Journal of Applied Statistics
Vol. 38, No. 1, January 2011, 47–55

Taylor & Francis
Taylor & Francis Group

Generating multivariate continuous data via the notion of nearest neighbors

Hakan Demirtas* and Donald Hedeker

Division of Epidemiology and Biostatistics (MC923), University of Illinois at Chicago, 1603 West Taylor Street, Chicago, IL, 60612, USA

(*Received 3 December 2008; final version received 1 August 2009*)

Taylor and Thompson [15] introduced a clever algorithm for simulating multivariate continuous data sets 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 data set, and the other generates pseudo-random data using this optimal number.

Keywords: simulation; random number generation; density estimation; bootstrap; nearest neighbors

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 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 genera-

*Corresponding author. Email:demirtas@uic.edu

ISSN 0266-4763 print/ISSN 1360-0532 online

DOI: 10.1080/02664760903229260
http://www.informaworld.com

tion techniques [5,8]. Creating simulated data sets that are generated around a real data set has been increasingly common in statistics [2], 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 [1,3,4].

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 data set. 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 [8].

In this article, we focus on the multivariate continuous data generation approach of Taylor and Thompson [15], which 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 article is as follows. The next section provides some key operational attributes of the method proposed by Taylor and Thompson [15]. In Section 3, we give information about the two R [13] functions that we provide in the Appendix 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 two applications to data from multivariate analysis literature [7] and psychiatric research [14]. 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, $j = 1, 2, \ldots, n$, is a vector of length p denoting the jth row of the data matrix of dimension $n \times p$. The goal is to generate a pseudorandom matrix from the

unknown underlying distribution that gives rise to the random sample. The steps of the algorithm that is proposed by Taylor and Thompson [15] are as follows:

(1) Select a row Y_j at random.
(2) Determine its m nearest neighbors under the Euclidean metric.
(3) Center the vectors $[Y_j]_{j=1}^m$ around their sample mean $\bar{Y} = 1/m \sum_{j=1}^m Y_j$ to yield $Y_j^* = [Y_j - \bar{Y}]_{j=1}^m$.
(4) Generate a random sample $u_1, u_2, \ldots, u_m$ 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 $Y_{\text{new}} = Y^{**} + \bar{Y}$. 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 data set is obtained.

This procedure yields a data set that resembles the characteristics of the original data set. 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 mechanism of this algorithmic procedure is outlined by Taylor and Thompson [15].

When $m = 1$, the procedure is simply classical bootstrap. As m increases, we arrive at something in the spirit of Efron's [6] smoothed bootstrap where a small amount of (usually normally distributed) zero-centered random noise is added on to each resampled observation. The selection of m can be performed by an iterative simulation for any given data set. More specifically, one can simulate a fairly large number of data sets 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 refer to 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

We provide the two R functions in the Appendix. 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 optimal 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 increase the defaults considering the specific nature of the problem (if erratic features are present in the data or when the number of variables and/or subjects is large). Changing these numbers did not lead to substantial 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.

With default arguments (nsim = 100, m.max = 5), it took 15 and 59 s for data of dimensions 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 data set using the function

generate.data.via.neighbors takes literally no time. Of note, the code is equally functional in Splus [12].

4. Real data examples

Our first data example comes from Fisher's famous data set that has been frequently cited in the multivariate analysis literature. The Iris flower data set was introduced by Fisher [7] as an example of discriminant analysis. The data set consists of 50 samples from each of three species of Iris flowers (setosa, versicolor, and virginica). Four features were measured from each sample, and 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 data set from the following 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 data sets for each of the species using the function *generate.data.via.neighbors* repeatedly. The method of Taylor and Thompson [15] was developed for continuous data. However, Fisher's data set is semi-continuous. For this reason, we added a negligibly small jitter to each observation in our application. The method is literally a weighted bootstrap procedure where the individual weights of each data row do not have to be equal, and it is based on identifying nearest neighbors via a statistical distance measure. Therefore, it does not allow identical observations within the same row of the data matrix since it would lead to multiple candidate rows whose number may exceed the specified number of neighbors. A refinement to circumvent this complication could be adding a random disturbance whose magnitude is extremely small, which is what we did in the current investigation.

Table 1 shows some descriptive statistics (mean, standard deviation, range, and mode) for Fisher's data. We present the simulation results for the average means and correlations across 1000 replications in Tables 2 and 3, respectively. In Tables 1–3, the numbers 1, 2, 3, and 4 in the first column stand for sepal length, sepal width, petal length, and petal width, respectively. In Table 2, μs represent the mean values, TV and EV stand for true value (the data mean) and

Table 1. Descriptive statistics for the Iris flower data.

Species	Setosa	Versicolor	Virginica
Variable	Mean (standard deviation)	Mean (standard deviation)	Mean (standard deviation)
1	5.006 (0.352)	5.936 (0.516)	6.588 (0.636)
2	3.428 (0.379)	2.770 (0.314)	2.974 (0.322)
3	1.462 (0.174)	4.260 (0.470)	5.552 (0.552)
4	0.246 (0.105)	1.326 (0.198)	2.026 (0.275)
	Range (mode)	Range (mode)	Range (mode)
1	4.3–5.8 (5.05)	4.9–7.0 (6.05)	4.9–7.9 (6.3)
2	2.3–4.4 (3.40)	2.0–3.4 (3.00)	2.2–3.8 (3.0)
3	1.0–1.9 (1.45)	3.0–5.1 (4.50)	4.5–6.9 (5.1)
4	0.1–0.6 (0.20)	1.0–1.8 (1.50)	1.4–2.5 (1.8)

Note: Means and standard deviations are shown in the upper part, ranges and modes are shown in the lower part.

Table 2. Comparison of true and empirical values for each of the three species.

Species	Setosa	Versicolor	Virginica
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)

Note: The parameters μs represent the mean values, TV and EV stand for true value (data mean) and empirical value (average simulated mean), respectively.

Table 3. Comparison of true and empirical values for each of the three species.

Species	Setosa	Versicolor	Virginica
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)

Note: Corr(x, y) represents the Pearson correlation between variables x and y, TV and EV stand for true value (data mean) and empirical value (average simulated mean), respectively.

empirical value (the average simulated mean), respectively. Table 3 shows Pearson correlations among four variables, and follows the same format.

The results tabulated in Tables 2 and 3 demonstrate that the procedure is working properly, yielding numbers that are in very close agreement to the data means and correlations.

Our second example comes from a study in psychiatric research [14]. It was fairly extensively discussed in Hedeker and Gibbons [11]. This study focused on the longitudinal relationships between imipramine (IMI) and desipramine (DMI) plasma levels and clinical response in 66 depressed inpatients. IMI is the prototypic drug in the series of compounds known as tricyclic antidepressants, and is commonly prescribed for the treatment of major depression.

The study design was as follows. Following a placebo period of 1 week, patients received 225 mg/day doses of IMI for 4 weeks. In this study, subjects were rated with the Hamilton Depression Rating Scale (HDRS) [10] twice during the baseline placebo week (at the start and end of this week) as well as the end of each of the four treatment weeks of the study. These HDRS scores represent the outcome variable that is measured across time. Higher scores on the HDRS represent higher levels of depression and lower scores indicate less depression. Plasma level measurements of both IMI and its metabolite DMI were made at the end of each week. The sex and age of each patient was recorded and a diagnosis of endogenous and nonendogenous depression was made for each patient.

For the purpose of this work, we focus on change in HDRS since the baseline, IMI and DMI measurements for the 4 drug weeks. In what follows, HDRS stands for the difference of HDRS with respect to the baseline measurement across 4 weeks. Although the total number of subjects in this study was 66, the number of subjects with all measures at each of the week fluctuated. Of the 66 subjects, only 52 had complete data at all time points for HDRS, IMI, and DMI. For the remainder of the article, we use this complete portion for reporting the descriptive statistics and subsequent simulation study in implementing the algorithm of Taylor and Thompson [15].

Table 4. Comparison of true and empirical values for HDRS, IMI and DMI.

Variable	HDRS	IMI	DMI
Parameters	TV (EV)	TV (EV)	TV (EV)
μ_1	−5.230769 (−5.231998)	3.888899 (3.883973)	4.549794 (4.550581)
μ_2	−6.634615 (−6.632238)	3.998663 (3.997256)	4.764238 (4.774197)
μ_3	−9.461538 (−9.469347)	3.989774 (3.983743)	4.804877 (4.812775)
μ_4	−11.115385 (−11.113386)	3.957733 (3.953614)	4.778593 (4.773755)
σ_1	5.143492 (5.153375)	0.710522 (0.711647)	0.760885 (0.759874)
σ_2	5.804085 (5.807556)	0.693643 (0.693647)	0.827967 (0.828067)
σ_3	6.800985 (6.807430)	0.693719 (0.693237)	0.795232 (0.795407)
σ_4	7.752682 (7.749825)	0.637670 (0.638978)	0.746944 (0.745752)

Note: The parameters μs represent the mean values, σs stand for standard deviations, TV and EV stand for true value (data mean) and empirical value (average simulated mean), respectively.

Table 5. Comparison of true and empirical values for HDRS, IMI and DMI.

Variable	HDRS	IMI	DMI
Parameters	TV (EV)	TV (EV)	TV (EV)
Corr(1,2)	0.707634 (0.708371)	0.933663 (0.933400)	0.961628 (0.965695)
Corr(1,3)	0.713255 (0.727403)	0.929230 (0.931986)	0.945416 (0.956169)
Corr(1,4)	0.518086 (0.513939)	0.876615 (0.882436)	0.927838 (0.931495)
Corr(2,3)	0.791680 (0.793227)	0.949733 (0.949567)	0.948490 (0.951829)
Corr(2,4)	0.628445 (0.631617)	0.885413 (0.882966)	0.906064 (0.908004)
Corr(3,4)	0.712241 (0.709372)	0.913355 (0.921669)	0.948834 (0.942866)

Note: Corr(x, y) represents the Pearson correlation between variables x and y, TV and EV stand for true value (data mean) and empirical value (average simulated mean), respectively.

Implications of missing data and possible extensions are briefly mentioned in the discussion section.

We proceeded by replicating the previously described process to the three subsets of data. Each of the three multivariate subsets of data (HDRS, IMI, and DMI which have been measured at four time points) are analyzed separately. As before, we found the optimal m using the function *find.optimal.m* with arguments nsim $= 50$ and m.max $= 5$. The computed m values are 1, 4, and 3 for HDRS, IMI, and DMI, respectively. Then, we created 1000 simulated data sets for each of the species using the function *generate.data.via.neighbors* repeatedly. In the HDRS part, there were ties at few data rows, so an infinitesimal jitter was added.

Table 4 shows true parameter values and simulation results for the mean and standard deviation across three variables. Table 5 tabulates the results of Pearson correlations. The format of Tables 4 and 5 are very similar to that of Tables 2 and 3. In Tables 4 and 5, the numbers 1, 2, 3, and 4 in the first column stand for the measurement week. As we demonstrate in Tables 4 and 5, differences between the true and simulated values are very close to zero, which can be considered a further support for the method of Taylor and Thompson [15].

5. Concluding remarks and discussion

There are some issues that need to be addressed. First, our evaluation was based on accuracy. As pointed out by a referee, precision issues are also important. From a conceptual standpoint, m acts like the bandwidth parameter in density estimation. In nonparametric density estimation, $m = 1$

corresponds to a situation where bias is minimal and variability is large, and as m gets larger, bias goes up (more smoothing, worse fit) and precision goes up (more smoothing, less variability). In a supplementary simulation study (not reported for brevity), we observed that there was a bias–variance trade-off to some extent, but differences with respect to the changing values of m were not substantial. The directions of bias and variance were somewhat similar; however, one would expect much more dramatic differences in density estimation where $m = 1$ is almost a catastrophic event. Despite the conceptual resemblance between the bandwidth parameter in density estimation and m in Taylor and Thompson [15], they operationally correspond to distinct paradigms, and differential effect of the choice of m is expected to be much less apparent with Taylor and Thompson [15] compared with the density estimation. Second, the optimal value of m was computed via the sum of absolute differences between the expected moments and empirical ones. One can modify the code in a way that uses the sum of squared differences or other plausible distance measures. Third, although the selection criterion for m was predicated upon the means and correlations, the algorithm as implemented in the given code adequately retains the higher-order moments such as skewness and kurtosis, as supported by our auxiliary simulation study. The usage of only the first two moments in the code certainly does not mean that higher-order moments are not reasonably captured. Somewhat related to this, one may raise the possibility of using simpler methods such as generating data with multivariate normal distribution whose parameters are determined by the underlying data. This approach would preserve the mean and variance–covariance structure; however, when nonnormal features (heavy tails, skewness, multimodality, boundary at the mode, etc.) are present, we can expect this method to fare miserably in terms of higher-order moments. Fourth, we do not claim that these R routines are the most efficient. Given sufficient time and energy, one can write more efficient routines. Fifth, an alternative way of generating pseudorandom data is fitting generalized classes of families (see [9] 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 [15] 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 data sets whose behavior mimic the original incomplete data set. This concept ideally suits 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.

References

[1] H. Demirtas, *Simulation-driven inferences for multiply imputed longitudinal datasets*, Statist. Neerlandica 58 (2004), pp. 466–482.

[2] H. Demirtas, *Multiple imputation under Bayesianly smoothed pattern-mixture models for non-ignorable drop-out*, Stat. Med. 24 (2005), pp. 2345–2363.

[3] H. Demirtas and D. Hedeker, *Gaussianization-based quasi-imputation and expansion strategies for incomplete correlated binary responses*, Stat. Med. 26 (2007), pp. 782–799.

[4] H. Demirtas and J.L. Schafer, *On the performance of random-coefficient pattern-mixture models for non-ignorable drop-out*, Stat. Med. 22 (2003), pp. 2553–2575.

[5] L. Devroye, *Non-Uniform Random Variate Generation*, Springer-Verlag, New York, 1986.

[6] B. Efron, *Bootstrap methods: Another look at the jackknife*, Ann. Statist. 7 (1979), pp. 1–26.

[7] R.A. Fisher, *The use of multiple measurements in taxonomic problems*, Ann. Eugen. 7 (1936), pp. 179–188.

[8] J.E. Gentle, *Random Number Generation and Monte Carlo Methods*, 2nd ed., Springer-Verlag, New York, 2003.

[9] M.G. Genton (ed.), *Skew-Elliptical Distributions and Their Applications: A Journey Beyond Normality*, Chapman and Hall/CRC, Boca Raton, FL, 2004.
[10] M. Hamilton, *A rating scale for depression*, J. Neurol. Neurosurg. Psychiat. 23 (1960), pp. 56–62.
[11] D. Hedeker and R.D. Gibbons, *Longitudinal Data Analysis*, John Wiley & Sons, Inc., Hoboken, NJ, 2006.
[12] Insightful Corporation, *S-PLUS version 7.2*, Seattle, WA, 2007. Available at http://www.insightful.com.
[13] R Development Core Team, *R: A Language and Environment for Statistical Computing*, Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3-900051-07-0. Available at http://www.R-project.org.
[14] N. Reisby, L.M. Gram, P. Bech, A. Nagy, G.O. Petersen, J. Ortmann, I. Ibsen, S.J. Dencker, and O. Jacobsen, *Imipramine: Clinical effects and pharmacokinetic variability*, Psychopharmacology 54 (1977), pp. 263–272.
[15] M.S. Taylor and J.R. Thompson, *A data based algorithm for the generation of random vectors*, Comput. Statist. Data Anal. 4 (1986), pp. 93–101.

Appendix

```
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[l,]<-apply(mydata,2,mean)
avg.cor[l,]<-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)}
```