Privacy in spatial data with high resolution and time invariance

Andreas Bjerre-Nielsen & Mikkel Høst Gandil

August 17, 2018

Abstract

This paper presents a graph-based algorithm to construct partitions of data which preserve k-anonymity while maximizing precision. We present the algorithm and apply it to the geographical population distribution of Denmark. The algorithm provides a time-stable geographic partition with seven times larger precision than the smallest partition available so far. All software is made available free of charge on the authors' websites.

1 Introduction

Researchers working with data on individuals or organizations often work under the constraint of having anonymized entities. The anonymization works by hashing names, social security numbers etc.. One major problem often faced by researchers is not having access to the exact address and home location of individuals in the data.[1] The lack of access to individual location implies that researchers have to resolve to crude measures e.g. administrative boundaries. These measures often suffer from lack of precision and instability over time. This prohibits computation of geographical patterns and meaningful interpretation of the effects of nearby amenities.

[1]The reason for lack of access to such information is that it can be used to reverse engineer the identity of an individual by combining searches on local address and phone directories and/or social media.

Much of the existing literature focuses on making computational systems that can anonymize user location data over time such as continual GPS data from smartphones. See Gedik & Liu (2004), Mokbel et al. (2006) for examples of such approaches. In this paper, we present an algorithm, which creates a mapping from individual location to an approximate location shared by at least k other individuals. This aim is shared by earlier work on anonymization but our approach differs by considering two additional constraints. The first extra condition is that the mapping of location to approximate location is constant over time. The second is that instead of using exact individual locations as input it takes an approximate location corresponding to squares in a grid.[2]

Our contribution is to provide a procedure that makes spatial clusters that are stable over time and contain a minimum number of inhabitants over time according to a local optimization. The local optimization ensures high precision while respecting privacy. We measure location privacy at the *k-anonymity* level, see Samarati & Sweeney (1998). This implies that the location of one individual is the same as for at least k-1 other individuals.

This paper is structured the following way:

- We provide a method that computes a locally optimized procedure that produces k-anonymous partitions using a pre-existing set of polygons. This method works also for requiring k-anonymity across multiple points in time.

- We apply this method in the context of Danish data in the context of the Danish Squarenet. This data is a division of Denmark into a grid of equally sized squares of a certain length. We focus on the setting where cells measure 100m by 100m which has the cartographical reference "DKN_100m_euref89".[3] We partition this grid into polygons which are collections of cells such that each collection polygon has at least 100 inhabitants each year it is inhabited.

- We illustrate the results with a handful Danish municipalities.

[2]This condition is necessary for working with Danish registry data as the number of people living at a given address is too sensitive for researchers to access.

[3]`http://www.dst.dk/da/TilSalg/produkter/geodata/kvadratnet`

One advantage of using the Danish Squarenet (Det Danske Kvadratnet) is that it is based on a precise metric coordinate system and is supported by various governmental organizations and can be merged unto other geodata sources such as Open Streetmaps. Moreover, the grid is stable over time and exogenous with respect to political and deistic partitions (such as municipalities and parishes) alleviating fears of endogeneity of geographic subdivisions. The choice of 100 k-anonymity for each year was chosen by Statistics Denmark to ensure a modest level of privacy given that researchers also have access to multiple other identifiers. Given this constraint, our approach improves our measure of spatial precision on average from 3700m for parishes to 510m; moreover at the median our measure of spatial precision is reduced to 300m from 3500m.[4]

The main thrust of the problem is that the constraint of a minimum number of inhabitants creates a trade-off between precision and inclusion. Larger polygons that may include more people, and thus make a larger share of the population available for analysis. However, geographical precision is a decreasing function of polygon size. In other words, the goal is to have as *many polygons* with as *few people* in them while upholding the restriction of *at least 100 people*are in each polygon, i.e. group of squares.

This paper documents our approach to this problem. The algorithm is implemented in Python and the associated code is available in this public git repository and distributed under MIT licensing. The algorithm presented here is simplified to clarify the main logic. In practice, a number of steps is taken to simplify the "geography" of the data. We refer to the code repository for these procedures.

This paper proceeds as follows. We begin in Section 2 by describing the general problem and the metrics, we use for diagnosis. We then proceed in Section 3 to describe the algorithm before giving an example of the result from running the algorithm. Section 6 concludes.

[4]We measure spatial precision as the square root of the area in the convex hull of the shape - see Section 4.

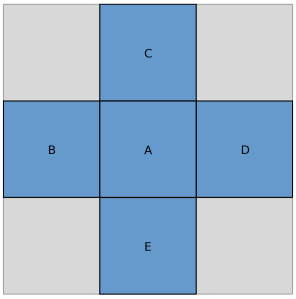


Figure 1: Possible neighbors to A

2 General approach

In this section, we present the problem and the logic of our algorithm. The basic data consists of a list of square cells. Each square is associated with a vector where each entry contains the population in a given year. The goal is to connect these squares to fulfill a restriction on minimum of the sum of population vectors. We restrict the possible connection by requiring that two squares must share a border in order to be connected as illustrated in figure 1. Thus each square has four possible connections (if figure 1 A can connect to B,C,D,E).

For a *given year* there are a number of different combinations. For simplicity, we restrict the example to a 3×3-square. An example of different feasible partitions for a given population distribution is illustrated in figure 2. There are many possible combinations and we need a method to rate these partitions against each other. Here we face a trade-off between inclusion of a large share of the population and geographical precision for those individuals included.

To be precise we introduce the following notation. Define the set of squares $\mathcal{S}$, indexed by j. Each square is defined by a coordinate of the lower left corner, (x_j, y_j), and a population n_j.[5] Let partition $\mathcal{P} \in \mathcal{S}$ be a set of non-overlapping polygons, indexed by p containing a number of squares.

As a measure of inclusion, we simply take the share of the population

[5]In this example we only have one year of analysis. If we need to consider the k-anonymity restriction in multiple years n_j would be a vector.

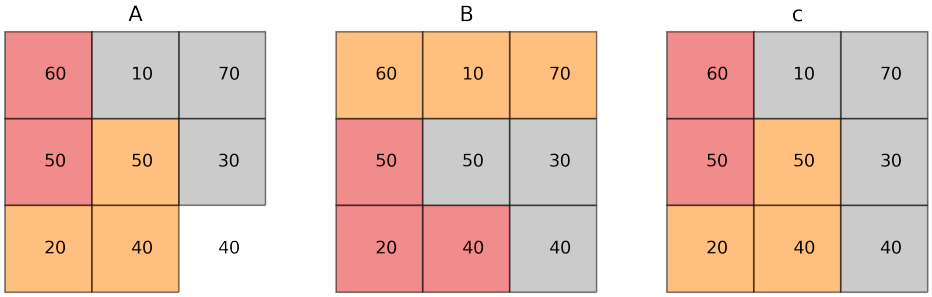


Figure 2: Examples of partitions

not included. We thus define:

$$non\_pop_{\mathcal{P}} = 1 - \frac{\sum_{p \in \mathcal{P}} \sum_{j \in p} n_j}{\sum_{j \in \mathcal{S}} n_j}, \tag{1}$$

where p is a polygon in partition $\mathcal{P}$, $p \in \mathcal{P}$. The polygon can itself be thought of as a set of geographically connected squares. As a measure of precision in a polygon, we use the bounding box of the polygon and take the length of the diagonal.

$$dist_p = \sqrt{[max_{j \in p}(x_j) - min_{j \in p}(x_j) + 1]^2 + [max_{j \in p}(y_j) - min_{j \in p}(y_j) + 1]^2} \tag{2}$$

When evaluating the precision of a partition $\mathcal{P}$ we calculate the weighted distance, simply expressed by:

$$weighted\_dist_{\mathcal{P}} = \frac{\sum_{p \in \mathcal{P}} \sum_{j \in p} dist_p \times n_j}{\sum_{j \in \mathcal{P}} n_j} \tag{3}$$

To evaluate the trade-off we define a simple linear loss-function:

$$L(\mathcal{P}) = -non\_pop_{\mathcal{P}} - \beta \times weighted\_dist_{\mathcal{P}}, \tag{4}$$

for some weight β. In table 1 it can be seen that with these diagnostics one should prefer partition B. One can see that partition C does cover the

P	dist	weighted_dist	pop_weight	loss
A	7.89	2.35	0.89	-0.1316
B	8.82	2.95	1.00	-0.0295
C	8.67	2.97	1.00	-0.0297

Table 1: Example of diagnostics of partitions in figure 2, $\beta = 0.99$

entire population it is punished by the relatively large distance in the gray group compared to partition B where the polygons are "more convex".

3 Method

In this section, we go into more detail about how data has been processed and how the algorithm works. For further details see the Python code available in our git repository.[6] The assignment of spatial cells to a collection consists of three overall stages:

1. Pre-processing the data
2. Application of the algorithm
3. Post-processing by merging the best partition for sub-areas into one large coherent partition

We move to briefly describe the stages and put an emphasis on the main algorithm.

3.1 Preprocessing

The first stage consists of processing data which is repeatedly used in the computation. A first stage is to split Denmark into sub-areas. Application of the algorithm below is time-consuming and it runs slowly, especially on larger areas. In order to speed up the process we to divide Denmark into sub-areas. We employ the (current) Danish municipalities as our basis for this division. An issue with using municipality

[6] `https://github.com/abjer/privacy_spatial`. For the practical implementation of our code we have relied on various open source projects, especially Blondel et al. (2008), Csardi & Nepusz (2006), McKinney (2011), Schult & Swart (2008), Van Der Walt et al. (2011)

data is that there is a big difference between the smallest municipality (Frederiksberg) and some of the larger rural ones. Therefore we begin our analysis by splitting the large municipalities (>25 square kilometers) into smaller chunks. This is done by expressing the square net cells as a network graph where links/edges exist between cells within a certain distance. This representation allows us to use the Louvain method, see Blondel et al. (2008), to make a pre-division of larger municipalities into smaller and manageable chunks. In other words, we construct connected components of cells and run the algorithm within these components.

The next pre-processing stage concerns identifying newly constructed dwellings. For the requirement by Statistics Denmark, to have at least 100 people in a populated polygon, to be fulfilled every year adds complexity. When cells in a polygon exhibit large variation in the population, this can become a problem. Especially for new construction or data-breaks where each cell can go from having no population to a possibly large population. When coupling a group of cells that contain both cells with new construction and cells without it becomes difficult to satisfy the population constraint in the initial years while maintaining high precision. To overcome this, we identify for each cell if there is a break of going from no population to some population. We then apply our method below to cells with such breaks separately. This greatly improves our precision in the resulting partition of the country.

3.2 Basic algorithm

Naturally, the possible valid partitions explode with the number of cells to consider and it is not feasible to check them all to achieve optimality. Instead, the basis algorithm runs from a random starting point. Prior to the run of the algorithm, a network graph is constructed wherein feasible neighbor cells are represented by edges.

Every population vector associated with each cell, $\mathbf{n}_i$, has the same length and may contain zeros. Thus on the outset, we have the following data available:

- $\mathcal{S}$: Set of squares, where each square, i is defined by the tupple $(x_i, y_i, \mathbf{n}_i)$.

- $\mathcal{G}$: A graph with information on which squares are neighbors.

Furthermore, we define a number of variables:

- R: an integer, the maximum attempts an algorithm should perform before exiting.
- r: number of attempts, initially set to 0.
- $\mathcal{U}$: A set of unassigned squares, initially equal to $\mathcal{S}$.

The basic algorithm is displayed in Algorithm 1. When all squares are assigned or the sum of the remaining squares is less than 100 the algorithm stops.

3.3 Application of algorithm

We apply our algorithm for each of the sub-areas described above. Each application of the algorithm consists of multiple stages. The first step is to apply the algorithm to year breaks caused by construction (see preprocessing above). If there are more than enough inhabitants collectively, these are added to the partition. The second is to apply the algorithm to all the squares of the sub-area, which are not assigned in the first stage (with year breaks). Finally, we apply an additional algorithm, that check for 1:1 feasible exchanges of squares between neighboring polygons. The condition for an exchange is that it improves the distance between the polygon cells without violating the feasibility constraints.

Upon termination, we compute various measures for the partition and these are stored together with the partition. The algorithm runs multiple times for each sub-area and we pick the partition with the numerically smallest value of our loss function defined in (4). The definition of $\mathcal{S}$ is important for the amount of time the algorithm takes to finish. Densely populated areas are partitioned quickly, but in geographically large and sparsely populated areas, the algorithm will take a very long time to complete as the algorithm does not scale well. Therefore, it is essential to split the spatial areas where the algorithm is executed into smaller parts.

Data: $\mathcal{S}, \mathcal{G}, \mathcal{U}, r, R, \mathcal{P} = \emptyset$
Result: $\mathcal{P}$
while $(r < R)\,\&\,(\min_{\mathcal{A}}(\sum_{l\in\mathcal{U}} \boldsymbol{n}_l) \geq 100)$ **do**
  Pick random square from $\mathcal{U}$ and define $p = \{i\}$;
  if $\min \boldsymbol{n}_i \geq 100$ **then**
    $\mathcal{P} \leftarrow \mathcal{P} \cup p$;
    $\mathcal{U} \leftarrow \mathcal{U} \backslash p$
  else
    Define $\mathcal{F}_p$ as components in $\mathcal{G}$ connected to the squares already in p;
    if $\mathcal{F}_p = \emptyset$ **then**
      $r \leftarrow r + 1$;
      Return to start
    else
      while $\left(\min_{\mathcal{A}}\left(\sum_{l\in p} \boldsymbol{n}_l\right) < 100\right)\,\&\,(r < R)$ **do**
        Pick random neighbor j from $\mathcal{F}_p$;
        $p \leftarrow p \cup \{j\}$;
        Update $\mathcal{F}_p$;
        if $\mathcal{F}_p = \emptyset$ **then**
          if $\min_{\mathcal{A}}\left(\sum_{l\in p} \boldsymbol{n}_l\right) \geq 100$ **then**
            $\mathcal{P} \leftarrow \mathcal{P} \cup p$;
            $\mathcal{U} \leftarrow \mathcal{U} \backslash p$;
          else
            $r \leftarrow r + 1$;
          end
          Return to start
        end
      end
      $\mathcal{P} \leftarrow \mathcal{P} \cup p$;
      $\mathcal{U} \leftarrow \mathcal{U} \backslash p$
    end
  end
end

Algorithm 1: Basic algorithm

3.4 Finalizing the output

After having executed the algorithm multiple times on all the sub-areas the output is collected. Among the output, the candidate sub-partition with the lowest value of the loss function trading off spatial precision versus missing data is chosen. Finally, the sub-partitions are merged into one partition.

4 Algorithm output

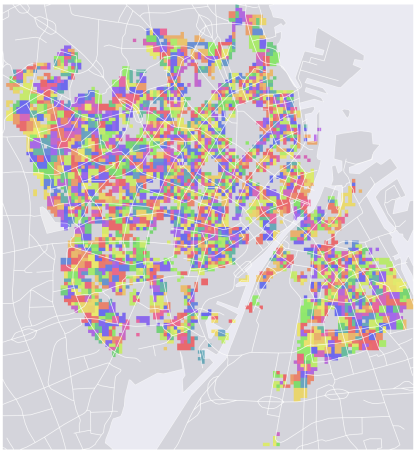
In figure 3 we apply the algorithm to different Danish municipalities. Coloring a map is no simple feature and some polygons will share color while not being joined. Thus, the partition into sub-areas will tend to look slightly worse than it really is.

The strength of the algorithm is evident for densely populated areas as seen in the urban areas in Figure 3a and 3b . It is also evident that in a suburban municipality such as Rudersdal in Figure 3c the algorithm performs well in dense areas, but it branches out once the population density falls and becomes more rural as evident in the North. The branching out, however, becomes more severe as the density falls as seen for the municipality of Vordingborg in Figure 3d. Fortunately, the branching-out covers a relatively small amount of people, and can thus be discarded in the analysis if more precision is called for.

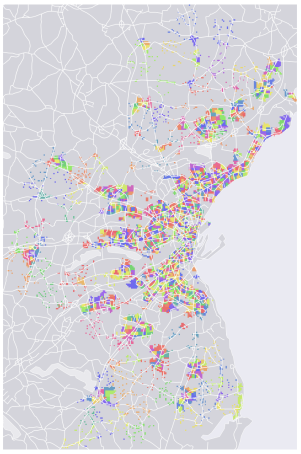
4.1 Descriptive statistics of the partition

Descriptive statistics of a partition of the whole of Denmark is provided in table 2 and Figure 4. The final partition produces more than 30.000 polygons, with a mean of 13 squares in each. It is, however, a right-skewed distribution and the median is thus only 8, as seen in Figure 4. Surprisingly the distribution of areas is bimodal as there is a peak at 1 cell and a peak around 45 cells. This is most likely due to a difference in the structure of large cities and smaller towns.

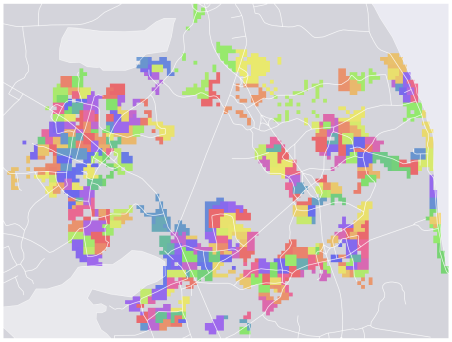
Looking at the scatter-plot between population and area, the scatter plot illustrates the result of the choice of cost function, as there is a



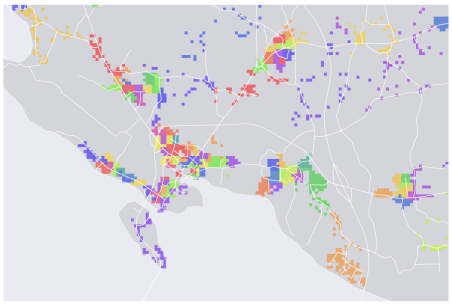
(a) Municipality of Copenhagen



(b) Municipality of Aarhus



(c) Municipality of Rudersdal



(d) Town of Vordingborg

Figure 3: Examples of spatial partitions of suburban and rural municipalities

trade-off between including the entire population and maintaining precision. The second peak in the distribution of area-size is due to polygons with very low populations. This indicates that the algorithm works as intended.

We can compare the output of our algorithm with the current standard practice for working with geographic data, namely to use the administrative boundaries for parishes. These areas divide Denmark into local districts for the Danish National Church (Folkekirken). The comparison is found in Figure 5. Our measure of distance across the areas shows a seven-fold reduction at the mean and eleven-fold at the median when comparing the population distribution of Danish Squarenet partition to

	count	mean	std	min	25%	50%	75%	max
area	30,604	13.84	15.35	1.00	4.00	8.00	17	344
density	30,604	43.01	67.12	0.94	10.56	20.00	39	789
population	30,604	180.85	82.46	100.00	127.00	159.00	208	1,778
convex_share	30,604	0.73	0.31	0.02	0.61	0.84	1	1

Table 2: Descriptive statistic of full partition

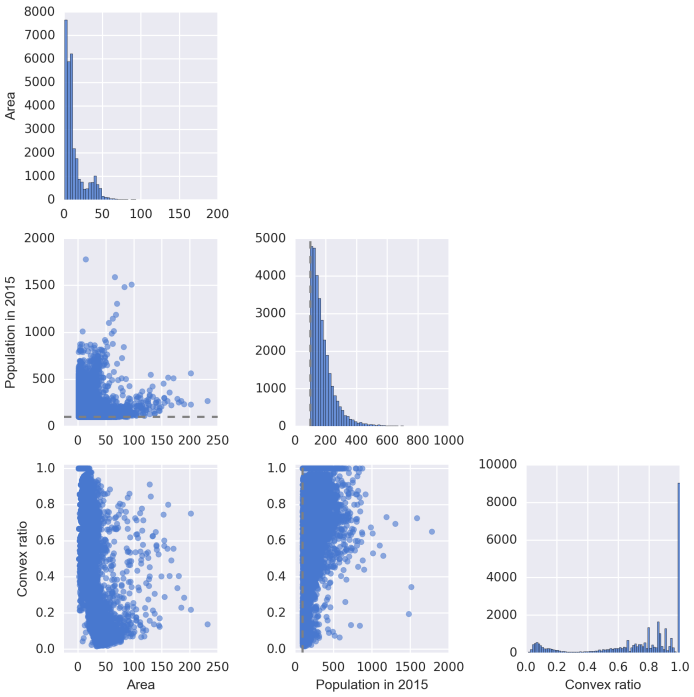


Figure 4: Descriptives of partition

parishes.[7]

As demonstrated above, the size of the polygons is a function of the population density. When performing spatial analysis it will sometimes be beneficial to further exclude the polygons, which are very large, as they diminish precision in a given measure. It turns out, that these are identified by having a very low ratio of polygon area to the area of the convex hull. The convex hull is the smallest convex set that contains the polygon. Figure 6 shows an example. A value of around 0.4 seems appropriate for excluding polygons without villages.[8]

[7]The numbers reflect the square root of the area for convex hull of the shape and this proxy for distance; thus for spatial area the improvement are thus respectively 50-fold at the mean and >100-fold at the median.

[8]This is sometimes referred to as "bizareness"

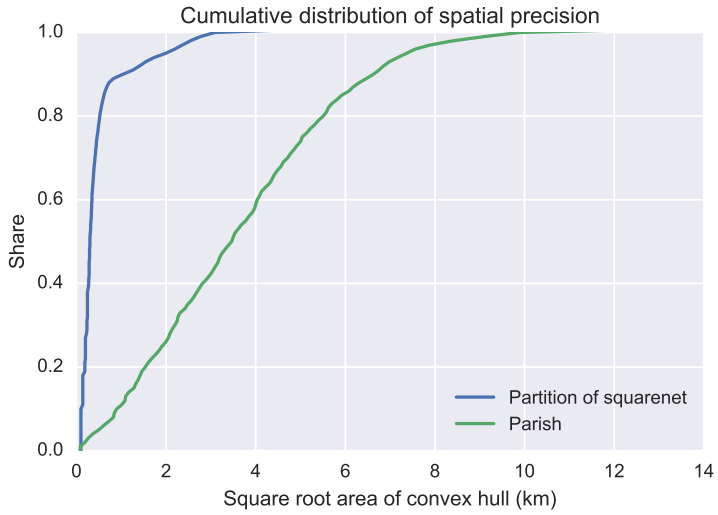
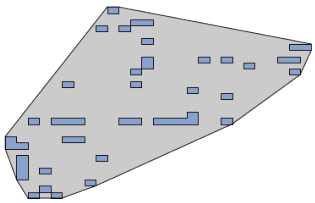
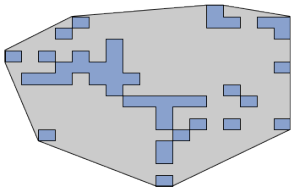


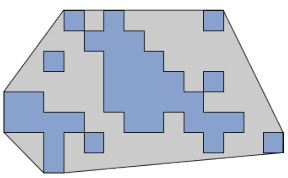
Figure 5: Comparison of spatial precision: square partition vs. Danish parishes



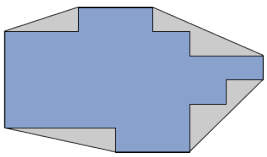
(a) Area/Convex=0.1



(b) Area/Convex=0.2



(c) Area/Convex=0.4



(d) Area/Convex=0.8

Figure 6: Examples of ratios of area to convex hull area

5 Example of usage

To briefly illustrate the potential of the algorithm we briefly present a map constructed from register data merged on our final partition of the square-net. Within each polygon, the share employed is registered as well as total population. To maintain a degree of anonymity only polygons with zero *or* at least ten individuals are used. Synthetic individuals are then generated and within each polygon given employment status according to the mean employment rate and a random location within the polygon. The result can be seen in Figure 7 where we use Datashader (Bednar et al. 2016) to aggregate the points to visible pixels. The spatial patterns are very visible. The Northern coastlines of Zealand (right-most island), for example, are dominated by cyan dots representing non-employment. We conjecture that this is mostly pensioners living in a summerhouse whereas bigger cities see a mix of both employed and unemployed. The coastal patterns would not be nearly as visible using other geographic entities, such as parishes, which do not delineate between coast and inland with the same degree of precision.

We note that our approach is limited by the fact that although the partitions are invariant people may relocate and some areas may then lose population. This could imply that some of the sub-areas violate the constraint of having at least k people living in them (100 in the context of Denmark) in the future. We also note that partitioning comes at the cost of a small number of households ($<$2 pct.) not being assigned an area. These households were mainly rural and they would still have a parish available so a more coarse measure of location would be available.

Figure 7: Employment in Denmark

Red dots represent employed individuals. Non-employed individuals are colored cyan. These individuals may be pensioners and students as well as unemployed. The individuals receive a random location within the polygon of residence. Data is aggregated using the Datashader-package.

6 Conclusion

We have introduced a method that produces a partition of addresses into sub-areas where people reside. The method works by specifying a desired level of privacy to bind while optimizing spatial precision. A key feature of our approach is that the shapes of the sub-areas are time-invariant and thus constant despite changes to administrative boundaries.

We have used our approach to make a partition of the Danish Squarenet. The resulting partition preserves the chosen level of privacy and simultaneously provide a much more accurate spatial precision measured with distance when compared with the de-facto standard measure of using administrative boundaries for parishes. We believe that our approach can be leveraged for similar endeavors in other contexts, e.g. applying it to other countries' addresses in research projects or in business analysis where preserving privacy is required.

References

Bednar, J. A., Crist, J., Cottam, J. & Wang, P. (2016), 'Datashader: Revealing the structure of genuinely big data', *15th Python in Science Conference (SciPy 2016)* .

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. (2008), 'Fast unfolding of communities in large networks', *Journal of statistical mechanics: theory and experiment* (10), P10008.

Csardi, G. & Nepusz, T. (2006), 'The igraph software package for complex network research', *InterJournal-Complex Systems* p. 1695.

Gedik, B. & Liu, L. (2004), A customizable k-anonymity model for protecting location privacy, Technical report, Georgia Institute of Technology.

McKinney, W. (2011), 'pandas: a foundational python library for data analysis and statistics', *Python for High Performance and Scientific Computing* pp. 1–9.

Mokbel, M. F., Chow, C.-Y. & Aref, W. G. (2006), The new casper: Query processing for location services without compromising privacy, *in* 'Proceedings of the 32nd international conference on Very large data bases', VLDB Endowment, pp. 763–774.

Samarati, P. & Sweeney, L. (1998), Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression, Technical report, Technical report, SRI International.

Schult, D. A. & Swart, P. (2008), Exploring network structure, dynamics, and function using networkx, *in* 'Proceedings of the 7th Python in Science Conferences (SciPy 2008)', Vol. 2008, pp. 11–16.

Van Der Walt, S., Colbert, S. C. & Varoquaux, G. (2011), 'The numpy array: a structure for efficient numerical computation', *Computing in Science & Engineering* **13**(2), 22–30.