Statistica Sinica **18**(2008), 1669-1687

DISCLOSURE RISK AND REPLICATION-BASED VARIANCE ESTIMATION

Wilson W. Lu and Randy R. Sitter

Acadia University and Simon Fraser University

Abstract: Protecting respondents from disclosure of their identity in publicly released survey data is of practical concern to many government agencies. Methods for doing so include suppression of cluster and stratum identifiers, and altering or swapping record values between respondents. Unfortunately, stratum and cluster identifiers are usually needed for variance estimation using linearization or replication methods. One might feel that releasing a set of replicate weights that also have stratum and cluster identifiers suppressed might circumvent this problem to some extent, especially using some random resampling such as the bootstrap. In this article, we first demonstrate that by viewing the replicate weights as observations in a high dimensional space one can easily use clustering algorithms to reconstruct the cluster identifiers irrespective of the resampling method even if the replicate weights are randomly altered. We then propose a fast algorithm for swapping cluster and strata identifiers of ultimate units before creating replicate weights without significantly impacting resulting variance estimates of characteristics of interest. The methods are illustrated by application to publicly released data from the National Health and Nutrition Examination Surveys, where such disclosure issues are extremely important.

Key words and phrases: Balanced repeated replication, bootstrap, confidentiality, jackknife.

1. Introduction

Protecting against inadvertent disclosure of respondent identity in publicly released data files from complex surveys is becoming increasingly important as web-access to census data and other auxiliary data becomes more prevalent, and computational resources become faster and require less skill. Most of the literature on the topic considers the masking of the released data itself (see for example, Lambert (1993) and the remainder of that issue of the *Journal of Official Statistics* which is dedicated to the topic; and also see Skinner and Elliot (2002) and Skinner and Carter (2003) for some nice discussion and a large set of related references). One common simple measure taken to help avoid identity disclosure is to suppress the stratum and primary sampling unit (PSU) identifiers (first-stage cluster identifiers in a multi-stage survey). This can make it more

difficult for an "attacker" to match a sampled cluster to a population cluster and thus narrow the search for respondent identity, but it can also make it difficult for direct variance estimation since stratum and PSU identifiers are typically needed to obtain asymptotically unbiased variance estimators.

One might feel that releasing a set of replicate weights together with the data which also have the PSU and stratum identifiers suppressed would circumvent the problem. Yung (1997) shows this may not be so. He proposes a repeated bootstrap in an attempt to create replicate weights from which it is more difficult to reconstruct PSU and/or stratum identifiers. As we will show, this method falls short of its goal and, in fact, one can quickly and easily reconstruct PSU identifiers from essentially any set of replicate weights if the problem is viewed appropriately.

Another method that is commonly used to mask the data in complex surveys is to change data values or swap data values between cases. In our setting, one might swap stratum and/or PSU identifiers before creating the set of replicate weights. This would only impact the variance estimation and could be done, such that the impact on resulting variance estimates is small, for the characteristics measured in the survey. This idea was introduced in Lu (2004), and a semi-manual method for implementing it in the 2001-2002 release of the National Health and Nutrition Examination Surveys (NHANES) was presented in Dohrmann, Lu, Park, Sitter and Curtin (2006). The basic idea is to match second-stage clusters on some demographic variables by using a record-linkage algorithm, and then to sort through the matches to decide on ones for which PSU identifiers would be swapped for all ultimate units in the second-stage cluster. The paper also gives some discussion on how the variables might be chosen and how to evaluate the impact on variance estimates and on the potential for disclosure of PSU/strata affiliations via graphical techniques.

The purpose of the current article is two-fold: (1) to illustrate the ease with which one can identify PSUs through replication weights; and (2) to propose a solution through a fast algorithm for swapping PSU identifiers. In section 2 we discuss the general structure of the problem and demonstrate how, by viewing the problem of reconstructing PSU and/or stratum identifiers from replicate weights as a high-dimensional clustering problem, one can easily reconstruct PSU identifiers from any of the usual replication methods without knowledge of which replication method was used, even if the replication weights are randomly perturbed, or altered at each replicate to account for non-response and/or post-stratification. We also demonstrate that despite the fact that bootstrap replicate weights seem at first glance to provide more protection, and the method of repeated bootstrapping of Yung (1997) even more so, in fact they are actually much easier

to reconstruct from than say the jackknife or the method of balanced repeated replications. In Section 3 we propose an extremely fast and simple algorithm for swapping PSU identifiers and discuss the inclusion of design weights in measures of distance. In Section 4, we compare the proposed algorithm in terms of performance and speed to a version of the match-and-swap approach used in Dohrmann et al. (2006) that we enhance to make more automatic. We finish with some concluding remarks and relegate proofs to an appendix.

2. Disclosure Risk from Replication Weights

2.1. Complex survey, design weights and replication

To introduce replication-based variance estimation methods, consider a stratified multi-stage design in which PSUs (clusters) are selected with replacement or are so treated for the purposes of variance estimation, with independent subsamples taken within clusters that are selected more than once. Suppose n_h PSUs are selected with probabilities p_{hi} with replacement or with inclusion probabilities $\pi_{hi} = n_h p_{hi}$, independently within each stratum. Let $\hat{\boldsymbol{Y}}_{hi}$ be a linear unbiased estimator of the vector of totals for the i-th PSU from stratum h based on sampling at the second and subsequent stages, so that $\hat{\boldsymbol{Y}}_h = \sum_{i=1}^{n_h} \hat{\boldsymbol{Y}}_{hi}/(n_h p_{hi})$ is a linear unbiased estimator of the vector of stratum totals $\boldsymbol{Y}_h$. A linear unbiased estimator of the total $\boldsymbol{Y} = \sum_h \boldsymbol{Y}_h$ is then given by $\hat{\boldsymbol{Y}} = \sum_h \hat{\boldsymbol{Y}}_h$. This can be written as

$$\hat{\boldsymbol{Y}} = \sum_{(hik)\in s} w_{hik} \boldsymbol{y}_{hik}, \tag{2.1}$$

where s is the total sample of elements, and w_{hik} and $\boldsymbol{y}_{hik} = (y_{1hik}, \ldots, y_{phik})'$, respectively, denote the sampling (or design) weight and the vector of item values attached to the hik-th sampled element ($k = 1, \ldots, n_{hi}; i = 1, \ldots, n_h; h = 1, \ldots, H$).

Often a survey estimator can be expressed as a function of a vector of estimated totals as in (2.1), $\hat{\theta} = g(\hat{\boldsymbol{Y}})$. The population distribution function can be estimated by $\hat{F}_n(t) = \sum_s w_{hik} I_{[y_{hik}\le t]}/(\sum_s w_{hik})$, where $I_{[\,\cdot\,]}$ is the indicator function. Some non-smooth estimators that are often of interest are the p-th sample quantiles, $\hat{F}^{-1}(p)$, where $\hat{F}^{-1}$ is the inverse of $\hat{F}$.

For replication methods, subsamples are repeatedly selected from the full sample, the statistic of interest is computed for each subsample, and the variability among the subsample or replicate estimates is used to estimate the variance of the full sample statistic. The delete-1 jackknife variance estimator is given by $v_J = \sum_{g=1}^{H} (n_g - 1) n_g^{-1} \sum_{j=1}^{n_g} (\hat{\theta}_{(gj)} - \hat{\theta})^2$, where $\hat{\theta}_{(gj)}$ is calculated as before but with w_{hik} replaced by $w_{hik(gj)} = b_{gj} w_{hik}$, where $b_{gj} = 0$ if $(hi) = (gj)$;

$= n_g/(n_g-1)$ if $h = g$ and $i \neq j$; $= 1$ if $h \neq g$. This estimator requires $n = \sum_h n_h$ replicates.

The method of balanced repeated replications (BRR) for $n_h = 2$ (McCarthy (1966)) forms a set of R balanced half-samples or replicates by deleting one PSU from the sample in each stratum simultaneously, where this set is defined by an $R \times H$ matrix $(\alpha_{rh})_{R\times H}$ with $\alpha_{rh} = +1$ or -1 according to whether the first or the second PSU of stratum h is in the r-th half-sample and $\sum_{r=1}^{R} \alpha_{rh}\alpha_{rh'} = 0$ for all $h \neq h'$; that is, the columns of the matrix are orthogonal. A minimal set of R balanced half-samples can be constructed from an $R \times R$ Hadamard matrix by choosing any H columns excluding the column of all $+1$'s, where $H+1 \leq R \leq H+4$. The estimator $\hat{\theta}_{(r)}$ is obtained using the same formula as for $\hat{\theta}$, with w_{hik} changed to $w_{hik(r)}$ which equals $2w_{hik}$ or 0 according to whether or not the hi-th PSU is selected in the r-th half-sample or not. The BRR variance estimator for $\hat{\theta}$ is then given by $v_{BRR} = \sum_r (\hat{\theta}_{(r)} - \hat{\theta})^2/R$ (for extentions to $n_h > 2$, see Gurney and Jewett (1975), Wu (1991) and Sitter (1993)).

A generalization that has some advantages is often termed the Fay BRR (Dippo Fay and Morganstein (1984) and Judkins (1990). Choose $0 \leq \epsilon < 1$, redefine $w_{h1k(r)} = [1 + \alpha_{rh}(1-\epsilon)]w_{h1k}$, $w_{h2k(r)} = [1 - \alpha_{hr}(1-\epsilon)]w_{h2k}$, and let $v_{FBRR} = \sum_r (\hat{\theta}^{(r)} - \hat{\theta})^2/[R(1-\epsilon)^2]$. Fay's BRR is the usual BRR when $\epsilon = 0$.

There have been a number of bootstrap methods proposed for stratified multi-stage sampling (Rao and Wu (1988), Sitter (1992a,b), Booth, Butler and Hall (1994) and Shao and Tu (1995)). One simple method is the rescaling bootstrap of Rao and Wu (1988) (see also Rao, Wu and Yue (1992)):

1. Draw m_h^* clusters with replacement from the n_h sample PSUs in each stratum.
2. Let n_{hi}^* be the number of times the hi-th sample PSU is selected ($\sum_i n_{hi}^* = m_h^*$).
3. Define bootstrap weights

$$w_{hik}^* = w_{hik}\left[\left\{1 - \left(\frac{m_h^*}{n_h - 1}\right)^{\frac{1}{2}}\right\} + \left(\frac{m_h^*}{n_h - 1}\right)^{\frac{1}{2}} \left(\frac{n_h}{m_h^*}\right) n_{hi}^*\right],$$

   where n_{hi}^* =number of times the hi-th PSU is resampled, and calculate $\hat{\theta}^*$ using the bootstrap weights.
4. Repeat a large number of times, R, to get $\hat{\theta}_{(1)}^*, \ldots, \hat{\theta}_{(R)}^*$ and let $v_B = \sum_{r=1}^{R} (\hat{\theta}_{(r)}^* - \hat{\theta})^2/R$.

Note that, if n_h^* is chosen as $n_h - 1$, $w_{hik}^* = w_{hik} n_{hi}^* n_h/(n_h - 1)$.

The variance estimators v_J, v_{BRR} and v_B have been shown to be consistent for smooth functions of estimated means or totals (see Krewski and Rao (1981)

and Rao and Wu (1985, 1988)) and v_{BRR} and v_B for nonsmooth estimators (see Shao and Tu (1995) and references therein).

2.2. Replicate weights and stratum/PSU identifiers

In this section, we suppress the triple index and write $\hat{\boldsymbol{Y}} = \sum_{j\in s} w_j \boldsymbol{y}_j$, where $j = (hik)$. For public release data where stratum and PSU identifiers are being suppressed, this is how the data would be released in some random order.

All of the replication methods of the previous section can be rewritten as selecting R subsets of the full sample $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m$, where $m = \sum_{(hi)\in s} n_{hi}$ is the total number of ultimate units in the sample, according to some re-sampling mechanism to form replicate estimates denoted as $\hat{\theta}_{(1)}, \hat{\theta}_{(2)}, \cdots, \hat{\theta}_{(R)}$. The r-th replicate estimate, $\hat{\theta}_{(r)}$, is calculated in the same way as $\hat{\theta}$ but using the r-th set of replicate weights $w_{j(r)}$, $j = 1, \ldots, m$. The variation among replicate estimates, $v(\hat{\theta}) = \sum_{r=1}^{R} c_r(\hat{\theta}_{(r)} - \hat{\theta})^2$, is then used to estimate $V(\hat{\theta})$, where the c_r are constants specific to the replication method.

Table 1. The matrix representation of design weights and replicate weights.

Sample	Characteristics	Design Weights	Replicate Weights			
1	$\boldsymbol{y}_1$	w_1	$w_{1(1)}$	$w_{1(2)}$	$\cdots$	$w_{1(R)}$
2	$\boldsymbol{y}_2$	w_2	$w_{2(1)}$	$w_{2(2)}$	$\cdots$	$w_{2(R)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m	$\boldsymbol{y}_m$	w_m	$w_{m(1)}$	$w_{m(2)}$	$\cdots$	$w_{m(R)}$

The typical form of the released data is given in Table 1. The end-user can then use the same program that calculates $\hat{\theta}$ from $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ and $w_1, \ldots, w_m$ to get $\hat{\theta}_{(1)}, \ldots, \hat{\theta}_{(R)}$ by applying it to $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ and $w_{1(r)}, \ldots, w_{m(r)}$ for $r = 1, \ldots, R$. To understand how one can use the replicate weights in Table 1 to reconstruct the PSU and/or stratum identifiers even if they are tabled in some arbitrary order, let $\delta_{j(r)} = w_{j(r)}/w_j$, the ratios of replicate weights and design weights, where $j = 1, \cdots, m$ and $r = 1, \cdots, R$, depicted in Table 2.

Table 2. The matrix representation of indicator variables

	Replicate			
Sample	1	2	$\cdots$	R
1	$\delta_{1(1)}$	$\delta_{1(2)}$	$\cdots$	$\delta_{1(R)}$
2	$\delta_{2(1)}$	$\delta_{2(2)}$	$\cdots$	$\delta_{2(R)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m	$\delta_{m(1)}$	$\delta_{m(2)}$	$\cdots$	$\delta_{m(R)}$

Suppose the replicate weights are full jackknife replicate weights. As each jackknife replicate is formed by deleting all the elements from a particular PSU and up-weighting those in the same stratum as the one removed, the relationship between jackknife replicate weights and the original design weights is obvious. In particular, the r-th column of Table 2, corresponding to the r-th replicate, will only consist of three different values: 0, 1 and a positive constant, say $k_r(> 1)$. If we know or deduce that these are jackknife replicate weights, we can easily conclude that the elements in column r with $\delta_{j(r)} = 0$, $\delta_{j(r)} > 1$ and $\delta_{j(r)} = 1$ are from the deleted PSU, the same stratum as the deleted PSU, and from a different stratum from the deleted PSU, respectively. Thus the PSU and stratum identifiers can be easily determined by examining all R columns in Table 2.

In the case of BRR, it may seem more difficult to reconstruct the stratum and/or PSU identifiers as each BRR replicate consists of half of the sampled PSUs from each stratum. However, if any two rows of Table 2 are identical, then the two corresponding sample units are from the same PSU;, and if any two rows of Table 2 are complementary to each other, meaning that the sum of two rows is a vector of 2's, the two corresponding units are from different PSUs in the same stratum. A similar idea will work for Fay's BRR (FBRR).

For bootstrap replicate weights, it seems even more difficult to reconstruct the stratum and/or PSU identifiers than for the BRR or jackknife. Yung (1997) considers the special case of the rescaling bootstrap with $m_h^* = n_h - 1$ PSUs randomly selected from each stratum with n_h original clusters to form bootstrap replicates. He notes that some partial cluster membership can be identified in this case. In particular, for any bootstrap replicate, at least one PSU from each stratum will be excluded from the replicate, resulting in replicate weights of 0 for all elements from these PSUs. Yung goes on to propose a mean bootstrap method (M-bootstrap) in an effort to hide the PSU identifiers from the end users. The key idea is to repeat the resampling procedure for each bootstrap replicate enough times, say b times and average bootstrap replicate weights in an effort to select each PSU at least once in each replicate. Therefore, all the final bootstrap weights will be nonzero for each replicate. We delay further discussion on this to the sequel.

2.3 A simple method for reconstructing PSU/stratum identifiers

In the previous section, we discussed the connection between replicate weights and stratum and/or PSU identifiers and argued that, by closely examining the relative change of replicate weights and knowing the replication method was a jackknife or BRR, we are able to reconstruct at least the original PSU identifiers and perhaps the stratum identifiers, provided that neither non-response nor post-stratification weight adjustment is applied. We could further investigate the

impact of such weight adjustments, which is likely to make the reconstruction procedure a little more complex. However, we find that reconstructing PSU identifiers is actually much simpler than it appears. In this section, we introduce a simple approach and then apply it to some data combined with a small numerical study to illustrate the simplicity and effectiveness of the proposed method, even when the weights are randomly perturbed or have been adjusted in some way.

To motivate the idea, treat the i-th row of Table 2 as an R dimensional object with entries $\delta_{j(1)}, \cdots, \delta_{j(R)}$, and define a distance, $d(j,l)$, of any pair of sample elements j and l. The following two lemmas demonstrate the potential advantages of viewing the problem in this way.

Lemma 1. *Let* $d(j,l) = \sum_{r=1}^{R} |\delta_{j(r)} - \delta_{l(r)}|$.

(a) *For jackknife replicate weights,* $d(j,l) = 0$, $2n_h/(n_h - 1)$, *and 4 if sample elements j and l are from the same PSU, different PSUs within the same stratum, and different strata, respectively.*

(b) *For FBRR replicate weights,* $d(j,l) = 0$, $2R(1-\epsilon)$, *and* $R(1-\epsilon)$ *if sample elements j and l are from the same PSU, different PSUs within the same stratum, and different strata (Note that* $\epsilon = 0$ *implies BRR), respectively.*

Proof. See Appendix in online Supplementary Document.

Lemma 2. *Let*

$$d(j,l) = \Big[\sum_{r=1}^{R} (\delta_{j(r)} - \delta_{l(r)})^2\Big]^{\frac{1}{2}}.$$

For bootstrap replicate weights, $d(j,l) = 0$, $E_*[d^2(j,l)] = 2n_hR/(n_h - 1)$ *and* $E_*[d^2(j,l)] = 2R$, *if sample elements j and l are from the same PSU, different PSUs within the same stratum, and different strata, respectively, where E_* denotes expectation under the resampling. In addition,* $V_*[d(j,l)^2] = 10n_h^2(n_h - 2)R(n_h - 1)^{-3}$, *if sample elements j and l are from different PSUs within the same stratum, and*

$$V_*[d(j,l)^2] = \left[20 - \frac{n_h(5n_h - 6)}{(n_h - 1)^2} - \frac{n_{h'}(5n_{h'} - 6)}{(n_{h'} - 1)^2}\right] R,$$

if sample elements j and l are from different strata. Here V_* denotes variance under the resampling.

Proof. See Appendix in online Supplementary Document.

Lemma 1 demonstrates that, for the jackknife and FBRR, if we treat the rows in Table 2 as points in an R dimensional space, the m rows will shrink to replicates of n distinct points in that space provided no weight adjustments have been made.

If some noise or perturbation is added to the original replicate weights, either to attempt to limit an attacker's ability to reconstruct the PSU and/or stratum identifiers or due to nonresponse and/or post-stratification adjustments redone for each replicate, it is very likely that we will still observe n distinguishable clusters in the R dimensional space as the magnitude of the perturbations should be small relative to that of the distance between any two originally distinct points; otherwise, intuitively, the magnitude of the added variability due to these (possibly) random perturbations will be comparable to the variability measured by the replication method and make its usefulness limited.

Lemma 2 suggests this will also be true for the bootstrap. To see this, note that $E_*[d^2(j,l)]$ is about $2R$ if j and l are from different PSUs and 0 if they are from the same PSU, while twice the standard deviation is about $2\sqrt{10R}$. Thus, the clusters associated with each PSU will be well-separated.

With the idea of clustering sampled units in mind, we need only find a clustering algorithm, preferably easy to access and use, and test whether or not the algorithm can identify the PSU membership with high accuracy. A quick search in the common packages R and SAS found function "hclust" and procedure "Proc FASTCLUS", respectively. Both accept multivariate data and form cluster trees. Though we found both to work well, the SAS procedure is much faster and handles very large data sets easily, so we restrict further discussion to "Proc FASTCLUS". We did not try anything more sophisticated to make the point that even someone with rudimentary skills and tools could do this.

We used a set of publicly released NHANES data with 42 sets of Fay BRR replicate weights. To investigate the impact of adding noise to the replicate weights, possibly for the purpose of limiting an attacker's ability to obtain PSU identifiers, we introduce a random noise $\varepsilon_{j(r)}$ to the replicate weights denoting the perturbed replicate weights as

$$w^*_{j(r)} = w_{j(r)}(1+\varepsilon_{j(r)}) = \delta_{j(r)}(1+\varepsilon_{j(r)})w_j = \delta^*_{j(r)}w_j, \tag{2.2}$$

where $\delta_{j(r)} = w_{j(r)}/w_j$ are the elements in Table 2, $\delta^*_{j(r)}$ the perturbed ones, and the $\varepsilon_{j(r)}$ are independent and identically distributed $U(-\Delta,\Delta)$. For values $\Delta = 0.1, 0.2, 0.3, 0.4, 0.5$, we applied "Proc FASTCLUS" (simple SAS code given in the Appendix in the online Supplementary Document) as described above to obtain the PSU membership. If we assumed the number of PSUs to be known, which is often so, in all cases the method correctly assigned units to PSU. If we set the maximum number of clusters to be larger than the true number of clusters, in all cases the results were a partition of the true set of PSUs. That is, "Proc FASTCLUS" yielded more clusters than the truth, but only by splitting PSUs.

To illustrate that the bootstrap or the M-bootstrap method provided even less protection, we designed the following numerical study: (1) create 100 sets of bootstrap weights, each of which was the average of 20, precisely as was done in the Yung (1997) paper; (2) apply the method using only 2,000 sets of the 9,965 replicate weights and only 2, 3, 4, and 5 of the replicates. It turns out that the error rate for assigning units to original PSUs was 2.5% using 2 replicates and 0 using 3 or more replicates, respectively. We repeated the simulation without averaging weights, that is using the ordinary bootstrap method with $m_h^* = n_h - 1$. The error rates were 47.5%, 28%, 5.5% and 1.5% using 2 to 5 replicates, respectively. The clustering algorithm performs very well in terms of reconstructing original PSUs in both cases, even if only a few sets of replicate weights are used.

In summary, it is evident that the replicate weights, no matter how they are created, with or without weight adjustments, can be used to reconstruct the original PSU identifiers quite easily, even for an unsophisticated user. In addition, randomly perturbing the replicate weights provides little protection. Note that, though it is true that the stratum identifiers are harder to reconstruct when little is known about the replication method, confidentiality of stratum indicators may be less important in many cases. For example, for NHANES the PSUs are counties or metropolitan areas. Thus PSU level census data is available and could be used to identify a sampled PSU.

3. Proposed Swapping Algorithm

3.1. Sequential swapping approach

The idea of this section is to swap the PSU identifiers of ultimate units (one could alternately swap PSU identifiers for second stage clusters (SSUs) of units in the case of multi-stages), before constructing the replicate weights or to create pseudo-PSUs for the purpose of variance estimation. We sometimes refer to this as swapping units between PSUs as the two are essentially equivalent. This should be done such that the variance estimates of key characteristics are disturbed as little as possible. Ideally, this should be done via some automatic fast algorithm. Such a swapping algorithm should meet the following criteria:

1. Since one of our major goals is to hide the original PSU identifiers from the end users, a considerable portion of units should be swapped from each original PSU, making it unidentifiable for any cluster analysis of replicate weight patterns. Furthermore, in any formed pseudo-PSU, the number of units from any one original PSU should not be inordinately large.
2. Another goal is to limit the resulting change in variance estimators.

Dohrmann et al. (2006) use a two-stage approach, where in the first stage units from different PSUs are paired together, and at the second stage a user-specified proportion of these paired units are swapped between PSUs. We delay

discussion of the details of their approach and our suggested improvements to make it more automatic to the next section, where we will compare its performance to our proposed algorithm.

Instead we propose a sequential swapping approach. The idea is to avoid swapping a proportion of matched pairs of units at the same time. Instead we establish a rule to determine the best single pair of units for swapping under some optimality criterion at the current step, swap them, and then repeat until enough units have been swapped. This is more in the spirit of the algorithms used for grouping of strata for combined-strata variance estimation in Lu, Brick and Sitter (2006), which was adapted from scheduling theory. The key question is, how to establish a good rule that only swaps units satisfying pre-determined requirements.

Assume we have defined an appropriate distance measure that reflects our preferences and requirements (see next section for discussion). That is, the smaller the distance between two units, the more likely we are to swap them. We first rank all $\binom{n}{2}$ possible pairs of units in ascending order of distance. After so ordering, we need only select the eligible pair of units with smallest distance at the current step and swap them.

Denote $u_{hi} = \lfloor \alpha * n_{hi} \rfloor + 1$ as the minimum number of units to be swapped from PSU hi, and $v_{hi} = \lfloor \beta * u_{hi} \rfloor$ as the maximum number of units to be swapped from PSU hi to any particular other PSU, where $\lfloor \cdot \rfloor$ denotes the largest integer less than or equal to, and β is a tuning parameter which prevents the swapping rate between any two PSUs from being inordinately large. Denote the set of all possible $\binom{n}{2}$ pairs of units as $\mathcal{A} = \{(j,l) : (h_j i_j k_j), (h_l i_l k_l) \in s\}$.

Proposed Sequential Algorithm

Step 1. Preparation

(a) Compute the distance for all pairs from $\mathcal{A}$ and order* them as: $d_1 \leq d_2 \leq \cdots \leq d_{\binom{n}{2}}$, where d_k is associated with $(j_k, l_k) \in \mathcal{A}$, $k = 1, \cdots, \binom{n}{2}$.

(b) Compute u_{hi}, v_{hi} for all PSUs. Let $U_{hi} = 0$ for all (hi); let $V_{hi}(h'i') = v_{hi}$ for all (hi) and $(h'i')$; let $\mathcal{A}' = \emptyset$ and $k = 1$.

Step 2. Swapping

(a) Assume $j_k \in (h_0 i_0)$ and $l_k \in (h_1 i_1)$. If $j_k \in \mathcal{A}'$, $l_k \in \mathcal{A}'$, $V_{h_0 i_0}(h_1 i_1) = 0$ or $V_{h_1 i_1}(h_0 i_0) = 0$, go to **Step 2(c)**.

(b) Let $U_{h_0 i_0} = U_{h_0 i_0} + 1$, $U_{h_1 i_1} = U_{h_1 i_1} + 1$, $V_{h_0 i_0}(h_1 i_1) = V_{h_0 i_0}(h_1 i_1) - 1$, $V_{h_1 i_1}(h_0 i_0) = V_{h_1 i_1}(h_0 i_0) - 1$ and $\mathcal{A}' = \mathcal{A}' \cup \{j_k, l_k\}$.

(c) Let $k = k + 1$.

Step 3. If $k = \binom{n}{2}$, or $U_{hi} \geq u_{hi}$ for all (hi), **stop**; otherwise, go to **Step 2**.

*We use a fast Fortran sorting algorithm available at http://www.fortran-2000.com/rank/index.html

The proposed algorithm is simple and very fast because, after the initial sorting there is very little computation left during the examining and swapping step. The only complexity is in tracking how many units have been swapped from each PSU and how many of these have been swapped to any particular other PSU. These are monitored in the algorithm by U_{hi} and $V_{hi}(h'i')$, respectively. The algorithm's simplicity also makes it very flexible to accommodate different constraints or requirements; for example, to treat high-disclosure-risk PSUs differently from low-disclosure-risk PSUs (discussed in the next section). These properties are preferable in practice because a data swapping procedure takes a great deal of time and effort, and needs to be repeated many times under different settings in order to obtain a satisfactory final swapping result, since typically only some of the characteristics are used for matching, and one must examine the resulting performance of variance estimators on all characteristics. In Section 4, we use a small simulation study to demonstrate these properties.

3.2. Distance measure

Both the proposed sequential swapping algorithm and the match-and-swap approach described in the next section require a distance measure to determine which units to swap or match, respectively. One could merely adopt the same type of approach as is typically used when masking the released data itself through swapping of record elements (see Takemura (2002)). That is, merely use a Euclidean distance measure for continuous data and some form of Hamming distance for categorical variables. For example, let the distance between units i_1 and i_2 be $d(i_1, i_2) = 0$ or 1 if they belong to the same category or not, for categorical variables, and $d(i_1, i_2) = |y_{i_1} - y_{i_2}|/\text{range}(y_i)$, for continuous variables. This rescales the measure to be in $[0, 1]$ for all variables. Different importance of variables can be handled by multiplicative weights.

We do consider this approach in the next section when evaluating performance (denoted D3). However, in this context of swapping PSU identifiers specifically for the purpose of then creating replicate weights for variance estimation, one can more directly rationalize a distance measure that captures the explicit desire to limit the impact on resulting variance estimates. To see this, first consider the linear estimator $\hat{\boldsymbol{Y}}$. All of the replication-based variance estimators given in Section 2 reduce to (approximately for the bootstrap)

$$v(\hat{\boldsymbol{Y}}) = \sum_{h=1}^{H} \frac{1}{n_h(n_h-1)} \sum_{i=1}^{n_h} \boldsymbol{y}_{hi} - \bar{\boldsymbol{y}}_h)(\boldsymbol{y}_{hi} - \bar{\boldsymbol{y}}_h)^T, \tag{3.1}$$

where $\boldsymbol{y}_{hi} = \sum_{k=1}^{n_{hi}} w_{hik}\boldsymbol{y}_{hik}$ and $\bar{\boldsymbol{y}}_h = \sum_{i=1}^{n_h} \boldsymbol{y}_{hi}/n_h$.

The impact of having swapped the PSU identifiers of units in some set $\{(hik) \in s_0 \subset s\}$ with the PSU identifiers of units $\{(hik) \in s_1 \subset s \cap s_0^c\}$ will be

$$v'(\hat{\boldsymbol{Y}}) = v(\hat{\boldsymbol{Y}}) + \boldsymbol{\Delta}_{01},$$

where v' is the variance estimator applied after swapping and Δ_{01} is the impact of the swapping given in Lemma 3.

Lemma 3. *Let* $\mathcal{A}_{01} = \{(j,l) : (h_j i_j k_j) \in s_0$ *swapped PSU identifiers with* $(h_l i_l k_l) \in s_1\}$ *and let* $\boldsymbol{\Delta}_{jl} = w_{h_j i_j k_j} \boldsymbol{y}_{h_j i_j k_j} - w_{h_l i_l k_l} \boldsymbol{y}_{h_l i_l k_l}$ *for ordered pair* $(j,l) \in \mathcal{A}_{01}$. *Then*

$$\Delta_{01} = \sum_{(j,l)\in\mathcal{A}_{01}} \left\{ \left(\frac{n_{h_j}-1}{n_{h_j}} + \frac{n_{h_l}-1}{n_{h_l}} \right) \boldsymbol{\Delta}_{jl} \boldsymbol{\Delta}_{jl}^T + \left(\frac{\boldsymbol{y}_{h_j i_j} - \bar{\boldsymbol{y}}_{h_j}}{n_{h_j}} - \frac{\boldsymbol{y}_{h_l i_l} - \bar{\boldsymbol{y}}_{h_l}}{n_{h_l}} \right) \boldsymbol{\Delta}_{jl}^T \right.$$
$$\left. + \boldsymbol{\Delta}_{jl} \left(\frac{\boldsymbol{y}_{h_j i_j} - \bar{\boldsymbol{y}}_{h_j}}{n_{h_j}} - \frac{\boldsymbol{y}_{h_l i_l} - \bar{\boldsymbol{y}}_{h_l}}{n_{h_l}} \right)^T \right\},$$

where the sum is over pairs (j,l) *so that each pair occurs only once in the sum.*

Proof. See Appendix in online Supplementary Document.

Lemma 3 emphasizes the importance of the design weights. One simple way to incorporate the weights is to apply the usual distance measures to $w_{hik}\boldsymbol{y}_{hik}$ rather than $\boldsymbol{y}_{hik}$ (denoted D1), that is, try to make all of the Δ_{jl} small. This should better ensure that the impact of the swapping on the variance estimation for the variables under consideration will be as small as possible. Another is to merely include the weights as an additional variable in the swapping (denoted D2), that is, make the distance measure between weights small and between y's small. D2 has the advantage that by making the weight change as little as possible should reduce the impact on variables not considered in the swapping. These are also considered in the next section.

Whichever of these ideas is adopted for determining a measure of distance, one must in addition alter or penalize the measure to take into account the desires represented in Criterion 1 of Section 3.1; that is, to ensure that units not be swapped within PSU (and to a lesser extent, strata). A simple way to do this is to add a penalty. Suppose $\gamma_1 + \gamma_2$ is greater than the largest distance between any two units. Then alter the distance measure as

$$d^*(i,j) = d(i,j) + \gamma_1 I_{[i,j\in \text{ same strata}]} + \gamma_2 I_{[i,j\in \text{ same PSU}]}. \tag{3.2}$$

Since $I_{[i,j\in \text{ same PSU}]} = 1$ implies $I_{[i,j\in \text{ same strata}]} = 1$, this penalizes swapping within PSU more than within stratum.

In some applications, there are PSUs that are of greater concern for disclosure risk. This can happen if the size of the PSU is small and/or has a distinct

demographic makeup. In such cases, it would be preferable to add a term to the distance measure that encourages swapping between low disclosure-risk PSUs and high disclosure-risk PSUs. For example, let $\delta_i = 1$ or -1 if the i-th PSU is high-risk or low-risk, respectively, and add term $\gamma_3(\delta_i\delta_j + 1)$. This adds a penalty of $2\gamma_3$ to the distance if PSUs i and j are both low-risk or both high-risk.

4. Evaluation of Performance

Before considering the speed and performance of the proposed algorithm with the various possible distance measures, we describe an enhanced version of the match-and-swap approach in Dohrmann et al. (2006) so that we can then include it in the comparisons.

4.1. Match-and-swap approach

One approach to swapping PSUs of units is to adapt methods for local record swapping which have been proposed for the purpose of disclosure control on the micro data set itself (Takemura (2002)). Then elements of the so paired records are swapped. To adapt this to our problem, add the PSU identifier as one of the components of the record. Then adjust the distance measure to a form such as (3.2) so that if two records are from the same PSU (stratum), the measure of distance becomes large. This prohibits records from the same PSU from being paired together. Apply the pairing algorithm, and once pairs have been formed, choose $\alpha\%$ of the pairs and switch their PSU identifier (see Dohrmann et al. (2006) for discussion). The idea being that the matching will help ensure Criterion 2 of Section 3.1. One possible matching algorithm is the publicly available implementation of a version of Edmonds' (1965) algorithm called WMATCH (see Gabow (1973)).

With the *matching* obtained, we propose a linear programming approach to choose which matched pairs to swap, aiming to balance the proportion of switched pairs of units over all PSUs. Let n_j be the number of units in the j-th PSU, $j = 1, \ldots, m$, and $n_0 = \sum_j n_j/2$ be the number of matched pairs. We then let $(a_{j1}, a_{j2}), j = 1, \ldots, n_0$, denote the matched pairs, (p_{j1}, p_{j2}) be their PSU identifiers and $d_1, \ldots, d_{n_0}$ their corresponding distance measures. Then solve the linear programming (LP) problem,

$$\min_{x_k \in \{0,1\}} d_1 \cdot x_1 + \cdots + d_{n_0} \cdot x_{n_0} \quad s.t. \quad \sum_{k \in P_j} x_k \geq \alpha \cdot n_j, \quad j = 1, \cdots, n_0. \tag{4.1}$$

The optimization procedure will be accomplished through indicator variables x_k, $k = 1, \ldots, n_0$, which determine whether the matched pairs of units are being swapped or not. It is easy to recognize that any feasible solution will satisfy the

requirement of $\alpha\%$ swapping for all PSUs and the optimal solution will further minimize the overall distance of switched pairs of units. This does not entirely satisfy Criterion 1 as there is no real guarantee of avoiding a large number of swaps between pairs of PSUs and thus this would have to be evaluated after the process is complete.

The match-and-swap strategy does encounter some difficulties. It may be difficult or impossible to match all units (termed complete matching) due to the extreme computational time when the number of units is large. Takemura (2002) suggests considering only a unit's K nearest neighbors as candidates for matching. This decreases computation time, but may not yield a complete matching. The free source code (WMATCH) we obtained did achieve complete matching in the cases considered in the next section. For cases where this was not possible and further discussion, see Lu (2004).

4.2. Application to NHANES

Because the proposed approaches are motivated by survey problems, we chose to apply the proposed algorithms and evaluate their performance using the NHANES survey, where the problems originally occurred. However, this involves some sensitive information. Thus, we apply the proposed algorithms to the NHANES 2003-2004 Sample Person Demographics and Examination Files (see NHANES links on http://www.cdc.gov/nchs/), currently released for public use, to compare the speed, similarity of swapped units and flexibility for both the sequential swapping and match-and-swap algorithms. To do so we pretend the pseudo-PSU and stratum identifiers given there are the true PSU and stratum identifiers, and then apply the various strategies to mask these. One should note that this could, in principle, bias the results somewhat.

We choose two groups of characteristics for our simulation. The first group of four demographic and five medical examination variables given in Table 3 are used to determine the distance of any paired records. The column headings correspond to those on the above website. The variables were chosen to be correlated with a broad range of health and nutrition variables. The second group of 26 laboratory variables and 2 medical examination variables given in Table 4 are used for evaluating the performance of the proposed algorithms in terms of variance estimation of variables not used to do the swapping (note laboratory variables are medical examination variables which had to be processed at a laboratory and are thus filed separately). The 2003-2004 full data set includes 10,122 individual records, each of which is associated with a PSU identifier numbered from 1 to 30. However, we use only the 6,217 records with no missing values at the swapping stage. Our goal is to apply the proposed algorithms for swapping the PSU identifiers for a certain percentage of individual records without noticeably changing the resulting variance estimators of all first and second group variables.

Table 3. NHANES variables used for swapping.

Item #	Data File	Item ID	Label
		Demographic Variables	
4	DEMO_C	RIAGENDR	Gender-Adjudicated
5	DEMO_C	RIDAGEYR	Age at Screening Adjudicated-Recode
8	DEMO_C	RIDRETH1	Race/Ethnicity-Recode
13	DEMO_C	INDFMPIR	CPS Family PIR
		Examination Variables	
378	BMX_C	BMXWT	Weight (kg)
384	BMX_C	BMXHT	Standing Height (cm)
386	BMX_C	BMXBMI	Body Mass Index (kg/m**2)
419	BPX_C	BPXSY1	Systolic: Blood pres (1st rdg) mm Hg
420	BPX_C	BPXDI1	Diastolic: Blood pres (1st rdg) mm Hg

Table 4. NHANES variables used for evaluation of swapping

Item #s	Data File	Component
	Laboratory Variables	
45-48	L16_C	Urinary Albumin and Creatinine
57-76	L25_C	Complete Blood Count
137,140	L40_C	Biochemistry Profile
	Examination Variables	
395,401	BMX_C	Body Measures

We applied the match-and-swap and the proposed sequential swapping algorithm for $\alpha = 10\%$, 20%, 30% and 40%, the required percentage of units to be swapped from any PSU. For the proposed sequential swapping algorithm, we have better control during the swapping process in terms of the source of swapped units in any specific PSU. In addition to α, we can define the quantity β as the upper limit on the percentage of units swapped from any other PSU to the target PSU. By introducing β, we tend to monitor the component for each formed pseudo-PSU such that the swapped units in it are from a variety of original PSUs. This is beneficial for confidentiality concerns. The result for each combination of α and β levels is attainable in our simulation. To measure the performance we use the average percent absolute relative difference (ARD) of the variance estimators before and after swapping, defined as

$$ARD = 100 \cdot q^{-1} \sum_{p=1}^{q} \frac{|v'(\hat{Y}_p) - v(\hat{Y}_p)|}{v(\hat{Y}_p)},$$

where the sum is over the q variables, and v and v' denote the variance estimator in (3.1) before and after swapping, respectively. The $q = 9$ variables in Table

Table 5. *Match-and Swap Approach*: time and ARD for the variables used in swapping.

		CPU Time	α%			
Dist.	K	in seconds	10%	20%	30%	40%
D1	5	481	0.188	0.293	0.763	1.543
	10	567	0.222	0.363	1.027	1.564
	20	668	0.173	0.227	1.105	1.664
	40	949	0.147	0.281	0.762	1.725
D2	5	290	0.288	0.157	0.369	0.288
	10	347	0.519	0.439	0.307	0.397
	20	447	0.399	0.485	0.313	0.581
	40	682	0.413	0.475	0.322	0.658
D3	5	285	1.156	1.105	1.875	2.930
	10	338	1.741	2.906	3.378	4.781
	20	448	2.191	2.739	1.721	1.317
	40	696	1.943	2.583	1.554	1.175

3 were used for the swapping with the 3 distance measures discussed in Section 3: D1) weights incorporated by applying the distance measure to $w_{hik}\boldsymbol{y}_{hik}$; D2) incorporating the weights by including them as an extra variable; and D3) no weights.

Table 5 gives the time in seconds (on a Dell Notebook D810 with a 2GHz processor and 1GB of RAM) and ARD over the variables used in the swapping for the various α using the match-and-swap approach with various values of K. Table 6 gives the ARD for the proposed sequential swapping algorithm using the same α's and various β's. As one can see, using D1 yields better results than using D2, which in turn yields better results than using D3. Also, the proposed sequential algorithm performs extremely well and much better than the match-and-swap approach using D1, while there is no clear winner using D2 or D3. However, one must remember that we introduce another control factor β in the proposed sequential algorithm to better limit disclosure. In addition, in all cases the sequential algorithm took between 20 and 36 seconds of CPU time, and thus is much faster than using a match-and-swap approach.

Table 7 and 8 give similar results for the $q = 28$ variables not used to determine the swapping. We also include random swapping of units for comparison. The random swapping was repeated 1,000 times and averaged. As one can see, the impact on these variables is also reasonably small.

Table 6. *Sequential swapping approach*: ARD for the variables used in swapping.

Dist.	$\beta\backslash\alpha$	10%	20%	30%	40%
D1	10%	0.052	0.144	0.359	0.468
	20%	0.055	0.172	0.284	0.410
	30%	0.047	0.173	0.288	0.435
	40%	0.049	0.173	0.288	0.435
D2	10%	0.406	0.413	0.355	0.665
	20%	0.408	0.384	0.474	0.823
	30%	0.408	0.384	0.474	0.823
	40%	0.408	0.384	0.474	0.823
D3	10%	1.560	2.938	2.170	1.145
	20%	1.289	2.843	2.183	1.030
	30%	1.289	2.843	2.183	1.030
	40%	1.289	2.843	2.183	1.030

Table 7. *Match-and Swap Approach*: ARD for the variables not used in swapping.

		α%			
Dist.	K	10%	20%	30%	40%
D1	5	3.732	6.530	8.491	9.136
	10	1.848	4.945	6.160	7.113
	20	4.199	5.472	6.028	8.276
	40	3.930	4.317	6.018	8.503
D2	5	1.558	3.056	3.766	3.285
	10	1.769	3.734	6.101	5.137
	20	1.707	3.725	6.093	5.451
	40	1.718	3.725	6.164	5.504
D3	5	2.954	5.720	8.092	8.781
	10	3.546	6.860	9.962	11.826
	20	4.514	7.282	9.352	9.934
	40	4.260	7.135	9.262	9.871
Random Swap		15.72	29.60	41.48	51.34

5. Concluding Remarks

After demonstrating the risk of disclosing PSU and stratum identifiers through the replicate weights in publicly released survey data, we propose a fast sequential swapping algorithm to exchange PSU identifiers between records before constructing replicate weights. In this way, the true PSU identifiers can be masked without major impact on resulting variance estimators. Application to the Na-

Table 8. *Sequential swapping approach*: ARD for the variables not used in swapping.

Dist.	$\beta\backslash\alpha$	10%	20%	30%	40%
D1	10%	0.42	1.72	2.34	4.07
	20%	0.44	1.78	2.26	4.05
	30%	0.38	1.77	2.23	4.01
	40%	0.38	1.77	2.23	4.01
D2	10%	2.59	3.54	7.59	4.85
	20%	2.40	3.50	7.70	4.64
	30%	2.40	3.50	7.70	4.64
	40%	2.40	3.50	7.70	4.64
D3	10%	4.06	9.11	9.59	10.93
	20%	4.17	8.88	9.66	11.09
	30%	4.17	8.88	9.66	11.09
	40%	4.17	8.88	9.66	11.09
Random Swap		15.72	29.60	41.48	51.34

tional Health and Nutrition Examination Surveys demonstrates the potential of the approach. The speed of the algorithm, even when handling thousands of records, allows the analyst within the releasing agency to apply and evaluate the method many times in exploring which variables to use for swapping, and the impact on variance estimation and on disclosure risk.

One cautionary note: if all of the weights within a PSU are the same (SRS) but differ between PSUs, swapping PSU identifiers will not help. This does give some indication of the overall difficulty of this problem.

Acknowledgement

Partially funded by a grant from the Natural Science and Engineering Research Council of Canada. We thank an associate editor and two referees for their encouragement and helpful comments.

References

Booth, J. G., Butler, R. W. and Hall, P. (1994). Bootstrap methods for finite populations. *J. Amer. Statist. Assoc.* **89**, 1282-1289.

Dippo, C. S., Fay, R. E. and Morganstein, D. H. (1984). Computing variances from complex samples with replicate weights. In the *Proc. Amer. Statist. Assoc., Sec. Surv. Res. Meth.*, 489-494.

Dohrmann, S., Lu, W. W., Park, I., Sitter, R. R. and Curtin, L. R. (2006). Variance estimation to protect confidentiality in the National Health and Nutrition Examination Surveys. Submitted to the *J. Off. Statist.*

Edmonds, J. (1965). Maximum matching and a polyhedron with (0,1) vertices. *J. Res. Nat. Bur. Stand.* **B69**, 125-130.

Gabow, H. N. (1973). Implementation of algorithms for maximum matching on non-bipartite graphs. Unpublished Ph.D. thesis, Stanford University, Department of Computer Science.

Gurney, M. and Jewett, R.S. (1975). Constructing orthogonal replications for variance estimation. *J. Amer. Statist. Assoc.* **70**, 819-21.

Judkins, D. R. (1990). Fay's method for variance estimation. *J. Off. Statist.* **6**, 223-239.

Krewski, D. and Rao, J. N. K. (1981). Inference from stratified samples: properties of the linearization, jackknife and balanced repeated replication methods. *Ann. Statist.* **9**, 1010-1019.

Lambert, D. (1993). Measures of disclosure risk and harm. *J. Off. Statist.* **9**, 313-344.

Lu, W. W. (2004). Confidentiality and variance estimation in complex surveys. Unpublished Ph.D. thesis, Simon Fraser University, Department of Statistics and Actuarial Science.

Lu, W. W., Brick, M. and Sitter, R. R. (2006). Algorithms for constructing combined strata variance estimators. *J. Amer. Statist. Assoc.* **101**, 1680-1692.

McCarthy, P. J. (1966). *Replication: An Approach to the Analysis of Data From Complex Surveys*, Vital and Health Statistics Ser. 2, No. 14, Washington, DC: U.S. Gov. Printing Office.

Rao, J. N. K. and Wu, C. F. J. (1985). Inference from stratified samples: second-order analysis of three methods for nonlinear statistics. *J. Amer. Statist. Assoc.* **80**, 620-630.

Rao, J. N. K. and Wu, C. F. J. (1988). Resampling inference with complex survey data. *J. Amer. Statist. Assoc.* **83**, 231-241.

Rao, J. N. K. and Wu, C. F. J. and Yue, K. (1992). Some recent work on resampling methods for complex surveys. *Surv. Method.* **18**, 209-217.

Shao, J. and Tu, D. (1995). *The Jackknife and Bootstrap*. Springer-Verlag, New York.

Sitter, R. R. (1992a). A resampling procedure for complex survey data. *J. Amer. Statist. Assoc.* **87**, 755-765.

Sitter, R. R. (1992b). Comparing three bootstrap methods for survey data. *Can. J. Statist.* **20**, 135-154.

Sitter, R. R. (1993). Balanced repeated replications based on orthogonal multi-arrays. *Biometrika* **80**, 211-221.

Skinner, C. J. and Elliot, M. J. (2002). A measure of disclosure risk for microdata. *J. Roy. Statist. Soc. Ser. B* **64**, 855-867.

Skinner, C. J. and Carter, R. G. (2003). Estimation of a measure of disclosure risk for survey microdata under unequal probability sampling. *Surv. Method.* **29**, 177-180.

Takemura, A. (2002). Local recoding and record swapping by maximum weight matching for disclosure control of microdata sets. *J. Off. Statist.* **18**, 275-289.

Wu, C. F. J. (1991). Balanced repeated replications based on mixed orthogonal arrays. *Biometrika* **78**, 181-188.

Yung, W. (1997). Variance estimation for public use files under confidentiality constraints. In the *Proc. Amer. Statist. Assoc. Surv. Res. Meth. Sec.*, 434-439.

Department of Mathematics and Statistics, Acadia University, Wolfville, NS, B4P 2R6, Canada.
E-mail: wilson.lu@acadiau.ca

Department of Statistics and Actuarial Science, Simon Fraser University, Burnaby, BC V5A-1S6, Canada.
E-mail: sitter@stat.sfu.ca

(Received June 2006; accepted May 2007)