Communications in Statistics - Simulation and Computation

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/lssp20

Bivariate Nonparametric Random Variate Generation Using a Piecewise-Linear Cumulative Distribution Function

W. Kaczynski [a], L. Leemis [b], N. Loehr [c] & J. McQueston [b]

[a] Department of Mathematical Sciences, United States Military Academy, West Point, New York, USA

[b] Department of Mathematics, The College of William & Mary, Williamsburg, Virginia, USA

[c] Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA

Available online: 20 Dec 2011

To cite this article: W. Kaczynski, L. Leemis, N. Loehr & J. McQueston (2012): Bivariate Nonparametric Random Variate Generation Using a Piecewise-Linear Cumulative Distribution Function, Communications in Statistics - Simulation and Computation, 41:4, 469-496

To link to this article: http://dx.doi.org/10.1080/03610918.2011.594532

Communications in Statistics—Simulation and Computation®, 41: 469–496, 2012

ISSN: 0361-0918 print/1532-4141 online
DOI: 10.1080/03610918.2011.594532

Bivariate Nonparametric Random Variate Generation Using a Piecewise-Linear Cumulative Distribution Function

W. KACZYNSKI[1], L. LEEMIS[2], N. LOEHR[3], AND J. McQUESTON[2]

[1]Department of Mathematical Sciences, United States Military Academy, West Point, New York, USA
[2]Department of Mathematics, The College of William & Mary, Williamsburg, Virginia, USA
[3]Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA

An extension of the univariate case of nonparametric random variate generation using a piecewise-linear cumulative distribution function is developed. The method is a blackbox variate generation technique requiring only data pairs from the modeler. The technique is a novel nonparametric approach to density estimation, and generating variates for simulation is accomplished without explicitly computing the estimated joint density, thereby speeding up random point generation. The method presented effectively captures marginal distributions with multiple modes. The algorithm presented uses the convex hull of the observed data as a preliminary support, then generates the first element of the two-dimensional random vector via inversion of the marginal piecewise-linear cdf, and the second element from a conditional weighted piecewise-linear cdf created from selected values of the second variable.

Keywords Density estimation; Marginal distributions; Modeling; Piecewise-linear functions.

Mathematics Subject Classification 62-04; 62-07; 62G99.

1. Introduction

Univariate random variate generation from parametric distributions is a well-established methodology providing the modeler dozens of distribution choices having a variety of statistical properties (Banks et al., 2001; Law, 2007; Leemis and Park, 2006). For parametric bivariate distributions, however, the number of

Received November 9, 2010; Accepted May 25, 2011
Address correspondence to W. Kaczynski, Department of Mathematical Sciences, United States Military Academy, West Point, NY 10996, USA; E-mail: william.kaczynski@usma.edu

distribution choices is much more limited. Additionally, the ability to generate observations from some bivariate distributions relies on the acceptance–rejection method, casting out the preferred inversion method. Recent literature in copula-based approaches indicates improvement in this area. Copula-based approaches have often been applied to finance and are becoming more prevalent in other areas such as actuarial science and hydrology. We did not consider these approaches as candidates for comparison because recent literature suggests that the method of model selection is not universally accepted (Genest and Rëmillard, 2006). Additionally, this approach is a two-stage estimation process (1. marginals, 2. copula function). There is promising recent work in nonparametric copula-based approaches, overcoming the two-stage estimation issue. Biller (2009), for example, used the copula-based approach for simulation input modeling. However, we do not compare the proposed algorithm to this work.

Kernel density estimation (KDE) is another popular method for density estimation. Hörmann and Leydold (2000) presented algorithms that generate variates directly from a sample via KDE for both the univariate and bivariate cases. In their approach, resampling occurs from a multivariate normal distribution with a covariance matrix that matches that of the observed data. In the univariate case, Bratley et al. (1987) and Law (2007) described variate generation methods using the linear interpolation of the empirical distribution function. Generating variates from KDE offers the advantages (Devroye and Györfi, 1985; Devroye, 1986; Silverman, 1986) of simplicity and well-established theory of density estimation. However, KDE requires the arbitrary (but necessary) step of fine tuning a smoothing parameter as well as choosing the appropriate kernel function. Hörmann and Leydold (2000) also noted that generating variates from KDE results in the "variance of the empirical distribution always being larger than the variance of the observed sample," and furthermore, since generating from KDE is not an inversion method, correlation induction for variance reduction is lost. Silverman (1986) presented an algorithm that corrects the KDE variance difference by forcing it to equal the sample variance.

Since the focus of this article is modeling bivariate dependencies in input data for simulation, we now review the literature in this area. In the parametric case, Devroye (1986) and Johnson (1987) devised strategies for generating from several multivariate distributions including the multivariate-normal and the multi-variate Johnson family. Wagner and Wilson (1995) developed techniques for the bivariate Bezier distribution. Taylor and Thompson (1986) formulated a semi-nonparametric method that comprises samples from a combination of a nearest neighbor technique and KDE. Matching moments occasionally occurs as an appropriate method for density estimation. Because the majority of these published methods assume a known population distribution, they are coupled with potentially unrealistic distribution properties such as the support, moments, etc. Additionally, many of these methods rely on the acceptance–rejection technique for variate generation, and thus are not synchronized. This loss in synchronization sacrifices the ability to implement variance reduction through the use of common random numbers, and carries the added expense of wasted $U(0, 1)$'s. All of the methods reviewed in the literature model a unimodal distribution well, but often fail to consistently model a bimodal distribution correctly. We were unable to find a flexible family capable of greater than two modes, therefore generating variates according to some parametric family may not be possible for data with more than two modes.

In this article, we intend to show three differences between the proposed bivariate nonparametric random variate generation and KDE. The differences are:

(1) no reliance on the selected kernel density function; (2) no reliance on the selected smoothing parameter; and (3) no production of unrealistic variates (i.e., negative values from a service time distribution). The notable strengths of the proposed algorithm are that it is synchronized and cannot produce unrealistic variates.

The article is organized as follows. Section 2 introduces a piecewise-linear cumulative distribution function (cdf) and explains how to sample from this cdf. It follows with a discussion of how to manipulate this estimator so that the first two moments of the estimator match the corresponding moments of the observed data. The section concludes with the proposed bivariate random variate generation algorithm, an applied example, and an interesting variant of the algorithm for selected data sets. Section 3 compares the proposed algorithm to KDE for bivariate data with unknown underlying bivariate densities, along with data generated from known bivariate densities. Where possible, the comparisons include visual representations, marginal means and variances, covariances, and squared error between the known and estimated cdfs. Section 4 describes limitations of the proposed generator. The last section summarizes the results and provides areas for future study on the topic.

2. Generating Variates from Bivariate Data

One obvious and simple technique for generating variates from a data set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is to sample from the empirical cdf, $\widehat{F}(x, y) = \frac{1}{n} I(x, y)$, where $I(x, y)$ is a function that counts the number of (x_i, y_i) pairs in the data set satisfying $x_i \leq x$ and $y_i \leq y$ (i.e., $\widehat{F}(x, y)$ is the fraction of the data pairs falling to the southwest of (x, y)). An algorithm for generating from the empirical cdf is equivalent to sampling with replacement from the data pairs (x_i, y_i):

1. Generate $U \sim U(0, 1)$.
2. $I \leftarrow \lceil nU \rceil$.
3. Return (x_I, y_I).

This random variate generation technique is fast and conceptually straightforward. The drawback with this method is that the random variates are limited to the data pairs—which is particularly problematic for a small sample size.

2.1. *The Piecewise-Linear cdf*

In the univariate case, the interpolation problem is easily solved by using a piecewise-linear approximation to the empirical cdf. The $n - 1$ gaps between the data values result in $n - 1$ piecewise-linear segments for the estimated cdf. If extrapolation in one or both tails is an issue, then the modeler can use Marsaglia's tail algorithm (Bratley et al., 1987) or kernel density estimation (Silverman, 1986).

In the bivariate case, the delineation of the support is less clear than in the univariate case. Using the rectangular support

$$\min\{x_1, x_2, \ldots, x_n\} \leq x \leq \max\{x_1, x_2, \ldots, x_n\},$$
$$\min\{y_1, y_2, \ldots, y_n\} \leq y \leq \max\{y_1, y_2, \ldots, y_n\},$$

for example, is likely to include regions of support that a modeler would want to exclude. In the method developed here, we use the convex hull of the data

values as a preliminary support. (This support can be modified using techniques described subsequently.) We define the convex hull traditionally as the minimum convex set containing the data set of interest in the two-dimensional plane. The variate generation algorithm (Law, 2007, p. 467) relies on conditioning:

1. Generate $U_1 \sim U(0, 1)$.
2. $X_0 \leftarrow F_X^{-1}(U_1)$.
3. Generate $U_2 \sim U(0, 1)$.
4. $Y_0 \leftarrow F_{Y|X_0=x}^{-1}(U_2)$.
5. Return (X_0, Y_0).

For a derivation of the joint density, see Devroye (1986).

The challenge associated with the development here is to find a reasonable nonparametric approximation to $F_{Y|X=x}^{-1}(\cdot)$. To illustrate the justification in using $F_{Y|X=x}^{-1}$, consider the scatterplot shown in Fig. 1 with $x = 8$. The data indicate a wide range of potential values to generate for the second element of the random pair, y. Depending on the unknown bivariate population distribution this might be acceptable. However, given the observed data, it appears the associated y value should not potentially occupy this entire range of the y values, and might more appropriately be represented by the limits naturally occurring at the lower and upper intersections of $x = 8$ and the convex hull.

2.2. *A Nonparametric Bivariate Generation Algorithm*

By combining strategies used in the univariate case, an algorithm is devised to generate bivariate random variates from observed data pairs using a nonparametric heuristic approach. This algorithm requires a random sample of bivariate data drawn from an unknown continuous population distribution. A good algorithm produces variate pairs that adequately mimic the distribution associated with the observed data. If appropriate, the marginal data are moment matched at the beginning of the algorithm. The moment-adjusted vectors are created by first stretching the marginal data so that the variance of the piecewise linear cdf estimator matches that of the sample data variance, and then shifting the resulting marginal

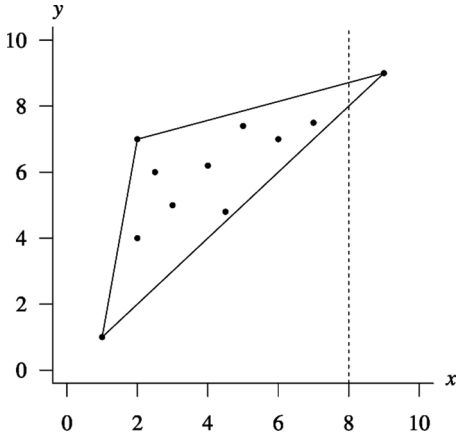

Figure 1. Intersection of a randomly generated $x = 8$ and the convex hull.

data values to match the marginal means. This process is only suitable in cases where the adjusted marginal values do not result in unrealistic data points, e.g., when service times are close to zero and adjusting them could produce impossible negative service times. The advantage of adjusting the data (when possible) is that the first two moments are conserved by the estimator, whereas, when the data is not adjusted, it is well known that the piecewise-linear cdf estimator's variance is less than the sample data variance. Matching the variances is important in computing the denominator in the correlation expression

$$Corr(X, Y) = \frac{Cov(X, Y)}{\sqrt{V(X)V(Y)}}.$$

Using the expressions derived in Kaczynski et al. (2012), reprinted here, the ordered moment-adjusted vector values $x'_{(j)}$ are calculated as

$$x'_{(j)} = x_{(1)} - \delta + w \sum_{j=1}^{i-1} g'_j,$$

where δ is the appropriate stretching parameter, w is the width of the support of the adjusted piecewise-linear cdf, and g'_j is a normalized gap value between sorted elements of the $\boldsymbol{x}$ vector. This calculation accomplishes matching the variance of the piecewise-linear cdf estimator to the sample data variance. We then match means by shifting each data value by

$$x''_{(i)} = x'_{(i)} - \left[\frac{x'_{(i)} + 2x'_{(2)} + \cdots + 2x'_{(n-1)} + x'_{(n)}}{2(n-1)} - \bar{x} \right].$$

The S-Plus/R code for this moment matching process (designated as the `mm(x)` procedure) is provided in Appendix A. For a more detailed explanation on matching the estimator's moments to the data, see Kaczynski et al. (2012).

The algorithm is separated into a setup portion, and a generation portion. The terms x_i and y_i represent the observed data pairs, x'_i and y'_i are the moment-adjusted data pairs, and, (x'', y'') is the generated variate pair produced by the algorithm. The corresponding vectors are set in boldface.

Setup

1. $\boldsymbol{x}' \leftarrow \mathtt{mm}(\boldsymbol{x}), \boldsymbol{y}' \leftarrow \mathtt{mm}(\boldsymbol{y})$ (optional stretching step).
2. $hull \leftarrow$ convex hull $(\boldsymbol{x}', \boldsymbol{y}')$.

Generation

1. Generate $U \sim U(0, 1)$.
2. $x'' \leftarrow F_X^{-1}(U)$ (F_X^{-1} is the inverse cdf of the piecewise linear estimator).
3. $y_{\text{lo}} \leftarrow$ minimum$\{hull(x'')\}$ (the height of lower intersection of the line $x = x''$ and the convex hull).
4. $y_{\text{hi}} \leftarrow$ maximum$\{hull(x'')\}$ (the height of upper intersection of the line $x = x''$ and the convex hull).
5. $A \leftarrow \{i \mid y_{\text{lo}} < y'_i < y_{\text{hi}}\}$, $i = 1, 2, \ldots, n$ (the index set of interior points).
6. $w_k \leftarrow \frac{1}{1+((x_k - x'')/s)^2}$ for $k \in A$, where s is the sample standard deviation of $\boldsymbol{x}_A$, the set of interior points (weights for y_{lo} and y_{hi} arbitrarily set to 1; all weights then normalized to sum to 1).

7. $F_{Y|X=x} \leftarrow$ weighted piecewise-linear cdf conditioned on $x = x''$ (see Kaczynski et al., 2012, for details on creating the weighted piecewise-linear cdf).
8. Generate $U \sim U(0, 1)$.
9. $y'' \leftarrow F^{-1}_{Y|X=x}(U)$.

In step 6 of the generation portion of the algorithm, we include s to standardize the weight calculation. Data pairs with x_i values closer to the line $x = x''$ receive higher weight. Dividing the absolute difference $x_i - x''$ by s scales the factors in terms of standard deviation units.

This algorithm is non conventional in the sense that it translates the data pairs directly into a variate generation algorithm, rendering the calculation of the estimated joint density unnecessary. There is, of course, an underlying joint probability density function associated with the algorithm.

2.3. *Examples*

Example 2.1. Consider the bivariate data set of size $n = 14$ random observations drawn from a continuous population: (4.1, 1.5), (6.2, 3.4), (8.3, 5.1), (7.8, 6.4), (5.2, 7.8), (2.0, 4.5), (1.9, 1.3), (2.7, 2.1), (3.5, 3.9), (4.0, 4.3), (3.6, 2.2), (4.4, 5.2), (5.0, 3.1), (5.3, 5.3).

Setup

1. Compute moment-matched adjusted $\boldsymbol{x}$ and $\boldsymbol{y}$ vectors, denoted as $\boldsymbol{x}'$ and $\boldsymbol{y}'$ for the data. Using the S-Plus/R `mm(x)` function, the adjusted vectors, to two decimal places, are: $\boldsymbol{x}' = (4.08, 6.48, 8.89, 8.32, 5.34, 1.67, 1.56, 2.47, 3.39, 3.96, 3.50, 4.42, 5.11, 5.45)$ and $\boldsymbol{y}' = (1.15, 3.35, 5.32, 6.82, 8.44, 4.63, 0.92, 1.85, 3.93, 4.39, 1.96, 5.44, 3.01, 5.55)$.
2. Find the convex hull of $\boldsymbol{x}'$ and $\boldsymbol{y}'$.

Generation

The S-Plus/R code provided in Appendix A combines the univariate strategies for generating from bivariate data using the proposed algorithm. Figure 2 presents the adjusted bivariate data and associated convex hull. Using the piecewise-linear cdf

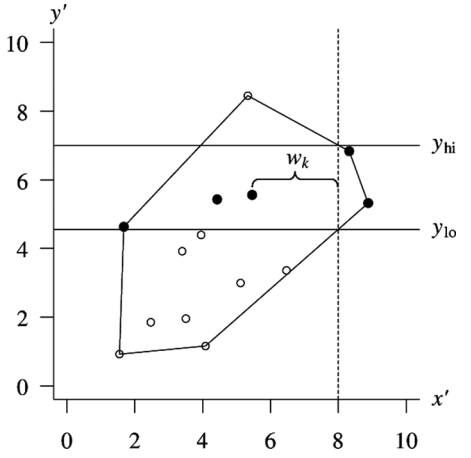

Figure 2. Plot of $\boldsymbol{x}'$ vs. $\boldsymbol{y}'$ and the convex hull.

created from the moment-matched $\boldsymbol{x}'$ vector, the variate x'' is generated at $x'' = 8$. The vertical dashed line at $x'' = 8$ intersects the convex hull in exactly two places, denoted as y_{lo} and y_{hi} in the algorithm. The horizontal lines at these intersection points establish the lower and upper limits capturing the interior original y data values used to create the weighted conditional piecewise-linear cdf for Y. The $|A| = 5$ interior values are the solid circles in Fig. 2. These corresponding y values are weighted by w_k based on their respective horizontal distance from the vertical dashed line associated with x''. One instance of w_k is shown in Fig. 2. Using the marginal piecewise-linear cdf created by the weighted interior y values, y'' is generated. Using this methodology, Fig. 3 displays 50 variates from the original $n = 14$ data values where both the mean and variance for the piecewise-linear cdf's of $\boldsymbol{x}$ and $\boldsymbol{y}$ match that of the data.

The previous example illustrates the workings of the algorithm and associated results. Figure 3 shows (and the algorithm requires) that generated variates must lie within the convex hull created by the original data (if the data is adjusted to match moments, we can generate slightly outside the original convex hull since matching moments requires stretching each endpoint by a positive distance δ, and the interior points by a corresponding proportional distance). Additionally, if the user desires bivariate data for a certain region not encompassed by the observed data, it is only necessary to adjust the convex hull as desired. This feature allows significant advantages for studying specific aspects of a data set. For example, the user could easily develop cases for data analysis that include regions of interest while also including observed data. The next example illustrates the algorithm's ability to replicate multi-modal data in terms of means, variances, and correlation. Hörmann and Leydold (2000) discussed KDE's difficulty in accurately estimating multi-modal data, which places adequate variate generation in question for such distributions. Although not studied in detail here, one open area of study for the proposed algorithm is the theoretical analysis of error bounds. Subsequent examples will show that in certain cases the estimated density converges to the true density when the size of the random data set increases and in other cases it may not.

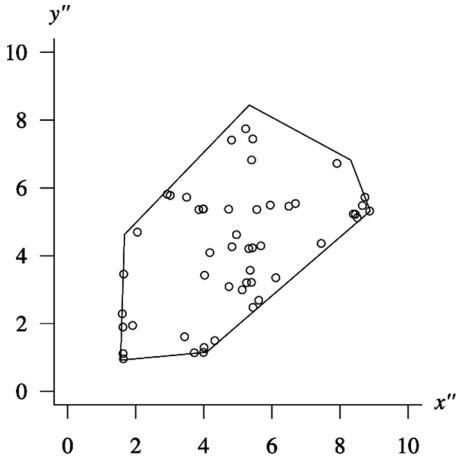

Figure 3. Plot of the convex hull and 50 random variates from $\boldsymbol{x}$ and $\boldsymbol{y}$.

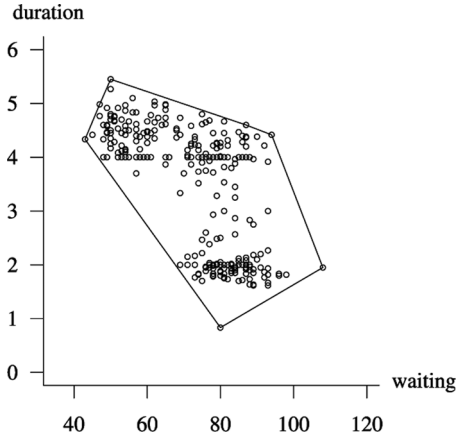

Figure 4. Plot of $n = 299$ waiting vs. duration times (minutes) for the Old Faithful Geyser.

Example 2.2. The Old Faithful geyser eruptions in Yellowstone Park are a commonly analyzed phenomenon. A data set from Weisberg (1980) consists of $n = 299$ data pairs, the waiting time between eruptions (x_i) and the eruption duration (y_i), and is displayed in Fig. 4, along with the convex hull. Although not easily visually distinguishable from the scatterplot, the data is tri-modal. Using a standard bivariate distribution to model this data set, such as the bivariate normal distribution, would not provide an adequate fit. For this data, it is appropriate to match the first two moments as doing so does not significantly change waiting nor duration times due to the large sample size. Additionally, matching the moments does not create any negative times.

Figure 5 shows the adjusted data and associated convex hull side-by-side with the 299 random variate pairs generated by our algorithm. The first numerical column of Table 1 provides the sample statistics associated with the data, and the second column shows that the first and second moments, and the covariance are adequately conserved in the generated variates. The third column provides p-values

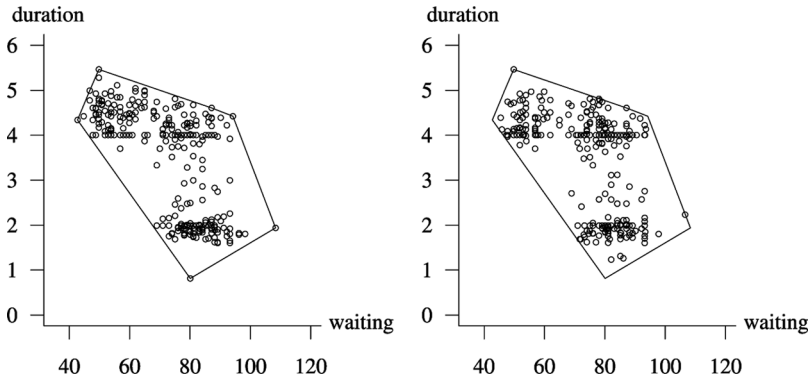

Figure 5. Sample adjusted observed data (left) and generated random variates (right) for waiting vs. duration times (minutes), for $n = 299$.

Table 1
Sample statistics for observed and generated data

	$n = 299$ observed data	$n = 299$ generated data	p-value
avg waiting	72.31	73.24	0.407
avg duration	3.46	3.39	0.465
var waiting	192.94	183.18	0.654
var duration	1.32	1.42	0.529
covariance	−10.28	−9.07	

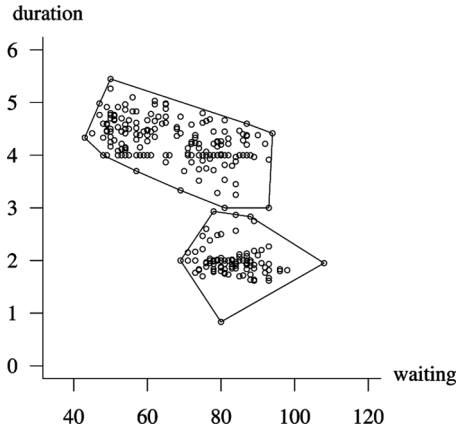

Figure 6. Plot of $n_1 = 105$ and $n_2 = 194$ waiting vs. duration times (minutes).

for the two-sided hypothesis tests with t-tests used for the means and F-tests for the variances.

It is apparent that the algorithm will occasionally generate variates in "white-space" (areas of the convex hull not represented by observed sample data values) of the convex hull as is expected. If this is problematic, we could fine-tune the appearance of the hull to avoid the possibility of these variates without significantly altering the algorithm. Alternatively, we could create two convex hulls as shown in Fig. 6, with $n_1 = 105$ data values in the lower hull and $n_2 = 194$ data values in the upper hull. The algorithm is modified so that a bivariate pair is generated from the lower hull with probability 105/299 and the upper hull with probability 194/299. The algorithm's run time change for this adjustment is negligible. A pathological example is provided in Appendix B where the proposed algorithm performs poorly due to the nature of the underlying population distribution.

3. Kernel Density Estimate Comparison

3.1. *Generating Variates via Kernel Density Estimation*

Perhaps the most widely accepted method of univariate density estimation is kernel density estimation (KDE). The kernel density approximation of the underlying true distribution is defined as $\hat{f}_X(x) = 1/(nb) \sum_{i=1}^{n} K((x - x_i/b)$, where K is the kernel

function, n is the sample size, and b is the bandwidth (smoothing) parameter. While several kernel functions exist in the literature, the most commonly used kernel function is the Gaussian kernel, $K(x) = 1/(\sqrt{2\pi})\, e^{-\frac{1}{2}x^2}$, $-\infty < x < \infty$, with mean zero and unit variance. These estimators provide a smooth density estimate with proven theoretical properties, making their choice of estimation a sound one. This estimator does require the additional step of calculating a smoothing parameter. We reference Hörmann and Leydold (2000) for use of kernels (and selection of a smoothing parameter) in generating bivariate data from an observed sample. They provide an efficient algorithm for sampling from a multi-dimensional kernel density estimate. Using their algorithm with a normal kernel function, generating variates is very fast. The algorithm is divided into a setup and generation portion.

Setup

For a random sample $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n$ of d length vectors, compute:

1. the mean vector $\overline{\boldsymbol{X}}$,
2. the estimated covariance matrix Σ,
3. the Cholesky factor $\boldsymbol{l}$ of Σ,
4. the smoothing parameter b,
5. the variance correction factor c_b.

Generation

1. Generate a random integer I uniformly distributed on $\{1, 2, \ldots, n\}$.
2. Generate a random vector $\boldsymbol{W}$ of d independent standard normal variates.
3. Return $\boldsymbol{Y} = \overline{\boldsymbol{X}} + (\boldsymbol{X}_I - \overline{\boldsymbol{X}} + \boldsymbol{l}(b\boldsymbol{W}))c_b$.

In this algorithm a full covariance matrix is specified from the observed data. Using the Old Faithful geyser data (Weisberg, 1980), the estimated covariance matrix, Σ, is

$$\Sigma = \begin{bmatrix} 192.94 & -10.28 \\ -10.28 & 1.32 \end{bmatrix}.$$

The joint probability density function is estimated as a sum of $n = 299$ translated versions of the chosen kernel function (bivariate normal in this case) multiplied by $\frac{1}{nb}$. Although there are many accepted versions of calculating b for the univariate case, the multidimensional case is more challenging. Silverman (1986) suggested a simple calculation for b as

$$b = \left(\frac{4}{(d+2)n}\right)^{1/(d+4)},$$

where d is the dimension of the data. The bivariate case results in $b = n^{-1/6}$.

Additionally, a variance-correction factor is included because the variance of the empirical distribution is always larger than the variance of the observed data (Silverman, 1986). Hörmann and Leydold (2000) defined the variance correction as c_b, where, $c_b = \sqrt{1 + b^2}$. For relatively large data sets, the choice of the kernel function only influences the tails of the estimated distribution. In the proposed algorithm the tails are always zero outside the convex hull, thus the Gaussian kernel should be advantageous for the KDE algorithm.

3.2. *Comparisons for Unknown Joint Densities*

To compare the two variate generation methods, 100 replications were made for each method, each of size $n = 299$ variates, using the geyser data introduced earlier. Prior to the study it was determined that a single replication is considered acceptable if it successfully captures the tri-modal KDE density appearing in Fig. 7. This density was computed directly from the $n = 299$ data pairs using S-Plus/R as described in Bowman and Azzalini (1997) for a normal kernel function and a normal optimal smoothing parameter. These estimated joint density plots are only used as a visual tool for comparing variate generation methods. The methods compared are: (1) nonparametric algorithm for unadjusted waiting and duration times; (2) nonparametric algorithm for adjusted waiting and duration times; and (3) Hörmann and Leydold's variance-corrected KDE algorithm. For each method the resulting three-dimensional estimated joint density plots, like the one shown in Fig. 7, was inspected for a tri-modal density. Methods one and two (those proposed in this paper) always captured the tri-modal appearance, while the KDE algorithm plots failed to capture the tri-modal density 35 times out of 100. An example of a failure instance is depicted in Fig. 8. Recognizing that the chosen smoothing parameter in the KDE algorithm is "oversmoothing" due to the multi-modality of the distribution, the parameter value was reduced by half as suggested in Hörmann and Leydold (2000) and the experiment was repeated. Doing so resulted in six failures out of 100 replications. This reduction in failures is evidence of the estimated density's sensitivity to the smoothing parameter selection. Using a more sophisticated parameter such as Bowman and Azzalini (1997) was purposefully not conducted, the reason being that the intent of the study was to compare a simple version of KDE to the proposed algorithm.

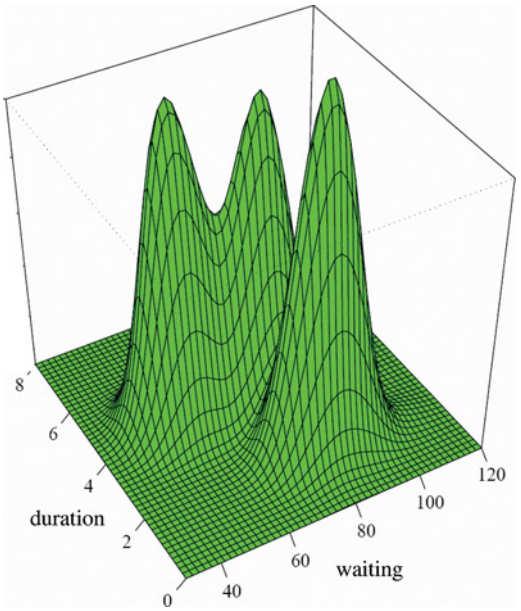

Figure 7. Joint density estimate of Old Faithful geyser data. (color figure available online)

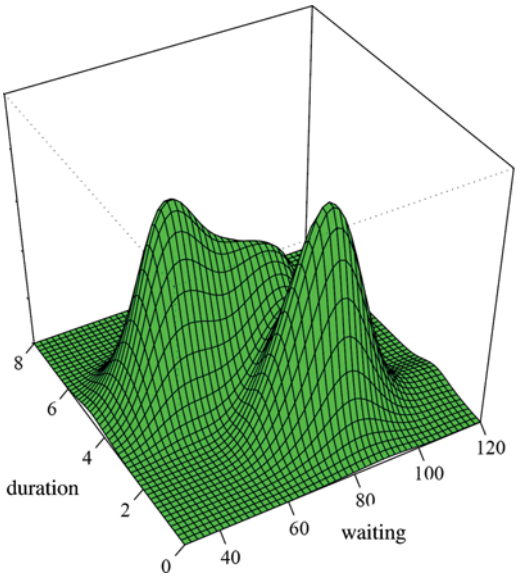

Figure 8. KDE joint density estimate failure for Old Faithful geyser data. (color figure available online)

The next example consists of warranty claim data provided by General Motors for model year 2000 cars sold in the month of December 2000. The bivariate data values are the mileage and the age of the vehicle at warranty claim. All vehicles share a three-year (1095 day), 36,000 mile warranty. This data set is unique because it is bounded below at zero and above at three years/36,000 miles. Given the lower and upper bounds on the data, it is inappropriate to stretch the data and match moments as on the geyser data. A scatterplot of the data is provided in Fig. 9, and the corresponding three-dimensional density estimate in Fig. 10. The figures

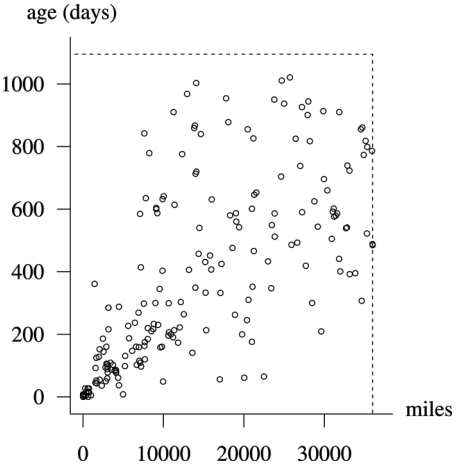

Figure 9. Scatterplot of miles vs. age (days) at warranty claim, $n = 259$.

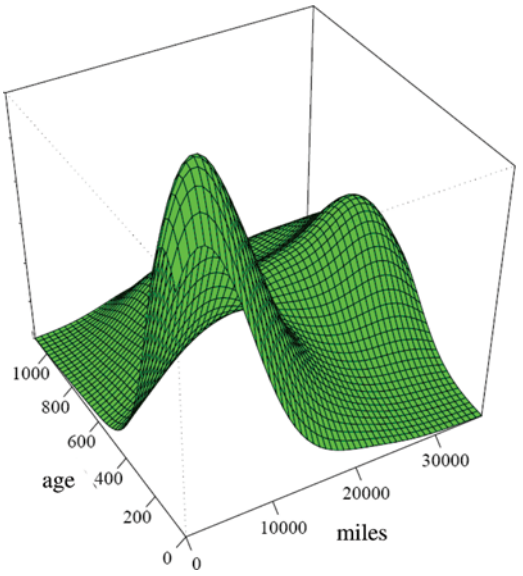

Figure 10. KDE joint density estimate of miles vs. age. (color figure available online)

depict a pronounced mode close to the origin and a less prevalent mode near the mileage axis upper bound. This is logical because a buyer might not recall when a three-year warranty will expire, but can easily notice the approaching 36,000 mile warranty limit. General Motors might be interested in the impact of adjusting warranty durations. Using the same type study as the geyser data, we test the proposed variate generation algorithm against both the variance-corrected KDE and reduced smoothing parameter variance-corrected KDE sampling techniques. Figure 11 depicts one resulting joint density comparison instance. Once again, it is apparent that variance-corrected KDE "oversmooths," while the reduced smoothing parameter KDE performs better in estimating the observed warranty

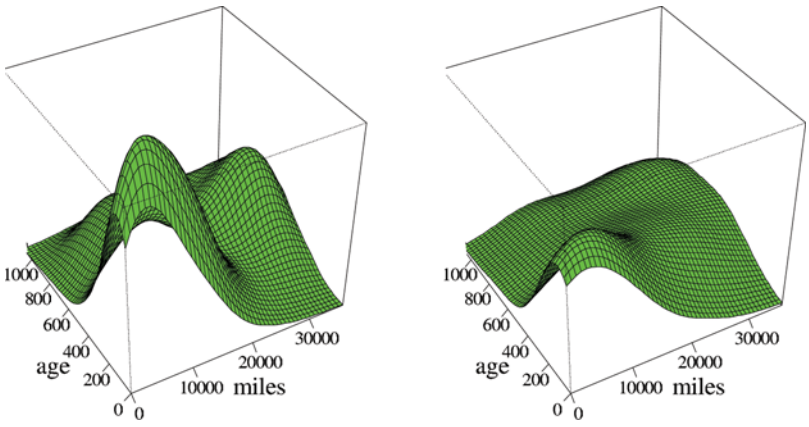

Figure 11. Variate generation for the proposed algorithm vs. variance-corrected KDE. (color figure available online)

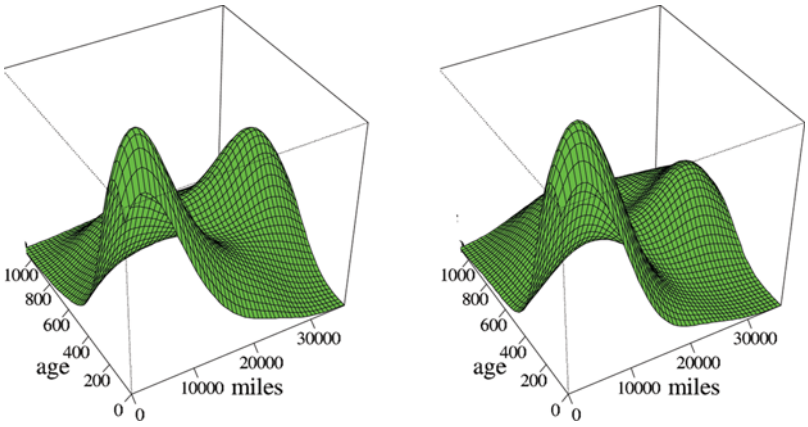

Figure 12. Variate generation for the proposed algorithm vs. reduced smoothing KDE. (color figure available online)

data as depicted in Fig. 12. A scatterplot of the KDE variance-corrected results, shown in Fig. 13, displays the tendency of KDE to generate more densely at the pronounced mode, further accentuating the possibility of variates outside of the support rectangle when the mode is close to zero, as is the case in this example. In addition, variates are also produced that lie outside the upper bounds for mileage and age. This behavior can be corrected by resorting to some type of acceptance–rejection or thinning method, however, both of these options ruin synchronization, which might be needed if a variance-reduction technique is employed.

The range of variates produced by the two approaches further accentuates their differences. Table 2 lists the minimums and maximums for each approach, along with the percentage of realizations falling outside the allowable warranty bounds. Given that all the generated variates for the proposed algorithm must (by construction) fall within the allotted bounds, impossible variates cannot occur. Consequently, using the KDE sampling method requires discarding impossible

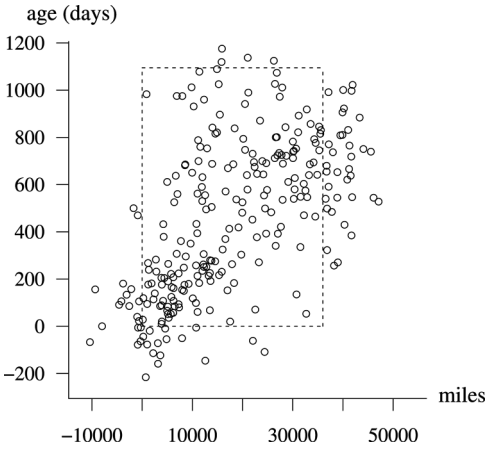

Figure 13. Scatterplot of variates generated via KDE.

Table 2
Range and percentage of variates outside allowable bounds

	min miles	max miles	min age(days)	max age(days)	percent <0	percent >3/36*K*
Observed data	8	35993	0	1056	0.0	0.0
Proposed algorithm	14	35983	0	1047	0.0	0.0
var. corr. KDE (standard *b*)	−11093	53178	−156	1104	18.5	7.0
var. corr. KDE (reduced *b*)	−2907	39984	−34	1179	8.5	6.2

variates. Finally, in 100 joint density visual comparisons the proposed algorithm was more successful than KDE in capturing the original data's depiction of customer warranty claims.

Using the normal kernel poses difficulty in modeling bounded data in two dimensions, as well as capturing multi-modal behavior. In months where sales numbers are higher, the upper limits of mileage and age are even more densely covered, further exhibiting multi-modal behavior.

In the proposed variate generation algorithm, the modeler has the choice between using the convex hull associated with the data pairs or using the rectangle with opposite corners (0, 0) and (36000, 1095). Figure 9 shows that there will be a significant difference between these two choices.

3.3. *Comparisons for Known Joint Densities*

We will now compare KDE and the proposed algorithm for two known joint densities, the first of which has infinite support and the second with bounded support.

The first example is an equiprobable mixture of three bivariate normal distributions, with parameters as indicated in Table 3.

Using this mixture as the underlying density, $n = 150$ variates were generated for use as the observed sample data. We then compare standard KDE with a Gaussian kernel (smoothing parameter is $b = n^{-1/6}$) and the proposed algorithm for 150 generated variates. Figure 14 illustrates the observed data in the left-hand plot, the KDE generated estimate on the right and the proposed algorithm's estimate in the center. A visual inspection indicates oversmoothing in the KDE case, a situation that could be remedied through manipulation of the smoothing parameter. Further work with the smoothing parameter did refine the KDE estimate suitably, and as

Table 3
Parameters for three equiprobable bivariate normals

	Bivariate normal parameters		
μ_X	2	4	8
μ_Y	1	8	4
σ_X	1	1	2
σ_Y	2	1	2
ρ	1/5	−1/5	−1/3

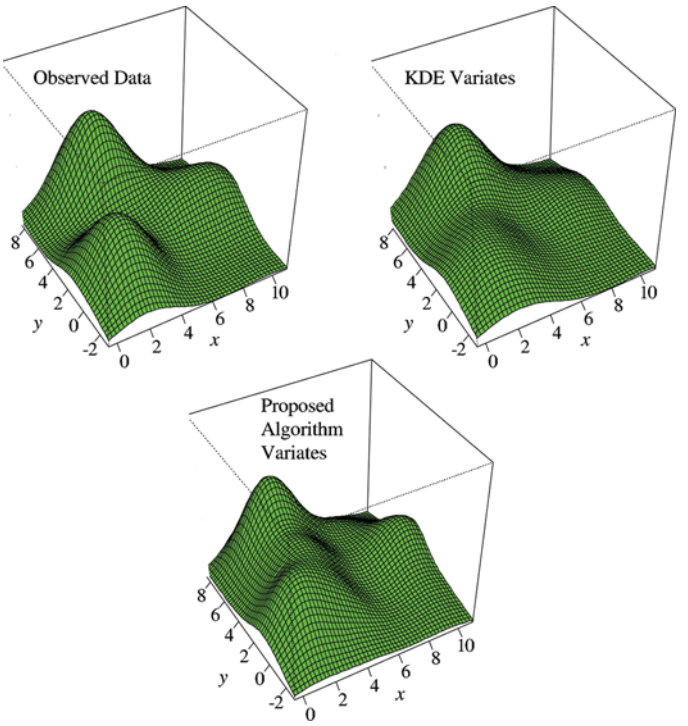

Figure 14. Observed data (left) and density estimate comparisons for KDE (right) and the proposed algorithm (center). (color figure available online)

expected, given a mixture of bivariate normals, KDE does well with proper selection of the smoothing parameter. As a second example, consider a uniform bivariate distribution with uniform support on the unit square. We will use this example to illustrate how our algorithm performs in the limit with regard to the marginals, which in this case are bounded by (0, 1). The experiment consists first of generating $k = 20, 50, 100$ data pairs from the bivariate uniform. Using these k data pairs, we then exercise the proposed algorithm and KDE, generating a single two-dimensional random variate pair for each. We repeat this experiment 100, 000 times and check the resulting marginal densities which we would like to converge to the theoretical marginals, each $U(0, 1)$. Figure 15 shows the resulting marginal densities for the proposed algorithm using $k = 50$ observed data pairs. The left-hand plot indicates that the density appears to converge to $U(0, 1)$ as desired. However, the conditioned density clearly does not. This result occurs because of the algorithm's tendency to designate more mass where the generated x value intersects the convex hull of the observed data. So there is the tendency to not adequately cover the vertical axis toward the upper and lower limits. A suitable manipulation of the algorithm allows us to partially correct this shortcoming by spreading the error equally between x and y. Since the vertical axis suffers in marginal estimation, we can modify the algorithm by alternating the roles of x and y on each subsequent (x, y) pair generated. Figure 16 depicts the estimated marginal densities for $k = 50$ observed data pairs after manipulating the algorithm. In the General Motors example, the support is rectangular, and furthermore, known. In this case we could have artificially created

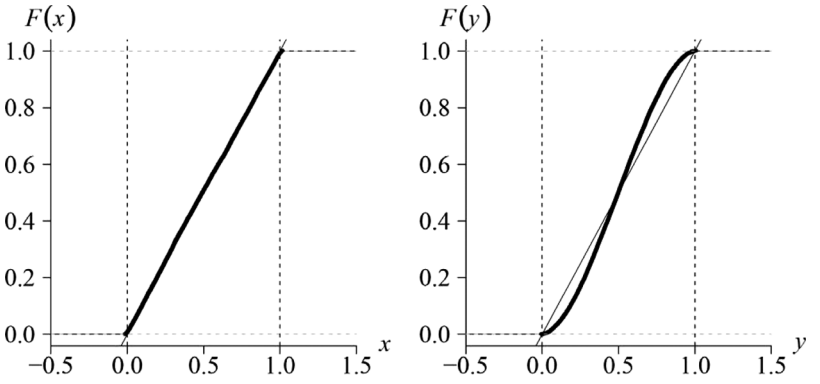

Figure 15. Estimated marginal densities for the unit square bivariate uniform distribution using the proposed algorithm.

the convex hull limits since the minimum and maximum for each marginal is known and fixed. For this example, we now fix the support as the unit rectangle, thus the convex hull is $(0, 1) \times (0, 1)$. We ran two cases for the fixed support, the proposed algorithm and the alternating algorithm. Given that the hull is fixed for both cases, the corresponding results do not differ significantly. Figure 17 shows the marginals for the first case, the proposed algorithm. Finally, we perform the same experiment for KDE, again using a Gaussian kernel and the same smoothing parameter used earlier. Figure 18 shows that although KDE does well over most of the support, it has the same trouble at the lower and upper end of the support. Furthermore, the KDE method generates impossible variates. Table 4 displays the squared error between the cdf and $N = 100,000$ generated data points, calculated as

$$\frac{1}{N}\sum_{i=0}^{N}(\widehat{F}(x_i) - F(x_i))^2,$$

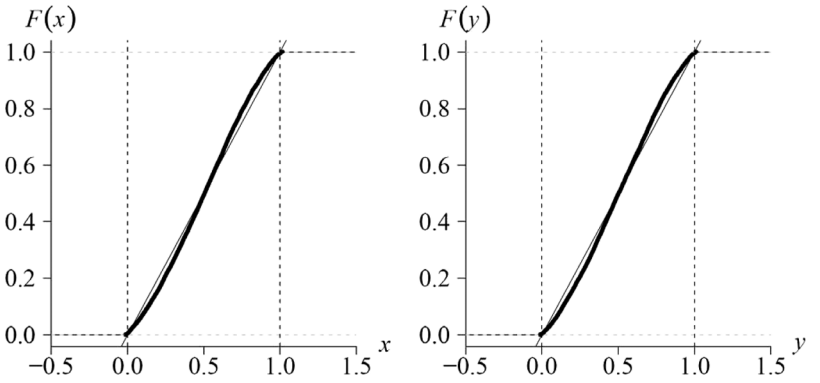

Figure 16. Estimated marginal densities for the unit square bivariate uniform distribution using the alternating algorithm.

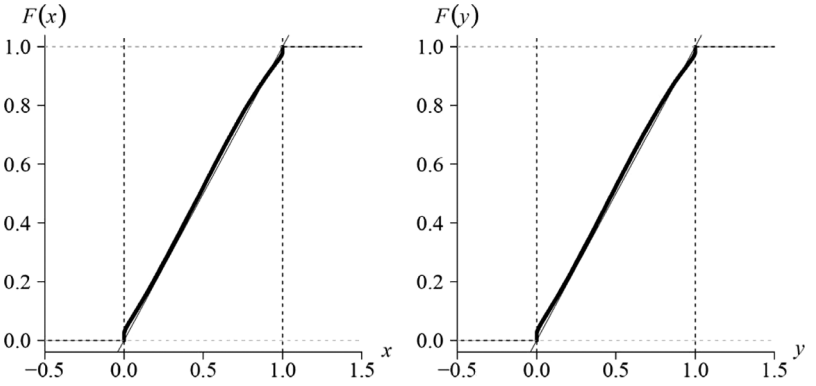

Figure 17. Estimated marginal densities for the unit square bivariate uniform distribution with fixed support $(0, 1) \times (0, 1)$.

where $\widehat{F}(x_i)$ is the estimated marginal cdf value at x_i and $F(x_i)$ is the theoretical cdf value at x_i. As another measure, we could include a quantile comparison, however, other than the lower and upper quantile discrepancies for KDE, there does not seem to be much difference across the board. As expected, KDE performs well throughout, except for the impossible variates generated. We could also change the kernel to a distribution with fixed support, which would reduce the extremity to which KDE produces impossible variates. However, the inclusion of such a comparison does not substantially change the overall results.

Since it is impossible to generate variates exactly from some data set without knowing the underlying distribution, questioning the quality of the variates generated from some known parametric distribution is justified. These hypothetical examples show that using the proposed algorithm exhibits quality at least as good as KDE. In terms of generation speed, KDE has the advantage over the proposed algorithm. In testing the vectorized version of the proposed algorithm's code versus KDE, excluding setup, we find that KDE runs about twice as fast, and given that

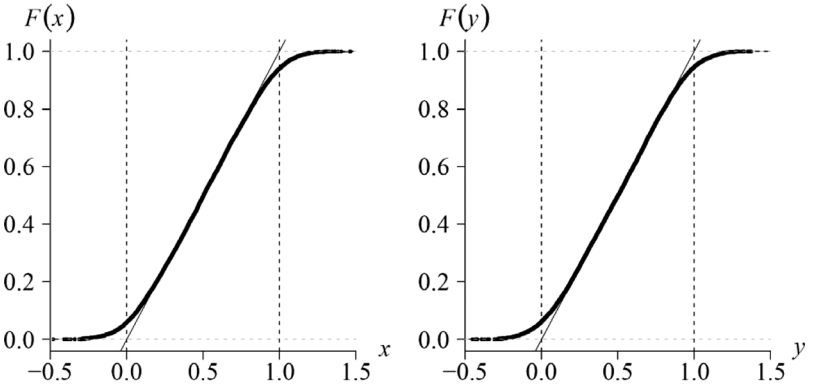

Figure 18. Estimated marginal densities for the unit square bivariate uniform distribution using KDE.

Table 4
Marginal cdf squared error for the estimates of the bivariate uniform distribution

	X error	Y error	Correlation
$n = 20$			
Proposed algorithm	3.7×10^{-5}	3.3×10^{-3}	0.007
Alternating algorithm	6.8×10^{-4}	6.7×10^{-4}	0.008
Fixed support algorithm	1.3×10^{-3}	2.2×10^{-3}	0.022
Alternating fixed support algorithm	3.9×10^{-4}	4.1×10^{-4}	0.012
KDE algorithm	5.7×10^{-4}	5.3×10^{-4}	−0.002
$n = 50$			
Proposed algorithm	1.1×10^{-5}	3.8×10^{-3}	0.001
Alternating algorithm	9.0×10^{-4}	8.5×10^{-4}	−0.001
Fixed support algorithm	2.8×10^{-4}	2.2×10^{-3}	0.015
Alternating fixed support algorithm	3.3×10^{-4}	3.0×10^{-4}	0.009
KDE algorithm	3.7×10^{-4}	3.6×10^{-4}	0.004
$n = 100$			
Proposed algorithm	3.2×10^{-6}	3.8×10^{-3}	−.001
Alternating algorithm	8.9×10^{-4}	9.1×10^{-4}	-9.2×10^{-5}
Fixed support algorithm	7.7×10^{-5}	2.6×10^{-3}	-6.6×10^{-4}
Alternating fixed support algorithm	4.7×10^{-4}	4.8×10^{-4}	-7.1×10^{-4}
KDE algorithm	2.6×10^{-4}	2.5×10^{-4}	−0.006

the proposed algorithm's run time is a function of n, KDE's advantage is more pronounced for large sample sizes.

4. Limitations

There are limitations associated with the proposed algorithm which we outline in this section. The two limitations discussed here are the algorithm's performance relative to KDE and the speed of the algorithm.

The first limitation is the algorithm's performance compared with that of KDE. There are few bivariate parametric distributions where variate generation is easy. We use the bivariate normal distribution to compare the impact of correlated random variables on the proposed algorithm and KDE. We expect KDE to perform extremely well since the underlying distribution is bivariate normal. The infinite tails associated with the bivariate normal distribution give an advantage to KDE, just as a bounded region, such as in the General Motors warranty data case, gives an advantage to the proposed algorithm. For the study, the underlying distribution is given by the parameters, $\mu_X = 1$, $\mu_Y = 2$, $\sigma_X = 4$, $\sigma_Y = 3$, $\rho = 0.01, 0.99$, where ρ varies from extremely low to high correlation. In addition to these two extreme values of ρ, the same study considered intermediate values of ρ, however including them here is not informative. For each value of ρ, we chose $n = 100$ and $n = 200$ as the observed sample sample sizes from the bivariate normal distribution. For each of the observed sample sizes, $N = 10$ and $N = 40$ variates were generated using KDE and the proposed algorithm (with and without moment matching). These variates were then used to calculate confidence intervals for the means, standard deviations,

Table 5
Confidence interval count for bivariate normal parameters and $\rho = 0.01$

n	N	μ_X	μ_Y	σ_X	σ_Y	ρ	Algorithm
100	10	9413	9304	**9496**	8722	**9513**	moment-matched proposed algorithm
		9384	9285	**9514**	8468	**9503**	proposed algorithm
		9392	9403	9112	9097	9396	KDE
100	40	9041	8652	9188	4921	9213	moment-matched proposed algorithm
		9038	8638	9089	3937	9204	proposed algorithm
		9113	9097	8068	8109	9079	KDE
200	10	9438	9418	**9528**	8879	**9554**	moment-matched proposed algorithm
		9421	9390	**9552**	8752	**9523**	proposed algorithm
		9445	**9447**	9205	9230	9400	KDE
200	40	9291	9111	9382	5331	**9451**	moment-matched algorithm
		9251	9100	9391	4759	9390	proposed algorithm
		9298	9300	8650	8568	9313	KDE

and correlation. The experiment for each (n, N) pair combination was conducted 10, 000 times and the count of the 95% confidence intervals containing each of the five parameters were tallied. Using the F distribution associated with the Clopper–Pearson confidence interval considered not significantly different from 9500 and $\alpha = 0.01$, the confidence interval counts are in the interval [9443, 9555]. Tables 5 and 6 contain the results of the simulation study, where boldface numbers are in [9443, 9555].

For low correlation, where we expected KDE to perform extremely well, the results do not indicate KDE dominating the proposed algorithm for producing variates that properly mimic the five distribution parameters. On the contrary, KDE performs rather poorly; in several instances it is outperformed by the proposed algorithm, regardless of whether moments are matched. One weakness of the proposed algorithm is apparent in the σ_Y column. This weakness can be partially overcome by implementing the alternating algorithm described in the previous section. There is no surprise that KDE had trouble inducing extremely high correlation, however, we did

Table 6
Confidence interval count for bivariate normal parameters and $\rho = 0.99$

n	N	μ_X	μ_Y	σ_X	σ_Y	ρ	Algorithm
100	10	9407	9388	**9483**	**9463**	9399	moment-matched proposed algorithm
		9400	9394	**9493**	**9468**	9393	proposed algorithm
		9420	9401	8711	9386	861	KDE
100	40	9063	9051	9161	9121	7801	moment-matched proposed algorithm
		9010	9001	9088	9033	7682	proposed algorithm
		9123	9034	7132	9106	0	KDE
200	10	**9458**	**9443**	**9549**	**9531**	9344	moment-matched proposed algorithm
		9432	9431	**9539**	**9521**	9347	proposed algorithm
		9450	**9466**	8967	**9478**	1304	KDE
200	40	9291	9255	9382	9334	7094	moment-matched proposed algorithm
		9224	9212	9336	9294	7082	proposed algorithm
		9319	9285	7756	9261	0	KDE

expect KDE to perform better in capturing the other distribution parameters. The proposed algorithm performs well in inducing correlation.

The second limitation noted for the proposed algorithm is generation speed. While generating the first element of the random variate pair is fast, generating the second element requires creating the conditional piecewise-linear cdf, which slows for large n. However, the algorithm benefits when high correlation exists in the observed variate pairs. High correlation results in a tight convex hull where once the first element of the random pair is generated, the conditional piecewise-linear cdf may involve only a small number of data points, or in the most extreme case, may be uniformly distributed between the points where the x variate intersects the convex hull. Additionally, it is possible to store y_{lo}, y_{hi}, A, and the conditional piecewise-linear cdf for efficiency if many pairs of random variates are required.

A more detailed study of each portion of the algorithm is now reviewed for complexity. Assume the input consists of n data points. The moment-matching function `mm(x)`, has running time $\Theta(n \log n)$ where the logarithmic factor comes from sorting the entries in the vector x. The setup portion has running time $\Theta(n \log n)$ where the logarithm comes from computation of the convex hull. The work to generate one simulated variate is either $\Theta(n)$ or $\Theta(n \log n)$, depending on whether the code needs to do any additional sorting of the points to calculate the weighted piecewise-linear cdf. This is because the code (as written) scans through all n data points to determine which ones are in the set A. Although not investigated in detail here, there are other methods available to speed up the proposed algorithm to include using a compiled language such as C, as well as the use of guide tables (Devroye, 1986).

5. Conclusions and Further Work

A nonparametric method of generating bivariate data was presented with examples. The method is blackbox, synchronized, and effectively captures multi-modal two-variable dependencies for most data sets. The method does not require any assumptions about the underlying distribution of the empirical data, nor does it ever compute an explicit formula for the estimated joint density as an intermediate step for variate generation. Thus, given an appropriate observed bivariate data set, a researcher or practitioner is capable of generating variates without the risk of introducing error associated with generating from some incorrect parametric distribution. Given continuous bivariate data, this method is capable of producing variates efficiently, and, in the case of observed data falling into recognizable groups, the algorithm can be easily altered for suitable employment. In a comparison study, the method performs at least as well as an accepted KDE generation algorithm in terms of estimation quality for selected data sets. Three significant contributions of the proposed algorithm are: (1) it is completely nonparametric and requires no parameters from the modeler; (2) it is simple to implement; and (3) it is a one-to-one (synchronized) variate generation algorithm whose resulting random vectors are capable of representing multi-modal bivariate distributions and will not produce impossible variates for fixed supports. In summary, the differences between the proposed algorithm and sampling from a KDE algorithm with a normal kernel are no reliance on selected kernel density function, no reliance on selected smoothing parameter, and no production of unrealistic variates (e.g., negative times from a service time distribution).

Three decisions are required from the modeler that are dependent on the data set. First, the modeler must decide if the data should be stretched in order to match moments. Second, the modeler must decide whether to use the convex hull associated with the (stretched or raw) data, or use a rational convex hull as in the case of the warranty data. Finally, the modeler must decide whether a single convex hull, as in Fig. 4, or multiple convex hulls, as in Fig. 6, is appropriate.

Five interesting potential areas of further work for the proposed algorithm are immediately evident. The first deals with studying how changes in the weighting function, w_k, of the interior points, $\boldsymbol{x}_A$, affect the resulting random pairs produced by the algorithm. Varying this same weighting function might also be explored to provide for asymptotic consistency for the estimator. The second area concerns the use of non convex hulls that allow for "dents" in the support. The third area concerns a two-dimensional extension of Marsaglia's tail algorithm. The fourth area concerns generation speed. The current algorithm generates the first element of the bivariate pair quickly and the second element slowly. The setup portion of the algorithm can be modified so as to generate both variates quickly by storing a set of conditional probability density functions. The fifth area concerns how a similar algorithm might be extended to higher dimensions.

As an illustration of the first area of further work, an extensive parallel study could be done with some manipulation of the weighting formula appearing in step 6 of the variate generation algorithm. Consider adding the smoothing parameter, b, to step 6 as

$$w_k \leftarrow \frac{1}{1 + ((x_k - x'')b/s)^2}.$$

The proposed algorithm sets $b = 1$, which satisfies the authors' intent to create a nonparametric, blackbox variate generation algorithm. The performance of the algorithm for $b = 1$ has been shown to be adequate. However, a separate study of the algorithm's performance vs. KDE for $b \neq 1$ deserves attention. This study should address optimization of the parameter b against an optimal smoothing parameter for KDE, as well as addressing error bounds associated with the proposed algorithm. Adding this new parameter, b, removes the blackbox property from the algorithm and may be expensive in execution, but might improve results just as Bowman and Azzalini's (1997) methods for the smoothing parameter and kernel selection improve KDE.

Appendix A

The R/S-Plus function `mm(x)` transforms the single argument $\boldsymbol{x}$, which is a vector, such that the returned vector values create a piecewise-linear cdf with a mean and variance equal to the unbiased sample mean and variance of the $\boldsymbol{x}$ argument vector.

```
mm <- function(x) {
 x   <- sort(x)
 n   <- length(x)
 xbar <- mean(x)
 xvar <- var(x)
 r   <- (2 * x - x[n] - x[1]) / (x[n] - x[1])
```

```
 rlo <- r[1:(n - 1)]
 rhi <- r[2:n]
 rmid <- r[2:(n - 1)]

 xlo <- x[1:(n - 1)]
 xhi <- x[2:n]
 xmid <- x[2:(n - 1)]

 aa  <- 1 / (3 * (n - 1)) * (sum(rlo * rlo) + sum(rlo * rhi)
       + sum(rhi * rhi))
      - 1 / (n - 1) ^ 2 * (sum(rmid) ^ 2)
 bb  <- 1 / (3 * (n - 1)) * (sum(2 * xlo * rlo)
      + sum(xlo * rhi + xhi * rlo)
     + sum(2 * xhi * rhi)) - 1 / (n - 1) ^ 2 * sum(rmid) * (x[1] + x[n]
      + 2 * sum(xmid))
 cc  <- 1 / (3 * (n - 1)) * (sum(xlo ^ 2) + sum(xlo * xhi)
       + sum(xhi ^ 2))
     - 1 / (4 * (n - 1) ^ 2) * ((x[1] + x[n] + 2 * sum(xmid)) ^ 2) - xvar

 del <- (-bb + sqrt(bb ^ 2 - 4 * aa * cc)) / (2 * aa)
 xp  <- x + r * del
 xpp <- xp - ((sum(xp) - xp[1] / 2 - xp[n] / 2) / (n - 1) - xbar)
 xpp
}
```

The R/S-Plus code below contains all the elements necessary to generate random bivariate pairs given the x and y vectors consisting of the observed data. The code is separated into three portions, setup, generation, and the main program. Indentation denotes nesting.

```
# SETUP PORTION
# xnew <- mm(x)
# ynew <- mm(y)

orderxnew <- order(xnew)
xnewlength <- length(xnew)

x <- xnew[orderxnew]
y <- ynew[orderxnew]

hullindex <- chull(x,y)
m <- length(hullindex)
xhull <- x[hullindex]
yhull <- y[hullindex]
hullorder <- order(xhull,yhull)
indexmin <- hullorder[1]
indexmax <- hullorder[m]

# determine the line separating the upper and lower hull
slope <- (yhull[indexmax] - yhull[indexmin]) / (xhull[indexmax]
```

```
    - xhull[indexmin])
intercept <- yhull[indexmax] - slope * xhull[indexmax]
count <- 0

# find length (segments) of upper and lower hulls
count <- length(which(yhull[hullorder]
    > slope * xhull[hullorder] + intercept))

# VARIATE GENERATION FUNCTIONS

# generate x from the piecewise-linear CDF from original
  (or moment matched)
# x vector

xpwl <- function(x) {
  u <- runif(1)
  i <- ceiling((xnewlength - 1) * u)
  x[i] + ((xnewlength - 1) * u - (i - 1)) * (x[i + 1] - x[i])
}

# generate y from the weighted piecewise-linear CDF created
  by conditioning
# on the x value generated

ywtpwl <- function(xgen) {

# find segments of hull lower and upper intersection with xgen,
  determine
# intersecting y values

  for (i in 1:length(upperx)) {
    if ((xgen >= upperx[i]) && (xgen <= upperx[i + 1])) {
    upperslope <- (uppery[i] - uppery[i+1]) /
     (upperx[i] - upperx[i + 1])
    upperint <- uppery[i + 1] - upperslope * upperx[i + 1]
    ymax <- upperslope * xgen + upperint
    }
  }

  for (i in 1:length(lowerx)) {
    if ((xgen >= lowerx[i]) && (xgen <= lowerx[i + 1])) {
      lowerslope <- (lowery[i] - lowery[i + 1])/
        (lowerx[i] - lowerx[i + 1])
      lowerint <- lowery[i + 1] - lowerslope * lowerx[i + 1]
      ymin <- lowerslope * xgen + lowerint
    }
  }

  # collect y values between ymin and ymax forming the set A
```

```
  j <- 0
  ybetweenindex <- 0
  for (i in 1:xnewlength) {
    if (y[i] <= ymax & y[i] >= ymin) {
      j <- j + 1
      ybetweenindex[j] <- i
    }
  }

 # create x and y vectors for interior points, augment with
     ymin, ymax
  ybetween <- y[ybetweenindex]
  xbetween <- x[ybetweenindex]
  ybetweenorder <- order(ybetween)
  yvec <- c(ymin, ybetween[ybetweenorder], ymax)
  xvec <- c(xgen, xbetween[ybetweenorder], xgen)

  # weight y values by distance from xgen, w(i)
     = 1 / (1 + ((x(i) - xgen) /
  #                            sqrt(var(xvec))) ^ 2)
  yweight <- 0
  for (i in 1:length(yvec)) {
   yweight[i] <- 1 / (1 + ((xvec[i] - xgen) / sqrt(var(xvec))) ^ 2)
  }

  # normalize weights
  ynmwt <- 0
  for (i in 1:length(yvec)) {
    ynmwt[i] <- yweight[i] / sum(yweight)
  }

  # find new y knot points of the weighted piecewise-linear CDF
  yknots <- matrix(0:0, length(yvec))
  yknots[1] <- 0
  for (i in 2:length(yvec)) {
    yknots[i] <- sum(ynmwt[1:(i - 1)]) + (i - 1) * (ynmwt[i]) /
            (length(yvec) - 1)
  }

  # generate y value pwl from knot point y values
  u1 <- runif(1)
  i <- 1
  while (u1 > yknots[i + 1]) {
    i <- i + 1
  }
  yvec[i] + (u1 - yknots[i]) * (yvec[i + 1] - yvec[i]) /
  (yknots[i + 1] - yknots[i])
}
```

```
# MAIN PROGRAM

# set N to the desired number of random variates here
# Generated <- matrix(0:0, N, 2) collects the resulting
# random variate pairs.

for (i in 1:N) {
  xgen <- xpwl(x)
  ygen <- ywtpwl(xgen)
  Generated[i, 1] <- xgen
  Generated[i, 2] <- ygen
}
```


Appendix B

We now present an isolated example where the proposed algorithm performs poorly due to the nature of the underlying population distribution. To support the illustration, a plausible formula for the limiting conditional density of $f_{Y \mid X=x_0}$ when the number n of data points goes to infinity is

$$f^*_{Y \mid X=x_0}(y) = \frac{1}{C} \int_{-\infty}^{\infty} \frac{f(x, y)}{1 + ((x - x_0)/\sigma)^2}\, dx,$$

where $\sigma^2 = \mathrm{Var}(X)$, $f(x, y) = f_{X,Y}(x, y)$ is the true joint density of the underlying population, and C is chosen so that $\int f^*(y)\, dy = 1$. The formula above for the conditional density $f^*_{Y \mid X=x_0}$ is a weighted average of the conditional densities at *all* x-coordinates, using the "damping" function $1/(1 + ((x - x_0)/\sigma)^2)$ to give less weight to x-coordinates that are at a greater distance from x_0. However, the densities for such x-coordinates still receive *some* weight. Thus, if $f_{Y \mid X=x}$ is very different from $f_{Y \mid X=x_0}$ for faraway x's (it's unlikely to be different for close x's, by the assumed continuity of the joint density f), then the estimate for $f^*_{Y \mid X=x_0}$ can be wrong. One way this can happen is if *the width of the support region varies significantly as* y *ranges from* $y_{\min}(x_0)$ *to* $y_{\max}(x_0)$. Intuitively, the extra data points in the wider part are misrepresented in the proposed algorithm giving more density to those y-coordinates.

Consider an L-shaped region with vertices $(0, 0)$, $(0, 1)$, $(1/M, 1)$, $(1/M, 1/M)$, $(1, 1/M)$, and $(1, 0)$, where M is some very large positive integer. The underlying joint density will pick (X, Y) uniformly from this shape. This L-shaped region has $f(x, y) = M^2/(2M - 1)$, so $f_X(x) = M^2/(2M - 1)$ for $0 < x < 1/M$ and $f_X(x) = M/(2M - 1)$ for $1/M < x < 1$. One can then compute $\mathrm{Var}(X) \approx 5/48 \approx 0.1$ for large M. For definiteness, we take $M = 100$ to illustrate the behavior of the proposed algorithm. Assume we generate the x-coordinate $x_0 = 0$. Then $f^*_{Y \mid X=0}(y)$ is proportional to

$$\int_0^1 \frac{1}{1 + 10x^2}\, dx \approx 0.40 \quad \text{for } 0 < y < 0.01,$$

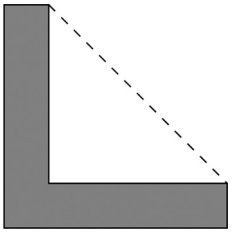

Figure 19. Joint density $f(x, y)$ for the L-shaped region with vertices $(0, 0)$, $(0, 1)$, $(1/M, 1)$, $(1/M, 1/M)$, $(1, 1/M)$, and $(1, 0)$.

whereas $f^*_{Y|X=0}(y)$ is proportional to

$$\int_0^{0.01} \frac{1}{1+10x^2}\,dx \approx 0.01 \quad \text{for } 0.01 < y < 1.$$

So, the algorithm will think that y-values less than 0.01 are 40 times more likely than y-values above 0.01 on the line $x = 0$. But, the true conditional distribution given $x = 0$ should be uniform on $(0, 1)$. By taking M larger and larger, we can create arbitrarily bad behavior that results from the difference in width in the two parts of the L in Fig. 19. It could be assumed that this behavior only occurs for $0 < x_0 < 0.01$, but that the proposed algorithm would get the conditional distribution correct for $x_0 > 0.01$. This would be the case if the simulation knew that the support of the density was the (non-convex) L-shape. In reality, the algorithm will draw the convex hull by connecting $(1/M, 1)$ to $(1, 1/M)$, and this will lead to other problems where variates can be generated outside the true support region.

Acknowledgments

We thank Jeff Robinson from the General Motors Research and Development Center for providing the data for this article and Bruce Schmeiser for his help on describing the two-dimensional empirical cdf. We acknowledge support for this research from the NSF via grant DUE-0123022 and the Omar Nelson Bradley Foundation.

References

Banks, J., Carson, J. S., Nelson, B. L., Nicol, D. M. (2001). *Discrete-Event System Simulation*. 3rd ed. New Jersey: Prentice Hall, Upper Saddle River.

Biller, B. (2009). Copula-based multivariate input models for stochastic simulation. *Operation Research* 57:878–892.

Bowman, A. W., Azzalini, A. (1997). *Applied Smoothing Techniques for Data Analysis*. Oxford: Oxford University Press.

Bratley, P., Fox, B. L., Schrage, L. E. (1987). *A Guide to Simulation*. 2nd ed. New York: Springer Verlag.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. New York: Springer Verlag.

Devroye, L., Györfi, L. (1985). *Nonparametric Density Estimation: the L_1 View*. New York: John Wiley.

Genest, C., Rëmillard, B. (2006). Discussion of "Copulas: Tales and Facts," by Thomas Mikosch. *Extremes* 9:27–36.

Hörmann, W., Leydold, J. (2000). Automatic random variate generation for simulation input. In: Joines, J. A., Barton, R. R., Kang, K., Fishwick, P. A., eds. *Proceedings of the 2000 Winter Simulation Conference*. Pistacaway, NJ: IEEE Press, pp. 675–682.

Johnson, M. E. (1987). *Multivariate Statistical Simulation*. New York: John Wiley.

Kaczynski, W. H., Leemis, L. M., Loehr, N. A., Taber, J. G. (2012). Nonparametric random variate generation using a piecewise-linear cumulative distribution function. *Communications in Statistics—Simulation and Computation* 41:449–466.

Law, A. (2007). *Simulation Modeling and Analysis*. 4th ed. New York: McGraw–Hill.

Leemis, L. M., Park, S. K. (2006). *Discrete-Event Simulation: a First Course*. Upper Saddle River, NJ: Prentice Hall.

Silverman, B. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

Taylor, M. S., Thompson, J. R. (1986). A data based algorithm for the generation of random vectors. *Computational Statistics & Data Analysis* 4:93–101.

Wagner, M. A. F., Wilson, J. R. (1995). Graphical interactive simulation input modeling with bivariate Bezier distributions. *ACM TOMACS* 5:163–189.

Weisberg, S. (1980). *Applied Linear Regression*. New York: Wiley.