Working Paper
ENGLISH ONLY

UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE (UNECE)

CONFERENCE OF EUROPEAN STATISTICIANS

EUROPEAN COMMISSION

STATISTICAL OFFICE OF THE EUROPEAN UNION (EUROSTAT)

Joint UNECE/Eurostat work session on statistical data confidentiality
(Ottawa, Canada, 28-30 October 2013)

Topic (i): New methods for protection of tabular data or for other types of results from table and analysis servers

Methodology for the Automatic Confidentialisation of Statistical Outputs from Remote Servers at the Australian Bureau of Statistics

Prepared by Gwenda Thompson, Stephen Broadfoot and Daniel Elazar, Australian Bureau of Statistics, Australia

Methodology for the Automatic Confidentialisation of Statistical Outputs from Remote Servers at the Australian Bureau of Statistics

UNECE Work Session on Statistical Data Confidentiality

Gwenda Thompson, Stephen Broadfoot, Daniel Elazar

Australian Bureau of Statistics,
45 Benjamin Way, BELCONNEN, ACT, 2617, Australia,
gwenda.thompson@abs.gov.au, stephen.broadfoot@abs.gov.au, daniel.elazar@abs.gov.au

Abstract. ABS has recently developed the TableBuilder and DataAnalyser remote server systems with automated confidentiality routines that allow users to build their own custom tables or undertake regression analyses on secured ABS microdata. This paper outlines the statistical methodology behind the perturbation and other protection methods used in these systems. The perturbation routines applied in TableBuilder and DataAnalyser are applied not at the unit record level, as is the case with confidentialised unit record files (CURFs), but at a level of aggregation relevant to the analysis. This results in lower levels of information loss by tailoring the perturbation both to the type of analysis requested and the nature of the underlying data. We firstly overview the functionality within TableBuilder and DataAnalyser, then discuss the range of possible disclosure attacks that remote servers may be susceptible to, and give details of how the perturbation and other confidentiality protections are implemented in each system.

Contents

1 Introduction

ABS has been working on the development of remote servers for a number of years. This has recently culminated in the production releases of TableBuilder for confidentialised tabular output and DataAnalyser for confidentialised data exploration, transformation and regression analysis. A driving force behind the commissioning of this work was the need to deliver on one of the ABS's key strategic objectives, namely the 'informed and increased use of statistics'. Remote servers contribute to delivering infrastructure for real time dissemination of ABS data, reducing the resources required, improving timeliness and growing the business through new statistical products and services.

The ABS currently spends considerable time and resources providing Confidentialised Unit Record Files (CURFs) for users. These require undertaking a set of complex manual confidentialisation procedures and clearance processes to ensure that our legal obligations under the Census and Statistics Act, 1905 are upheld regardless of the type of user or the kind of analysis being undertaken on the CURF. This results in a one-size-fits-all approach that is required to provide a sufficient level of confidentiality protection across a multitude of users and purposes.

ABS, along with other national statistical institutes (NSIs), has built up a high level of expertise and capability in confidentiality procedures for output micro and aggregate data. The Australian experience is that many other government agencies are wanting to make their data available for purposes such as cross agency data integration, but lack the knowledge and expertise in confidentialisation. ABS is taking on a leadership role as an integrating authority for dynamically confidentialised linked data and the deployment of infrastructure for remote servers is integral to achieving this strategic goal.

Of paramount importance for any statistical release, is reducing the risk of disclosure for an individual or business to an acceptable level under the Act. Under the Act, the Australian Statistician is firstly required to publish or disseminate compilations and/or analyses of statistical information collected under this Act, and secondly, ensure that this is done in a manner that is not likely to enable the identification of a particular person or organisation. Remote servers provide the additional benefit of ensuring that confidentiality is protected in an automated and consistent manner. Importantly, they form an important part of a suite of dissemination products that address the differing levels of sophistication and analytical requirements of users.

The move towards remote servers strategically positions the ABS to minimise perceived barriers to accessing ABS data holdings through increased ability for external users to analyse richer microdata from an expanded range of collections. This includes access to associated metadata and machine to machine web services. Users are also becoming more sophisticated in their adoption of the latest technologies and in exploiting the deep statistical content of linked, longitudinal and hierarchical datasets, that generally require more advanced statistical techniques.

Another strong business driver is increasing international collaborations with other NSIs in the confidentialisation of data and use of data management standards. One of the key focuses of our work on remote servers has been the use of DDI/SDMX and machine to machine interfaces such as application programming interfaces (APIs). Some of this work has lead to increased integration and automation with internal data feed systems (microdata and metadata for release).

Currently, the remote servers, and indeed CURFs, have been mostly limited to household survey collections rather than business surveys. This is due to concerns about the higher risks associated with outliers in business data and sensitivities in the business sector concerning the divulgence of business intelligence.

2 Current Data Services

ABS's first foray into remote execution services resulted in the development of the Remote Access Data Laboratory (RADL) which allowed users to remotely submit code in either SAS, SPSS or STATA. The user submitted code is run within the ABS's secure environment, with the output returned to the user. The submitted code and the output are subject to a range of automated or manual checks, with certain input commands not allowed (including graphical displays of data) and some output not released if it does not meet the ABS confidentiality requirements. Access to this service is limited to authorised users for statistical purposes only and the data is made available under Clause 7 of the Statistics Determination 1983 with users being required to sign an undertaking regarding their use of this data.

There is increasing demand from users for flexible access to richer and richer microdata datasets such as more detailed household datasets, linked datasets, longitudinal datasets and business survey datasets. This cannot be easily facilitated in the existing RADL as the automated protections are not 'bullet proof', in the sense that they are not designed to be run on unconfidentialised microdata. Rather, they were designed to facilitate protection of outputs generated from confidentialised datasets (Expanded CURFs). The viability of the existing RADL based approach is also under threat from increasing risk of identification due to increased computing power (both hardware and software) and better managed external datasets.

For more sophisticated users, the ABS Data Laboratory (ABSDL) enables on-site access for approved users, to undertake their own analysis of a specialist CURF which provides a richer source of data. However, this requires the user to travel to an ABS office and the price of a specialist CURF is not insubstantial in order to recover the cost of producing a confidentialised file tailor-made to the user and their analytical requirements.

For users with suitable analysis requirements, specialist data services, such as consultancies, allow their analysis to be undertaken on the unconfidentialised unit record data by ABS officers and then returned in a suitably confidentialised manner. The user may provide their own code or ABS officers may develop code, subject to requirements, on a fee for service basis.

Across this range of data services offered by the ABS, options that give the user access to greater information content and detail come with higher levels of restriction. For example, basic CURFs generally have lower information content but the user has greater flexibility over the computing environments in which the CURF can be installed, subject to prescribed security undertakings. Users of the ABSDL, on the other hand, get access greater information content, however, the analysis must be undertaken only in specified

ABS offices, with clearance of all outputs supervised by a presiding ABS officer.

There still remains a significant business need for microdata solutions for linked microdata and business microdata to maintain the relevance of the ABS. Progress is being made in developing solutions for accommodating linked data files in the ABS remote server, however more research is planned to develop more enhanced solutions that minimise the loss of utility for users.

The ABS continues to develop strategies to improve research user access to unit record data for statistical purposes. This is an important strategic direction, ensuring that the ABS maintains a valued and responsive service in Australia and internationally. TableBuilder and DataAnalyser will deliver online analytical capability for users, while mitigating disclosure risk in real time and complementing the existing suite of access modes. The provision of this infrastructure better positions the ABS to meet the increasing demand from users for flexible access to richer microdata datasets including social and household datasets, hierarchical, synthetic, linked / administrative, and longitudinal datasets. TableBuilder and DataAnalyser are designed to facilitate the exchange of information and analysis by implementing international standards and processes, including DDI and SDMX. This approach provides value for users to search metadata associated with microdata through integration with ABS metadata registries/repositories being developed as part of the ABS future corporate infrastructure, which will facilitate access by international statistical agencies.

3 TableBuilder and DataAnalyser

ABS has now developed the infrastructure for TableBuilder and DataAnalyser, and TableBuilder for survey data has been released in production to external users for over a year. TableBuilder provides a menu driven interface for producing confidentialised tables of count or continuous variables, as well as quantiles. Requested tables are produced and confidentialised on-the-fly, with each cell value estimated using the survey estimation weights. Relative standard errors are calculated and displayed in real time simply by checking a box.

DataAnalyser is a system that allows users to undertake analyses of ABS microdata using a menu driven user interface. DataAnalyser allows users to carry out data transformations and manipulations, basic exploratory data analysis (EDA), summary tables and regressions analysis including, linear (robust), logistic, probit and multinomial. Confidentialised outputs from EDA, summary table or regressions can be either viewed on screen or downloaded to the user's own computer.

The microdata in both systems sit within the ABS, protected by a series of firewalls. As the ABS is in the early phase of bedding down the automated perturbation of TableBuilder and DataAnalyser, a low level of manual confidentialisation is applied to the microdata before it is loaded into the system. This may include removing high risk variables, top coding sensitive variables or applying a categorisation to certain continuous variables. The level of confidentialisation applied to the microdata prior to input to these systems, is far less than that required to produce a CURF. Hence, compared with CURFs, TableBuilder and DataAnalyser outputs have higher utility with substantial cost savings and improvements in timeliness.

Both products allow external users to login remotely over the internet through an interface. Users first have to register to use the system. Once registered, access is controlled by a user registration system that authenticates the user's credentials and ensures that the user is only given access to the microdata that they are registered to use. TableBuilder currently has 16 datasets available, including: Education and Work, 2011, 2012; Characteristics of Recent Migrants, 2010; Disability, Ageing and Carers, 2003, 2009 and specific results form the Australian Health Survey, 2011-2012. The production infrastructure for DataAnalyser has been deployed, however release of datasets into DataAnalyser is currently subject to an approval process.

The key objective for DataAnalyser is to enable analyses to be undertaken remotely in real time on detailed data, in a reasonably flexible manner while ensuring that confidentiality is automatically protected with minimum loss of utility for users. It is simply not possible to achieve this objective for all users, regardless of level of sophistication, without some trade-offs. For this reason, it was decided to develop DataAnalyser for users with a medium level of sophistication. This 'market segment' comprises policy analysts and social and economic researchers who form a core component of our user base.

A menu-driven system is well suited to these users and makes fully automated confidentiality protection achievable for realistic cost. It is much more complicated and costly to automatically parse freely written computer code, written in one of a number of different languages, in order to detect the possibility of a sophisticated attack involving complex manipulations. Naturally there are downsides to a menu-driven system. Firstly, it is more likely to deter sophisticated analysts who use a variety of advanced statistical techniques. Secondly, a menu-driven system can become cumbersome for users who carry out numerous dataset transformations and manipulations in order to prepare the data for analysis. These users may or may not be at the higher levels of sophistication when it comes to the actual statistical analysis.

In the short to medium term, the needs of sophisticated users will be addressed by the provision of other analysis services such as RADL and ABSDL. Once DataAnalyser has bedded down and we have experience with it, we can look towards extending its capability to handle the needs of more sophisticated users. An incremental, agile approach like this also ensures that we have resources into the future to allow TableBuilder and DataAnalyser to evolve in line with users' feedback and unfolding analytical requirements, rather than commit to a costly and complex build that ends up not meeting user needs. ABS is committed to extensive long term user consultation, from which high priority requirements will be identified and deployed in future versions of DataAnalyser.

The development of automated methods and tailored outputs to ensure that respondents are not likely to be identified (to comply with ABS legislation), has been critical in building TableBuilder and DataAnalyser. In particular, this has required the development of more or less generalised perturbation algorithms that can be incorporated into analysis methods whether they be summary tables, statistical regressions or other summary statistics such as quantiles. Automated algorithms mean that the on-going cost and time required for manual checking and clearance is greatly reduced. Likewise, the inevitable inconsistencies in the application of manual checks of RADL and ABSDL output, by ABS officers, is eliminated.

Additionally, the perturbation algorithms ensure that the size of the perturbation (or noise) applied is tailored for each analysis being undertaken. Compared to one-size-fits-all confidentialisation, as in the case of CURFs, the amount of information loss for the user is considerably less. This has been demonstrated by Chipperfield and Lucie (2011), who have shown that input perturbation, where the perturbation is applied at the microdata record level, leads to substantially higher total variances than does output perturbation, where the perturbation is applied at the aggregated level. Thus for regression analysis, perturbation is applied to the score function, rather than the microdata, as it captures the sufficient statistics from which the model parameters are estimated (see Section 13 for more detail). Note that in the case of regression models, the perturbation is not simply applied to the final parameter estimates as this can lead to inconsistent estimators.

4 Statistical Attacks

A range of potential identification attacks have been discussed in the literature, involving either tabular or analysis outputs in the context of remote servers (O'Keefe and Chipperfield, 2013). The context in which an attack or identification attempt is carried out is important to take into account. At one end of the spectrum is the deliberate attack on a targeted record by circumventing confidentiality protections in order to obtain additional information about the individual or business. This may involve linking the record to other data sources with identifying information.

At the other end of the spectrum is spontaneous recognition, where a well intentioned user encounters a record that is sufficiently rare and the user believes that s/he knows the identity of the person or business in question. In between are attacks where the attacker has little prior information and trawls through the unit record file in order to identify a record with a rare combination of characteristics.

Most attacks can be adequately protected by a suitably chosen perturbation scheme, however some require additional levels of protection which are discussed below. For unit record files based on survey data, the risk of identification also depends upon whether the record is a sample unique, and if it is, the number of the units in the population with the same characteristics (see for example, Elamir and Skinner, 2006; Skinner and Schlomo, 2008).

For most attacks, significant amounts of time, knowledge and perseverance are required to effect the attack and this needs to be taken into account when assessing the risk. For a statistical attack to succeed, the attacker needs to have many of the following skills, knowledge and qualities:

1. Reliable knowledge of the characteristics of the target unit that make it rare in the sample and that this unit is likely to have high statistical leverage.

2. For many attacks, even if the attacker has the reliable knowledge of the key characteristics of the target unit, considerable time and perseverance is necessary to effect the attack. If the attacker wishes to recover the microdata values for the entire record of a unit, then this effort would be multiplied by the number of variables in

the record.

3. Sufficient statistical knowledge to be able to choose a model that would have adequately high predictive power to reliably predict the desired characteristic of the target unit.

A brief summary is given in Subsection 4.1 of some of these attacks identified for either TableBuilder or DataAnalyser.

4.1 Tabular Attacks

Averaging
A user makes repeated requests for the same table, or the same cell embedded in different tables, at different times and then averages these to obtain an estimate that has higher precision than the perturbation would permit.
Differencing
Typically undertaken by creating two tables that are different by only one or two units to obtain additional characteristics about the targeted units. This attack is closely related to the scope-coverage attack referred to below.
Sparsity
An attack that is undertaken by requesting a table in which the number of 'small' cells exceeds a given threshold. The definition of 'small' and the threshold value are configured within the business logic of the system.
Scope-Coverage
Scope refers to the logical meaning of a query submitted by a user to the DataAnalyser. Coverage, on the other hand, refers to the set of records resulting from the application of the query to the dataset. A perturbation methodology based on coverage only allows an attacker to change the scope of the query slightly and examine the outputs for changes in order to identify a unit. An implementation of scope-based perturbation that does not fix the perturbation seed or fails to recognise logically equivalent scopes will still be open to averaging attacks.
Comparison of TableBuilder output with the CURF
This involves comparing TableBuilder output with the corresponding publicly released CURF file to find out more details on an individual. This was found by an internal investigation to not be a sufficiently high level of risk.

4.2 Regression Attacks

O'Keefe and Chipperfield (2013) provide an overview of the kinds of attacks possible for fully automated remote analysis systems and identify confidentiality protection measures to mitigate the risk.

Leverage
A model fitted to data containing a high leverage unit (having unusual and rare characteristics) returns a more accurate predicted value which increases the disclosure risks.
Influence

Similar to the leverage attack, but the main difference is that the rare characteristics of the unit force the estimated model to almost fail which leads to ineffective perturbation protection in DataAnalyser. Failure to estimate occurs, for example, in the case of quasi separation of the data in logistic regression, and is more likely to occur with small sample sizes (sparsity) and units with rare characteristics.

Perturbation Averaging

An attacker firstly identifies a target unit for attack, and conducts repeated regression analyses by excluding a single unit (other than the target unit) each time. The attacker then averages these model predictions in anticipation of identifying a key characteristic of the target unit.

Solving Model Equations

An attacker fits a range of different types of models (e.g. logistic, Poisson and linear etc.) on the same dataset and solves the mathematical equations corresponding to the score functions simultaneously in order to recover the actual values in the dataset (Chipperfield, 2013).

Part I
Protections for TableBuilder

5 Perturbing Tables of Categorical Data

A table of categorical data, such as persons by sex and marital status, contains the number of individual units within the available categories. The confidentiality routine applied to categorical tables is based on adding a random amount to each non-zero cell of the table. For weighted datasets, e.g. surveys, the confidentiality routine is applied to the cell counts before the weights are applied. The perturbation method is repeatable, that is, when the same units are together in a particular cell, they will always be perturbed to the same value. This, and other confidentiality features, protect against differencing attacks and other attempts at identification. Some further details of how this has been implemented at the ABS is given in Leaver (2009). The paper by Schlomo (2007) reviews common statistical disclosure control (SDC) methods, including record swapping and cell rounding.

5.1 Definitions

A table consists of cells, both inner cells and total or marginal cells. Each cell is comprised of a number of units or contributors. When these contributors or units are summed, this quantity can be referred to as the unweighted count, defined as

$$Unweighted\, Count \quad = \quad n. \tag{1}$$

When the dataset is weighted, then each i^{th} unit has an associated weight, defined as

$$Weight\ of\ i^{th}\ Unit \quad = \quad w_i \tag{2}$$

for $i = 1, 2, \ldots, n$. The confidentiality formulas also hold for datasets without weights, such as an administrative dataset. In this case, unit weights are applied, that is $w_i = 1$ for all values of i.

When each unit's weight is taken into account, each cell of a table has an associated weighted count, defined as

$$WC = \sum_{i=1}^{n} w_i \tag{3}$$

and it follows that a cell's average weight is therefore

$$CellAW = \frac{\sum_{i=1}^{n} w_i}{n}. \tag{4}$$

To prepare a dataset for perturbation, a pseudo-random number, called a record key, is assigned to the microdata.

$$Record\ Key\ of\ i^{th}\ Unit = RKey_i. \tag{5}$$

Record keys are positive integers less than 2^{32} and are assigned randomly to each unit. They are the main driver for determining the perturbation amount that gets applied to a particular cell. When a table is constructed, the record keys are summed over each cell, to give the cell key

$$CKey = \sum_{i=1}^{n} RKey_i\,(mod\,bigN) \tag{6}$$

where $bigN$ is a large prime number chosen such that when represented as a 32-bit values, has a sufficiently random distribution of $0's$ and $1's$.

5.2 Count Perturbation Method

The perturbation amount is looked up in the perturbation table, $pTable$, which is a 2-dimensional array with 256 rows and $pTableSize$ columns, where $pTableSize$ is an unsigned integer, usually in the range of 15 to 100. The rows of the $pTable$ are indexed from 0 to 255 and the columns are indexed from 1 to $pTableSize$.

The perturbation amount, p, is looked up in the $pTable$ as

$$p = pTable\left[p_{row\_index}, p_{col\_index}\right]. \tag{7}$$

The perturbation table row lookup value is determined as

$$p_{row\_index} = A \oplus B \oplus C \oplus D \tag{8}$$

where $\oplus$ refers to the bitwise XOR operator, or exclusive OR, which is a logical operation of two numbers, defined as $(A \oplus B)_i = A_i + B_i \pmod 2$, where subscript i refers to the i^{th} bit. The values A, B, C, D are the four 8-bit binary components derived from representing $CKey$ as a 32-bit binary number.

The perturbation column lookup index is determined as

$$p_{col\_index} = \begin{cases} n & \text{if } n \leq pTableSize - smallN \\ pTableSize - smallN & \\ +n \,(mod\ smallN) + 1 & \text{otherwise} \end{cases} \tag{9}$$

where $smallN$ is a parameter that controls how the columns of the $pTable$ are scrolled, or recycled, through for cells of different sizes. This recycling directs sample sizes that are larger than $pTableSize$ to a specific column on the right hand side of the table.

The perturbation algorithm has been designed to incorporate a number of important properties:

- protect against differencing, that is where two large cells can be differenced to produce smaller cell estimates;
- ensures that the value for two cells with the same contributors, receives the same perturbation. This applies if the cells are in different tables. (This prevents averaging attacks)
- does not perturb zero cells;
- will not produce negative values;
- applies relatively more noise to smaller cells; and
- does not add bias to the final table.

The methodology behind the $pTable$ design is detailed in Fraser and Wooton (2003).

Once the perturbation amount is obtained using (7), the unweighted count, (1), is perturbed to give the perturbed unweighted count

$$pUWC = n + p. \tag{10}$$

If the dataset has weights, then this value is weighted, using the cell average weight (4), and the perturbed weighted count is given as

$$pWC = (n + p) \times \frac{\sum_{i=1}^{n} w_i}{n}. \tag{11}$$

As mentioned, perturbation has the greatest relative impact on small cells. However, less reliance should be placed on any small cell data as they are impacted by other errors including random adjustment, respondent, processing errors and, for survey data, sampling error. The effect of the introduced random error can be minimised if the statistic required is obtained directly from tabulation rather than from aggregating more finely classified data. Similarly, rather than aggregating data from small areas to obtain statistics for a larger area, published data for the larger area should be used wherever possible.

Since perturbation is applied independantly to every non-zero cell of a table, table additivity is lost. For example, consider a table with two inner cells and one total cell. Before perturbation, the cells are additive, as $(3) + (2) = (5)$. After perturbation, we may have $(3 + 2) + (2 + 4) \neq (5 - 2)$. As this simple example shows, the table is no longer additive. The ABS has developed an algorithm which restores additivity to the table by forcing the sum of the inner cells to equate to the table margins. The benefit of additivity is that tables will be internally consistent, however published estimates may differ across tables and ABS publications. Some TableBuilder datasets have additivity activated while others do not.

It is not possible to determine which individual figures have been affected by perturbation. For example, a zero value in TableBuilder can be due to the confidentiality process or it can be a logical or structural zero (when a non-zero value for a particular cell is not possible). For survey data, a zero can also occur when the quantity being measured can be non-zero but the unit with this characteristic was not included in the sample.

In TableBuilder, the final perturbed estimate, can be published with or without additivity and also be adjusted by a scale factor, displayed to a specified precision.

5.3 Example of Count Perturbation

Consider the following example where we perturb a cell of a table of categorical data.

A particular cell has $n = 31$ contributors, see (1), and each contributor has an individual record key, (5), and weight, (2). We can calculate the weighted count as $WC = 6258.8$, using (3), and the cell key as $CKey = 234606226$, using (6), for a particular $bigN$ and the cell average weight as $CellAW = 6258.8/31 = 201.897$, using (4).

We will use the perturbation table shown in Table 1. This $pTable$ has $pTableSize = 25$ and we assume that $smallN = 9$.

	col 1	col 2	col 3	...	col 21	...	col 25
row 0	-1	-2	5		-5		4
row 1	-1	-2	0		0		1
row 2	-1	-2	0		-3		-3
...				...		...	
row 170	-1	-2	3		**-4**		2
...				...		...	
row 255	-1	-2	-3		-2		-1

Table 1: Example perturbation table $pTable$

The perturbation table row lookup, using (8), resolves to $p_{row\_index} = 170$ and the perturbation table column lookup, using (9), resolves to $p_{col\_index} = 25 - 9 + mod\,(31, 9) +$

$1 = 21$. Using these values to lookup the *pTable* gives a perturbation amount of $p = -4$.

The cell count is perturbed, using (10), to give the perturbed unweighted count, $pUWC = 31 - 4 = 27$ and then the perturbed weighted count, using (11), to give $pWC = 27 \times 201.897 = 5451$.

Note that the *pTable* shown in Table 1 has the property that any cell with one or two contributors will always be perturbed to zero.

6 Perturbing Tables of Continuous Data

The methodology used to perturb tables of continuous means and sums, developed by the ABS, is referred to as the Top Contributors Method. For example, when income is a continuous field, a table of continuous data could be average income for sex by marital status. The Top Contributors Method consists of pseudo-random multiplicative adjustments made to the top contributors of each non-zero cell of the table.

The design of this method ensures that individual contributions are masked, as well as contributors to small cells and cells dominated by a small number of contributors. As with count perturbation, when the same units contribute to a particular cell, they will always be perturbed to the same value.

6.1 Definitions

Consider any continuous, or magnitude, variable for the i^{th} unit defined as

$$Continuous\_Value \quad = \quad y_i \tag{12}$$

where $i = 1, 2, \ldots, n$.

For a weighted database, we consider the sum and mean over all units in the cell.

The weighted sum is defined as

$$WY \quad = \quad \sum_{i=1}^{n} w_i y_i \tag{13}$$

The weighted mean is defined as

$$W\bar{Y} \quad = \quad \begin{cases} \frac{\sum_{i=1}^{n} w_i y_i}{\sum_{i=1}^{n} w_i} & \text{if } \sum_{i=1}^{n} w_i \neq 0 \\ 0 & otherwise \end{cases} \tag{14}$$

where n is given by (1) and w_i is given by (2). Note that in (14), the weighted mean is zero only in the case of empty cells.

As previously mentioned, the perturbation formulas also hold for datasets without weights as unit weights, $w_i = 1$ for all values of i, are applied.

6.2 Continuous Perturbation Method

The continuous perturbation in the Top Contributors Method is applied to selected units within the cell. The number of units that are selected is controlled by the *topK* parameter and the units are selected based on their absolute magnitude. As such, we rank the *topK* units by their absolute descending value, $|y_1| > |y_2| > \ldots > |y_{topK}|$, without considering their weights, if they have any. Any ties are resolved by choosing the first record in the database.

The perturbation amount, applied to each of the *topK* contributors, is composed of three components with complementary characteristics: the magnitude (or size), the direction (either positive or negative) and pseudo-random noise.

The magnitude component, m_i, where $i = 1, 2, \ldots, topK$, determines the size of the perturbation and is defined as

$$m_i \quad = \quad mTable\,[i] \tag{15}$$

where $mTable$ is a 1-dimensional array of length *topK*. We typically choose decreasing values for the $mTable$ so that largest to smallest contributors will receive decreasing amounts of perturbation. For example, if $topK = 2$ and $mTable = [0.5, 0.4]$, the magnitude component applied to the largest contributor is 0.5, and 0.4 is applied to the second largest contributor.

The direction component, d_i, where $i = 1, 2, \ldots, topK$, determines whether the perturbation amount is a positive or negative quantity and is defined as

$$d_i \quad = \quad \begin{cases} +1 & \text{if } 9^{th}\ bit\ of\ RKey_i = 1 \\ -1 & \text{if } 9^{th}\ bit\ of\ RKey_i = 0 \end{cases} \tag{16}$$

where $RKey_i$ is given by (5). Using $RKey_i$ to control the direction means that it is possible for each of the *topK* contributors to be assigned positive or negative perturbations.

The noise component, s_i, where $i = 1, 2, \ldots, topK$, adds a pseudo-random factor to the perturbation and is defined as

$$s_i \quad = \quad sTable\,[s_{row\_index_i}, s_{col\_index}] \tag{17}$$

The noise component is looked up in the $sTable$ which is a 2-dimensional array with 256 rows and $smallC + 32$ columns. The small cell parameter, $smallC$, sets the threshold value of the number of contributors in a small cell.

The perturbation table row lookup index determined as

$$s_{row\_index_i} \quad = \quad 1^{st}\ 8\ bits\ of\ RKey_i \tag{18}$$

which means that each of the *topK* contributors (are highly likely to) read a different row of the $sTable$.

The perturbation noise table column lookup index is determined as

$$s_{col\_index} = \begin{cases} n+32 & \text{if } n \le smallC \\ 1^{st}\, 5\, bits\, of\, CKey + 1 & \text{if } n > smallC \end{cases} \tag{19}$$

which means that each of the *topK* contributors will read the same column of the *sTable*. The perturbation was designed to allow different noise distributions to apply to small and large cells. The *sTable* can be visualised as two tables side by side, where large cells lookup the 32 left hand side columns and small cells lookup the remaining right hand side columns. As the column lookup is the same for each of the *topK* cell contributors, if the table is rebuilt and the cell has fewer contributors, then a different $CKey$ value will be calculated, usually corresponding to a different column in the *sTable*. The distributions used to model the noise in the *sTable* should be symmetric so as not to introduce bias.

Once the various continuous perturbation components have been calculated, the cell is initially perturbed as

$$pWY = \sum_{i=1}^{n} y_i w_i + \sum_{i=1}^{topK} y_i w_i m_i d_i s_i. \tag{20}$$

As (20) shows, the magnitide, m_i, direction, d_i, and noise, s_i, perturbation components are only applied to the *topK* contributors in the cell.

6.3 Mean Before Sum

The perturbation methodology in TableBuilder has been designed to ensure consistency between the categorical and continuous estimates. That is, the perturbed, weighted or unweighted, estimates of count, mean and sum will agree such that given two of these values, you will be able to derive the third. To maintain this consistency, the continuous calculation method must be configured with either the mean before sum or sum before mean method. The choice of method will depend on whether more accuracy is desired in the mean or the sum. We detail the mean before sum method, which allows greater accuracy in the calculation of the mean, and is the method currently employed in TableBuilder.

In the mean before sum method, we first calculate the perturbed weighted mean by taking the perturbed quantity, (20), and dividing it by the weighted count, (3), to give

$$\begin{aligned} pWY_m &= \frac{pWY}{WC} \\ &= \frac{\sum_{i=1}^{n} y_i w_i + \sum_{i=1}^{topK} y_i w_i m_i d_i s_i}{\sum_{i=1}^{n} w_i} \end{aligned} \tag{21}$$

The perturbed weighted sum is then obtained by multiplying the perturbed weighted mean, (21), by the pertubed weighted count, (11), to give

$$
\begin{aligned}
pWY_t &= pWY_m \times pWC \\
&= \frac{\sum_{i=1}^{n} y_i w_i + \sum_{i=1}^{topK} y_i w_i m_i d_i s_i}{\sum_{i=1}^{n} w_i} \times (n+p) \times \frac{\sum_{i=1}^{n} w_i}{n} \\
&= \left(\sum_{i=1}^{n} y_i w_i + \sum_{i=1}^{topK} y_i w_i m_i d_i s_i \right) \times \frac{(n+p)}{n}. \qquad (22)
\end{aligned}
$$

In the sum before mean method, the perturbed weighted sum is simply pWY and the perturbed weighted mean is obtained by dividing (20) by (11), the perturbed weighted count.

6.4 Example of Continuous Perturbation

Consider the following example where we perturb one cell of a continuous table using the Top Contributors Method.

We set $topK = 4$, and specify the magnitude table as $mTable = [0.6, 0.4, 0.3, 0.2]$. Table 2 shows the values of some key variables for the 8 contributors in this cell.

	Continuous field	Weight	Magnitude	Direction	Noise		
	y_i	w_i	m_i	d_i	s_i	$y_i w_i$	$y_i w_i m_i d_i s_i$
1	72.1	458.2	0.6	1	0.95	33,036	18,831
2	65.3	185.7	0.4	-1	1.02	12,126	-4,947
3	65.3	752.7	0.3	-1	1.54	49,151	-22,708
4	50.1	612.6	0.2	-1	1.54	30,691	-9,453
5	49.2	977.5				48,093	
6	45.4	458.7				20,825	
7	36.9	896.3				33,073	
8	36.9	995.2				36,723	
Total		5336.9				263,719	-18,278

Table 2: Example of continuous perturbation

Table 2 shows that the $TopK = 4$ units have been ranked by $|y_i|$ and assigned the corresponding magnitude value, (15). The direction value, (16), is calculated from $RKey_i$ and the noise values, (17), obtained from the $sTable$, given in Table 3. It has $smallC = 12$ and hence a dimension of 256 rows and $32 + 12 = 44$ columns.

Since the cell we are perturbing is small, $n = 8 \leq smallC = 12$, we lookup the $32 + 8 = 40^{th}$ column, using (19). The row lookup values are determined using (18), and resolve to three different row values. With these values, we can perturb the top contributors to give $-18,278$, and add to the weighted count for this cell, $263,719$, to give a perturbed weighted quantity of $245,441$, using (20).

	col 1	...	col 32	col 33	...	col 40	...	col 44
row 0	0.51		1.67	0.44		0.66		1.2
row 1	0.58		0.88	1.66		***1.54***		1.56
row 2	1.59		0.69	1.98		***0.95***		0.99
...		...			...		...	
row 207	0.48		0.64	1.01		***1.02***		0.88
...		...			...		...	
row 255	1.26		1.47	1.57		0.58		0.54

Table 3: Example perturbation (noise) lookup table *sTable*

7 Perturbing Tables of Quantile Data

7.1 ABS Method to Estimate Quantiles

Quantiles or percentiles indicate the value below which a specified proportion of the ranked dataset lies. For example, the median is the value below which 50% of the ranked data lies. TableBuilder allows quantiles to be constructed using two different types of distribution variables: a weight or a continuous field. For example, quartiles for income, distributed by person weight, equally distribute the number of persons into four groups, with each group having 25% of the total number of persons. Alternatively, quartiles for income, distributed by income, equally distribute the total income into four groups, with each group having a 25% share of the total of income.

The percentiles are calculated based on the ABS standard method of weighted cumulative proportions. After the units are ranked according to the value of the continuous field, the two records with levels above and below the desired percentile are isolated and a weighted interpolation between the two proportions is calculated. If the dataset is unweighted, then unit weights are used.

Once the quantile has been created, it can be used in a table just like any other categorical variable.

7.2 Quantile Perturbation Method

The quantile levels available in TableBuilder are configurable, with the most common being medians, quartiles, quintiles and deciles. The confidentiality risk, in particular the risk of differencing quantiles, is mitigated in a number of ways, such as the minimum sample size, quantile boundary thresholds and perturbation.

For any quantile requested in TableBuilder, the filtered subpopulation must exceed the minimum number of contributors, based on the fineness of the quantile. For example, if quintiles are requested for the subpopulation of unemployed males over 55 years, but the number of contributors falls below the minimum, the quantile is not produced. The minimum values can be configured per quantile type, for each dataset in TableBuilder.

Range restrictions are applied to user requested quantile range. If a quantile range satisfies the minimum number of contributors, the quantile boundaries are checked to ensure they do not fall below the minimum or above the maximum defined threshold. If one of these thresholds is breached, then the quantile is not produced. The thresholds can be configured per continuous field, for each dataset in TableBuilder.

When these safeguards are satisfied, the requested quantile levels are perturbed. A random amount is added to the quantile level and the corresponding quantile value is calculated. For example, if a median, 0.50, is requested, it may be perturbed to 0.62, and the quantile value displayed corresponds to 0.62, not 0.50. This is illustrated in a later example.

The perturbation amount is looked up in the quantile perturbation table, *qTable*, which is a 2-dimensional array with 128 rows and 100 columns, as

$$U_q = qTable\left(u_{row\_index}, u_{col\_index}\right). \tag{23}$$

So, for example, $U_{0.75}$ represents the perturbation amount applied to the 75th percentile.

The quantile perturbation table row lookup index is determined as

$$u_{row\_index} = 1^{st}\, 7\, bits\, of\, CKey \tag{24}$$

where $CKey$ is given by (6).

The quantile perturbation table column lookup index is determined as

$$u_{col\_index} = 100 \times q \tag{25}$$

where q is the quantile value such that $0 < q < 1$. So for example, for the 25th percentile, we would look up the 25th column for the first quartile.

Having obtained the quantile perturbation amount, the quantile level is perturbed by adding the perturbation amount to the original quantile as

$$pq = \begin{cases} q + \frac{U_q}{n} & \text{if } n \neq 0 \\ not\ calculated & \text{otherwise} \end{cases} \tag{26}$$

where $0 < pq < 1$, n is given by (1) and U_q is given by (23). As mentioned, the quantile is not calculated if the minimum number of contributors or the minimum and maximum thresholds are breached.

Once the quantile level has been perturbed, we need to calculate its value, as per the ABS standard method. We identify the unit, j, which lies below the perturbed quantile, defined as

$$j = max\left\{i : a_i < pq\right\}. \tag{27}$$

where pq is the perturbed quantile, (26), a_i, $i = 1, 2, \ldots, n$, is given by

$$a_i = \frac{1}{2}(e_i + e_{i-1})$$

and e_i, $i = 1, 2, \ldots, n$, represents a scaled weight function, given as

$$e_i = \frac{\sum_{k=1}^{i} w_k}{\sum_{k=1}^{n} w_k}$$

with $e_0 = 0$. Here the e_i are the empirical cumulative weights, and the a_i are the consecutive averages of the e_i.

The weighted interpolation between the two proportions, below and above each quantile, can then be calculated as

$$X_{pq} = \frac{(pq - a_j)\, y_{j+1} + (a_{j+1} - pq)\, y_j}{a_{j+1} - a_j} \tag{28}$$

An important issue to note is that of perturbed quantile cross-over. This means that quantiles are perturbed to the extent that a higher quantile falls below a lower quantile. For example, the first decile might be perturbed to $pq = 0.16$ and the second decile might be perturbed to $pq = 0.14$ which is inconsistent. Quantile cross-over can be prevented by ensuring the minimum number of contributors parameter and the maximum perturbation values, in the quantile perturbation table, are chosen with regards to one another. For example, if the minimum contributors is set to 200 for deciles, the *qTable* should not contain perturbation values greater than 10. This is determined by using (26) and solving $pq = 0.10 + {}^{x}\!/_{200} < 0.15$ for x. This means the first decile should not cross over mid-way to the second decile.

7.3 Example of Quantile Perturbation

The process of calculating the perturbed median for a cell with 30 contributors is illustrated in Table 4.

Without perturbation, the median could be calculated using (26), to give $pq = q = 0.5$. It occurs at the level $j = max\{i : a_i < 0.5\} = 16$, using (27), and has a corresponding value of $X_{0.5} = 16.13$, using (28).

To perturb the median, we look up the perturbation amount in the *qTable*, to give $U_{0.5} = 3.5$. The median level is perturbed to $pq = 0.5 + {}^{3.5}\!/_{30} = 0.6167$, occurs at the level $j = max\{i : a_i < 0.6167\} = 19$ and has a corresponding value of $X_{0.6167} = 18.74$.

8 Custom Ranges

TableBuilder also allows custom ranges to be built for each continuous variable on the dataset. As per quantiles, range restrictions are applied as described in Subsection 7.2. Once the custom range has been created, it can be used in a table just like any other categorical variable. The custom range works exactly as constructed and is not perturbed. When a custom range is added to a table, the disclosure protection consists of the count or continuous perturbation methods detailed in this document.

	Continuous Field	Weight	Scaled Weights	Averaged Scaled Weights
i	y_i	w_i	e_i	a_i
1	2	2.5	$2.5/180 = 0.014$	$0.5\,(0.014) = 0.007$
2	2	0.8	$(2.5+0.8)/180 = 0.018$	$0.5\,(0.014 + 0.018) = 0.016$
3	3	3	$(2.5+0.8+3)/180 = 0.035$	$0.5\,(0.018 + 0.035) = 0.027$
...			...	...
16	16	30	0.578	0.494
17	18	3	0.594	0.586
...			...	...
19	18	0.5	0.609	0.607
20	20	8.6	0.633	0.633
...			...	...
30	28	5	$(2.5+0.8+...+5)/180 = 1$	0.986
Total		180		

Table 4: Example quantile perturbation data

9 Other Tabular Confidentiality Routines

TableBuilder has some other confidentiality routines, in addition to perturbation, that provide further protection to the dataset. These routines include the Sparsity algorithm and the Field Exclusion Rule.

9.1 Sparsity

The sparsity routine ensures that there are not too many cells in a table with one or two contributors. When a table is suppressed, no other confidentiality modules are run on the table and all the cell values are set to zero.

A table is suppressed by the sparsity routine if any of the following conditions are true

$$\begin{aligned} \frac{c_1}{c - c_0} &> ThresholdA \\ \frac{c_1 + c_2}{c - c_0} &> ThresholdB \\ c - c_0 &\neq 0 \end{aligned}$$

where c = total number of inner table cells, c_i = number of inner table cells with i contributors, $i = 0, 1, 2$, and $ThresholdA$ and $ThresholdB$ are configurable parameters.

For tables of continuous values, the Sparsity routine is applied to the responding sample. That is, the units in the sample that have a valid continuous value and not any special codes, such as not applicable or not known.

9.2 Field Exclusion Rules

The Field Exclusion Rules prevent certain combinations of fields appearing on a table. The rule is constructed by a list of field names and a number. For example, if the rule is Cat, Dog, Frog and Mouse with max = 2, then no more than two of these fields can appear on a table at any one time. This rule was developed for combinations of variables that have a higher risk of identifying rare units or where the same concept is represented by multiple variables. An example of the latter is the case of slightly differing geographic variables.

10 Relative Standard Errors

The Relative Standard Errors (RSEs) are calculated using the ABS standard method of sample replication. TableBuilder allows for sample replication using either the Jackknife or Bootstrap, which are different types of sampling methods. The RSEs published in TableBuilder include variance components from a number of sources. All survey estimates will include the variability due to obtaining sample estimates of counts. The perturbed estimates (counts, means, totals and quantiles) will also include the variability introduced from the different perturbation processes. The RSE formulas and their derivations are provided in a future paper to be released.

Part II
Protections for DataAnalyser

11 Hex Bin Plots

In DataAnalyser, raw scatter plots are represented by hex bin plots, using a modification of the *hexbin* package in R (R Core Team, 2013). These plots are displayed within the hex bin plot page, where users can plot continuous variables against each other, and in various diagnostic plots from regressions.

Hex bin plots were originally developed to display high-density scatter plots; however, DataAnalyser uses them as disclosure risk method for scatter plots. A hex bin plot divides the area in a graph into tessellating hexagons, then shades each hexagon depending on the number of observations that occur in that hexagon. An example of such a plot is shown in Figure 1.

11.1 Protections within Hex Bin plots

A hex bin plot can be interpreted as a table of unweighted counts where each hexagon represents a cell with a custom definition; however, each unweighted count is presented to the user as a colour representing a range of values. The protections for hex bin plots were developed with this interpretation in the fore.

- When accessing the hex bin plot page, users may plot two continuous variables against each other. If either of these two variables have range restrictions (Subsection 7.2), any observations are discarded if they lie outside these restrictions for either of the two variables.

- A parameter, minCount, controls the minimum number of observations that a hexagon must contains for it to appear coloured. When a hexagon contains observations but is not plotted, we call this suppression.

  For plots of Cook's distance vs. Leverage, the minimum count protection is not applied as the purpose of these plots is to identify outlying points.

Perturbation methodology is not applied to the counts within hexagons that contribute to the plot as the colours already represent a range of values, and the boundaries of the hexagons are only graphically depicted.

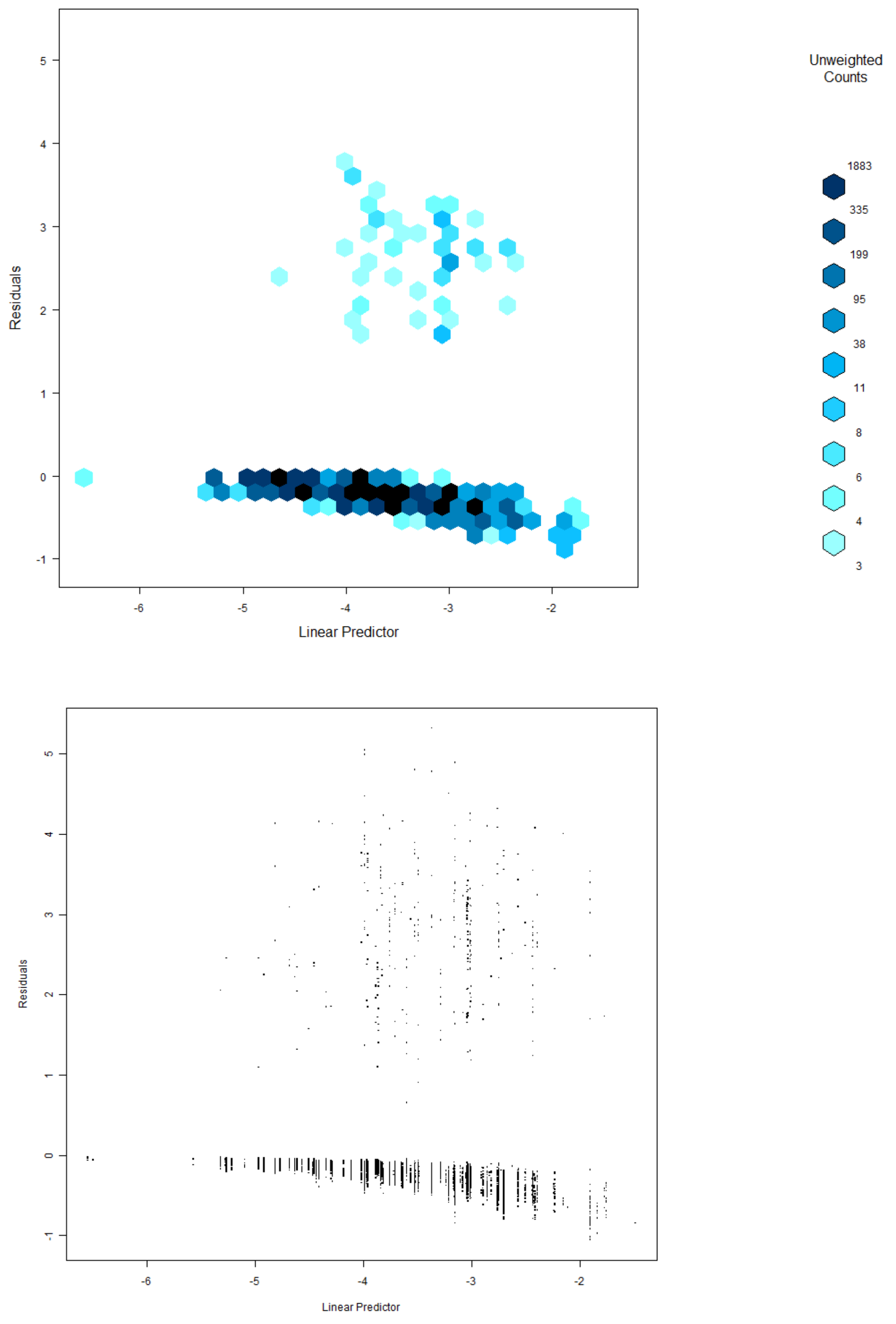

Figure 1: Example hex bin plot with equivalent scatter plot

11.1.1 Determining mesh size

The granularity of the tessellating hexagon mesh can be thought of as the resolution at which the scatter plot is blurred. Too low a resolution, and the hexagons are too large and not enough detail can be discerned; too high a resolution, and many of the hexagons will contain single observations and so will be suppressed. The algorithm to determine mesh size depends on a parameter, maxPropSuppressed, the maximum proportion of hexagons that have observations within them that are not coloured. The algorithm begins with the coarsest possible mesh, and increases the resolution of the mesh until the proportion of suppressed hexagons reaches maxPropSuppressed.

11.1.2 Determining colour scale

The colour scale, i.e. the ranges of counts that each colour represents, is chosen so that there are approximately the same number of hexagons of each colour/range in the final plot. This is achieved by taking deciles of the vector of counts that each hexagon represents. It was found that determining the colour scale in this fashion, as opposed to equispaced ranges, gave a much better representation of the initial scatter plot, especially when the plot had a combination of very dense areas and very sparse areas.

12 Scope Based Perturbation in DataAnalyser

Under coverage-only based perturbation, a scope-coverage attack may be undertaken by requesting two tables, each containing at least one cell with only slightly different scopes. By observing whether or not there is a difference between the corresponding cell values, an attacker is able to determine whether a unit falls within the non-common scope between the two cells. This is illustrated in Figure 2, where a potential attacker requests two tables defined by scopes A and B, which involve classification by 'Age' and a combination of any number of other classificatory variables (schematically represented by the vertical height of the boxes in Figure 2). Scope A involves the age range 15-95 whereas scope B involves the age range 15-96. The only difference between scopes A and B are the 96 year olds.

Note that the example given in Figure 2 is hypothetical, and is given for the purpose of illustration, only. In practice, ABS applies top-coding or broader categorisation to variables such as this, as part of the up-stream confidentialisation referred to in the third paragraph of Section 3.

A difference in the perturbed values indicates that at least one unit has been isolated, and if the difference in scopes is very slight, the attacker may infer the presence of only a single unit with sufficiently high probability. In Case 1 of Figure 2, the attacker looks at the perturbed counts for the two tables and notes that there is a difference, hence there must be at least one unit in the scope defined by B without A. In Case 2, the attacker observes that the perturbed cell counts are exactly the same and hence the scope B without A is empty.

The premise behind the perturbation scheme used in DataAnalyser is that each cell in a table is defined by a logical scope. The definition of this scope should contribute

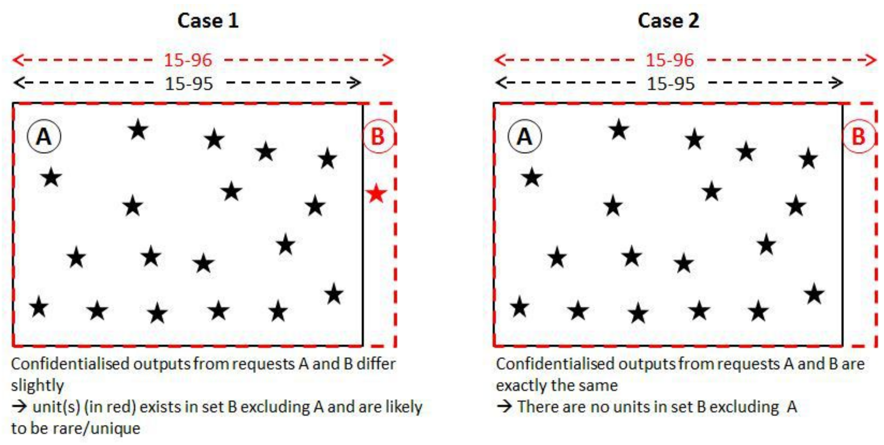
Case 1
15-96
15-95
A
B
Confidentialised outputs from requests A and B differ slightly
→ unit(s) (in red) exists in set B excluding A and are likely to be rare/unique
Case 2
15-96
15-95
A
B
Confidentialised outputs from requests A and B are exactly the same
→ There are no units in set B excluding A

Figure 2: Scope Coverage Attack

towards the perturbation applied to that cell, and not just the coverage of that scope (i.e. what units happen to fall into that cell). In DataAnalyser the scope of a cell in a table is determined by three contributing factors:

- The by-variables included in the table request
- Custom definitions of categorical variables (created with the Create New Variable page)
- Restriction of the scope of the dataset (created with the Drop Records page)

In the TableBuilder product, and in DataAnalyser prior to the implementation of scope-based perturbation, the system calculates a CKey based on coverage only, and this, together with the unperturbed cell value, was the main factor used to derive the perturbation to be applied. Scope-based perturbation mitigates the risk of a scope-coverage attack by determining the perturbation applied to a cell value from both the scope of the variable categories used in the request as well as the actual units that fall into the cell (coverage). Figure 3 illustrates schematically how the scope of a user request defines coverage, from which the CKey and the unperturbed value are determined. In TableBuilder, this is then used to identify the perturbation to apply, from a look-up table.

Under the new scheme, which has been deployed in DataAnalyser, an SKey (scope key) and SKey adjustment are calculated based on the scope of a cell. These two values are combined with the CKey to produce a TKey (total key) which is then used to derive a perturbation. This is illustrated schematically in Figure 4.

To preserve consistency between TableBuilder and DataAnalyser, the SKeyAdjustment term is introduced to cancel out the SKeys for any single categories that the scope is restricted to, or to put it another way, to remove the SKeys coming from the rectangular component of the shape. The Total Key is calculated as:

$$TKey = CKey + SKey - SKeyAdjustment \tag{29}$$

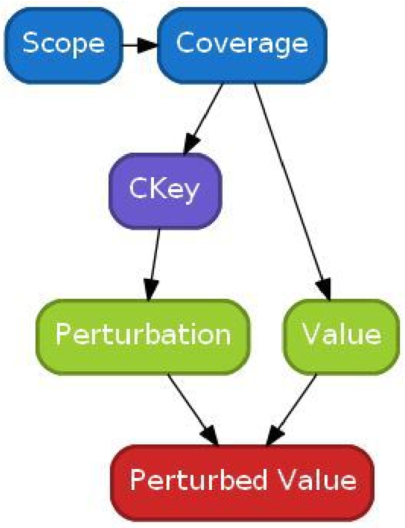

Figure 3: Schema for Coverage Based Perturbation

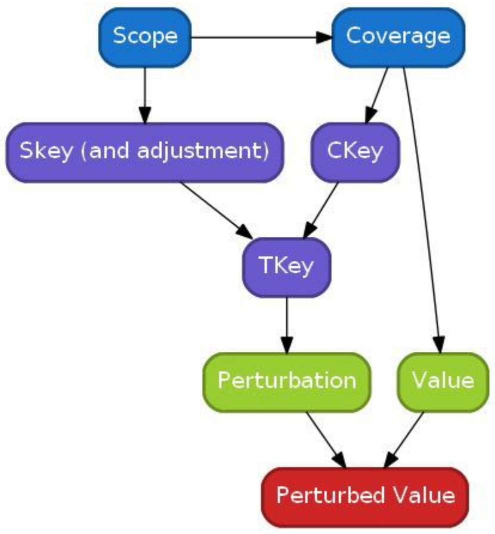

Figure 4: Schema for Scope and Coverage Based Perturbation

The look-up procedure for determining the perturbation value from CKey, shown in Figure 3, is identical to that for TKey, shown in Figure 4. Thus, if $SKey = SKeyAdjustment$, the same perturbation is derived, thus ensuring consistency. The calculation of SKeyAdjustment is shown in Subsection 12.3.

For the remainder of this section, **all arithmetic is done modulo** $bigN$ **(which, for the sake of example, is chosen to be 100)**.

A primitive SKey is assigned to all categories of existing categorical variables when the dataset is first loaded to the system. SKeys for custom created categorical variables

	Unemployed	Employed	NILF
Male	🟦	🟦	🟦
Female		🟦	

Table 5: Simple User Requested Table

are derived from these primitive SKeys.

12.1 Calculation of SKeys for Scopes Involving Categorical Variables Only

Scopes for cells can be considered geometrically as N-dimensional shapes where N is the number of variables involved and as mentioned previously these shapes are a union of rectangular cuboids. We can derive an SKey for any particular shape by defining it as the standard Lebesgue measure in N-dimensional Euclidean space (modulo $bigN = 100$) of the particular shape, ie area for $N = 2$ and volume for $N = 3$.

The side-lengths of these cuboids are fixed values called primitive SKeys that are assigned prior to loading the dataset. Similar to RKeys, SKeys must be elements of $\mathbb{Z}_{bigN}$. To ensure normalisation of primitive SKeys, we also require that for each categorical variable, the generated primitive SKeys sum to one. The primitive SKeys are generated for each categorical variable upon loading the dataset into into DataAnalyser.

This definition will become clear after an example.

12.1.1 Example of the Calculation of SKeys for Scopes Based Only on Existing Categorical Variables

A user may wish to create a new custom variable using the logical expression "Sex = Male OR LFS = Employed", resulting in a new categorical variable with categories: Unemployed Male, Employed Male, Not in the Labour Force (NILF) Male, Employed Female. This can be visualised as in Table 5.

Suppose the primitive SKeys are assigned to the relevant categories as follows:

	Primitive SKey
Male	22
Female	79
Unemployed	29
Employed	90
NILF	82

Then the derived SKey for the above scope would be calculated for each cell by multiplying the primitive SKey for the corresponding row category by the primitive SKey for the corresponding column category and taking the result modulo $bigN = 100$. This results in the SKeys shown in blue in Table 6.

Note that the column widths and row heights are shown proportional to the primitive SKey for the respective category. Therefore, the derived SKey for the scope restriction based on the categories coloured in blue is $SKey = 38 + 80 + 10 + 4 = 32$. As shown in

		Unemployed	Employed	NILF
	SKey	29	90	82
Male	22	38	80	4
Female	79		10	

Table 6: SKeys for Example 12.1.1

(29), the calculation of the final Total Key (Tkey) requires the calculation of the SKeyAdjustment. We give an example of this in Subsection 12.3.

Note that for any scope of the form "Variable=Category", for example "Sex = Male", the derived SKey is necessarily equal to the primitive SKey for the category in question. Thus, we can drop the distinction between primitive SKey and derived SKey and simply refer to them as SKeys.

12.2 SKeys for Scopes Involving Continuous Variables

The above schema works well for new categorical variables created by crossing categories from pre-existing categorical variables, but what about creating new custom categories from pre-existing continuous variables. In such a case, it is impossible to assign, ahead of time, an SKey to all possible intervals. Instead we define a hashing function H that maps every simple interval to an SKey. A requirement placed on this hashing function is that the sum of the hashed values for all partitions I of some partition P of the real line, is unity.

$$\sum_{I \in P} H(I) = 1$$

This can be achieved by defining H in terms of another hashing function $h : [-\infty, \infty] \to \mathbb{Z}_n$ with the requirement that $h(-\infty) = h(\infty) = 0$ such that if interval I is any one of (a, b), $[a, b)$, $(a, b]$ or $[a, b]$ $(a < b)$ then

$$H(I) = h(a) + adj_a - [h(b) + adj_b] \tag{30}$$

where

$$adj_a = \begin{cases} 1 & \text{if } I \text{ is } \textbf{open} \text{ at } a, \\ 0 & \text{otherwise.} \end{cases}$$

$$adj_b = \begin{cases} 1 & \text{if } I \text{ is } \textbf{closed} \text{ at } b, \\ 0 & \text{otherwise.} \end{cases}$$

and h(x) is based on the IEEE 802.3 CRC-32 checksum, modulo bigN.

h

		Age		
		$(-\infty, 18)$	$[18, 65]$	$(65, \infty)$
	SKey	57	7	37
Male	22	54	54	14
Female	79		53	

Table 7: SKeys for Example 12.2.1

12.2.1 Example of the Calculation of SKeys for a Scope Involving a Continuous Variable

Suppose the SKey for the following scope were to be calculated;
$Sex =$ Male OR (Age ≥ 18 AND Age ≤ 65).

Furthermore, suppose that the hashing function h gives $h(18) = 44$ and $h(65) = 36$; then the SKey for this new categorical variable can be calculated as:

$$\begin{aligned} H\left((-\infty, 18)\right) &= (0+1) - (44+0) && = -43 \pmod{100} = 57 \\ H\left([18, 65]\right) &= (44+0) - (36+1) && = 7 \\ H\left((65, \infty)\right) &= (36+1) - (0+0) && = 37 \end{aligned}$$

Note that the convention used for the modulus of a negative number is to take the value in excess of the multiple of 100 less than or equal to the given number. The SKeys for each new category can now be calculated as shown in Table 7.

The SKey for the custom variable with this scope is therefore $(54 + 54 + 14 + 53) \pmod{100} = 175 \pmod{100} = 75$.

12.3 Example of the Calculation of SKeyAdjustment

The SKey adjustment is calculated by considering every categorical variable that defines the scope, and for each of those categorical variables that restricts the scope to a single category, the SKey adjustment is defined as the product of the SKeys attached to those categories. In the example of Subsection 12.1.1, the scope is not restricted to one particular sex category nor one particular labour force category. In the example of Subsection 12.2.1, the scope is not restricted one particular sex category nor one particular age category. In uninteresting examples like these, if no categorical variables restrict the scope to a single category, then the SKey adjustment is set to 1.

If no merging of categories is allowed, and no intervals of continuous variables are allowed (this excludes all operations involving "My custom data" or "custom ranges" in TB, and "Create new variable" and "Subset Dataset" in DataAnalyser), then the SKeyAdjustment is necessarily equal to the SKey for any constructible cell. In this situation, users could only construct conditions involving conjunctions of the AND logical operator, such as Sex=Female AND LFS=Emp AND Occupation=Welder. In this example, both the SKey and the SKeyAdjustment will be the product of the SKeys for 'Female', 'Emp' and 'Welder'. The effect of this is that for any such cell, the TKey is just simply the CKey, which provides consistency between DataAnalyser and TableBuilder for such tables.

Below is a schematic representation (outlined in red) of a more interesting example of the calculation of the SKeyAdjustment. In this example a user defines the following scope: Smoker Status = Smoker AND (Sex = Male OR LFS = Unemployed). For each category, the SKey value is shown in brackets after it and the intervals are scaled accordingly to visually represent the SKey scale. The defined scope is not restricted to male or female, or any particular labour force status, but it is restricted to the Smoker category. Therefore in this example, the SKeyAdjustment will be equal to the SKey for the Smoker category.

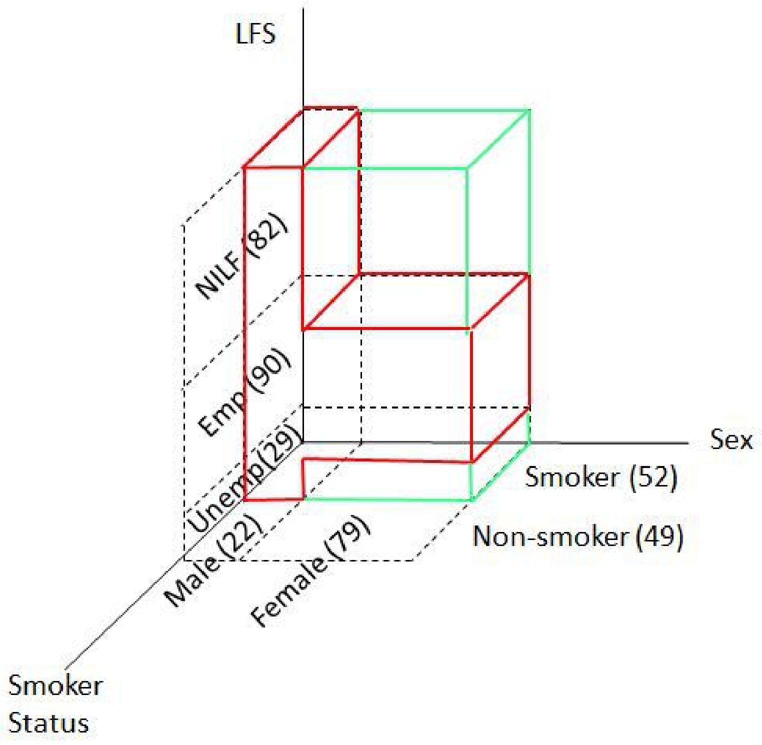

Figure 5: Example of Calculating SKeyAdjustment

If we let S = the SKey for Smoker, M = SKey for Male, F = SKey for Female, N = SKey for NILF, E = SKey for Emp and U = SKey for Unemp, and noting that by definition $N + E + U = 1$, we can see mathematically that:

$$\begin{aligned} SKey - SKeyAdjustment &= S*M*N + S*M*E + S*M*U + S*F*E - S \\ &= S*M + S*F*E - S \end{aligned}$$

We can see geometrically that for this example, the result of SKey - SKeyAdjustment is equivalent to the space indicated in light green which is the complement of the defined scope in red relative to the scope defined by Smoker.

12.4 Practical Considerations for Implementation

12.4.1 Resolution of Redundancies in Expressions Defining Scopes

In DataAnalyser, scopes are defined via logical expressions, and it is possible to build in redundancies into these expressions. In DataAnalyser, these redundancies are resolved before calculating an SKey. For example, a user might define a scope as $Age > 20\ AND\ Age > 15$, in which case, this should be resolved to $Age > 20$. This can also occur for categorical variables, eg $Sex \in (Male, Female)\ AND\ Sex = Male$ is resolved to Sex = Male. This step is crucial to preventing easy averaging attacks.

12.4.2 Computational Efficiency

As the SKey calculation can be thought of as a measure in n-dimensions, it follows the relationship SKey(A OR B) = SKey(A) + SKey(B) - SKey(A AND B). This can be used to break down more complicated scopes into simpler scopes that are represented by n-cubes. The SKeys for n-cubes can be calculated by summing up side lengths (sum of SKeys of the categories involved) and taking the product of these. This method is more efficient than the breakdown method shown in the examples above. For example, for the scope Industry $\in$ (Agriculture, Mining, Construction, Wholesale, Retail, Transport) AND Education $\in$ (Postgraduate, Diploma, Bachelor), one need not compute the SKeys for the $3 \times 6 = 18$ different cross-classifications and sum these values together; rather, one need only calculate the side lengths represented by the SKey for each variable, and multiply these side lengths together, e.g. (Agriculture + Mining + Construction + Wholesale + Retail + Transport) $\times$ (Postgraduate + Diploma + Bachelor).

13 Regression Perturbations

The application of perturbations to regressions in DataAnalyser is based on Chipperfield and Lucie (2011) and Chipperfield (2013) and uses a generalisation of the perturbed weighted quantity (20) to define a perturbed value for the weighted matrix transpose product $C = A^\top \operatorname{diag}(\mathbf{w})B$. Expanding this matrix product, the $(j,k)^{th}$ element of C is given by

$$c_{j,k} = \sum_{i=1}^{n} a_{i,j} b_{i,k} w_i.$$

So, if for each value of j, k we define $y_i := a_{i,j} b_{i,k}$, then the perturbed value, C_{pert}, of

C is a matrix of the same size whose $(j,k)^{th}$ element is given by

$$c'_{j,k} = \sum_{i=1}^{n} a_{i,j} b_{i,k} w_i + \sum_{i=1}^{topK} a_{i,j} b_{i,k} w_i m_i d_i s_i. \tag{31}$$

This allows us to apply this perturbation methodology to regressions. Recall that estimating the coefficients of a regression involve solving the equation generated by setting the score function equal to zero $\mathbf{\Theta}(\boldsymbol{\beta}) = \mathbf{0}$. In the case of generalised linear models, the score function can be written in the form

$$\mathbf{\Theta}(\boldsymbol{\beta}) = X^\top \operatorname{diag}(\mathbf{w}) Z(\boldsymbol{\beta}).$$

where X is the design matrix, $\mathbf{w}$ is a vector of weights, $\boldsymbol{\beta}$ is the vector of parameters to be estimated and Z is a matrix function of $\boldsymbol{\beta}$, in the case of Poisson models, $(Z(\boldsymbol{\beta}))_i = y_i - \log(X_i\boldsymbol{\beta})$. The following algorithm is used to calculate the perturbed parameter estimates $\hat{\boldsymbol{\beta}}_{pert}$.

1. Begin with an initial guess $\boldsymbol{\beta}^{(0)}$
2. Solve $\mathbf{\Theta}(\boldsymbol{\beta}) = \mathbf{0}$ using IRLS to get an unperturbed maximum likelihood estimate $\hat{\boldsymbol{\beta}}$
3. Calculate the perturbed score function evaluated at $\hat{\boldsymbol{\beta}}$ using 31 and letting $A = X$, and $B = Z(\hat{\boldsymbol{\beta}})$. Let the resulting vector of perturbations be $\boldsymbol{\epsilon}$.
4. Solve $\mathbf{\Theta}(\boldsymbol{\beta}) = \boldsymbol{\epsilon}$ using IRLS with initial guess $\boldsymbol{\beta}^{(0)}_{pert} = \hat{\boldsymbol{\beta}}$ to get a perturbed maximum likelihood estimate $\hat{\boldsymbol{\beta}}_{pert}$

14 Drop-k Units

Another protection applied to regression analyses is the random removal of a number of units. For each explanatory variable that is categorical, one record is removed for each category. The random number generator that determines which record is removed is seeded based on the scope key (SKey) (see Section 12) corresponding to that category and any modifications that have been made to the dataset. In this way, the removal of a unit is random but reproducible, as the randomness is essentially seeded on the request. This is an important point as it prevents averaging attacks whereby a user may request the same regression repeatedly and average the outputs to counter the perturbation protection.

15 Restrictions on Allowed Variables

There are two protections that restrict the variables that are allowed to be used in regressions: Field Exclusion Rules (FERs) and X-Only variables.

FERs work the same as in TableBuilder and each rule that is specified prevents certain combinations of variables being selected in the same query. These will typically be variables representing the same concept, but with a different coding, for example two different geographical codings, or age grouped in single year and in five year age groups.

Variables can be marked as X-Only variables meaning that they are restricted to only be used as explanatory variables in all regressions. These will typically be exogenous variables such as gender or age.

16 Other Regression Protections

The remaining protections that are applied to regressions relate to the leverage and sparsity of the requested models. A model is rejected if a unit has a leverage above a given value, or if two units have leverages that sum to above a given value.

Sparsity checks are performed to ensure that the analysis is based upon a sufficient number of records to reduce the risk of disclosure to acceptable levels. A requested model is rejected if:

- there are fewer than a minimum number of observations,
- greater than a maximum number of parameters, or
- there are fewer than a minimum number of observations for each parameter.
- A summary table is run with the response variable against each categorical explanatory variable in turn; if any of these tables contain a zero, then the model is rejected. Note that this sparsity check is based on the perturbed values of the tables, mitigating the risk of a user using the sparsity check to look for non-zeros that have been perturbed to zero.

17 Overall Conclusions

ABS has developed the TableBuilder and DataAnalyser remote server systems with automated confidentiality routines that allow users to build their own custom tables or undertake regression analyses on secured ABS microdata. A key issue has been in addressing the risk versus utility trade-off, to ensure that the level of protection is sufficient to ensure that ABS legislative requirements have been fulfilled, while at the same time delivering a system with sufficient flexibility for users and perturbed outputs that have minimal impact upon statistical inferences given the level of risk.

No theory exists in the literature that identifies the totality of all possible confidentiality risks for remote servers. We therefore had to take the approach of building into the systems protections against those attack risks, already identified in the literature, plus those we identified ourselves. With every new feature or functionality that will be added to the system in the future, it will be necessary to consider any new risks that arise from that added functionality, as well as how existing protections apply to the new feature. It is also necessary to consider the totality of protections being implemented as it is possible for a new protection introduced to eliminate one type of risk may increase another type of risk. This was found when the drop-k units protections was introduced (without constant seeding) as a preliminary solution to the scope-coverage attack, only to find that it can in some circumstances increase the risk of a perturbation averaging attack.

A substantial amount of development and infrastructure work was undertaken in developing R routines (R Core Team, 2013) for each functionality in DataAnalyser. If we are to extend the functionality of DataAnalyser to include more sophisticated statistical analysis techniques, such as multilevel models, significant development costs would be involved in writing the R code to incorporate the relevant perturbation and other protection routines. One possible strategy for tackling the future development of remote servers is for NSI's to demonstrate to proprietary statistical software vendors, such as SAS, SPSS and Stata, that there is a strong business case for developing confidentiality preserving analytical software that can be readily deployed in remote server systems. Regardless, it will be extremely beneficial for NSI's to work together in international collaborations in the building of more enhanced versions of analysis remote servers.

An important source of protection is the menu based user interface that impacts upon users' flexibility but makes the systems safer against a range of possible threats (Sparks et al, 2008). The menu based interface makes it time consuming and onerous to effect most attacks which require numerous requests. Apart from making it tedious for attackers, it increases the chance that such an attack will be detected from the audit system.

The availability of interactive metadata is almost as important as data itself in a remote server system. The structure of the dataset, the meaning and context of variables and the possible values the variables can take, is difficult to communicate to users through an interface, but is essential for usability. There is much more work to be done to improve this aspect in TableBuilder and DataAnalyser and this greatly depends on corporate solutions coming to fruition that enable better handling of metadata and machine to machine capability. Additionally, building a user interface that automatically handles all possible data structures in a flexible, responsive and easy-to-use way is difficult. In order to meet resource and timeframe restrictions, DataAnalyser has been built to handle only relatively simple hierarchical dataset structures. Subject to future funding opportunities, this will be an important area of future development.

User consultation is very important to ensure that TableBuilder and DataAnalyser remains relevant to users. ABS is planning to undertake extensive user consultation following the production release of DataAnalyser. Feedback from the consultation process will be assessed and prioritised for future system enhancements and development.

A high priority for ABS is making linked survey administrative datasets available in TableBuilder and DataAnalyser. There is strong analytical demand from the user community for linked datasets, however these pose a increased disclosure risk due to the fact that another agency has full access to the unconfidentialised administrative unit record file over which that agency has custody (Chipperfield, 2013). This provides detailed data with which an attacker within the external agency can launch an attack. ABS is currently undertaking a program of research work to develop confidentialisation methods for linked datasets. Another aspect of linked datasets is that they can sometimes be quite large, which impacts adversely upon system performance. Work is currently being undertaken to fine tune system performance in order to accommodate large datasets.

So far, the confidentiality routines within TableBuilder and DataAnalyser have largely been developed with household surveys in mind. Further work is needed to develop automated confidentiality methods for the dissemination of business and longitudinal datasets via remote analysis servers. Although TableBuilder and DataAnalyser already have confidentiality routines for handling continuous and highly skewed variables, there are increased confidentiality risks arising from cells in which a few businesses account for a high proportion of the cell total. Current rules for confidentialising data concerning people and households, may not be sufficient for businesses.

Another area of future research is the provision of information loss measures to researchers, which given a measure of the level of impact the perturbation may have had on inferences made from analytical outputs. This may be achieved through the use of either a verification server (Reiter et al, 2009), in which a user requests a report indicating the level of impact upon inferences, or information loss measures routinely derived for all analysis outputs using the missing information principle (Elazar and Chammas, 2011).

It seems clear that recent events in the development of remote analysis servers herald the dawn of a new era in automated confidentiality protection for analysis and we look forward to invigorated research collaborations among NSI's and academic institutions to further this research, particularly in extensions in the advanced analysis of linked, multilevel and longitudinal datasets. This in turn will hopefully lead to an opening up of government and corporate data holdings to their full analytic potential for the betterment of society.

References

Chipperfield, J. and Lucie, S. (2011) "Analysis of Micro-data: Controlling the Risk of Disclosure", *ABS Methodology Advisory Committee*, **MAC110**, June 2010.

Chipperfield, J. (2013) "Disclosure-Protected Inference with Linked Micro-data using a Remote Analysis Server", *Journal of Official Statistics* (Accepted subject to revision).

Elamir, E. A. H. and Skinner, C. J. (2006) "Record level measures of disclosure risk for survey microdata", *Journal of Official Statistics*, **22 (3)**, 525-539.

Elazar, D. N. and Chammas, J. (2011) "Application of the Missing Information Principle to the Analysis of Perturbed Data", *ABS Methodology Advisory Committee*, June 2011, ABS Cat. No. 1352.0.55.118

Fraser, B. and Wooton, J. (2005) "A proposed method for confidentialising tabular output to protect against differencing", *Joint UNECE Eurostat work session on statistical data confidentiality at Geneva*, Switzerland, 9-11 November 2005

Leaver, V. (2009) "Implementing a method for automatically protecting user-defined Census tables", *Joint UNECE/Eurostat work session on statistical data confidentiality*, Bilbao, Spain.

O'Keefe, C. and Chipperfield, J. (2013) "A summary of attack methods and confidentiality protection measures for fully automated remote analysis systems", *International Statistical Review* (Accepted).

O'Keefe, C. and Good, N. (2009) "Regression output from a remote analysis system", *Data and Knowledge Engineering*, **68**, 1175-1186.

R Core Team (2013) "R: A Language and Environment for Statistical Computing", *R Foundation for Statistical Computing*

Reiter, J. P., Oganian, A. and Karr, A. F. (2009), "Verification servers: enabling analysts to assess the quality of inferences from public use data", *Computational Statistics and Data Analysis*, **53**, 1475 - 1482.

Shlomo, N. "Statistical Disclosure Control Methods for Census Frequency Tables", *International Statistical Review*, **75 (2)**, 199-217.

Skinner, C. J. and Shlomo N. (2008) "Assessing Identification Risk in Survey Microdata Using Log-Linear Models", *Journal of the American Statistical Association*, **103:483**, 989-1001

Sparks, R., Carter, C., Donnelly, J., O'Keefe, C., Duncan, J. and Keighley, T. (2008) "Remote access methods for exploratory data analysis and statistical modelling: Privacy-preserving Analytics", *Computer Methods and Programs in Biomedicine* **91**, 208-222.