Catalogue no. 11-522-XIE

Statistics Canada International Symposium Series - Proceedings

Symposium 2006 : Methodological Issues in Measuring Population Health

2006

Statistics Canada Statistique Canada

Canada

Proceedings of Statistics Canada Symposium 2006
Methodological Issues in Measuring Population Health

Disclosure Risk and Variance Estimation

W. Wilson Lu and Randy R. Sitter[1]

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 and for replication methods as resampling is typically done on first-stage sampling units within strata. 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 resampling 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: 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. One very 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 their search for respondent identity, but it can also make it difficult for direct variance estimation as stratum and PSU identifiers are typically needed to obtain asymptotically unbiased variance estimators.

It might be thought that by instead 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) notes that this may not be the case. He proposes a repeated bootstrap in an attempt to create a set of 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.

Another method that is commonly used to mask the data itself 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 so as to limit the impact on resulting variance estimates for the characteristics measured in the survey. This idea is 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) is presented in Dohrmann et al. (2006). The basic idea is to match second-stage clusters on some demographic variables by using a record-linkage algorithm and then sort through the matches to decide on ones for which PSU identifiers would be swapped for all ultimate units in the second-stage cluster.

[1]W. Wilson Lu, Department of Mathematics and Statistics, Acadia University, 12 University Ave., Wolfville, NS, Canada, B4P 2R6; Randy R. Sitter, Department of Statistics and Actuarial Science, Simon Fraser University, 8888 University Dr., Burnaby, BC, Canada, V5A 1S6.

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 replicate weights.

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 $\hat{Y}_{hi}$ 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 $\hat{Y}_h = \sum_{i=1}^{n_h} \hat{Y}_{hi} / (n_h p_{hi})$ is a linear unbiased estimator of the vector of stratum totals Y_h. A linear unbiased estimator of the total $Y = \sum_h Y_h$ is then given by $\hat{Y} = \sum_h \hat{Y}_h$. This can be written as $\hat{Y} = \sum_{(hik)\in s} w_{hik} y_{hik}$, where s is the total sample of elements, and w_{hik} and $y_{hik} = (y_{1hik}, y_{2hik}, \ldots, y_{phik})'$ 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, i.e. $\hat{\theta} = g(\hat{Y})$. 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 $I_{[\bullet]}$ is the indicator function and the *p*-th sample quantiles can be estimated by $\hat{F}^{-1}(p)$, and $\hat{F}^{-1}$ is the inverse function of $\hat{F}$.

2.2 Replicate Weights and Stratum/PSU Identifiers

In this section, we suppress the triple index and write $\hat{Y} = \sum_{j\in s} w_j y_j$, 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.

Table 1. The Matrix Representation of Design and Replicate Weights

SAMPLE	*CHARACTERISTICS*	*DESIGN WEIGHTS*	*REPLICATE WEIGHTS*
1	y_1	w_1	$w_{1(1)}\ w_{1(2)}\ \cdots\ w_{1(R)}$
2	y_2	w_2	$w_{2(1)} w_{2(2)}\ \cdots\ w_{2(R)}$
…	…	…	… … … … …
m	y_m	w_m	$w_{m(1)} w_{m(2)}\ \cdots w_{m(R)}$

All of the usual replication-based variance estimators (jackknife, bootstrap, Fay's BRR) can be rewritten as selecting *R* subsets of the full sample $y_1, y_2, \ldots, y_m$, where *m* is the total number of ultimate units in the sample, according to some re-sampling mechanism to form replicate estimates denoted as $\hat{\theta}_{(1)}, \ldots, \hat{\theta}_{(R)}$. 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 c_r are constants specific to the replication method.

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 $y_1, y_2, \ldots, y_m$ and $w_1, w_2, \ldots, w_m$ to get $\hat{\theta}_{(1)}, \ldots, \hat{\theta}_{(R)}$ by applying it to $y_1, y_2, \ldots, y_m$ and $w_{1(r)}, w_{2(r)}, \ldots, w_{m(r)}$ 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 $j = 1, \ldots, m$ and $r = 1, \ldots, R$ depicted in Table 2.

We now demonstrate how easy it is to reconstruct PSU identifiers from Table 2 and then apply it to some real 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 $\delta_{j(1)}, \ldots, \delta_{j(R)}$ and define a distance, $d(j,l)$ of any pair of sample elements j and l. It is not difficult to see that if one applies a clustering algorithm to these rows, units from the same PSU will end up in the same cluster.

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 multi-variate 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 will 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 $w^*_{j(r)} = w_{j(r)}(1 + \varepsilon_{j(r)}) = \delta_{j(r)}(1 + \varepsilon_{j(r)})w_j = \delta^*_{j(r)} w_j$, where $\delta_{j(r)} = w_{j(r)} / w_j$ are the elements in Table 2, $\delta^*_{j(r)}$ the perturbed ones and $\varepsilon_{j(r)}$ are independent and identically distributed $U(-\delta, \delta)$. For various values of δ, we applied “Proc FASTCLUS” (simple SAS code available from the authors) 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 of Yung (1997) provide 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) then apply the method using only 2000 sets of the 9965 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 is 2.5% using 2 replicates and 0 using 3 or more replicates, respectively. We repeat the simulation without averaging weights, that is, using the ordinary bootstrap method with $m^*_h = n_h - 1$. The error rates are 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. It is also the case that randomly perturbing the replicate weights provides minimal protection.

3. Proposed Swapping Algorithm

3.1 Sequential Swapping Approach

The idea of this section is to swap the PSU identifiers of ultimate units, before constructing the replicate weights or to create pseudo-PSUs for the purpose of variance estimation. We will sometimes refer to this as swapping units between PSUs as the two are essentially equivalent. This should be done so as to disturb the variance estimates of key characteristics 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. 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. 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.

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 $n(n-1)/2$ 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 $\lfloor \bullet \rfloor$ denotes the largest integer less than or equal 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 $n(n-1)/2$ pairs of units as $A = \{(j,l) : (h_j i_j k_j), (h_l i_l k_l) \in s\}$

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. Its simplicity also makes it very flexible to accommodate different constraints or requirements if needed.

3.2 Distance Measure

This sequential swapping algorithm requires a distance measure to determine which units to swap. One could use, for example, $d(i_1, i_2) = 1$ or 0 if units i_1 and i_2 belong to the same category or not, for categorical variables; and $d(i_1, i_2) = | y_{i_1} - y_{i_2} | / range(y_i)$, 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 D_3). 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 $\hat{Y}$. All of the usual replication-based variance estimators reduce to (approximately for the bootstrap)

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

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

The impact of having swapped the PSU identifiers of units $\{(hik) \in s_0 \subset s\}$ with the PSU identifiers of units $\{(hik) \in s_1 \subset s \ \cap s_0^c\}$ will be $v'(\hat{Y}) = v(\hat{Y}) + \Delta_{01}$, where v' is the variance estimator applied after swapping and Δ_{01} is the impact of the swapping given in Lemma 1.

Lemma 1. Let $\mathrm{A}_{01} = \{(j,l) : (h_j i_j k_j) \in s_0 \text{ swapped cluster identifier with } (h_l i_l k_l) \in s_0\}$ and let $\Delta_{jl} = w_{h_j i_j k_j} y_{h_j i_j k_j} - w_{h_l i_l k_l} y_{h_l i_l k_l}$ for ordered pair $(j,l) \in \mathrm{A}_{01}$. Then

$$\Delta_{01} = \sum_{(j,l)\in \mathrm{A}_{01}} \{ (\frac{n_{h_j}-1}{n_{h_j}} + \frac{n_{h_l}-1}{n_{h_l}}) \Delta_{jl} \Delta_{jl}^T + (\frac{y_{h_j i_j} - \bar{y}_{h_j}}{n_{h_j}} - \frac{y_{h_l i_l} - \bar{y}_{h_l}}{n_{h_l}}) \Delta_{jl}^T + \Delta_{jl} (\frac{y_{h_j i_j} - \bar{y}_{h_j}}{n_{h_j}} - \frac{y_{h_l i_l} - \bar{y}_{h_l}}{n_{h_l}})^T \},$$

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

Lemma 1 emphasizes the importance of the design weights. One simple way to incorporate the weights is to apply the usual distance measures to $w_{hik} y_{hik}$ rather than y_{hik} (denoted D_1), 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 impacted as little as possible. Another is to merely include the weights as an additional variable in the swapping (denoted D_2), that is, make the distance measure between weights small and between *y*'s small. These are also considered in the next section.

Whichever of these ideas are adopted for determining a measure of distance, one must in addition alter or penalize the measure so as 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 follows. Use

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

Since $I_{[i,j \in same\ PSU]} = 1$ implies $I_{[i,j \in same\ stratum]} = 1$, 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

We apply the proposed algorithm and the match-and-swap of Dohrmann et al. (2006) 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.

We choose two groups of characteristics for our simulation. The first group of 4 demographic variables, Gender, Age, Ethnicity and Family income, and 5 medical examination variables, Weight, Standing height, Body mass index, Systolic blood pressure and Diastolic blood pressure are used to determine the distance of any paired records. The variables were chosen to be correlated with a broad range of health and nutrition variables. The second group of 26 laboratory variables, measuring urinary albumin and creatinine, complete blood count and biochemistry profile, and 2 medical examination variables for body measures are used for evaluating the performance of the proposed algorithms in terms of variance estimation of variables not used to do the swapping. The 2003-2004 full dataset 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.

We applied the match-and-swap and the proposed sequential swapping algorithm for α = 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 will be beneficial for confidentiality concerns. To measure the performance we use the average percent absolute relative difference (ARD) of the variance estimators before and after swapping, defined as $ARD = \sum_{p=1}^{q} | v'(\hat{Y}_p) - v(\hat{Y}_p) | / [q v(\hat{Y}_p)]$, where the sum is over the q variables and v and v' denote the variance estimator before and after swapping, respectively. The $q=9$ variables in the first group were used for the swapping with the three distance measures discussed in section 3: D_1) weights incorporated by applying the distance measure to $w_{hik} y_{hik}$; D_2) incorporating the weights by including them as an extra variable; and D_3) no weights.

Table 2. *Match-and-Swap Approach*: ARD for the variables **used** and **not used** in swapping

Dist	*K*	*Variables **used** in swapping*					*Variables **not used** in swapping*			
		CPU Time (seconds)	*α=10%*	*20%*	*30%*	*40%*	*α=10%*	*20%*	*30%*	*40%*
D1	5	481	0.188	0.293	0.763	1.543	3.732	6.530	8.491	9.136
	10	567	0.222	0.363	1.027	1.564	1.848	4.945	6.160	7.113
	20	668	0.173	0.227	1.105	1.664	4.199	5.472	6.028	8.276
	40	949	0.147	0.281	0.762	1.725	3.930	4.317	6.018	8.503
D2	5	290	0.288	0.157	0.369	0.288	1.558	3.056	3.766	3.285
	10	347	0.519	0.439	0.307	0.397	1.769	3.734	6.101	5.137
	20	447	0.399	0.485	0.313	0.581	1.707	3.725	6.093	5.451
	40	682	0.413	0.475	0.322	0.658	1.718	3.725	6.164	5.504
D3	5	285	1.156	1.105	1.875	2.930	2.954	5.720	8.092	8.781
	10	338	1.741	2.906	3.378	4.781	3.546	6.860	9.962	11.83
	20	448	2.191	2.739	1.721	1.317	4.514	7.282	9.352	9.934
	40	696	1.943	2.583	1.554	1.175	4.260	7.135	9.262	9.871

The first part of Table 2 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, while the second part gives the ARD for the $q=28$ variables not used to determine the swapping. As one can see, using D_1 yields better results than using D_2 which in turn yields better results than using D_3.

Table 3. *Sequential Swapping Approach*: ARD for the variables **used** and **not used** in swapping

Dist	*β/α*	*Variables **used** in swapping*				*Variables **not used** in swapping*			
		10%	*20%*	*30%*	*40%*	*10%*	*20%*	*30%*	*40%*
D1	10%	0.052	0.144	0.359	0.468	0.42	1.72	2.34	4.07
	20%	0.055	0.172	0.284	0.410	0.44	1.78	2.26	4.05
	30%	0.047	0.173	0.288	0.435	0.38	1.77	2.23	4.01
	40%	0.049	0.173	0.288	0.435	0.38	1.77	2.23	4.01
D2	10%	0.406	0.413	0.355	0.665	2.59	3.54	7.59	4.85
	20%	0.408	0.384	0.474	0.823	2.40	3.50	7.70	4.64
	30%	0.408	0.384	0.474	0.823	2.40	3.50	7.70	4.64
	40%	0.408	0.384	0.474	0.823	2.40	3.50	7.70	4.64
D3	10%	1.560	2.938	2.170	1.145	4.06	9.11	9.59	10.93
	20%	1.289	2.843	2.183	1.030	4.17	8.88	9.66	11.09
	30%	1.289	2.843	2.183	1.030	4.17	8.88	9.66	11.09
	40%	1.289	2.843	2.183	1.030	4.17	8.88	9.66	11.09

Similarly, Table 3 gives the ARD for the proposed sequential swapping algorithm using the same α's and various β's. Apparently, the proposed sequential algorithm performs extremely well and much better than the match-and-swap approach using D_1, while there is no clear winner using D_2 or D_3. 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.

We also include random swapping of units for the *q=28* variables not used to determine the swapping for comparison. The ARD for α=10%, 20%, 30% and 40% are 15.72, 29.60, 41.48 and 51.34, respectively. The random swapping was repeated 1000 times and averaged. As one can see, the impact on these variables is also reasonably small.

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 National 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.

References

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 *Journal of Official Statistics.*

Lu, W.W., (2004), "Confidentiality and Variance estimation in Complex Surveys", unpublished PhD thesis, Simon Fraser University, Canada.

Yung, W. (1997), "Variance Estimation for Public Use Files Under Confidentiality Constraints", *Proceedings of the Survey Research Methods Section, American Statistical Association*, pp. 434-439.