# Low-Resolution Summaries for Spatial Point Data With Shifted Grid Cells

Category: Research


Figure 1: A point cloud showing ca. 2 000 locations in Europe and groupings into equal-sized cells for summarization with our technique (A, left) and a grid (A, right). Better use of negative space reveals spatial features and makes it possible to distinguish, e.g., Great Britain from Norway. B) Comparison of (A) with 7 summaries showing median value per cell.


## Abstract

Choropleth maps are useful to investigate global patterns of a single variable, and also multiple variables or variables over time, when used in small multiples. Grids can be seen as a choropleth-like technique for point data, with the advantage that they do not assume what advanced computations may or may not be carried out with the data at hand and how. They have the main drawback of not looking much like the point cloud, especially in low resolutions, the smaller size of which being beneficial for small multiples. We propose an adaptive approach to construct a grid-like structure from point data. Our experiments suggest that it resembles the point cloud better and is more efficient with regard to the number of cells used.


**Index Terms:** Human-centered computing—Visualization—Visualization techniques;

## 1 The Problem

Choropleth maps are useful for analysis tasks such as investigating global patterns of a single variable [6]. When shown side by side, comparison of global patterns is enabled (e.g., Fig. 1B). Small multiples of choropleth maps then allow comparison of many spatial variables, possibly over time if the dataset is spatio-temporal. Choropleth maps are usually created from existing polygons depicting the areas in question, but the analysis tasks are applicable to point data, too. In this paper we consider if and how we can build a choropleth map from point data, without use of metadata, like zip codes or assumptions about point volume, distribution or data domain. Such design restrictions might at first seem unnecessary, needlessly complicating the issue, or even unorthodox as visualizations are usually designed for a particular combination of data, users and tasks [13, 14]. But it is beneficial, e.g., when designing a visual analytics (VA) solution around a domain-independent numeric method. A widely used example for that is, e.g., Principal Component Analysis. As domain-independent implies, such methods allow data analysis on any suitable dataset, be it from geosciences or psychology. In such a case, visualization designers need to find the lowest common denominator that suits as many domains and datasets as possible. If designers do not, their designs will be less useful for datasets where the assumptions do not hold.

We describe our requirements using the design triangle by Miksch and Aigner [13]. *Users* are data analysts in any domain. They work with multivariate spatial point *data*, where all variables are numeric and we expect 2–100 variables. Their *task* for the purpose of this paper is to get a coarse overview of a single variable in space, i.e., analysts are interested in where values exist, areas of low and high value, whether values change smoothly in one direction, etc. Furthermore, they want to compare these global patterns between variables. Hence we are looking for a way (adhering to the restrictions above) to show many individual variables in low resolution, so that they can be compared, such that patterns of single variables are still visible.

Our contributions in this paper are specifically:

- We propose an alternative choropleth-like technique for spatial point data, similar to grids, that is suited for low-resolution renderings. We explain our focus on grids in Sect. 3.
- We provide a simple greedy algorithm to obtain such a visualization in Sect. 4.
- We evaluate our technique by computing two metrics and comparing them to grids (Sect. 5). These experiments suggest that cell arrangements from our technique resemble the point cloud better and use less cells.

## 2 Related Work

Spatial point data is relatively common in the visualization literature. Hogräfer et al. [9] recently surveyed and classified map-like visualizations, which categorizes grids as an *imitation* technique using

geometric hulls. He et al. [8] specifically survey point data, but focus on multivariate analysis, which is not our goal in this paper.

Goodwin et al. [6] propose a framework to visualize variables across scales and geography. In that, our analysis goal is the *Global-Uni* case as we are interested in global patterns of a single variable. The authors suggest a choropleth map to support this analysis. How to compute such a map using points instead of areas and without metadata such as political boundaries is the topic of our paper.

Zhou et al. [21] propose the point grid map. Based on a grid map [12], which is commonly used for area data, they use it for point data. Marks are aligned on a grid such that they retain directional relations. Since we do not know in advance the number of points or how they are distributed, there is a chance this visualization would produce too small marks. This is also the case when using pixel-oriented techniques [10], which show each datapoint as a single pixel. If executed well, these can support our analysis tasks, but they require at least as many pixels as data points, which might be too many, and show spatial patterns in an unintuitive way due to the prevailing use of space-filling curves.

Attribute signatures by Turkay et al. [18] is close to what we want, as it shows values of variables along a user-drawn curve as small multiple line charts. This approach is flexible and works for any point data, but requires both a larger supporting map and the analyst to choose a curve. For our purposes, this offsets its benefits.

Several techniques exist that deal with density of categorical point data. Micro diagrams [7] show the distribution of categories as pie charts via small multiples in geographical space. Similarly, TopoGroups [19] show the distribution of categories on the outline of point clusters. Phoenixmap [20] encloses each category with a concave hull and encodes the point density with line thickness on the hull. BinSq [2] aligns points on a quadtree and ensures that the perceived density of categories is accurate by removing points as necessary. As all approaches focus on categorical data and not numerical, and density instead of value, they cannot be used in our case.

## 3 Design Choices

As outlined in Sect. 1 we look for a visualization that assumes the least about what can be done with the data at hand and supports global pattern analysis even in low-res renderings, so that it can be used in small multiples, which enables comparison over variables or time. We discuss alternative approaches to our technique, that we did not pursue, in the following. These include showing raw data as points or lines, data aggregation without grids, and data mining.

The most straightforward approach is to show raw data. Dot maps, where each point is encoded by one mark and its value by color or size, are a common choice for point data. In our case they cannot be used as-is, because many or large points in small space lead to occlusion (Fig. 2A1).

Another option are isolines, where one curve joins points of equal value, and multiple curves are combined, thereby representing multiple values. When the curve's area is filled with color, we obtain a heat map. Such isolines are often computed with the Marching Squares algorithm, which requires a regular grid. Hence as a pre-processing step the variable is assumed to be a scalar field and interpolated onto such a grid. In our very general setting, we do not know if interpolation is admissible or useful, and if so, which interpolating function should be used.

The most simple approach for aggregation is to overlay a regular grid of fixed size. Then, desired summary statistics such as count, mean, median, or variance can be computed per cell. The advantage of grids with summary statistics is they do *not* assume a scalar field. But two disadvantages of grids make them less attractive: First, spatial features tend to be barely visible in low resolutions, as cells are perfectly aligned and no negative space is allowed between them (see Fig. 1A, right). Second, for summary statistics we ideally want

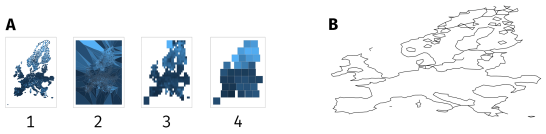



Figure 2: A) Issues with design alternatives in small sizes: Dot map (1) suffers from overdrawing, Voronoi diagram (2) and quadtree (3) are too detailed. Our approach (4) is shown for comparison. B) An $\alpha$-shape of the point cloud where all of continental Europe and Great Britain are joined in one hull.

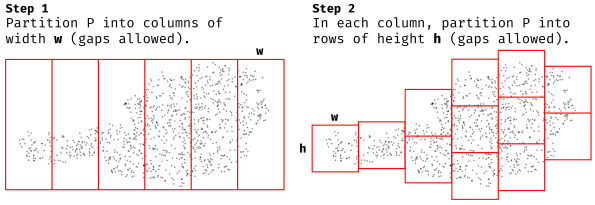



Figure 3: The idea for our technique. Neither columns nor rows are required to be contiguous. Both steps are carried out by the same function `place_intervals` (algorithm 1).

many points. But because grid cell locations are mostly independent from point locations, more cells than necessary are used, lowering the average number of points per cell.

There are many techniques that similarly subdivide the plane into cells, examples would be quadtrees (Fig. 2A3), kd-trees, Voronoi diagrams (Fig. 2A2) or Delaunay triangulations [3]. However, they do not improve upon the mentioned issues: They do not group multiple points (Voronoi, Delaunay), or generate cells of varying size that may be too small in low resolutions (quadtree, kd-tree). As none of the approaches allow negative space, it would be implicitly visible as larger cells and not show spatial features well.

Other means to group multiple points exist, like algorithms to compute various geometric hulls [3, 4, 16]. But even if some may in principle return multiple hulls, we might for a given unlucky dataset end up with one big hull enclosing all points (cf. Fig. 2B). Then the summary statistics equal the global statistics. A similar argument can be made about using density-based clustering algorithms such as DBSCAN [5]. Grouping not in spatial space but data space, e.g., with k-means [11], is a possibility, but more useful when the goal is only multivariate analysis.

In conclusion, regular grids seem to be our best option, but still have drawbacks. How we intend to overcome these is described in the next section.

## 4 Our Approach

In Sect. 3 we identified grids as our best option, but with two drawbacks:

D1 Spatial features of the point cloud are not retained well. These can be, e.g., holes, islands, or concave sections along the outline.

D2 Grids are less efficient than necessary, i.e., use too many cells.

The idea is now to overcome these by relaxing the definition of a grid. If we allow negative space between cells, we would expect this leads to better resembled spatial features, as they are often defined by it. For example, holes are literally negative space, islands have lots of it around them and concave sections can be thought of holes on the boundary of a polygon.

There is a simple way to compute such a cell arrangement, outlined in Fig. 3. As input we take the point cloud $P$ and the desired cell dimensions $w, h$. First, the X coordinates in $P$ are segmented into columns of width $w$, with gaps if possible. Using the same logic, the Y coordinates of points in each column are then again segmented into rows of height $h$. The left column edges, top row edges and cell dimensions together define the cell arrangement. We use a heuristic to determine the processing order of dimensions: The longer side of $P$'s bounding box is processed first to hopefully avoid cases where, e.g., a tall point cloud is just too wide for a single column.

### 4.1 The Algorithm

Because we use the dimensions independently for our computation, the problem boils down to placing intervals of length $k$ over points on a line. Intervals are left-closed and right-open. We use a greedy algorithm where we scan the line from left to right and insert a $k$-interval as late as possible, move to the end of the just placed interval and repeat. The intervals contain all points as required, but are not centered over their points, which distorts spatial features. Hence we proceed with a postprocessing loop in which we try to center the intervals, or groups thereof, over their points until no improvement can be made. This function is called `place_intervals` and its pseudocode is shown in algorithm 1. Our implementation is available as supplementary material.

**Data:** List of points $P$, interval length $k$
**Result:** List of interval left endpoints $L$
Sort $P$, set $L$ to empty list
**while** $\exists p \in P$ *that is unprocessed* **do**
  Let $p$ be smallest value in $P$
  Place interval $l$ with left endpoint at $p$
  Calculate trailing space, i.e., the difference between largest $q \in P$ in $l$ and $l$'s right endpoint
  Move $l$ to the left, at most $1/2$ of trailing space
  Snap to previous interval's right endpoint if gap is too small
  Add $l$ to $L$, mark all points in $l$ as processed
**end**
**repeat**
  Find groups $G$ of contiguous intervals in $L$
  **for** $g \in G$ **do**
    Let $l_g$ be the union of all intervals in $g$
    **if** *there is a gap to the interval left of* $l_g$ *and there is more trailing than leading space in* $l_g$ **then**
      Move $l_g$ to the left
    **end**
  **end**
**until** *no* $l_g$ *was moved*
**return** $L$

Algorithm 1: Pseudocode for `place_intervals`

## 5 Experiments

In the following, we investigate if our proposed technique improves the drawbacks mentioned in Sect. 4. Does its cell arrangement reflect spatial features better (cf. D1 in Sect. 4) and does it use less cells (cf. D2) than a grid? To answer these questions, we compared two quality metrics for grids and our technique using realistic and diverse point clouds. We provide the source code used for our experiments as supplementary material.

Regarding to D1 we expect that, if our technique shows spatial features better, then its outline would be more similar to the point cloud's outline than the outline of the grid arrangement. We use a popular similarity metric for polygons, the Fréchet distance [1], to compare outlines. The cell arrangement outline is obtained by taking the union of all adjacent cells. The point cloud outline is obtained by a concave hull algorithm based on ideas by Park and Oh [16], implemented in `https://github.com/mapbox/concaveman`.

(a) Thailand (b) Poland (c) Mauritania (d) Panama

Figure 4: The four countries used in our experiments.

Regarding to D2 we count the cells to determine if our technique is more efficient.

We wanted realistic point cloud shapes to increase confidence in our results. So we picked four countries which we think have complex and diverse outlines (Fig. 4): Mauritania has some near-45-degree angles, Panama has large and small concave sections, Thailand has two larger landmasses connected by an isthmus and Poland is quite round. Country outlines were obtained from R's `rnaturalearth` package and projected to the plane in Equal Earth projection [17]. To obtain point clouds, we uniformly sample points into their outlines such that all point clouds have the same density.

We use square cells with decreasing size defined by the point cloud's diameter $D$ (maximum distance between two points). The side length $k = D/i$ ranges from $1/2$ to $1/10$th of the diameter. We stop at $1/10$th as our focus is on low resolutions and because with smaller cells we would at some point obtain multiple outlines that require more effort to compare as they need matching.

There is one additional problem to take care of: A grid for a point cloud is not uniquely defined by cell dimensions, it also matters where the origin point is. We figured that the bottom left of $P$'s bounding box is a natural place to put it, because it is simple and quick. To reduce the impact of our choice on the results we aggregate over five random offsets between 0 and $k/2$ subtracted from it.

### 5.1 Results

Fig. 5 shows the results of our experiments. The Y axis encodes the relative value of the metric, the X axis is the cell size with decreasing cells from left to right, rows are the metrics and columns the countries. If a line is below the horizontal black line, our technique performed better.

The large improvement in Fréchet distance with big cells stems from unfavorable origins for grids which lead to additional poorly aligned cells (Fig. 6, top row). After that, from maybe $k = D/5$, follows a phase in which the grids get closer to our technique, but are still worse (Fig. 6, second row).

At this point we would like to point out something the Fréchet distance does not show. It computes the smallest sufficient "leash" so a human and their dog can walk on either curve. As soon as there is anywhere a large distance between cell and point cloud outline, our technique and grids will have similar Fréchet distance, independent of how well the rest of the outline fits. Examples are shown in the two bottom rows of Fig. 6, where our technique misplaced only one (bottom row) or a two (third row) cells but arguably tracks the rest of the point outline better than the grid.

Furthermore, the two metrics should not be analyzed in isolation. We ran experiments only up to $k = D/10$ cell size in order to compare single outlines. This does not mean that grids perform the same as our technique from this value on, as the top row of Fig. 5 might suggest. Because the number of cells seems to be consistently lower

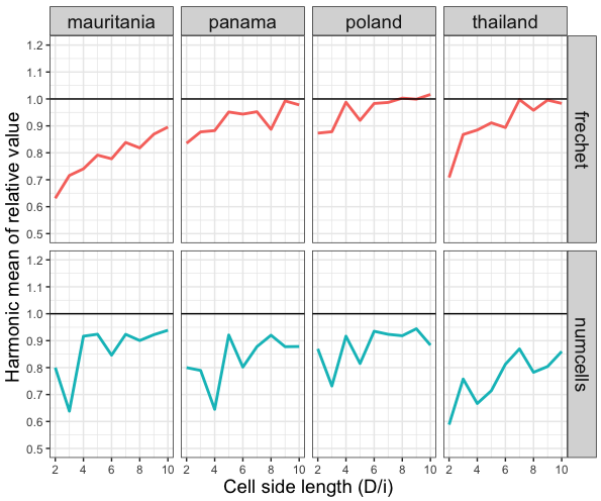



Figure 5: Evaluation results for our technique. If lines are below the black line, our technique performs better than a grid.

(Fig. 5, bottom row), our visualization will show more negative space, which is important information.

## 6 Discussion and Conclusion

We presented a visualization technique for spatial point data that is related to grids. It works by greedily placing fixed-length intervals in each dimension separately, thus skipping over empty space. Although we presented our technique with geospatial points throughout this paper, it is applicable to any point cloud in the plane. Our experiments with low resolutions suggest it produces better cell arrangements in terms of similarity to the point cloud outline and number of cells, alleviating two drawbacks we identified with our preferred option in Sect. 3. Other advantages are that analysts do not need to pick an origin point and can defer the choice of dimension order to a heuristic, leaving only the cell size as a parameter. Since grouping points inherently suffers from the modifiable areal unit problem [15], we do not expect to introduce any new biases with our approach. If anything our technique makes nonsensical groupings, e.g., points on opposite coasts of a sea, less likely.

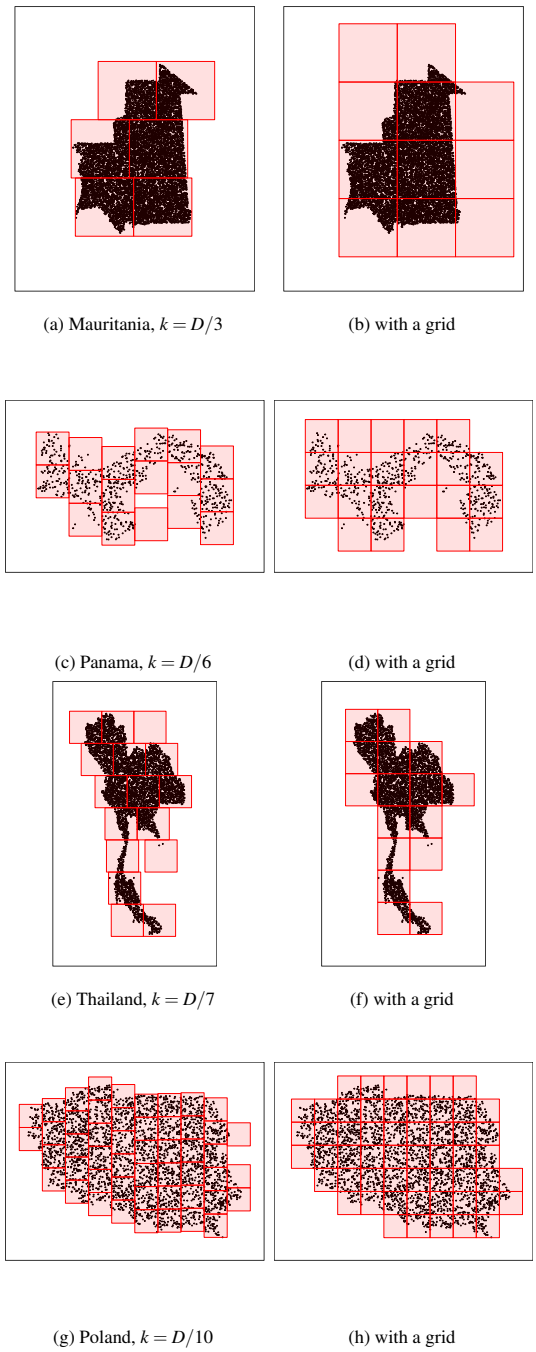
In future work we would like to conduct a user study to validate if people find cell arrangements by our technique more representative of the point cloud, as the Fréchet distance is smaller than with grids but can still be relatively large. It would also be interesting if our technique can help with specific tasks, such as lookups of known locations ("which cell contains London?"). We would like to test if we can remove the dependency on the point set's orientation, e.g., by running our algorithm along principal components. Finally, for effective use in practice it would be helpful to support or automate the selection of cell dimensions.

## References

[1] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 05(01n02):75–91, Mar. 1995. doi: 10.1142/S0218195995000064

[2] A. Chua and A. V. Moere. BinSq: Visualizing geographic dot density patterns with gridded maps. *Cartography and Geographic Information Science*, 44(5):390–409, Sept. 2017. doi: 10.1080/15230406.2016.1174623

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer, 2008.



(a) Mauritania, $k = D/3$ (b) with a grid

(c) Panama, $k = D/6$ (d) with a grid

(e) Thailand, $k = D/7$ (f) with a grid

(g) Poland, $k = D/10$ (h) with a grid

Figure 6: Examples from the experiments. Our technique is on the left and a grid with same cell size on the right.

[4] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, July 1983. doi: 10.1109/TIT.1983.1056714

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc.\ KDD*, pp. 226–231. AAAI Press, Palo Alto, CA, USA, Aug. 1996.

[6] S. Goodwin, J. Dykes, A. Slingsby, and C. Turkay. Visualizing Multiple Variables Across Scale and Geography. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):599–608, Jan. 2016. doi: 10.1109/TVCG.2015.2467199

[7] M. Gröbe and D. Burghardt. Micro diagrams: Visualization of categorical point data from location-based social media. *Cartography and Geographic Information Science*, 47(4):305–320, Mar. 2020. doi: 10.1080/15230406.2020.1733438

[8] X. He, Y. Tao, Q. Wang, and H. Lin. Multivariate spatial data visualization: A survey. *Journal of Visualization*, 22(5):897–912, Oct. 2019. doi: 10.1007/s12650-019-00584-3

[9] M. Hogräfer, M. Heitzler, and H.-J. Schulz. The State of the Art in Map-Like Visualization. *Computer Graphics Forum*, 39(3):647–674, June 2020. doi: 10.1111/cgf.14031

[10] D. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, Jan. 2000. doi: 10.1109/2945.841121

[11] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Mar. 1982. doi: 10.1109/TIT.1982.1056489

[12] W. Meulemans, M. Sondag, and B. Speckmann. A Simple Pipeline for Coherent Grid Maps. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1236–1246, Feb. 2021. doi: 10.1109/TVCG.2020.3028953

[13] S. Miksch and W. Aigner. A Matter of Time: Applying a Data-users-tasks Design Triangle to Visual Analytics of Time-oriented Data. *Computers & Graphics*, 38:286–290, Feb. 2014. doi: 10.1016/j.cag.2013.11.002

[14] T. Munzner. *Visualization Analysis and Design*. A.K. Peters Visualization Series. CRC Press, 2014.

[15] S. Openshaw and P. J. Taylor. A Million or so Correlation Coefficients: Three Experiments on the Modifiable Areal Unit Problem. *Statistical Applications in the Spatial Sciences*, pp. 127–144, 1979.

[16] J.-S. Park and S.-J. Oh. A New Concave Hull Algorithm and Concaveness Measure for n-dimensional Datasets. *Journal of Information Science & Engineering*, 28(3):587–600, May 2012.

[17] B. Šavrič, T. Patterson, and B. Jenny. The Equal Earth map projection. *International Journal of Geographical Information Science*, 33(3):454–465, Mar. 2019. doi: 10.1080/13658816.2018.1504949

[18] C. Turkay, A. Slingsby, H. Hauser, J. Wood, and J. Dykes. Attribute Signatures: Dynamic Visual Summaries for Analyzing Multivariate Geographical Data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2033–2042, Dec. 2014. doi: 10.1109/TVCG.2014.2346265

[19] J. Zhang, A. Malik, B. Ahlbrand, N. Elmqvist, R. Maciejewski, and D. S. Ebert. TopoGroups: Context-Preserving Visual Illustration of Multi-Scale Spatial Aggregates. In *Proc.\ CHI*, pp. 2940–2951. ACM, New York, NY, USA, May 2017. doi: 10.1145/3025453.3025801

[20] J. Zhao, X. Liu, C. Guo, C. Qian, and Y. V. Chen. Phoenixmap: An Abstract Approach to Visualize 2D Spatial Distributions. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2000–2014, Mar. 2021. doi: 10.1109/TVCG.2019.2945960

[21] M. Zhou, J. Tian, F. Xiong, and R. Wang. Point grid map: A new type of thematic map for statistical data associated with geographic points. *Cartography and Geographic Information Science*, 44(5):374–389, Sept. 2017. doi: 10.1080/15230406.2016.1160797