

Topic (ii): software and computing developments

FINGERPRINTS IN MICRODATA SETS

Submitted by Statistics Netherlands¹

Contributed paper

I. INTRODUCTION

1. The approach taken at Statistics Netherlands to protect a microdata set starts with identifying the unsafe records in this set. There are general rules that (more or less) prescribe which combinations of identifying variables to consider. Each combination of values that appears less than a prespecified threshold value in the microdata set is considered unsafe. The next step is to modify the data by the application of data protection techniques such as global recoding and local suppression so as to “eliminate” these combinations, in such a way that as much information as possible is preserved in the microdata set (see e.g. [6]).

2. In the present paper we consider a somewhat different approach. Instead of checking prespecified combinations of identifying variables, we are interested in records with many short fingerprints. A *fingerprint* is a combination (set) of values of identifying variables that are unique in the microdata set at hand, and besides contain no proper subset with this property (so it is a minimum set with the uniqueness property). So whether a combination of values is a fingerprint depends on the microdata set in which it is contained. The idea is that records with “many short” fingerprints are “risky” and should not be published. Of course, when a fingerprint is “short” and when a record contains “many” fingerprints is a matter for the data protector to decide. We consider these as parameters to be specified. In this way a more exible measure than the “unsafe” definition in the preceding paragraph is provided.

3. As part of the checking procedure of a file prior to its release, a data protector could use the fingerprinting option and identify the records that are risky (according to this criterion) in the data file. On the basis of the fingerprint distribution in the file, he can decide what measures to take, i.e. what variables to recode globally and what values to suppress. We shall not be dealing with this latter process in the present paper, as it is described elsewhere. As is clear from the discussion below the fingerprinting procedure is ideal for incorporation in a general purpose data protection package such as μ -ARGUS (see [5]).

¹ Prepared by Leon Willenborg and Jan Kardaun. The views expressed in this paper are those of the authors and do not necessarily reflect the policies of Statistics Netherlands.

4. The application of this more exible definition of risk of re-identification is *not* in the area of so called *public use files* where a requirement is that *all* records are absolutely safe (no matching, no spontaneous recognition, no succesful fishing), but in the area of microdata sets for research, which are supplied to well-respected research institutions, under a contractual arrangement which includes a non-record-matching obligation among others. Here, records that are unique over all variables are allowed, but the number of variables needed to make a record unique should be sufficient to prevent spontaneous recognition.

5. Below we shall consider the algorithmic problem how to identify the set of “risky” records, i.e. with too many fingerprints, or approximations to this set, in a microdata set. The algorithms we consider derive more or less naturally from a basic, straightforward, algorithm for identifying the fingerprints in a microdata set.

II. THE PROBLEM

6. We have a microdata set M with n_r (individual) records r . Each record (i.e. each observation) consists of key variables, which can be used by an intruder for re-identification, and other variables. We do not consider these “other variables”, except for mentioning here that these need often larger storage space than the key variables. We suppose that the data set is split into two parts, namely a key variable part and the part consisting of the other variables, linked by a further uninformative record identifier (RID). In fact, for our purposes we can assume that our data set M consists of key variables only and a RID , which latter variable we can ignore in our discussion.

7. We want to find the set of risky records in M . We denote this set by $S(M, K, l, n_f)$, in which

- M denotes a microdata file, viewed as a set of records. With the size of M we mean the number
- of records it contains and denote it by $|M|$ or n_r .
- K is the set of key variables in M ; its size is denoted as $|K|$.
- l is an integer that denotes the maximum number of key variables to consider in one record.
- n_f is the minimum number of fingerprints in a record before it is called risky.

8. A fingerprint F is a *minimum* combination of variable *values*, i.e. the subset K_F of K for which the values of K_F make a certain record unique in M^2 . Note that the minimality requirement means that a fingerprint does not contain a shorter fingerprint. So a fingerprint has a unique value in M , and therefore appears in one record of M only. The same subset of *variables* may have other unique values (which appears in another record of M), or non-unique values (appearing in yet other records of M).

9. So the primary aim is to calculate the set $S(M, K, l, n_f)$ or a good approximation thereof. Secondary, if possible, for each record r in $S(M, K, l, n_f)$ we would like to have the (at most) n_f fingerprints F that the algorithm has identified during its search. We shall not require from practically useful algorithms to calculate more than n_f fingerprints, in particular that they calculate *all* fingerprints in a record r . In order to delineate the problem, however, we start with considering how all fingerprints in r can be calculated, before introducing some shortcuts or simplifications.

10. To fix our minds, we will give an example with realistic, numbers and will investigate some of the consequences. Consider a microdata set M containing 40 key variables ($|K| = 40$) and 100,000 records, which would be typical for a Labour Force Survey, for example. We would like to consider at most 10 subsets (n_f) of at most 8 variables (l) in each record that would make this record unique in the data set. If there are more than 10 subsets that make the record unique we call the record (too) risky

² An extension of this concept would be to allow for fingerprints that are not unique, but occur 2, 3 or a few times in M . We shall not consider this conceptual extension in the present paper.

anyhow. There are $\binom{|K|}{1} = \binom{40}{8} = 76,904,685$ combinations to consider of length 8. Often l will be smaller than 8, say at most 3, 4 or 5. This may depend on the number of categories of the combinations of variables to be considered. The fact that we will limit ourselves to 10 fingerprints in a record helps, but we cannot predict how much, as it is dependent upon the actual composition of the data.

11. Even if this example of limited size appears to be hard to solve, we should not despair, as it can be useful to start with smaller examples, where, say, $|K| = 20$ and $l = 6$, yielding $\binom{20}{6} = 28,760$ combinations of length 6. These can be calculated easily on present day computers. Such small instances will give us insight into the behaviour and application of the fingerprint approach.

12. On the other hand we do not seriously have to consider data sets that have much larger values of $|K|$. If we have more key variables most records will be unique. As many records will be unique on a subset of 10 or so variables, this approach has reached the limits of its usefulness³.

13. Assume for a moment that the key variables have on (a rather conservative) average three different values. This conservatism is necessary for the negligence of correlations between some variables. Then with $K = 40$, we have 10^{19} possible cells, and for $K = 50$ this is: 10^{24} . Even if we have, not to be skimpy, 10^{10} observations (the World Census, for instance), still our “cells” will be filled quite sparsely. In such a situation (i.e. for large $|K|$ ’s) it is not practically meaningful to know that a record has a fingerprint for a combination of at most 10 variables, say. It is clear that the maximum length of the fingerprints to consider depends on the number of possible values that one can expect: if this is too big almost any record will be unique, whereas if it is too small almost no fingerprint will exist in a file M . So, generally speaking, fingerprinting is of interest in case fingerprints have small to medium sized lengths.

III. ALGORITHMIC ASPECTS

14. In this section we consider various algorithms to calculate the set of risky records in a microdata set. There are two basic approaches to calculate fingerprints, namely the bottom-up (or expansion) method and the top-down (or reduction) approach. In the former the idea is to start looking at short fingerprints and increase their size, whereas in the latter one start looking at the longer one and then descent to shorter ones by aggregation. In the present section we discuss various modules that are needed in calculations in at least one of these approaches.

III.1 Combination generator

15. At the heart of a fingerprinting algorithm lies a generator of combinations of variables (subsets of K), which we provisionally will consider one by one. We have the choice of generating these subsets starting with small subsets and *expanding* or starting with the maximum subsets and *reducing*. If we search exhaustively, the number of combinations considered is the same for expansion or reduction (i.e. bottom-up and top-down, respectively), but if we make shortcuts there are differences. Furthermore, there is a difference between expansion and reduction with regard to the information that we have available from the previous step. Especially, when reducing, we have all information at hand to generate the smaller subsets, without the need to read the full dataset again.

³ We do not have empirical data yet to verify whether the limits of this approach are sooner reached by constraints in computer resources or by the interpretation of its results for daily statistical disclosure control practice.

16. We can also choose the sequence of the variables for inclusion in the subsets. It is intuitively clear that combinations of variables which have a large number of possible combinations as values are more likely to yield unique combinations (in the file) than in case this number is not so large. This heuristic can be used to arrive more quickly at the limit n_f .

III.2 Aggregation

17. Another basic ingredient is an *aggregation* routine. Aggregation — sometimes called tabulation, collapsing or redesigning (a table) — of a data set (or previous aggregate) over a set of variables S_v , is counting the number of occurrences (f) of records with the same value for these variables. This gives information whether a record is unique for S_v . Information in other variables (those not in S_v) is usually lost, but in the process of aggregation some other information may be added, e.g. a *RID* which has only meaning if $f = 1$.⁴ The actual process of aggregation can be implemented in different ways, e.g. (full sized, rectangular) arrays, sparse arrays, hash tables, trees, or by sorting (cf. [1]). The preference depends upon a number of parameters, the most unpredictable of which is the distribution of the higher dimensional data. Other important factors are the size of the data set, available memory and the behaviour of virtual memory managers as compared to explicit disk I/O.

III.3 Storing the results

18. We need also a routine to store and keep the relevant results for us, while traversing the search space. More specifically, we need:

- **Per record:** The number of fingerprints that are contained in a record, and which fingerprints actually appear in the record. This is up to the threshold n_f . If the number of fingerprints turns out to be larger than the threshold, one only gets a subset of these fingerprints.
- **Per variable:** Which combinations with other variables are “particularly” rare, i.e. occur less than would be expected from the marginal distributions.⁵ This is a help for global recoding decisions, and also for gaining experience about what variables are problematic to combine. If the search space is not completely traversed, this information will be incomplete, but may still be useful.
- **Per table:** Which records have become unique (when using the expansion approach) or have become non-unique (when reducing), and the fraction of cells and/or records that have become so. The latter is of use for heuristically not following a branch of the search space if a variable with many values is in- or excluded.

III.4 I/O and memory considerations

19. Whatever approach we take, we have to “read the data” quite often, ten thousand or a million times. As disk I/O is relatively slow, we will consider here for a moment how far we can come if we want to keep all data in memory. Assume that a machine with 1 GB internal memory is available (this amount of memory costs currently approximately ECU 1500). Our example data set consists of 100,000 records, 40 key variables, i.e. 10 MB say. Even if the other required information per record (*RID*, n_f times K_F) would multiply this by a factor of 10, we still have sufficient memory left (90%) to store temporarily quite a number of intermediate aggregates. The aggregates that are of interest (not too many unique records) are much smaller. If an aggregate is larger than 10%, say, of the original single-record data set, we know that almost certainly there are so many risky records, that we do not have to consider the single records. So, a typical size of an aggregate of interest is 1 MB or much less. There is space for hundreds if not thousands of those. We conclude that it is worthwhile to concentrate on an

⁴ Or up to n *RID*'s which have only meaning if $f \leq n$.

⁵ As a side effect, this information is also valuable for analysis of the data set for other purposes than statistical disclosure control.

approach that does only disk I/O at the beginning and at the end of the program. This is also relevant for considerations of parallelisation (cf. [3, 4]). If there is no intermediate I/O, it is much simpler to distribute part of the search space over multiple (in the order of several tens of) machines, increasing both available CPU power and memory. The data and a description of a partial search space have to be sent out to all participating machines, and the results have to be collected. Both the reduction style and the expansion style approach are equally suited for parallelisation: they traverse the search space along a tree structure, which can be independently done for each (sub-) branch. Paradoxically, it is the heuristics and short-cuts that may hamper parallelisation, unless we confine ourselves to short-cuts that are *local* to a branch.

If the size of *all* objects to be kept in memory *at one time* can be known in advance (or closely estimated), then the theory of bin-packing (cf. [2]) can be applied to cram as many objects in memory as possible for a run through the microdata file. For most objects the exact size can either hardly be predicted, or only a wasteful upper bound can be calculated⁶. Thus the savings of bin-packing might as well be limited, and an approach with a language and operating system with good dynamic memory management, garbage collection and virtual memory management might as well be as efficient. In that case one does not have to bother about it and can one leave this memory management problem to the operating system.

III.5 Heuristics and shortcuts

20. In order to avoid traversing the full, vast search space, it will be necessary, except for small (in terms of K) data sets, to skip parts that seem to be uninteresting, even if we take a small chance of skipping a relevant part of the search space. It is best to try to gain insight in the behaviour of the heuristics with *several* small data sets that can be searched exhaustively.

21. In general, either *variables* or *records* can be excluded for further processing, if we have already sufficient information in an early stage that these variables (or better variable combination, say region with religion) gives too much fingerprints. We have to admit that now in fact we reduce our search task to a *different* data set.

22. Also, we can exclude *aggregates* from further processing, if we have seen that there are too many cells with fingerprints (10 or 30%, for instance). Now we are not reducing the search task to a different data set, but our knowledge will be reduced as we have “white spots” in the results. Checking whether an aggregate has more than a certain fraction of risky records can be done by examining (more or less) randomly 100 cells or so⁷. Omitting aggregates from further processing not only saves “run time”, but also the memory space to store the results (which is particularly in such a case extensive). As a price, we have to think of a notation and display of the fact that this part of the search space was considered “barren land”.

23. If we know that a certain set of variables gives many fingerprints, then we there is no need to examine supersets of this set. This seems to be a valid approach, both if the work is done on a per record basis, or on a per aggregate basis. In the latter case we must be more tolerant before rejecting a superset: the threshold for “too many fingerprints” in a table is less precise than in a single record.

24. If we process per record, then we can stop if n_f is reached. If we work per table, we use this limit only for saving storage space of the results.

⁶ Unless data structures such as arrays are used to store the tables, where the size of each table can be calculated from the categories of each of the variables spanning a table, although this may be wasteful. For other data structures this might be less straightforward, if not impossible.

⁷ Granted, this is an optimisation within an optimisation, but with some of the data structures that were mentioned above, we get this one “for free”.

III.6 Presentation of selected, relevant results

25. The results of our search for “risky records” are intended for “human consumption”, for making recoding or suppression decisions. This means that the results must be presented in an easy to grasp way, and without too much detail. The last constraint is especially challenging, as we can expect “oodles of detailed results”. What level of detail we can show, depends upon the actual outcome.

26. If we have only a few (less than a few hundred) risky records, we can show the records (perhaps only the key variables), ordered by number of variables in the fingerprint, and within by variable that has most risky records. Optionally, ordered by user-specified variable sequence.

27. If we have more risky records, the single records are likely not (yet) interesting. For several orderings of the variable-sets involved, it is sufficient to present the number or fraction of risky records. In a graphical display, the variable-sets involved may be represented each as a bitmap = pixelmap, adjacent to a field which color represents this fraction (class). The interactive sorting of the many bitmaps will show some patterns, while some aid from algorithmic preprocessing can be expected.

28. If we have even too many risky records for a display of the above sort, we can look at the data by even a stronger “helicopter view”: for each subset of variables processed, the frequency distribution of identical records can be used to guess whether the ‘tail’ of this distribution is small enough that a limited number of fingerprints can be expected. Only results for these subsets are shown, and the remainder of the results are discarded as too risky on aggregate level, not at record level.

IV. CONCLUSION

29. The paper addressed a number of issues, that we can summarise as follows.

- In larger microdata sets for research, to be supplied under non-matching contract, many records are unique. We need a measure for the “degree of uniqueness”⁸.
- We proposed a more exible definition of risky records, by considering rare combinations instead of unique ones.
- We discussed the selection of risky records for further processing, e.g. to protect the microdata set by global recodings and local suppressions.
- We indicated that the method proposed allows one to study the properties of the variables in combination
- We gave a preliminary discussion of some design criteria for an implementation of the ideas sketched.
- We stipulated some heuristics that might be useful. Whether they are really of practical value has yet to be determined experimentally.

References

- [1] A.V. Aho, J.E. Hopcroft, J.D. Ullman (1983). Data structures and algorithms. *Addison-Wesley*, Reading.
- [2] M. Hofri (1987). Probabilistic analysis of algorithms. *Springer-Verlag*, New York.
- [3] J. J´ aJ´ a (1992) An introduction to parallel algorithms. *Addison-Wesley*, Reading.
- [4] G.C. Fox, R.D. Williams, P.C. Messina (1994). Parallel Computing Works! *Morgan Kaufmann Pub.*

⁸ Some of us are more unique than others!

[5] Leon Willenborg and Anco Hundepool (1999). ARGUS: Software from the SDC project. *This conference*.

[6] Leon Willenborg and Ton de Waal (1996). Statistical disclosure control in practice. Lecture Notes in Statistics, Vol. 111. *Springer-Verlag*, New York.