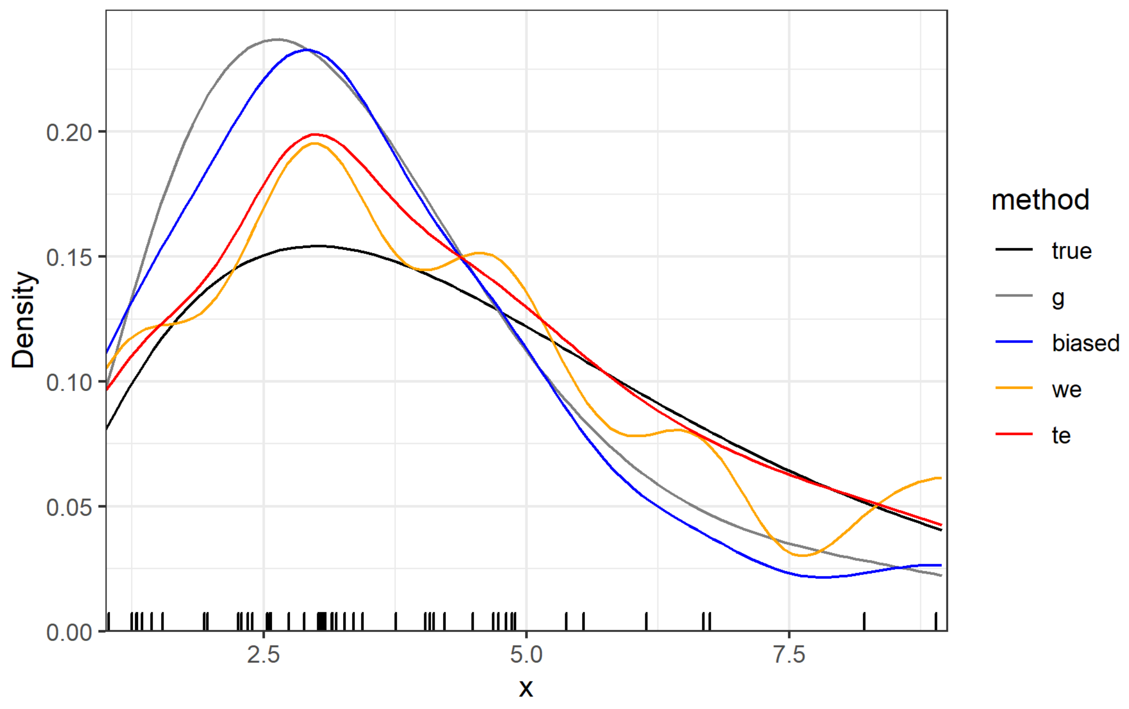

Figure 1: The true PDF is a chi-square distribution with 5 degrees of freedom is and is plotted in black. The distribution from which observations are drawn is in grey and reflects the influence of selection bias. A random sample of 50 observations from the grey PDF is plotted as a rug at bottom. The kernel density estimates based on the sample are drawn in colour: the blue curve is a standard KDE, the orange curve is a weighted KDE, and the red curve is a transformed KDE.

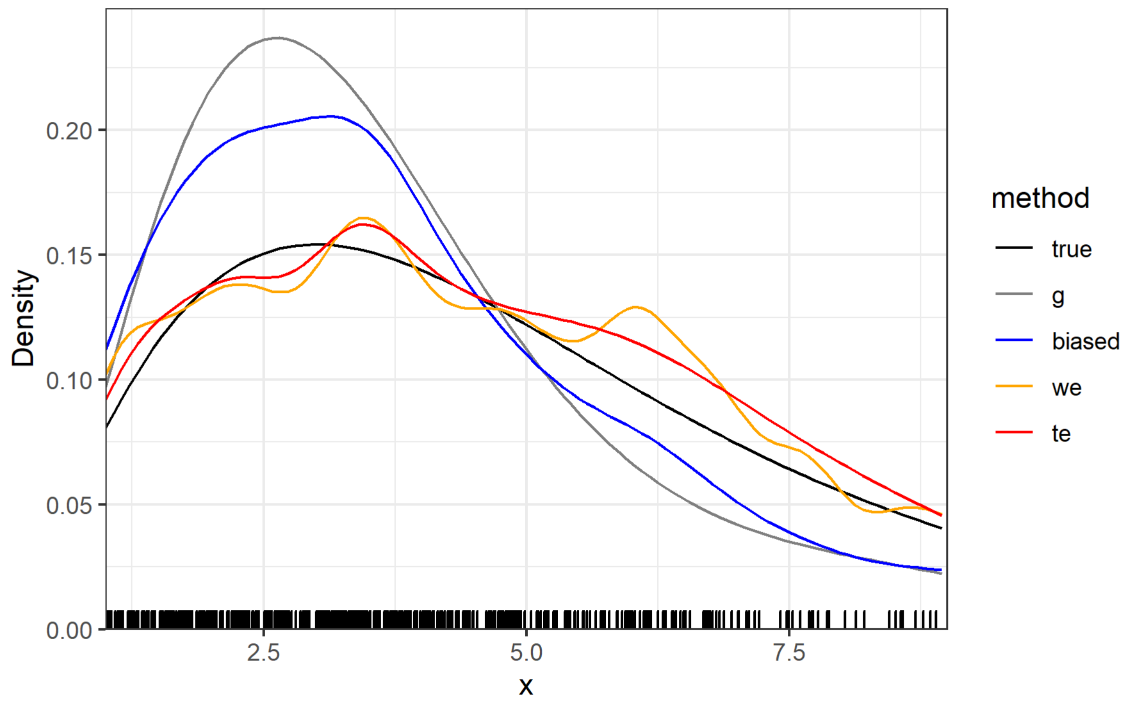

Figure 2: The PDF colours are as in Figure 1. The sample size in this case is 500 observations instead of 50.

References

Barmi, Hammou El, and Jeffrey S Simonoff. 2000. "Transformation-Based Density Estimation for Weighted Distributions." Journal Article. *Journal of Nonparametric Statistics* 12 (6): 861–78.

Borrajo, Maria Isabel, Wenceslao González-Manteiga, and María Dolores Martínez-Miranda. 2017. "Bandwidth Selection for Kernel Density Estimation with Length-Biased Data." Journal Article. *Journal of Nonparametric Statistics* 29 (3): 636–68.

Bose, Arup, and Santanu Dutta. 2013. "Density Estimation Using Bootstrap Bandwidth Selector." Journal Article. *Statistics & Probability Letters* 83 (1): 245–56.

Cox, David. 1969. "Some Sampling Problems in Technology." Book Section. In *New Developments in Survey Sampling*, edited by NL Johnson and H Smith, 506–27. New York: Wiley.

Cristóbal, José A, and José T Alcalá. 2001. "An Overview of Nonparametric Contributions to the Problem of Functional Estimation from Biased Data." Journal Article. *Test* 10 (2): 309–32.

Deng, Henry, and Hadley Wickham. 2011. "Density Estimation in R." Unpublished Work. http://www2.cs.uh.edu/~ceick/7362/T2-4.pdf.

Frair, Jacqueline L, John Fieberg, Mark Hebblewhite, Francesca Cagnacci, Nicholas J DeCesare, and Luca Pedrotti. 2010. "Resolving Issues of Imprecise and Habitat-Biased Locations in Ecological Analyses Using Gps Telemetry Data." Journal Article. *Philosophical Transactions of the Royal Society B: Biological Sciences* 365 (1550): 2187–2200.

Hall, Peter, and Terence Tao. 2002. "Relative Efficiencies of Kernel and Local Likelihood Density Estimators." Journal Article. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 64 (3): 537–47.

Jones, Michael C. 1991. "Kernel Density Estimation for Length Biased Data." Journal Article. *Biometrika* 78 (3): 511–19.

Pavia, Jose M. 2015. *Testing Goodness-of-Fit with the Kernel Density Estimator: GoFKernel. Journal of Statistical Software, Code Snippets*. Vol. 66. http://www.jstatsoft.org/v66/c01/.

Silverman, Bernard W. 1986. *Density Estimation for Statistics and Data Analysis*. Book. Monographs on Statistics and Applied Probability. Bristol: Chapman; Hall Ltd.

Vardi, Yehuda. 1982. "Nonparametric Estimation in the Presence of Length Bias." Journal Article. *The Annals of Statistics* 10 (2): 616–20.

Wasserman, Larry. 2006. *All of Nonparametric Statistics*. Book. Springer Texts in Statistics. New York: Springer. http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=170713&site=ehost-live&authtype=ip,uid.