

Processing aggregated data: the location of clusters in health data

Kevin Buchin · Maïke Buchin · Marc van Kreveld ·
Maarten Löffler · Jun Luo · Rodrigo I. Silveira

Received: 12 February 2011 / Revised: 19 September 2011 /
Accepted: 26 September 2011 / Published online: 18 October 2011
© The Author(s) 2011. This article is published with open access at Springerlink.com

Abstract Spatially aggregated data is frequently used in geographical applications. Often spatial data analysis on aggregated data is performed in the same way as on exact data, which ignores the fact that we do not know the actual locations of the data. We here propose models and methods to take aggregation into account. For this we focus on the problem of locating clusters in aggregated data. More specifically, we study the problem of locating clusters in spatially aggregated health data. The

This research was initiated during the GADGET Workshop on Geometric Algorithms and Spatial Data Mining, funded by the Netherlands Organization for Scientific Research (NWO) under BRICKS/FOCUS grant number 642.065.503, and has been also partially funded by NWO under the project GOGO.

K. Buchin (✉) · M. Buchin
Department of Mathematics and Computer Science,
TU Eindhoven, Eindhoven, The Netherlands
e-mail: k.a.buchin@tue.nl

M. Buchin
e-mail: m.e.buchin@tue.nl

M. van Kreveld
Department of Computer Science, Utrecht University, Utrecht, The Netherlands
e-mail: m.j.vankreveld@uu.nl

M. Löffler
Computer Science Department, University of California, Irvine, CA, USA
e-mail: mloffler@uci.edu

J. Luo
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Beijing, China
e-mail: jun.luo@sub.siat.ac.cn

R. I. Silveira
Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya,
Catalunya, Spain
e-mail: rodrigo.silveira@upc.edu

data is given as a subdivision into regions with two values per region, the number of cases and the size of the population at risk. We formulate the problem as finding a placement of a cluster window of a given shape such that a cluster function depending on the population at risk and the cases is maximized. We propose area-based models to calculate the cases (and the population at risk) within a cluster window. These models are based on the areas of intersection of the cluster window with the regions of the subdivision. We show how to compute a subdivision such that within each cell of the subdivision the areas of intersection are simple functions. We evaluate experimentally how taking aggregation into account influences the location of the clusters found.

Keywords Cluster · Aggregated data · Algorithm · Public health

1 Introduction

Spatial data often do not include exact location information. Instead data are aggregated into areas corresponding to regions like counties, zip codes, census blocks or enumeration districts, or come from sources like anonymous questionnaires where only approximate locations (like partial zip codes) are provided. Even if information about exact locations is available, there may be privacy and confidentiality considerations for not disclosing for instance exact address information of patients [1, 5, 6]. Aggregated data is frequently used in application areas such as criminology [23], sociology [26], political science [4, 13], geography [21], and public health [12]. A known problem with aggregated data analysis is that it suffers from the *modifiable areal unit problem*: it can be problematic to use aggregated data when the aggregation units have nothing to do with the phenomenon being analyzed (e.g., spreading of disease is oblivious to municipality boundaries) [21]. However, it is often the case that the only data available for analysis is aggregated data.

In this paper we focus on aggregated data, and in particular, on locating clusters in disease data. The health application is relevant and interesting in its own, but in addition, it serves to illustrate the models and methods that we propose to deal with aggregated data. Our methods can be applied to many other situations where aggregated spatial data is used, and reduce the effect of the modifiable areal unit problem when compared to existing methods.

The study of geographical patterns of diseases is an important aid for the investigation of outbreaks. Analyzing the geographic nature of disease cases has been a key factor in finding the source of many outbreaks. Since the famous case of geographical analysis of John Snow in 1854 [3, 29], numerous examples have been documented in the literature of several fields like epidemiology, public health, preventive medicine, and medical geography.

Investigation of outbreaks due to both infectious and non-infectious causes (e.g., toxic exposure) can greatly benefit from the use of spatial information. Even though the role played by geography in the identification of outbreaks depends entirely on the disease, spatial factors have a major importance for many outbreaks related to exposure to pollution or radiation sources (for a wide range of diseases, from respiratory illnesses to different types of cancer), as well as for airborne diseases like *Legionella* [7] or Q fever [8].

In the public health domain, detecting clusters in aggregated data is done by statistical methods [27, 31]. These methods, however, typically represent the aggregation

regions by their centroids. One of the most widely used approaches, which also uses centroids, is the *spatial scan statistic* [14, 16]. Another well-known method for cluster detection is the Geographical Analysis Machine [22]. This method assumes non-aggregated point data and tests cluster centers based on grid points. In this paper we argue and verify that such point-based methods may not perform well, and area-based alternatives can be used instead.

1.1 Aggregated data

The centroids of regions are not necessarily good representatives for the regions. Large parts of a region might be closer to the centroids of other regions than to the centroid of their own region. This happens for instance for regions that are larger than neighboring regions, or for elongated regions. The centroid of the elongated region in Fig. 1(right), which corresponds to the municipality of Rotterdam in the Netherlands, does not even lie in the region. The figure is a snapshot from our experiments, which are detailed in Section 4. For now, we distributed an imaginary illness at random over the inhabitants of the Netherlands, and created an artificial cluster by giving everyone inside a given circle (shown dashed in the figure) a higher probability of being ill. The figure also shows the cluster found by a standard centroid-based method (SaTScan¹). At first sight, it may be surprising that the cluster does not include the municipality of Rotterdam, although most of the cases of the cluster lie there. The reason for this is that the centroid of Rotterdam is not close to the centroids of the other regions in the cluster, and moreover, the centroid-based method, in order to include the cases in Rotterdam, is forced to include the whole municipality, which has a relatively large population and makes the likelihood of the cluster decrease. The figure also shows the result found by one of our methods, which tries to take the actual area of the regions into account (thus we refer to it as *area-based*). Figure 2 shows another example using real data from an outbreak of Salmonella that took place in the Netherlands in 2006 [11]. Also in this case, the area-based method is able to provide a more precise location for the outbreak.

Clearly, the problems that occur when using centroids to represent whole regions also occur for any other *representative point* one might use, since they are inherent to concentrating area-based information into one single point. Despite being a practical simplification, representing regions by arbitrary points may lead to artifacts or distorted results, especially if we are interested in geometric properties of the data.

1.2 Contribution

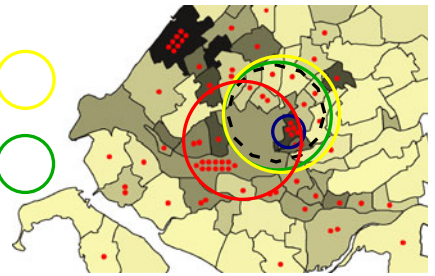
The issues described in the previous section arise from representing areal—two-dimensional—features by single points (e.g. centroids). It is clear that representing whole regions by single points can lead to a loss of valuable information, especially when analyzing aggregated data for cluster location. Therefore we propose alternatives to point-based methods that take the area of the regions into account. We refer to our methods as *area-based*, because they treat aggregated regions as areas and not as points. Taking the full geometry of the regions into account, as opposed to considering data on regions simply as data concentrated at single points, implies

¹<http://www.satscan.org>

Artificial case data

Poisson Likelihood Ratio Test

- centroid-based method, discrete area-based method, homogeneous
- centroid-based method, continuous area-based method, non-homogeneous
- original cluster • = 5 cases



population density 0/km² 5000/km²

Fig. 1 Estimating the location of a cluster on artificial data aggregated into Dutch municipalities, with two point-based method (*centroid-based*, discrete and continuous variants) or with the two proposed area-based methods (*area-based*, homogeneous or non-homogeneous variants). All methods use a Poisson likelihood test

Salmonella Typhimurium 560 per municipality in 2006

Poisson Likelihood Ratio Test

- centroid-based method, discrete
- centroid-based method, continuous
- area-based method, homogeneous and non-homogeneous (virtually same result)

• = 5 cases

population density 0/km² 5000/km²

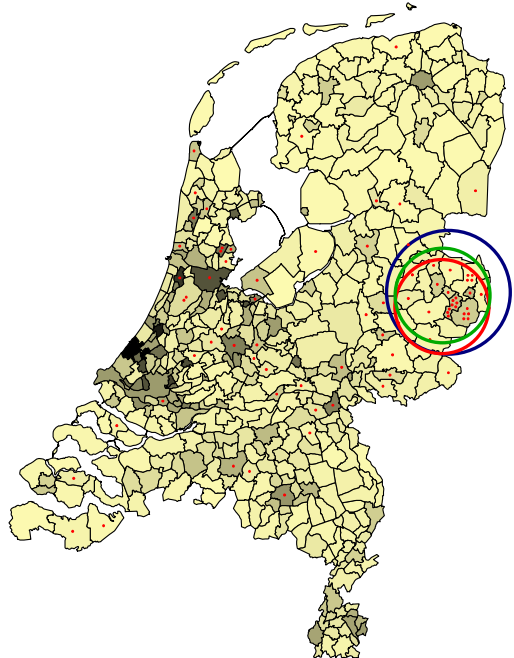


Fig. 2 Estimating the location of a cluster on real data from the Netherlands, with population and cases either concentrated at the centroids of regions (*centroid-based*) or uniformly distributed (*area-based*). All methods use a Poisson likelihood test

that algorithms can become more complex and require a higher running time than algorithms for point-based methods. We show how to efficiently handle these more complex models by using tools from computational geometry [2].

We begin by formulating the problem as a polygon placement problem (for illustration a rectangle will be used at first; later we will explain how to handle any polygonal shape). This polygon represents a possible location for the cluster, and a placement that contains many cases compared to the population is an indication that we may have found the cluster. By choosing regular polygons, we can approximate a circular cluster region arbitrarily well. We refer to the polygon as the *cluster window*.

When the cluster window contains a region completely, then it contains the complete population and all cases of that region. When the cluster window contains only part of a region, we face the problem that the data was aggregated and must make assumptions on how the data (in our case, the population at risk and the disease cases) are distributed inside the regions. We first propose a *homogeneous model*, where we assume that both the population and the cases are uniformly distributed inside each region. In some situations, assuming a uniform case distribution can be inappropriate, thus we also propose a second, *non-homogeneous model*, where cases have the tendency to cluster. We present these models in the next section.

The cluster window placement that we seek is one that maximizes some cluster function depending on the cases and population inside the window; the precise function depends on the model. We refer to this window placement as the *optimal window*. To find the optimal window we need to compute an arrangement of the combinatorially different placements of the window and optimize the cluster function within each cell of the arrangement.

This allows us to consider *all* possible placements of the cluster window over a subdivision with n regions. This is an important difference to previous approaches, which restrict the search to a finite number of locations. The optimization per cell can be performed exactly for simple functions such as the case-to-population ratio, or numerically for more complex functions. Assuming the optimization takes constant time, the total worst-case running time of the resulting algorithm is $O(N^2)$, where N is the number of vertices in the subdivision. However, we prove that under reasonable, practical assumptions on the resolution of the regions and the cluster window, the running time is only $O(N \log N)$.

While the focus of this paper is on algorithmic techniques for aggregated data, we also provide an experimental comparison of area-based versus centroid-based cluster detection. The goal of the experiments is to study which methods are best in estimating the location of a cluster. To compare the methods we use the same statistical test for all experiments but different methods to obtain case and population counts. The results suggest that area-based models indeed capture the geometry of the data better than centroid-based methods. In other words, they suffer less from the modifiable areal unit problem than centroid approaches.

2 Modeling the problem

In this paper we abstract the problem of finding the location of a cluster as a polygon placement problem. We will first represent the cluster area by a rectangle W and later extend our results to general polygons. The objective is to find a placement

of W such that a *cluster function* is maximized. The cluster functions will depend on the population at risk and the cases covered by W . The number of cases in W will be estimated using various models. We assume that we have access only to aggregated location and population data, meaning that the exact location of the cases and population is not known. The rectangle W will have some fixed size and we will assume that it is axis-aligned. Moreover, we address the situation with different subdivisions for the case and population data.

2.1 Models for the case distribution

In the models proposed next, we are given one subdivision of the plane, consisting of a set \mathcal{P} of n regions P_1, \dots, P_n , and for each region P_i in the subdivision we are given two values c_i and p_i . The first value c_i represents the number of disease cases within P_i , whereas the second value p_i represents the population at risk of P_i (for example, the number of people at risk for the disease in question).

We propose two basic models. The first model assumes that the distribution of the cases is *homogeneous* inside each region. The second model assumes a more *non-homogeneous* distribution of the cases, that is, for a region that is partially covered by the cluster window, we take into account that the case density in the cluster window might be higher than the case density outside the cluster window.

The problem of estimating the number of cases and the population at risk in the cluster window is closely related to the problem of areal interpolation. Areal interpolation is the problem of determining—based on aggregated data from a source subdivision—the aggregated values of the data on a target subdivision. It is needed when data aggregated on different subdivisions is integrated. Our problem is similar in the sense that we have a source subdivision and target areas, i.e., the cluster windows. There are volume-preserving and non-volume-preserving areal interpolation methods, with the volume-preserving methods considered to be superior [18]. Common areal interpolation methods are *areal weighting*, the *pycnophylactic method* [30], and methods using ancillary data. Areal weighting assumes a homogeneous distribution of the data within the cells of the source subdivision. In contrast the pycnophylactic method assumes that the data value is a smooth function of the location, on the domain of the subdivision. Areal interpolation can be improved by using ancillary data like satellite imagery or the road network. See [9] for a comparison of different areal interpolation methods.

In the following we will always assume that the population is distributed uniformly in each region. The cases will be assumed to be distributed uniformly for the homogeneous model, while in the non-homogeneous model cases will be assumed to cluster. Thus, in spirit this is very similar to areal weighting. Nonetheless, the techniques proposed in this paper can be combined with further assumptions on the population and case distribution by preprocessing the data to redistribute the population and cases to a finer subdivision according to the assumptions made.

2.1.1 Homogeneous model

We will assume for the first model that the distributions of both the cases and the population are uniform.

More formally, let (x, y) be the position of the center of window W . We will write $W(x, y)$ to refer to the window W translated in such a way that its center lies at (x, y) .

See Fig. 3 for an illustration. Finding a placement of W is equivalent to finding a value for x and y . We can express the number of cases $c(x, y)$ and the population $p(x, y)$ in the cluster window as:

$$c(x, y) = \sum_{i=1\dots n} c_i \cdot f_i(x, y) \quad \text{and} \quad p(x, y) = \sum_{i=1\dots n} p_i \cdot f_i(x, y) \quad (1)$$

where $f_i(x, y) \in [0, 1]$ denotes the fraction of the area of P_i intersected by $W(x, y)$: $f_i(x, y) = \text{Area}(P_i \cap W(x, y)) / \text{Area}(P_i)$.

This model seems reasonable given that we are dealing with aggregated data and the location of the cases is not known. However, there are situations in which the result obtained is not what one would expect. As an example, consider the example depicted in Fig. 4. Assume we would want to find the window with the highest cases-to-population ratio. Under the homogeneous model, the optimal window lies completely inside the most dense region, like W_1 . However, the situation suggests that the window with the highest ratio might actually be close to the intersection of the three shaded regions since such a window W_2 can contain all twelve cases, whereas W_1 cannot. The next model we propose handles such situations better.

2.1.2 Non-homogeneous model

There are various ways to address the situation illustrated in Fig. 4. From a global perspective the situation can often be avoided by a suitable model for the overall case (and population) distribution. For instance, one might assume that the cases-to-population ratio changes continuously. Using the pycnophylactic method for areal interpolation this would result in a finer subdivision (i.e., grid) which might give preference to a cluster window closer to the intersection of the regions depending on the global situation. As mentioned earlier, we see the techniques proposed in this paper independent to this global view of redistributing population and cases, since this redistribution is independent of the clustering and can be done in a preprocessing step.

From a more local perspective, we might want to know by how much our homogeneous model is off in a pessimistic scenario, i.e., if the cases in a region are clustered such that a larger proportion than expected falls into the currently inspected cluster window. In a worst-case scenario this could mean that all cases of a region are in the cluster although the cluster window only partially covers the region. In this approach the total number of cases in a region is preserved, but how they are distributed is decided only after querying for a cluster window.

Fig. 3 Example for the homogeneous model: for each region P_i , (c_i, p_i) is shown; shading shows the cases-to-population ratio. The cases and population are assumed to be uniformly distributed inside each region. The location of W maximizes the density of cases under the homogeneous model

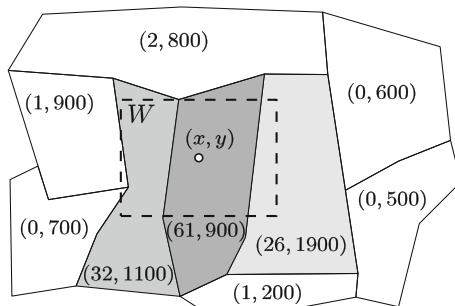
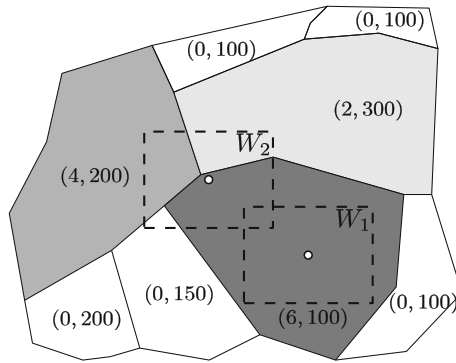


Fig. 4 The numbers in each region indicate number of cases and population. The homogeneous model will result in a window like W_1 , whereas the non-homogeneous model will yield one like W_2



In our second model we give a formalization of this local perspective. It is worth stressing that this model represents only one possible way to address the assumption of a non-homogeneous distribution, but many others are possible. In the homogeneous model the number of cases contributed by a region to a cluster is proportional to the area of intersection between the region and the cluster window. To allow for a larger proportion of cases in the cluster window, we express the proportion by an increasing function $g: [0, 1] \rightarrow [0, 1]$ with $g(z) \geq z$. The population is assumed to be distributed uniformly as in the homogeneous model. For instance, if the cluster window covers half of a region, the homogeneous model would take half of the cases. The second model would instead take a fraction $g(1/2)$ of the cases, which by assumption is at least $1/2$. We can express the number of cases $c(x, y)$ and the population $p(x, y)$ in the cluster window as:

$$c(x, y) = \sum_{i=1 \dots n} \min \{c_i \cdot g(f_i(x, y)), p_i \cdot f_i(x, y)\} \quad \text{and} \quad p(x, y) = \sum_{i=1 \dots n} p_i \cdot f_i(x, y) \tag{2}$$

where again $f_i(x, y) \in [0, 1]$ denotes the fraction of the area of P_i intersected by $W(x, y)$.

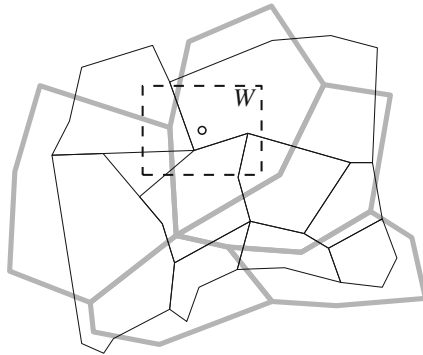
In the formula for the number of cases, the minimum accounts for situations in which the estimated number of cases would actually be higher than the estimated population. Taking the minimum guarantees that these two estimates are consistent. For computational reasons we will restrict ourselves to piecewise linear functions g . Piecewise linear functions can be used to approximate other choices of g .

2.2 Different subdivisions for population and cases

Population information and case information might be aggregated differently. In this case, we have one subdivision for the population data and another one for the number of cases. See Fig. 5 for an example. In fact, such asymmetry between the case and population data is not uncommon, since case data is often aggregated to larger geographical units than population data (see e.g. [10, 19]).

We handle this situation by areal interpolation in a preprocessing step. More specifically, we use areal weighting, i.e., we determine the overlay of the two subdivisions and distribute population and cases by area. In this section we address the following problem: areal weighting might result in inconsistent estimates for the

Fig. 5 A variant of the problem is where two different subdivisions are given, one for the case data and one for the population data



population and the number of cases in regions of the overlay, in particular, the number of cases might be larger than the population. In the following we explain how to compute numbers for the cases and the population of the overlay that are consistent and in accordance with the input data.

We are given a subdivision \mathcal{P} of the plane comprised of n regions P_1, \dots, P_n , and for each region P_i we are given a population value p_i , and in addition, we are given a second subdivision \mathcal{C} comprised of m regions C_1, \dots, C_m , each with a case count value c_i . Let \mathcal{B} be the overlay of the subdivisions \mathcal{P} and \mathcal{C} . All regions in \mathcal{B} are such that they are part of only one region from \mathcal{P} and only one region from \mathcal{C} . Let B be some region of \mathcal{B} , and assume it is part of a region $P(B)$ in \mathcal{P} and $C(B)$ in \mathcal{C} . Assuming equal population density in the regions of \mathcal{P} implies that the population of B is equal to the population of $P(B)$ times the ratio of the areas of B and $P(B)$, and a similar statement holds for the cases. However, the two equal density assumptions are conflicting if this implies that the number of cases in B is greater than its population. We will define a reasonable way to assign a population and a number of cases to B . The following three conditions define a valid assignment of population and cases to the regions of \mathcal{B} .

- If a region P from \mathcal{P} is partitioned into a set of regions in \mathcal{B} , then the total population in these regions must be equal to the population of P .
- If a region C from \mathcal{C} is partitioned into a set of regions in \mathcal{B} , then the total number of cases in these regions must be equal to the number of cases in C .
- For each region in \mathcal{B} , its population is at least the number of cases.

These conditions do not uniquely determine how to assign a population and case count to the regions. In fact, in general, it could happen that these three conditions cannot be fulfilled at the same time, but this can only occur if the data are inconsistent. If the data come from a real situation, there will always be at least one solution, namely the real population and the real number of cases for the regions of \mathcal{B} . However, since these values are not given, we still need to determine a solution.

To choose the “most likely” solution, we will try to spread population and cases as uniformly as possible. Recall that for any region $B \in \mathcal{B}$, $P(B)$ is the region of \mathcal{P} that contains B , and $C(B)$ is the region of \mathcal{C} that contains B . Furthermore, let $Pop(P(B))$ and $Case(C(B))$ be their population and case counts, to be computed.

In order to distribute the population and cases as uniformly as possible, we formulate the problem as a minimization problem. The minimization is over all

possible population and case counts of all regions in \mathcal{B} , restricted to the three conditions specified before. The minimization problem yields zero exactly when population and cases are uniformly distributed in the subregions of each region from \mathcal{P} and \mathcal{C} .

We can solve the minimization problem optimally by linear programming, although fractional population and cases may arise. To obtain a linear programming formulation we will introduce extra variables to be able to remove the absolute-value aspect in the current formulation. For any region $B \in \mathcal{B}$, let $ErrPop(B)$ be the deviation of the population $Pop(B)$ in B from the uniform spreading over $P(B)$:

$$ErrPop(B) = \left| Pop(B) - \frac{Area(B)}{Area(P(B))} Pop(P(B)) \right| .$$

Similarly, let

$$ErrCase(B) = \left| Case(B) - \frac{Area(B)}{Area(C(B))} Case(C(B)) \right| .$$

Then we can use the following linear inequalities in our linear program:

$$\begin{aligned} ErrPop(B) &\geq Pop(B) - \frac{Area(B)}{Area(P(B))} Pop(P(B)) ; \\ ErrPop(B) &\geq \frac{Area(B)}{Area(P(B))} Pop(P(B)) - Pop(B) ; \\ ErrPop(B) &\geq 0 . \end{aligned}$$

Analogous inequalities are used for $ErrCase(B)$. The equalities and inequalities that arise from the three conditions stated before in the list are also linear. Our formulation of the minimization problem is now:

$$\min \left(\sum_{B \in \mathcal{B}} ErrPop(B) + w \cdot \sum_{B \in \mathcal{B}} ErrCase(B) \right) .$$

The parameter w is used to weigh uniformly distributed population more heavily than uniformly distributed cases, which appears to make sense. It should then be chosen smaller than 1. The minimization (objective function of the linear program) is linear, and hence we conclude that a linear programming formulation is obtained. Due to minimization, any optimal solution must set the extra variables $ErrPop$ and $ErrCase$ to match their definition, and hence the linear programming formulation is equivalent to the original formulation that uses the absolute-value function.

It may be possible to avoid fractional population and case counts in the regions $B \in \mathcal{B}$, but this implies using *integer programming* instead of *linear programming* [28]. Since linear programming is computationally much easier, and good solvers are available, a linear programming formulation is definitely preferable [34]. Intuitively, it seems that a fractional solution does not have a negative effect on the cluster eventually found, and may even have a positive effect. Also observe that the optimal location of the cluster window generally also corresponds to fractional case and population counts, and hence we do not consider fractional values to be a real problem.

2.3 Cluster functions

So far, our models up to now describe how the population and cases are distributed over the plane, given the aggregated input data that we have. In order to actually detect clusters, we will try to find a *cluster window*, which is a region in the plane (such as a circle or a square) that is most likely to define a cluster. The distributions of population and cases allow us to compute the number of each quantity inside a candidate cluster window. To determine whether it really is a cluster, though, we also need a so-called *cluster function*. A cluster function takes these two numbers, the population and case count, as input. It returns a number that should be higher if the population and case count indicate a cluster.

A very simple cluster function would consist in simply dividing the number of cases by the population in the window. But this quotient has little statistical significance. A better cluster function is the likelihood of a cluster under a statistical model. For instance, in the experiments presented in Section 4, the likelihood test assumes a Poisson model [14]. Several other tests, based on different statistical models, exist, and our methods can be adapted to most of them, as long as the test itself can be performed efficiently. A thorough comparison of several clustering models used for disease clustering can be found in [17].

3 Geometric structure and algorithm

To solve the cluster finding problems defined in the previous section, we can compute the arrangement of combinatorially different placements of the window W . Sections 3.1 and 3.2 detail what this arrangement looks like, and how to compute it efficiently. In Section 3.3 we will use the arrangement to compute a placement of the window that maximizes the case-to-population ratio. We will then describe how to adapt the method to work with other polygonal cluster windows. Finally, we discuss alternative algorithms and extensions.

3.1 Arrangement of placements

Given a subdivision \mathcal{P} and a window W , we say two placements of W are *combinatorially different* if the set of pairs of edges of W and \mathcal{P} that intersect are different. For example, the placements in Fig. 6a and b are combinatorially the same, but the

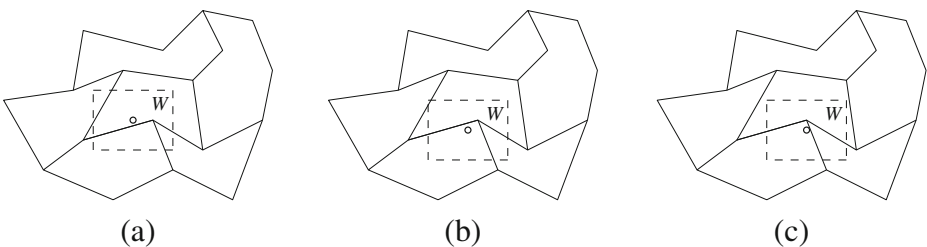


Fig. 6 Placements in **a** and **b** are combinatorially the same, but in **c** it is combinatorially different: the set of regions intersected by W is different

one in Fig. 6c is different because the top left corner of W has moved from one cell to another. When two placements are the same, we can write the area of overlap between W and any cell $P \in \mathcal{P}$ as a closed-form function in x and y (the coordinates of the center of W), which will allow us to optimize functions that involve this area of overlap efficiently for all combinatorially equal placements.

This combinatorial relation between placements subdivides the *placement space*—the set of possible positions for the center point of W —into a number of regions such that inside each region, all placements are combinatorially equal. We can define and compute this arrangement for each cell $P \in \mathcal{P}$, and the total arrangement will be the overlay of these.

For a given cell P , a placement of W changes whenever a corner of W moves across an edge of P (as in Fig. 6), or when an edge of W moves across a vertex of P . This means that the edges of the arrangement are defined by the line segments that arise when sliding a corner of W along an edge of P , as in Fig. 7b, or an edge of W along a vertex of P , as in Fig. 7c. The arrangement of this region is then the overlay of those two, as shown in Fig. 7d. Note that the final arrangement of placements, as shown in Fig. 7d, is composed of the overlay of a number of translated copies of the window W and a number of translated copies of the cell P .

If we compute this arrangement for all cells P_1, \dots, P_n of the subdivision, and compute the overlay of all of them, then this gives a partition of the whole plane referred to as the arrangement \mathcal{A} . For any cell of \mathcal{A} , if the center of W is in that cell, the pairs of one edge from \mathcal{P} and one edge from W that intersect are fixed (placements of W are combinatorially the same as long as the centers lie in the same cell of \mathcal{A}). To compute this arrangement, we collect all translated copies of the cells and of W , and note that their total number of vertices is $O(N)$ (as long as W has constant complexity). We can compute the overlay of all these polygons in $O(N \log N + K \log N)$ time using standard methods (see for example [2], Section 2.4), where K is the complexity of the final arrangement. In the worst case, K can be $\Theta(N^2)$. In the situation where the population and cases are given in two separate subdivisions \mathcal{P} and \mathcal{C} , things get even worse: since the overlay of the subdivisions can have quadratic complexity, the algorithm may take $O(N^4 \log N)$ time in the worst case. However, we will show next that in practice, we expect much better running times.

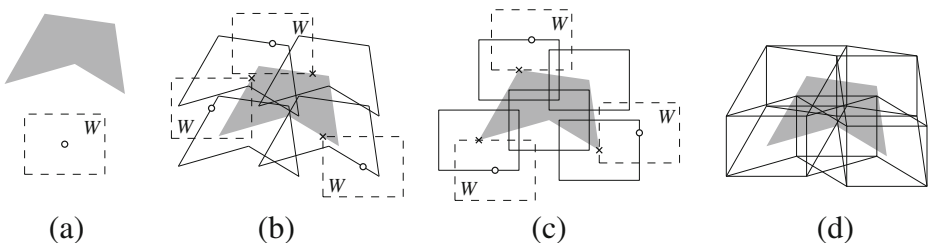


Fig. 7 **a** A cell of the subdivision (*shaded*), and a window W (*dashed*). **b** The positions of the center point of W as a vertex of the window slides along an edge of the cell. **c** The positions of the center point of W as an edge of the window slides along a vertex of the cell. **d** The total arrangement is the overlay of the previous two figures

3.2 Efficiency analysis in practical situations

The near-quadratic complexity obtained in the previous section is based on a worst-case analysis. Most likely, such analysis will not reflect the actual complexity of the arrangement \mathcal{A} of placements in practice. Therefore in this section we consider a more realistic scenario. In particular, we will make a *resolution assumption*. Define r as the shortest distance between any two vertices of the region subdivision \mathcal{P} , so r is a measure for the detail level of the data. Our *resolution assumption* states that there are positive constants c_1, c_2, c_3 , and c_4 such that (i) the distance between any vertex and any edge not incident to that vertex in \mathcal{P} is at least c_1r , (ii) the length of any edge in \mathcal{P} is at most c_2r , and (iii) the diameter of W is at least c_3r and at most c_4r . This assumption will allow to prove that in practice, the algorithms have a considerably better running time than what is provable otherwise. The algorithm itself and its correctness do not depend on the resolution assumption.

The assumption essentially states that the distances between the vertices and edges of the subdivision and the window W are all of the same order of magnitude. There are therefore two factors that contribute to the resolution assumption. First, the regions themselves should not be modeled using too many edges and vertices. That is, the input subdivision needs to be chosen at an appropriate scale. Second, the extent of the cluster that the algorithm is searching for should be at most a few factors larger than the extent of the regions. For example, it would be impractical to have regions that are city neighborhoods with an outbreak region of the size of the whole country. Moreover, if the cluster window becomes very large compared to the regions, then many regions will actually lie completely within the window and as a consequence the problem of spatial misalignment becomes less relevant. Therefore, the more interesting case is indeed the one in which the region and cluster extents do not differ too much.

Lemma 1 *The resolution assumption implies that any angle between two adjacent segments of \mathcal{P} is bounded from below by a positive constant.*

Proof Let v be a vertex of \mathcal{P} , suppose there are two edges with angle α that have v as an endpoint, and let the shorter have length l . Then the distance d between the endpoint of the shorter and the longer edge will be $d = l \sin \alpha$. But we know that $l \leq c_2r$ and $d \geq c_1r$, which implies that $\sin \alpha \geq \frac{c_1}{c_2}$. Since c_1 and c_2 are positive constants, the lemma follows. \square

Note that Lemma 1 implies that vertices in \mathcal{P} have constant degree. Under the resolution assumption, we can prove that the complexity of the arrangement \mathcal{A} is actually $O(N)$.

Lemma 2 *The complexity of \mathcal{A} under the resolution assumption is $O(N)$.*

Proof As noted, \mathcal{A} is formed by the set of line segments that are translated copies of edges of \mathcal{P} and W . Every vertex of \mathcal{P} gives rise to four line segments in the arrangement which are translates of the edges of W , and every edge of \mathcal{P} gives rise to four translates due to the four corners of W . The arrangement is therefore formed by $O(N)$ line segments.

Take any line segment l that defines \mathcal{A} : it is either a translate of an edge of \mathcal{P} , or a translate of one of the four edges of W . In either case, it has length $\Theta(r)$ since c_1, c_2, c_3 , and c_4 are constants. Assume it is a translate of an edge e of \mathcal{P} ; the case where it is a translate of an edge of W is handled in exactly the same way.

We analyze the number of intersections of l in the arrangement (with translates of other edges of \mathcal{P} and W), and will bound it from above by a constant depending on c_1, c_2, c_3 , and c_4 . Observe that any intersection on l is with a line segment generated by a vertex or edge of \mathcal{P} that is at most $O(r)$ away from e . The region within cr from e (for some constant c) is a region of area $O(r^2)$. Since the resolution assumption holds inside this region, a straightforward packing argument shows that only $O(1)$ vertices and edges can intersect this region. Hence, l contains at most $O(1)$ intersection points in \mathcal{A} . Since this argument holds for all $O(N)$ edges that define the \mathcal{A} , it has $O(N)$ complexity. \square

When the information about the population and the disease cases are given in separate subdivisions \mathcal{P} and \mathcal{C} , we can compute the overlay of the two and treat this as if it was a single subdivision. As mentioned, the algorithm now takes $O(N^4 \log N)$ time. However, this will hardly occur in practice. In fact, under the resolution assumption we can prove that the complexity of the arrangement \mathcal{A} is still linear.

Lemma 3 *In the two-subdivision variant of the problem, under the resolution assumption, the complexity of \mathcal{A} is $O(N)$.*

Proof Let e be an edge of \mathcal{P} . We will show that e intersects at most a constant number of edges of \mathcal{C} .

We know that the length of e is at most c_2r . Sort the edges of \mathcal{C} according to the order in which they intersect e . If two consecutive edges do not share an endpoint, then the distance between them is at least c_1r . If they do share an endpoint, then their distance can be small, but there can be only a constant number of such edges sharing the same endpoint, because the resolution assumption for \mathcal{C} and Lemma 1 imply that \mathcal{C} has constant vertex degree. Therefore there are $O(1)$ edges of \mathcal{C} intersecting e .

Now the same arguments used to prove Lemma 2 can be applied. In particular, since the overlay of the two subdivisions splits each edge in \mathcal{C} and \mathcal{P} into a constant number of pieces, it is still the case that the window W contains a constant number of vertices and edges. Therefore the result follows in a similar way as in the proof of Lemma 2. \square

3.3 Computing the optimal placement

Using the arrangement \mathcal{A} of combinatorially different placements we can find the placement of the cluster window maximizing the cluster function by maximizing the function in each cell of the arrangement. Since we are considering cluster functions that depend on the number of cases and the population in the cluster window, it is important to efficiently compute the expression for the number of cases and the population for each cell of the arrangement. In the following we will show how to efficiently compute these expressions for all cells of the arrangement.

3.3.1 Computing the optimal placement in the homogeneous model

The optimization within a cell depends on which cluster function is used. We will illustrate the optimization by a simple function, the ratio of cases-to-population. Note that for more complex functions it might be infeasible to compute the exact optimum. In such a case we numerically approximate the optimum.

To optimize the ratio $c(x, y)/p(x, y)$ using Formula 1 over the arrangement, we first need to be able to optimize it inside a single cell. The only part of the formula that depends on x and y is the fraction $f_i(x, y)$ that describes which part of each region P in \mathcal{P} is covered by W . We can decompose the intersection between the cluster window and a subdivision cell into trapezoids, see Fig. 8. The locations of the corners of these trapezoids are linear functions in x and y , so the area of each trapezoid is a quadratic function. In other words, $f_i(x, y)$ is a quadratic function in x and y for every i . Recall from Formula 1 that $c(x, y) = \sum_{i=1..n} c_i \cdot f_i(x, y)$ and $p(x, y) = \sum_{i=1..n} p_i \cdot f_i(x, y)$, where c_i and p_i are constants that do not depend on x and y , so $c(x, y)$ and $p(x, y)$ are also quadratic functions in x and y . Then the ratio $c(x, y)/p(x, y)$ is a fraction of two quadratic functions, i.e., is of this form:

$$\max_{(x,y) \in \mathbb{R}^2} \frac{a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6}{b_1x^2 + b_2xy + b_3y^2 + b_4x + b_5y + b_6} \tag{3}$$

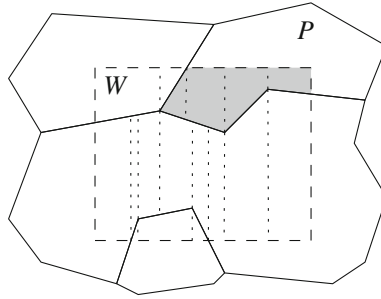
This formula can be optimized in constant time by using standard algebraic methods, or it can be numerically approximated fast, for instance by using a computational software package like Mathematica² [33].

To find the optimum over the whole arrangement \mathcal{A} , we need to determine Formula 3 for every cell, which means that we determine the six a_i and the six b_i coefficients. Of course, we can just do this from scratch for each cell individually, but that could require up to $O(N)$ time per cell, since the cluster window belonging to each cell could intersect up to $O(N)$ edges of \mathcal{P} .³ This then leads to a total of $O(NK)$ time (where K is the complexity of the arrangement). We can do better by traversing the cells of the arrangement from cell to adjacent cell while maintaining some information. We maintain the formulas for the number of cases and the population separately (represented by six coefficients each), and update both when we move the center point over the arrangement to an adjacent cell. This approach was used previously by [20]. Recall that the combinatorial structure of the intersection changes when a corner of W moves over an edge of P , or vice versa. When this happens, most of the subdivision cells do not change, and of the ones that do, most of the trapezoids that make up the intersection stay the same. We only need to subtract the functions of the trapezoids that are no longer valid, and add the functions of the new ones that came into existence. This number is typically a constant and therefore the update takes constant time. Even in the unrealistic case that it is not a constant, we can precompute some information that still allows us to update the function in constant time (the ideas are the same as used by [24]). The basic idea is to subtract the contribution of quadratic functions that no longer give a trapezoid, and add the contribution of quadratic functions that give a new trapezoid. Since we can do

²<http://www.wolfram.com>

³This bound applies without making the resolution assumption.

Fig. 8 The intersection between W and a subdivision cell P is the union of four trapezoids

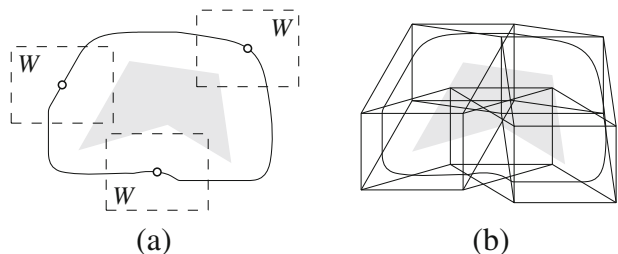


updates in constant time, we spend $O(K)$ time in total. Therefore, we spend only $O(K)$ time to determine the expressions for the number of cases and the population in Formula 3 for all cells, to find the maximum. With the resolution assumption, we spend only $O(N \log N)$ time in total.

3.3.2 Computing the optimal placement in the non-homogeneous model

To solve the problem in the non-homogeneous model, our arrangements become slightly more complicated, because there is another event where the functions involved change. Recall that the model depends on a function g that describes how many cases there can be among a certain percentage of the population. When the window starts containing enough area to go from one linear part of g to the next, we also have an event. This means we must add some extra curves to the placement space: the curves where the area of overlap has this value. Generally this gives one closed curve for each value at which a new linear part of g begins (as W moves around P_i , keeping the area of overlap constant), but it could also be a collection of curves. Figure 9a shows how this looks in our example. In fact, the points where the pieces of this curve change coincide with the lines of the other parts of the arrangement, since this happens exactly when the combinatorial structure of the area of overlap changes. Within one piece, the curve is a level-set (or iso-contour) of a quadratic function, so it is a quadratic curve in the plane. Figure 9b shows the total arrangement. The functions we need to optimize over the new arrangement remain the same, only now we have to optimize them over cells with non-linear boundaries. Standard numerical methods, as the ones provided by computational software packages like Mathematica, can be used to solve this problem.

Fig. 9 **a** New curves introduced in the second model. **b** Total arrangement



3.4 The shape of the cluster window

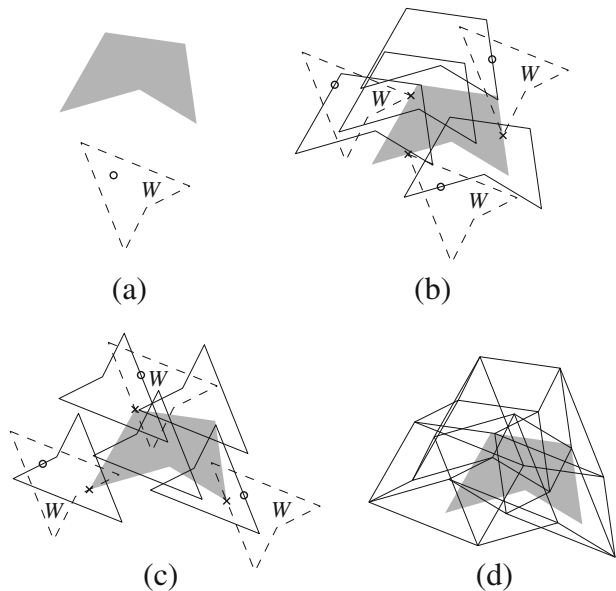
When the window W is not a rectangle, but some polygon with a constant number of sides, the algorithm still works without any modifications. In the algorithm the shape of the cluster window is used in the construction of the arrangement and for calculating the area of overlap between the window and a region. In the computation of the arrangement we simply replace the rectangle by the polygon as can be seen in Fig. 10. We used the center point to specify the location of a (rectangular) window, but we can in fact choose any point as its reference point. The number of vertices of W will appear in the running time, but not asymptotically if it remains constant. For calculating the area of overlap we again decompose the intersection of the window with subdivision cells into trapezoids. The adaptations needed for this extension are straightforward, and hence, we do not describe them any further.

If the window is not a polygon, we approximate it by a polygon. For instance, a circle can be approximated arbitrarily well by regular polygons. In our experiments we approximate circular cluster windows by regular 16-gons.

3.5 Alternative algorithms and extensions

Using the arrangement of combinatorially different placements we can find the optimal placement of the cluster window. The main tool needed to implement the algorithm is a robust implementation of arrangements, which is available in several software libraries. For example, the *Computational Geometry Algorithms Library*

Fig. 10 **a** A cell of the subdivision (*shaded*), and a polygonal window W (*dashed*). **b** The positions of the reference point of W as a vertex of the window slides along an edge of the cell. **c** The positions of the reference point of W as an edge of the window slides along a vertex of the cell. **d** The total arrangement is the overlay of the previous two figures



(CGAL)⁴ is an open-source library that provides easy access to efficient and reliable geometric algorithms in the form of a C++ library. It provides a class for arrangements of line segments that offers all necessary methods to deal with the construction and manipulation of arrangements in a transparent and simple way [32]. In particular, the arrangement class provides the two methods our algorithm requires, namely inserting line segments and traversing the faces of the arrangement.

It is also possible to use our models without the arrangement of combinatorially different placements, but the resulting algorithms will no longer give the exact location of the optimal clustering window. We expect that in many cases a nearly optimal placement can be found by sampling.

A sampling-based algorithm for finding an approximate solution can use the same framework as the Geographical Analysis Machine and SaTScan: The cluster window is first placed at a discrete set of regularly spaced grid points covering the subdivision. At locations where the cluster function scores high, we test nearby locations using a finer grid. The refinement can be repeated several times in order to improve the quality of the solution (see for instance [22] for more details). Another simple approach is to rasterize the subdivision, which requires areal interpolation of both the population and case data. We can then test different locations of the cluster windows efficiently using standard image operations on graphics hardware. However, for the areal interpolation we need to fix the distribution of the cases within the regions in advance. Thus, we can use this algorithm with the homogeneous model, but not with the non-homogeneous model.

Our algorithm locates spatial clusters with respect to a cluster window of fixed size. Two important extensions are locating spatial clusters with respect to a variable-size window and locating space-time clusters. Such extensions can be incorporated by adding extra dimensions (that is, window size, time) to the arrangement. Thus the corresponding arrangements of combinatorially different locations are no longer two-dimensional. However, most available practical algorithms that deal with arrangements are restricted to two dimensions. Therefore we suggest to use sampling in this case as well. For instance, in our experiments we use a discrete set of radii for the (circular) cluster window. To locate space-time clusters, a possibility is to use the *space-time permutation scan statistic* framework by [15], but using areas of overlap instead of centroids. The locations for the cluster window should be sampled over the subdivision as described above (and not only placed at centroids).

4 Experiments

In this section we test how well our methods estimate the location of the center of the cluster. We compare our area-based models with centroid-based models.

4.1 Data set

The experiments were run on real regions with artificially generated case data. The region subdivision consists of a map with the 458 municipalities of the Netherlands in

⁴<http://www.cgal.org>

the year 2006. The population of each region consists of the municipality's population in 2006, whereas the cases are randomly generated for each run.

The artificial disease clusters created are circles of 20 km radius, located at a random location in the Netherlands. This radius was chosen so that typically, a small number of regions is completely inside a cluster. The median number of completely covered regions over all runs was six. The size of clusters is also similar to the estimated size of the cluster on real case data in the introductory example (Fig. 2).

For each run, we first assigned relative risks to regions and then assigned 6,000 cases to regions independently from each other. We assigned a relative risk R to the cluster, i.e., a person in the cluster has an R times higher probability to contribute a case. Regions partially covered by the circle were assigned a relative risk corresponding to the situation that the population in the part of the region covered by the circle had the relative risk R , while the remaining part had a relative risk of 1. This assumes a homogeneous distribution of the population in the region. To allow us to focus on a comparison of the estimated cluster location, the combination of cluster radius and relative risks was chosen in such a way that the methods tested are very likely to detect the existence of a cluster. Furthermore, we also comment on the statistical power of the methods. We used three different values for the relative risk R , namely 2.5, 4.0 and 10.0. For a power comparison we also tested a radius of 10 km with $R = 4.0$. For each combination of radius and relative risk, we generated 10,000 test sets. Additionally, we generated 10,000 test sets without a cluster to estimate the statistical power of the methods tested.

4.2 Methods

Since we want to evaluate the effect of area-based in contrast to centroid-based calculations, we use the same cluster function for all methods, namely the likelihood function assuming a Poisson model [17]. This is one of the functions used in *SaTScan*, which is one of the most widely used software packages for disease cluster detection. Thus, the choice of this function allows us to compare our methods directly with methods provided by *SaTScan*.

In order to compare area-based and centroid-based methods, we select four different methods, two belonging to each group. To ease the presentation, we refer to the methods by the following keyword combinations: “area, homogeneous”, “area, non-homogeneous”, “centroid, discrete”, and “centroid, continuous”. The two area-based methods correspond to the homogeneous and non-homogeneous models of Section 2.1. The method “area, non-homogeneous” uses our non-homogeneous model with $g(x) = \min\{ax, 1 - 1/a + x/a\}$, for $a = 3/2$, that is, if a fraction x of the region is covered, a fraction $g(x)$ of the cases can be included. Increasing the parameter $a \geq 1$ allows to include a larger fraction of the cases. For $a = 1$ the function g corresponds to the homogeneous model. With the value $a = 3/2 = 1.5$ that was used in the experiments the model is still close to the homogeneous model, but notably different: it allows to include up to 50% more cases than in the homogeneous model, for the same fraction of area. This is the value chosen for our experiments. Note that larger values of a (i.e. 1.75, 2, etc.) would result in allowing even more cases to be included (i.e. 75% more, 100% more, etc.). Further studies are needed to determine what the best options are for a (and, more generally, for $g(x)$), although we expect them to depend on the specific nature of the data.

The centroid-based methods include whole regions depending on whether their centroids are included or not in the cluster window. The difference between the two methods considered, “centroid, discrete” and “centroid, continuous”, is whether the cluster windows can be placed only at region centroids—thus there is a *discrete* set of possible locations—, or it can be freely *moved* to any point. Both centroid-based methods are part of SaTScan. The “centroid, continuous” method in SaTScan tests a prespecified set of grid points as circle centers. To allow for a more continuous placement of the circle, we re-implemented this method such that after testing a set of grid points the method is re-run with a finer grid around the location of the cluster found in the first step as described in Section 3.5. Thus, our implementation finds the same solution as SaTScan in the first step but then searches locally for better solutions. Since the centroid-based methods report a set of regions as cluster, we take the centroid of the union of these regions as cluster center.

For all methods we use a circular cluster window. For the area-based methods we approximate the circle by a regular 16-gon. Our area-based methods are designed to look for clusters of a fixed radius. Therefore our implementation—which does not know the real cluster radius, of course—tries several different potential cluster radii, and for each run it chooses the result with the highest likelihood.

4.3 Results

The results of the experiments are shown in Tables 1 and 2. Table 1 shows the mean value and the standard deviation of the distance between the real cluster center and the cluster center found by the methods. Table 2 shows the mean value and the standard deviation of the radius of the cluster found by the methods. All methods detected all clusters at a confidence level of 0.05, except for the “area, non-homogeneous” method, which missed 3 of the 10,000 clusters. For these three cases we nonetheless included the location returned by the method.

We performed several additional tests. We used a signed rank test for a pairwise comparison between the methods [25]. For each method we again considered the distance between the real and the estimated cluster center and for every pair of methods we took the difference between these distances. As alternative hypothesis we used that this difference is positive, where the order of the methods is chosen according to the mean values in Table 1. All of the pairwise differences are statistically significant with p -values below 10^{-14} . We also tested for the estimated size of the clusters. All methods slightly overestimate the size of the cluster as can be seen in Table 1. Only for relative risk 2.5 the method “area, non-homogeneous” slightly underestimates the radius (19.94 km). For comparing the statistical power of the methods we performed

Table 1 Distance between the real and the estimated cluster center (mean and standard deviation)

Method	Distance to cluster center (km)		
	$R\ 2.5$	$R\ 4$	$R\ 10$
Centroid, discrete	4.19 ± 2.52	3.84 ± 2.22	3.72 ± 2.11
Centroid, continuous	3.13 ± 1.86	2.71 ± 1.54	2.59 ± 1.43
Area, homogeneous	2.64 ± 1.56	2.34 ± 1.28	2.28 ± 1.22
Area, non-homogeneous	2.51 ± 1.53	1.82 ± 0.98	1.76 ± 0.89

Table 2 Radius of estimated cluster (mean and standard deviation)

Method	Distance to cluster center (km)		
	$R\ 2.5$	$R\ 4$	$R\ 10$
Centroid, discrete	20.81 ± 2.94	21.09 ± 2.47	21.81 ± 2.33
Centroid, continuous	20.60 ± 2.04	20.98 ± 1.79	21.65 ± 1.74
Area, homogeneous	21.29 ± 1.78	21.66 ± 1.61	22.49 ± 1.66
Area, non-homogeneous	19.94 ± 1.54	20.40 ± 1.01	21.21 ± 1.12

an additional set of experiments with radius 10 km and $R = 4.0$. All methods have a power above 0.99, except “area, non-homogeneous” which has a power of 0.91.

We conclude that in our experiments the area-based methods outperform centroid-based methods in determining the center of the cluster. The method “area, homogeneous” is a reliable method to determine the location of the cluster with a high power.

The best estimates for the location of a cluster were obtained by the “area, non-homogeneous” method. However, this method has a low power. We conjecture that this low power is due to the fact that this method has an in-built clustering of the cases. This makes sense in the case that there is a cluster but otherwise not. This method should therefore not be used to detect whether there is a cluster in the data. We suggest to use a different method to test whether there is a cluster. If this test is positive then the “area, non-homogeneous” method can help to get a better estimate for the location of the cluster.

5 Discussion

In this paper we studied the problem of locating clusters in aggregated health data. Unlike most existing approaches that represent the regions as centroids, we proposed models and methods that take the area of the regions into account. We introduced two area-based models: a homogeneous model, which assumes that case data is distributed uniformly within regions, and a non-homogeneous model, which assumes that cases have a tendency to cluster.

Treating regions as geometric objects requires geometric algorithms to handle them efficiently. Based on analyzing the problem geometrically as a placement problem, we provided an exact algorithm that is based on computing the arrangement of possible locations for the cluster window. As opposed to most existing methods, which only try a discrete number of locations for the cluster, our method considers all possible placements.

We performed experiments over a real geographic area using real population data and synthetic case data. In these experiments the proposed area-based methods give a better estimate for the location of a cluster than the corresponding centroid-based method. This paper shows that it is possible, and worthwhile, to take the shape of the regions into account when dealing with aggregated data.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Armstrong MP, Rushton G, Zimmerman DL (1999) Geographically masking health data to preserve confidentiality. *Stat Med* 18:497–525
2. de Berg M, Cheong O, van Kreveld M, Overmars M (2008) *Computational geometry: algorithms and applications*, 3rd edn. Springer
3. Brody H, Rip MR, Vinten-Johansen P, Paneth N, Rachman S (2000) Map-making and myth-making in broad street: the London cholera epidemic, 1854. *Lancet* 356:64–68
4. Cleave N, Brown P, Payne C (1995) Methods for ecological inference: an evaluation. *J R Stat Soc A* 158:55–75
5. Cox LH (1996) Protecting confidentiality in small population health and environmental statistics. *Stat Med* 15:1895–1905
6. Cromley E, McLafferty S (2002) *GIS and public health*. The Guilford Press, New York
7. Den Boer, JW, Verhoef L, Bencini MA, Bruin JP, Jansen R, Yzerman EP (2007) Outbreak detection and secondary prevention of legionnaires disease: a national approach. *Int J Hyg Environ Health* 210:1–7
8. Gilsdorf A, Kroh C, Grimm S, Jensen E, Wagner-Wiening C, Alpers K (2008) Large Q fever outbreak due to sheep farming near residential areas. *Epidemiol Infect* 136(8):1084–1087. doi:10.1017/S0950268807009533
9. Hawley K, Moellering H (2005) A comparative analysis of areal interpolation methods. *Cartogr Geogr Inf Sci* 32(4):411–423
10. Henry K, Boscoe F (2008) Estimating the accuracy of geographical imputation. *Int J Health Geogr* 7(1):3. doi:10.1186/1476-072X-7-3. <http://www.ij-healthgeographics.com/content/7/1/3>
11. Isken L, Kostalova B (2007) Salmonella typhimurium 560, 2006. RIVM. In: *Volksgezondheid Toekomst Verkenning, Nationale Atlas Volksgezondheid*
12. Johnson PJ, Thiede Call K, Blewett LA (2010) The importance of geographic data aggregation in assessing disparities in american indian prenatal care. *Am J Publ Health* 100(1):122–128
13. King G (1997) *A solution to the ecological inference problem*. Princeton University Press, Princeton
14. Kulldorff M (1997) A spatial scan statistic. *Commun Stat Theory Methods* 26:1481–1496
15. Kulldorff M, Heffernan R, Hartman J, Assunção R, Mostashari F (2005) A space-time permutation scan statistic for the early detection of disease outbreaks. *PLoS Med* 2:216–224
16. Kulldorff M, Nagarwalla N (1995) Spatial disease clusters: detection and inference. *Stat Med* 14:799–810
17. Kulldorff M, Tango T, Park PJ (2003) Power comparisons for disease clustering tests. *Comput Stat Data Anal* 42(4):665–684
18. Lam N (1983) Spatial interpolation methods: a review. *Am Cartogr* 10:129–149
19. Luo L, McLafferty S, Wang F (2010) Analyzing spatial aggregation error in statistical models of late-stage cancer risk: a monte carlo simulation approach. *Int J Health Geogr* 9(1):51. doi:10.1186/1476-072X-9-51. <http://www.ij-healthgeographics.com/content/9/1/51>
20. Mount D, Silverman R, Wu A (1996) On the area of overlap of translated polygons. *Comput Vis Image Underst* 64(1):53–61
21. Openshaw S (1984) The modifiable areal unit problem. *CATMOG*, no 38. Geo Books, Norwich
22. Openshaw S, Charlton M, Wymer C, Craft A (1987) A Mark 1 geographical analysis machine for the automated analysis of point data sets. *Int J Geogr Inf Syst* 1:335–358
23. Phillips P, Lee I (2007) Areal aggregated crime reasoning through density tracing. In: *Workshops proc. 7th IEEE internat. conf. data mining*. IEEE Computer Society, pp 649–654
24. Reinbacher I, van Kreveld M, Benkert M (2006) Scale dependent definitions of gradient and aspect and their computation. In: Riedl A, Kainz W, Elmes GA (eds) *Proc. 12th international symposium on Spatial Data Handling (SDH'06)*. Springer, pp 863–879
25. Rice JA (2006) *Mathematical statistics and data analysis*. Duxbury Press
26. Robinson W (1950) Ecological correlations and the behavior of individuals. *Am Sociol Rev* 15:351–357
27. Rogerson P, Yamada I (2008) *Statistical detection and surveillance of geographic clusters*. Interdisciplinary Statistics Series, Chapman & Hall/CRC, Boca Raton
28. Schrijver A (1986) *Theory of linear and integer programming*. John Wiley & Sons, Chichester
29. Snow J (1854) *On the mode of communication of cholera*, 2nd edn. Churchill Livingstone, London
30. Tobler WR (1979) Smooth pycnophylactic interpolation for geographical regions. *J Am Stat Assoc* 74:519–536

31. Waller LA, Gotway CA (2004) Applied spatial statistics for public health data. Wiley Series in Probability and Statistics, Wiley, Hoboken
32. Wein R, Fogel E, Zukerman B, Halperin D (2010) 2D arrangements. In: CGAL user and reference manual, 3.6 edn. CGAL Editorial Board
33. Wolfram S (2003) The mathematica book. Wolfram Media, Incorporated
34. Wolsey LA, Nemhauser GL (1999) Integer and combinatorial optimization. Wiley-Interscience



Kevin Buchin received his Ph.D in computer science from the Free University Berlin in Germany. He was a postdoctoral researcher at Utrecht University, Netherlands, and the Technical University Eindhoven, Netherlands. Currently he is assistant professor at the Technical University Eindhoven. His main research interest lies in geometric and graph algorithms with the focus on geographic applications.



Maïke Buchin received her Ph.D in computer science from the Free University Berlin in Germany. She was a postdoctoral researcher at Utrecht University, Netherlands, and is currently an assistant professor at the Technical University Eindhoven, Netherlands. Her research interest lies mostly in geometric algorithms with the focus on algorithms for GIS.



Marc van Kreveld received his Ph.D. in computer science from Utrecht University in the Netherlands. He was a postdoctoral researcher at the school of computer science of McGill University, Montreal, Canada, from 1992 until 1993. Since 1993 he works at the department of information and computing sciences of Utrecht University, currently as an associate professor. His main research interests are computational geometry and algorithms for GIS.



Maarten Löffler received his Ph.D in computer science from Utrecht University in the Netherlands. He is a postdoctoral researcher at the Bren School of information and computer sciences of the University of California, Irvine. His research interest lies mostly in computational geometry, and in related fields such as geographical information analysis and graph drawing.



Jun Luo received his Ph.D in computer science from the University of Texas at Dallas, USA, in 2006. Then he spent two years as postdoc in Utrecht University, the Netherlands. He is currently an assistant professor in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interests are in algorithm design and analysis, computational geometry, GIS and data mining.



Rodrigo I. Silveira received his Ph.D in computer science from Utrecht University in the Netherlands. He is a postdoctoral researcher at the Universitat Politècnica de Catalunya, Spain. His main research interests are computational geometry, geographic information science and graph drawing.