CONTEXTUAL VARIABLES VIA GEOGRAPHIC SORTING: A MOVING AVERAGES APPROACH

Alan Saalfeld, Laura Zayatz, Erik Hoel, Bureau of the Census*
Alan Saalfeld, Bureau of the Census, Washington, DC 20233

Key Words: microdata, spatial analysis, geographic context.

Abstract

Microdata (data records from individuals) are released by the Bureau of the Census to the public after they have been stripped of geographic and other identifiers. Users in the public would like to know *relative* geographic information about the records to be able to perform spatial analysis on the data. Contextual variables provide the user with local information such as averages of surrounding or neighboring individuals. Contextual variables must be assigned carefully to avoid compromising the exact location of the data records. This paper examines some computationally efficient methods of calculating and assigning useful, but non-compromising, contextual variables, focusing on a do-it-yourself technique based on taking moving averages of a nicely sorted data set.

1 What is Context?

Users of Census Bureau microdata have requested that additional information be computed and placed on the data records to permit the users to perform spatial data analysis. Attaching detailed geographic identifiers to the data records is out of the question because protecting geographic location is a key element of the Bureau's confidentiality strategy. An alternative to disclosing a record's geographic location is to summarize certain values of variables of all of the record's geographic neighbors and attach that summary to the record itself. The summary datum, for example, the average income of all wage earners on the same block, is called a *contextual variable*. The neighborhood used to compute that variable is called the *context*. There are several immediately obvious concerns with creating contextual variables:

1. What are the alternative definitions of neighborhoods of interest (contexts)?
2. When might summary variables implicitly reveal or compromise geographic location?
3. What (new) disclosure protection methods may be applied to contextual variables?
4. How difficult and costly is the computation of contextual variables?

There are two fundamentally different ways to build contexts; and each approach has its advantages and disadvantages.

1.1 Partitions Versus Windows

A *partition* of space is a decomposition of space into mutually exclusive, non-overlapping component pieces. The counties of the US form a partition of the country. The states form another partition. We may sub-divide the country into non-overlapping rectangles (or parts of rectangles along the national boundary) by laying a rectangular grid over the nation to create yet another partition of space (see figure 1). If we partition our space, then all data points belonging to a cell component will have that cell as their context. The contextual variable for all points belonging to the same component will have the identical value. If the contextual variable is the state average, for example, then all Illinois residents will have the same value for that contextual variable. Partition contexts are computationally easier to generate, but they engender special disclosure problems that must be addressed.

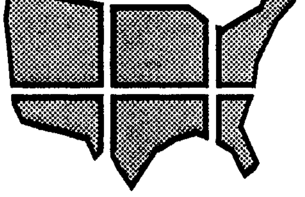
Figure 1: Partitions are Non-Overlapping

*This paper reports the general results of research undertaken by Census Bureau staff. The views expressed are attributable to the authors and do not necessarily reflect those of the Census Bureau.

A *window* is a context that is defined for each data point; and windows may overlap and need not be mutually exclusive (see figure 2). "All neighbors within one mile of a data point" constitutes a window-based neighborhood. Points which are close will have similar, but not necessarily identical, neighborhoods. Window contexts are computationally more complex, but they afford better disclosure protection than partition contexts.

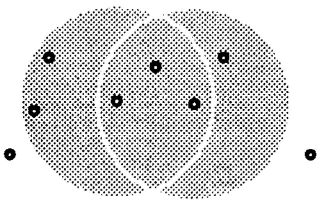

Figure 2: Windows May Overlap

1.2 Standard Geography

Often summary data are presented by common geographic units—counties, census tracts, blocks, congressional districts. A data user may desire that summary contextual variables also correspond to these same units to ensure data comparability. Many of these geographic units are too small to be identified explicitly on Census Bureau microdata files. It would not be acceptable, therefore, to reveal these units inadvertently through a partitioning scheme that permits a user to recover a small unit's identity or, equivalently, its location. A partitioning scheme that uses small but standard geographic units presents too many disclosure risks to be acceptable for building contexts. A partitioning scheme that builds nonstandard geographic regions—for example, a heuristic procedure that groups blocks according to some randomized (or at least non-replicable) method—offers somewhat more protection and is slightly more acceptable.

1.3 Other Partitioning Schemes

A space may be partitioned into uniform regions (gridded) or into regularly shaped regions whose size depends on the density of data points in such a way that each component region contains (approximately) the same number of data points. Two such adaptive partitions are the quadtree and the k-d-tree, shown in figure 3.

Both circumstances offer better disclosure protection than recognizable standard geographic partitions. Every partitioning scheme, however, has two inherent weaknesses. The first is that all elements of

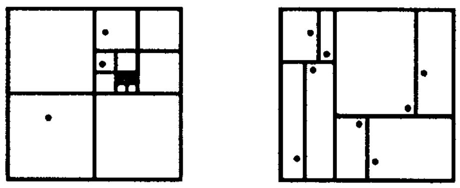

Figure 3: Adaptive Partitions

the same component share identical context values, thus linking all elements of a single component together geographically. The second weakness is called *boundary effects*. Points near to a partition boundary have no effect or influence on points just across the boundary—they belong to a different context by definition.

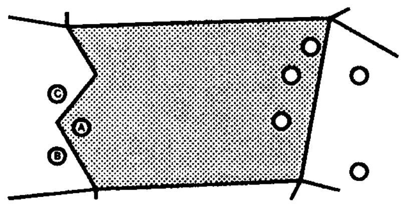

Figure 4: Boundary Effects

The context of point A in figure 4 does not include points B and C, for example. The placement of context boundaries plays an enormous role in the determination of local effects. These effects disappear under windowing methods.

1.4 Sampling and Confidentiality

The Bureau of the Census recognizes that samples enjoy additional measures of confidentiality protection. The uncertainty of inclusion in the sample makes an intruder's activity more difficult, especially when the intruder is looking for a specific population member (who may fail to be in sample!) A unique combination of some characteristics among sample data points (or even a match on those characteristics) by no means isolates an individual from the rest of the total population: there may be several out-of-sample individuals with the same traits; and an intruder has *no access* to the values for the whole population.

Sampling was not instituted to protect data confidentiality, however. Its main benefit lies in greatly reduced cost of surveys coupled with the useful measures of uncertainty that the users have grown to accept and incorporate into their analysis. Information can be compiled quickly from samples; and only a satisfactory level of precision is required for analysis and decision-making. Because users have learned to work with statistical uncertainty, the Census Bureau should not hesitate to introduce similar uncertainty into new census data products, such as contextual variables. Sampling data from partition contexts can eliminate the linking effect of identical context values. Very large subsets can be used as samples to introduce perturbations that are of little significance for analytical procedures, but sufficiently large to reduce the potential for data linkage. Finally we note that sampling from a data file can be accomplished very efficiently—no costly field operations are involved whatsoever.

We are suggesting the use of sampling in this section as a deliberate, independent, and final operation in the series of steps to generate a context from which to compute contextual variables. We will see in section 5 that a kind of sampling may occur naturally at other stages in our context-building operations as we try to increase their efficiency. We will nonetheless try to exploit the disclosure protection effects of this embedded sampling to the fullest.

2 Computational Issues

We assume that contextual variables will be computed by building contexts. In other words, we must successively create groups of records that form the neighborhoods in question. The number of different neighborhoods and the efficiency with which we can move from one neighborhood to another will determine the complexity of our context-building strategy. For example, to build a state-level context set, we merely create 50 "buffers" in which to store records. Then we read through our microdata file *once*, copying each record into the buffer that corresponds to its state. When we are finished, we have a subfile for each context, and we compute summary variables for each subfile. We could also have been carrying out 50 accumulation processes (one for each state) as we read the microdata records the first time; or we can do that as soon as each context is built. For efficiency, we would like to attach the contextual variable to the record while we are processing its context. This is generally straightforward since each record belongs to its context (for the most intuitive definitions of context, anyway).

We illustrate the importance of efficient data structures and algorithms by analyzing the problem of computing contexts in one dimension (the data are points on a line). Our illustration highlights some techniques for building contexts as well as revealing the enormous difference in complexity between the 1-D and 2-D problem formulations.

3 Contexts in 1-D

Suppose our data points reside on a line instead of on a two-dimensional surface. The mathematical analysis of neighborliness is extremely simplified: there are only two directions in which to search instead of infinitely many. We can even *compute* the contents of *all* possible windows of a fixed size in time proportional to the number of data points in our set (once our data points have been sorted along the line). We do this by maintaining a right-sliding interval of the fixed size (keeping track of the interval's end points). Our sliding interval can only add points by adding on the next data point to the right of the active interval's rightmost point and can only lose points by removing the current leftmost data point within the sliding interval. If we consider the *events* that can change the contents of our right-sliding interval, they only occur when the left endpoint bumps into a point already in the interval, (in which case, we delete that point), or when the right endpoint of the moving interval bumps into a new point outside the interval, (in which case, we add the point to our interval). This simple analysis also proves that for a given set of n points on the line, there can be at most $2n$ distinct sets of contents of the infinite number of possible intervals of a fixed size—contents can only change when there is an endpoint "bumping" event; and there can be at most $2n$ such events. It is not difficult to construct examples that achieve this upper limit of $2n$ different neighborhoods or contexts. We suggest that the reader try this as an exercise.

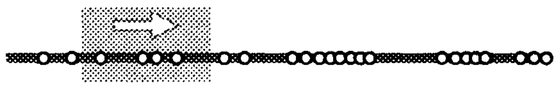

Figure 5: Sliding Interval

An alternative to constructing windows of a fixed size is to construct windows whose size depends on data density. An example of such a data-driven context is the window consisting of the k nearest neighbors. In 2-D (with the usual Euclidean metric), the k-nearest-neighbors context would still be a round window, but the window's size would vary with the density of the data points. In 1-D, the k nearest neighbors fill out an interval, but the interval's width varies with density. It is interesting to note that in 2-D, the strategies for attacking the fixed-size-window and the k-nearest-neighbors problems are quite distinct; and the k-nearest-neighbors problem is harder (computationally more complex). In the 1-D case, the same approach that we outlined above becomes even easier to implement for the k-nearest-neighbors problem—since the neighborhoods all contain the same number k of points, and since they fill out whole intervals, there can only be n-k+1 such contexts. An algorithm to find all such contexts begins with the left-most k points, then successively adds a point to the right while removing one from the left.

Various windowing algorithms in 1-D have linear-time worst-case complexity (after sorting the data—and the complexity of sorting is $O(n \log n)$). Moreover, we can compute different sized contexts at the same time by sweeping through the data just once (and updating different variables for accumulating the values corresponding to the different sized contexts). We cannot do this simultaneous computation in 2-D. Unfortunately, the easy windowing algorithms in 1-D do not generalize to 2-D. No linear-time complexity windowing algorithms are known for 2-D. We can do fixed-size window context computation in expected-time $O(n\sqrt{n})$. The best known algorithms for solving the all-k-nearest-neighbors problem (including the building of a complex data structure known as the kth-order Voronoi diagram) run in $O(k(n-k)(\sqrt{n}\log n + k))$ worst-case time (Edelsbrunner, 1987). Our 1-D techniques, however, can find some application to 2-D data **if we can compress those 2-D data down to 1-D in some way**. And we can, but not without some information loss.

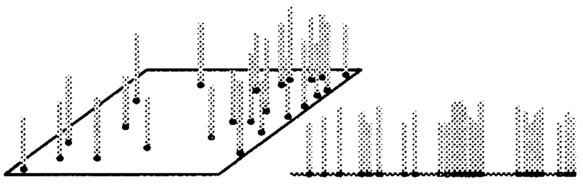

Figure 6: Mapping 2-D data to 1-D

In the next section, we discuss the extent of the information loss; and in section 5 we suggest how we might actually use that loss to our advantage in terms of improving data disclosure protection.

4 Mappings from 2-D to 1-D

Mathematicians have known about bijective continuous mappings or functions from a line (or line segment) to a plane (or rectangle) for over 100 years. They know that the inverse of such a mapping can never be continuous—nearness in the plane can never translate perfectly into nearness on any 1-dimensional representation of the plane. Computer scientists have expanded the mathematicians' research in their pragmatic development of spatial databases—the problem of storing spatially referenced (planar) data in sequential (linear) files to facilitate proximity queries can be solved by constructing a (preferably continuous) bijection Φ from 1-D to 2-D such that for every query set U in 2-D, there is a small, easily computable collection of intervals in 1-D which covers $\Phi^{-1}(U)$.

Two examples of such bijections, shown in their discretized versions in figure 7, are the Peano key (shown on the right), used by the Census Bureau to provide locational access to its 19 gigabyte digital cartographic database known as the TIGER System, and the Hilbert curve, favored by some researchers at the University of Maryland (Faloutsos, 1989) because of its continuity, which is lacking in the Peano key.

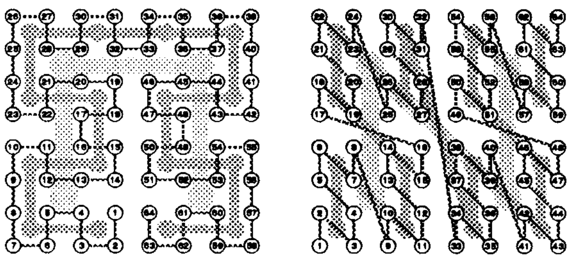

Figure 7: Hilbert Curve and Peano Key

Both the Hilbert curve and the Peano key are quadrant-recursive (Mark, 1990), which permits range queries to be covered by at most 4 interval accesses to the linear representation of the spatial data. These so-called *space filling* curves fill up space one quadrant at a time; they are recursive because, within any quadrant, they fill up space one subquadrant at a time, and within any subquadrant, they fill things up one subsubquadrant at a time, etc. They differ only in the order in which the choose the subquadrants to fill.

The Hilbert curve, because it is continuous, will always move into an adjacent quadrant or subquadrant after exhausting a quadrant or subquadrant. If we compare the space filled by an interval of the curve with the smallest square neighborhood containing that space, we see that the curve always fills at least 50% of the square. This value will be important later when we try to regard the curve as *sampling* points from a neighborhood.

4.1 Ordering Data for Sampling

In addition to the computer science techniques for ordering spatial data, the Census Bureau has examined some new procedures for numbering or ordering spatial data to facilitate systematic sampling (Saalfeld, 1991). One new procedure guarantees nearness, but not necessarily contiguity, for consecutively numbered data points. Nearness, it turns out, is sufficient to assure reduced travel costs for cluster samples extracted from the linearly-ordered list. We observed earlier that nearness cannot be preserved by both a bijection from 1-D to 2-D and its inverse. The new ordering methods can generate (by taking a consecutively numbered interval of points) a non-probability sample of neighboring points whose expected sampling fraction may be estimated. We describe and illustrate here briefly the recently developed numbering scheme for points in the plane. Details of its implementation may be found in (Saalfeld, 1991). The steps of the algorithm are depicted in figures 8 through 12:

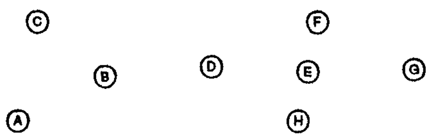

Figure 8: Start with $n(=8)$ labeled points

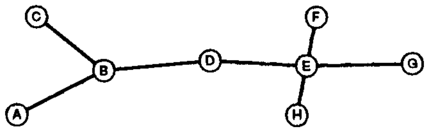

Figure 9: Build an EMST connecting the points

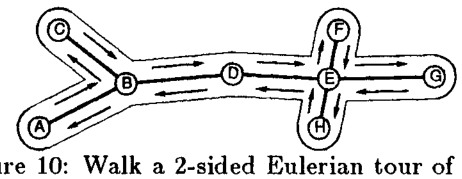

Figure 10: Walk a 2-sided Eulerian tour of the tree

Point Ordering Algorithm

Input: `n labeled points in the plane.`

Output: `A cyclic ordering of the n points.`

```
I. Build the Euclidean Minimum Spanning Tree
   of the points.
II. Tree-order the vertices of the EMST:
    1. Walk a 2-sided Eulerian tour of the
       EMST.
    2. Build a uniform weight circular
       sampling interval of labeled
       subintervals.
    3. Select points from the sampling
       interval, recording the labels of
       containing subintervals.
```

In order to evaluate the relative density of points in the plane that correspond to an interval on the linear order (and thereby assess the sampling fraction that the interval represents), we may make use of the following properties of a minimum spanning tree.

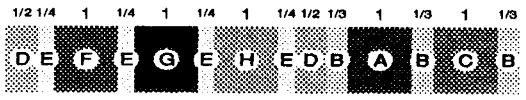

Figure 11: Order weighted vertex visits

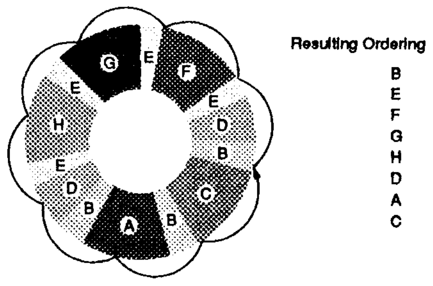

Figure 12: Select a cyclic ordering

1. A Euclidean minimum spanning tree is a subset of the Delaunay triangulation of the data points, which in turn is the dual graph of the Voronoi diagram for the points.

2. The Voronoi diagram breaks the plane into *regions of influence* consisting of all points which are closer to the data point that determines the particular cell than to any other data point. The dual graph, the Delaunay triangulation, simply describes the adjacency relations of regions. The EMST, a subgraph of the Delaunay triangulation, captures only *some* of the adjacency relations. A data point may be weighted by the size of its Voronoi region, since that is the region of influence of the data point.

3. As we suggested doing with Hilbert curves, one may compare the area of Voronoi regions of a consecutive sequence of data points to the size of the smallest circle containing those regions to obtain a measure of the sampling fraction defined by the interval.

5 Sampling Effects

We have been trying to assess how 1-D to 2-D mappings cover space in order to treat that embedding as a sample. All of our mappings have been deterministic without any random element or randomization procedure. There are several opportunities for randomizing that correspond approximately to the notion of a random start in a list under systematic sampling. As with systematic sampling, our random start concept does not eliminate other theoretical shortcomings of the procedure itself (Kish, 1965). Nevertheless, insofar as systematic sampling is used widely and successfully in spite of unverifiable assumptions about randomness and independence, the following steps should elevate our 1-D ordering procedures to a similar kind of random sampling.

5.1 Hilbert Curve Orderings

We may randomize the coordinate axes upon which the Hilbert curve is split. If the origin is placed randomly in the plane of the data set; and if the coordinate axes are rotated in a random fashion, then if two quadrants are selected by an interval in the linear order, they can be any adjacent quadrants with any orientation.

5.2 Tree Orderings

The new orderings for sample selection described in section 4.1 depend on linking all of the data points in a tree (a minimal connected graph). There are many ways to introduce randomization to build a tree on those points, even a tree that is a subgraph of the Delaunay triangulation. Such a measure would, in effect, randomize the choice of which truly neighboring point to visit on the Eulerian tour.

In this instance, as in the section on randomizing the Hilbert curve, the randomization is done at the start of the process; and every step thereafter is completely deterministic. A linear order is established; and the computation of moving averages gives contextual variables whose interpretation as a sampled set will depend on the linear order used.

6 Conclusions

For publishing contextual variables, we recommend that an appropriate 2-D to 1-D mapping be selected, a 1-D windowing context be computed, and an analysis of variance of the 2-D-to-1-D-mapping's sampling properties be conducted to permit the user to obtain or compute measures of uncertainty of the published contextual variables.

We base our recommendations on the following observations:

1. Partitioning according to standard geography is unacceptable from a disclosure standpoint.

2. Any partitioning scheme will have data grouping effects and boundary effects.

3. Windowing in 2-D is computationally complex and costly.

4. Windowing (even simultaneous computation of different sized windows) in 1-D is straightforward, but not directly extensible to 2-D.

5. Mappings from 2-D to 1-D lose information, but that loss may be made systematic and sampling effects may be exploited.

6. Sampled contexts can protect confidentiality and give the user a mathematically quantifiable measure of uncertainty.

7 References

1. Aho, Alfred, John Hopcroft, and Jeffrey Ullman, 1985, **Data Structures and Algorithms**, Addison-Wesley, Reading, MA.

2. Edelsbrunner, Herbert, 1987, **Algorithms in Combinatorial Geometry**, Springer-Verlag, New York, NY.

3. Faloutsos, Christos and Yi Rong, 1989, *Spatial Access Methods Using Fractals: Algorithms and Performance Evaluation*, University of Maryland Computer Science Technical Report Series, UMIACS-TR-89-31, CS-TR-2214.

4. Kish, Leslie, 1965, **Survey Sampling**, John Wiley, New York.

5. Mark, David M., 1990, *Neighbor-based Properties of Some Orderings of Two-Dimensional Space*, **Geographical Analysis**, April, 22(2), 145-157.

6. Saalfeld, Alan, 1991, *New Proximity-Preserving Orderings for Spatial Data*, **Proceedings of the Tenth International Symposium on Computer-Assisted Cartography (Auto-Carto 10)**, ACSM/ASPRS, Baltimore, MD, 59-76.

7. Wolter, Kirk, and Rachel Harter, 1989, *Sample Maintenance Based on Peano Keys*, presented at Statistics Canada Symposium on Analysis of Data in Time, Ottawa, Canada.