

**University of Illinois at Chicago
School of Public Health
Division of Epidemiology and Biostatistics**

*Technical report#:2007-008
December 2007*

Title: Generating multivariate continuous data via the notion
of nearest neighbors

Authors: Hakan Demirtas and Donald Hedeker

**Affiliation(s): University of Illinois at Chicago, Division of Epidemiology and
Biostatistics.**

Generating multivariate continuous data via the notion of nearest neighbors

Hakan Demirtas and Donald Hedeker*

December 11, 2007

Abstract

Taylor and Thompson (1986) introduced a clever algorithm for simulating multivariate continuous datasets that resemble the original data. Their approach is predicated upon determining a few nearest neighbors of a given row of data through a statistical distance measure, and subsequently combining the observations by stochastic multipliers that are drawn from a uniform distribution to generate simulated data that essentially maintain the original data trends. The newly drawn values are assumed to come from the same underlying hypothetical process that governs the mechanism of how the data are formed. This technique is appealing in that no density estimation is required. We believe that this data-based simulation method has substantial potential in multivariate data generation due to the local nature of the generation scheme which does not have strict specification requirements as in most other algorithms. In this work, we provide two R routines: One has a built-in simulator for finding the optimal number of nearest neighbors for any given dataset, and the other generates pseudo-random data using this optimal number.

Key Words: Simulation, Random number generation, Density estimation, Bootstrap.

1 Introduction

Describing a real phenomenon by generating an environment within which the process under consideration operates is not uncommon and is often the only feasible way of evaluation. In this regard, researchers resort to the simulation paradigm that is driven by the idea of creating imperfect proxies of what is believed to be the truth, and then refining the perceived truth in an iterative fashion. Generating mirror images of the underlying

*Hakan Demirtas (e-mail:demirtas@uic.edu) is Assistant Professor and Donald Hedeker is Professor of Biostatistics, Division of Epidemiology and Biostatistics (MC923), University of Illinois at Chicago, 1603 West Taylor Street, Chicago, IL, 60612.

mechanism that leads to the observed data serves as a handy operational tool to explore actual data trends, especially in the absence of analytically tractable solutions. Even when such solutions exist for a given problem, it is constructive to verify their properties via an empirical examination of some key statistical features such as unbiasedness, efficiency, and consistency, among others.

Random number generation is an indispensable component of simulation studies that are designed to replicate the characteristics of what has been observed or measured. For this reason, scientists from a broad range of disciplines often employ multivariate data generation techniques (Devroye, 1986; Gentle, 2003). Creating simulated datasets that are generated around a real dataset has been increasingly common in statistics (e.g. Demirtas, 2005), with the rationale being re-producing the real data trends with compatible distributional properties. Because there is usually no consensus among statisticians about which of the competing methods is best, many advocate sensitivity analyses that could be performed by trying a variety of methods, or varying the model parameters over a plausible range to see what happens. This approach is valuable, but limited. Instead, we advocate the idea of simulating the performance of a method by proposing a variety of populations that are capable of producing data like those actually seen, simulating behavior of various methods over repeated samples from each population, and subsequently identifying methods that seem to perform well for most of the populations. To elaborate further, suppose we identify a family of models that, from a likelihood standpoint, fit the data equally well. If our basic conclusions about effects of interest do not change drastically over this family, then the scientific validity of these conclusions is enhanced. Conversely, if the answers do exhibit great variation, drawing firm conclusions seems unwise. Robustness of results over the domain of parameters is desirable and fortunate when it occurs. Yet there is another type of analysis which may lead us to prefer one model, M_1 , to another, M_2 , even when M_1 and M_2 achieve the same likelihood for the current data set. Suppose that we devise a variety of plausible population models, different in nature but all tending to produce samples that resemble the observed data. If, by simulation, we discover that M_1 performs better than M_2 across many of these populations, then we may be more inclined to trust M_1 than M_2 (Demirtas and Schafer, 2003; Demirtas and Hedeker, 2007).

Suppose that we have a sample and need to generate random numbers from the unknown distribution that yielded it. Specifically, we have a set of observations, and we wish to generate a pseudorandom sample from the same distribution as the given dataset. This kind of method is called data-based random number generation. If it is assumed that the

distribution is continuous, but no other assumptions are made other than some general assumptions of existence and smoothness of the probability density, the problem can be thought of as two steps. The first step is to estimate the density, and the second step is to generate random deviates from that density (Gentle, 2003).

In this article, we focus on the multivariate continuous data generation approach of Taylor and Thompson (1986), that does not require estimation of the underlying density although some concepts are borrowed from density estimation. Their method uses the m nearest neighbors of a randomly selected data row (in fact, $m - 1$ nearest neighbors and the original row itself), where m can be regarded as a smoothing parameter in the density estimation literature. This approach is particularly useful for multivariate data given its computational simplicity and ease of implementation in comparison to the methods that assume strict parametric forms. The details of their algorithm are given in Section 2.

The organization of this paper is as follows. The next section provides some key operational attributes of the method proposed by Taylor and Thompson (1986). In Section 3, we provide two R (Development Core Team, 2007) functions for empirically computing the optimal number of the nearest neighbors through a built-in simulator, and for generating the data using this number. In Section 4, we present an application to Fisher’s famous flower dataset (Fisher, 1936). Section 5 includes concluding remarks and discussion.

2 Data-based simulation methodology

Suppose we have a random sample $[Y_j]_{j=1}^n$ of size n from a multivariate distribution of dimension p . In other words, each Y_j , $j = 1, 2, \dots, n$, is a vector of length p denoting the j^{th} row of the data matrix of dimension $n \times p$. The goal is to generate a pseudorandom matrix from the unknown underlying distribution that gave rise to the random sample. The steps of the algorithm that is proposed by Taylor and Thompson (1986) are as follows:

1. Select a row Y_j at random.
2. Determine its m nearest neighbors under the Euclidean metric.
3. Re-code the vectors $[Y_j]_{j=1}^m$ around their sample mean $\bar{Y} = \frac{1}{m} \sum_{j=1}^m Y_j$ to yield $Y_j^* = [Y_j - \bar{Y}]_{j=1}^m$.
4. Generate a random sample u_1, u_2, \dots, u_m from the uniform distribution $U(1/m - \sqrt{3(m-1)/m^2}, 1/m + \sqrt{3(m-1)/m^2})$.

5. Form the linear combination $Y^{**} = \sum_{k=1}^m u_k Y_k^*$, and go back to the original scale by the translation $Y_{new} = Y^{**} + \bar{Y}$. Here, Y_{new} is a vector of length p .
6. Sample another row from the original data matrix with replacement, and repeat the above steps $n - 1$ times. Eventually, an $n \times p$ matrix of simulated draws whose dimension is the same as the original dataset is obtained.

This procedure yields a dataset that resembles the characteristics of the original dataset. The bounds in Step 4 are chosen to ensure that the drawn set of uniform deviates functions as stochastic multipliers of each neighboring row. It can be shown that the empirical first and second order marginal and product moments (mean, variance, and covariance) are compatible with the original data, on average. A mathematical proof that suggests the mechanics of this algorithmic procedure is outlined by Taylor and Thompson (1986).

When $m = 1$, the procedure is simply classical bootstrap. As m increase, we arrive at something in the spirit of Efron's (1979) smoothed bootstrap. The selection of m can be performed by iterative simulation for any given dataset. More specifically, one can simulate a fairly large number of datasets for competing values of m , and compare the sum of average absolute deviations between observed means and correlations and the empirical ones, and pick an m value that minimizes these differences. In the next section, we present an R routine that serves as a simulator to find an optimal m , and another routine that generates pseudodata for the selected m .

3 Implementation in R

The function **generate.data.via.neighbors** has two input arguments: *data* is the original data matrix, and m is the number of nearest neighbors. The output is *new.data* which represents the simulated pseudodata. The object names are chosen consistently with the algorithm outlined in Section 2.

The function **find.optimal.m** is a simulator that finds the optimal number of neighbors, as its name suggests. It simulates data repeatedly by calling the function **generate.data.via.neighbors** for competing values of m . The required input argument is *data*, and there are two optional arguments: *nsim*, the number of simulation replicates; and *m.max*, the upper limit of m . The default values are 100 and 5, respectively. One can alter these numbers considering the specific nature of the problem. Increasing these numbers did not lead to significant differences in the examples we have tried. The output is *m.optimal*,

the optimal value for m that minimizes the sum of absolute differences between expected moments and empirical ones, as mentioned in Section 2.

```

generate.data.via.neighbors<-function(data,m){
  nrow.data<-nrow(data) ; ncol.data<-ncol(data)
  distance.matrix<-matrix(0,nrow.data,nrow.data)
  new.data<-matrix(0,nrow.data,ncol.data)
  for (j in 1:nrow.data){
    distance.matrix[j,-j]<-sqrt(apply((matrix(rep(data[j,],nrow.data-1),
      nrow.data-1,ncol.data,byrow=T)-data[-j,])^2,1,sum)))}
  dist<-distance.matrix
  for (i in 1:nrow.data){
    index<-sample(1:nrow.data,1)
    neighb<-(1:nrow.data)[1*(rank(dist[index,])<m+1)& (rank(dist[index,])>0)==1]
    ybar<-apply(as.matrix(data[neighb,],1,ncol.data),2,mean)
    yj.star<-data[neighb,]-matrix(rep(ybar,m),m,ncol.data,byrow=T)
    lower.limit<-1/m-sqrt(3*(m-1)/(m^2))
    upper.limit<-1/m+sqrt(3*(m-1)/(m^2))
    unif.no<-runif(m,lower.limit,upper.limit)
    y.doublestar<-apply(unif.no*yj.star,2,sum)
    y.new<-y.doublestar+ybar
    new.data[i,]<-y.new}
  new.data}

find.optimal.m<-function(data,nsim=100,m.max=5){
  data.mean<-apply(data,2,mean)
  data.cor<-cor(data)
  avg.mean<-matrix(0,nsim,ncol(data))
  overall.mean<-matrix(0,m.max,ncol(data))
  avg.cor<-matrix(0,nsim,ncol(data)*ncol(data))
  overall.cor<-matrix(0,m.max,ncol(data)*ncol(data))
  deviation<-numeric(m.max)
  for (k in 1:m.max){
    for (l in 1:nsim){
      mydata<-generate.data.via.neighbors(data,k)

```

```

avg.mean[1,]<-apply(mydata,2,mean)
avg.cor[1,]<-as.vector(t(cor(mydata)))}
overall.mean[k,]<-apply(avg.mean,2,mean)
overall.cor[k,]<-apply(avg.cor,2,mean)
deviation[k]<-sum(abs(data.mean-overall.mean[k,]))
+sum(abs(data.cor-matrix(overall.cor[k,],ncol(data),ncol(data))))/2}
min.dev<-min(deviation)
m.optimal<-(1:m.max)[deviation==min.dev]
return(m.optimal)}

```

With default arguments ($nsim = 100$, $m.max = 5$), it took 15 and 59 seconds for data of dimension 20×2 and 50×5 , respectively, on a 3 GHZ Dell Optiplex GX270 machine that has Intel Pentium 4 processor and 1 GB RAM. Once an optimal m is found, creating a dataset using the function **generate.data.via.neighbors** takes literally no time.

4 Real data example

Our real data example comes from Fisher's famous dataset that has been frequently cited in the multivariate analysis literature. The Iris flower dataset was introduced by Fisher (1936) as an example of discriminant analysis. The dataset consists of 50 samples from each of three species of Iris flowers (setosa, versicolor, and virginica). Four features were measured from each sample, they are the length and the width of sepal and petal. Based on the combination of the four features, Fisher developed a linear discriminant model to determine which species they are. We have downloaded this public-domain dataset from the website

http://en.wikipedia.org/wiki/Iris_flower_data_set

We applied the nearest-neighbor approach to the data for each of the three species separately. First, we identified the optimal m using the function **find.optimal.m** with default input arguments ($nsim = 100$, $m.max = 5$). m value turned out to be 2, 1, and 3 for species setosa, versicolor, and virginica, respectively. Then, we created 1000 simulated datasets for each of the species using the function **generate.data.via.neighbors** repeatedly. The method of Taylor and Thompson (1986) was developed for continuous data. However, Fisher's dataset is semi-continuous. For this reason, we added a negligibly small jitter to each observation in our application. We present the average means and correlations

Table 1: Comparison of true and empirical values for each of the three species. μ 's represent the mean values, TV and EV stand for true value (data means) and empirical value (average simulated means), respectively.

<i>Species</i>	<i>Setosa</i>	<i>Versicolor</i>	<i>Virginica</i>
Parameter	TV (EV)	TV (EV)	TV (EV)
μ_1	5.006 (5.005983)	5.936 (5.935831)	6.588 (6.587394)
μ_2	3.428 (3.428045)	2.770 (2.770245)	2.974 (2.973599)
μ_3	1.462 (1.461963)	4.260 (4.260060)	5.552 (5.551918)
μ_4	0.246 (0.245948)	1.326 (1.328587)	2.026 (2.026039)

Table 2: Comparison of true and empirical values for each of the three species. $Corr(x, y)$ represents the Pearson correlation between variables x and y , TV and EV stand for true value (data means) and empirical value (average simulated means), respectively.

<i>Species</i>	<i>Setosa</i>	<i>Versicolor</i>	<i>Virginica</i>
Parameter	TV (EV)	TV (EV)	TV (EV)
$Corr(1, 2)$	0.742547 (0.741690)	0.521615 (0.522371)	0.457228 (0.456983)
$Corr(1, 3)$	0.267176 (0.268328)	0.754049 (0.755126)	0.864225 (0.862959)
$Corr(1, 4)$	0.278098 (0.277729)	0.546461 (0.547002)	0.281108 (0.280961)
$Corr(2, 3)$	0.177700 (0.178612)	0.560522 (0.559843)	0.401044 (0.400847)
$Corr(2, 4)$	0.232752 (0.234109)	0.663999 (0.663580)	0.537728 (0.537264)
$Corr(3, 4)$	0.331630 (0.327826)	0.786668 (0.787016)	0.322108 (0.322862)

across 1000 replications in Tables 1 and 2, respectively. Let 1, 2, 3, and 4 stand for sepal length, sepal width, petal length, and petal width, respectively. In Table 1, μ 's represent the mean values, TV and EV stand for true value (the data mean) and empirical value (the average simulated mean), respectively. Table 2 shows Pearson correlations among four variables, and follows the same format.

The results tabulated in Tables 1 and 2 demonstrate that the procedure is working properly, yielding numbers that are in very close agreement to the data means and correlations.

5 Concluding remarks

There are some issues that need to be addressed. First of all, we do not claim that these R routines are the most efficient. Given sufficient time and energy, one can write more

efficient routines. Secondly, an alternative way of generating pseudorandom data is fitting generalized classes of families (see Genton, 2004 for a review). These families can accommodate a broad range of distributional shapes, but the price to be paid is having to estimate many parameters through complex computational algorithms. Furthermore, even with model parameters at hand, simulating random data matrices from these distributions is often a challenging task. In this regard, the method of Taylor and Thompson (1986) is far more flexible and easier to implement as well as it does not require strict distributional assumptions to hold. It is also more “real” than distribution-based methods in the sense that simulated data rows are a weighted average of the original data themselves. Finally, we believe that this data-based simulation method has substantial potential in many areas of research, especially in missing-data problems. For example, one can sort the data with respect to missingness patterns and apply this approach separately for each pattern, then combine them to generate incomplete datasets whose behavior mimic the original incomplete dataset. This concept ideally suits to the simulation paradigm we described in Section 2. The R functions we provide can be useful tools from practitioners’ point of view to carry out this procedure.

6 References

- Demirtas H. and Schafer J.L. (2003). On the performance of random-coefficient pattern-mixture models for non-ignorable drop-out. *Statistics in Medicine* 22:2553–2575.
- Demirtas, H. (2005). Multiple imputation under Bayesianly smoothed pattern-mixture models for non-ignorable drop-out. *Statistics in Medicine* 24:2345–2363.
- Demirtas, H. and Hedeker, D. (2007). Gaussianization-based quasi-imputation and expansion strategies for incomplete correlated binary responses. *Statistics in Medicine* 26:782–799.
- Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Annals of Statistics* 7:1–26.
- Devroye, L. (1986). *Non-Uniform Random Variate Generation.* , Springer-Verlag, New York.
- Fisher, R.A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics* 7: 179–188.

Gentle, J.E. (2003). *Random Number Generation and Monte Carlo Methods*. Second Edition, Springer-Verlag, New York.

Genton, M.G. (Ed) (2004). *Skew-Elliptical Distributions and Their Applications: A Journey Beyond Normality*. Boca Raton, FL: Chapman and Hall/CRC.

R Development Core Team. (2007). *R: A Language and Environment for Statistical Computing*. Foundation for Statistical Computing, Vienna, Austria, URL <http://www.R-project.org>. ISBN 3-900051-07-0.

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