Fast flow-based algorithm for creating density-equalizing map projections

Michael T. Gastner[a,1], Vivien Seguy[b], and Pratyush More[a]

[a]Division of Science, Yale-NUS College, Singapore 138527; and [b]Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

Edited by Michael F. Goodchild, University of California, Santa Barbara, CA, and approved January 11, 2018 (received for review July 16, 2017)

Cartograms are maps that rescale geographic regions (e.g., countries, districts) such that their areas are proportional to quantitative demographic data (e.g., population size, gross domestic product). Unlike conventional bar or pie charts, cartograms can represent correctly which regions share common borders, resulting in insightful visualizations that can be the basis for further spatial statistical analysis. Computer programs can assist data scientists in preparing cartograms, but developing an algorithm that can quickly transform every coordinate on the map (including points that are not exactly on a border) while generating recognizable images has remained a challenge. Methods that translate the cartographic deformations into physics-inspired equations of motion have become popular, but solving these equations with sufficient accuracy can still take several minutes on current hardware. Here we introduce a flow-based algorithm whose equations of motion are numerically easier to solve compared with previous methods. The equations allow straightforward parallelization so that the calculation takes only a few seconds even for complex and detailed input. Despite the speedup, the proposed algorithm still keeps the advantages of previous techniques: With comparable quantitative measures of shape distortion, it accurately scales all areas, correctly fits the regions together, and generates a map projection for every point. We demonstrate the use of our algorithm with applications to the 2016 US election results, the gross domestic products of Indian states and Chinese provinces, and the spatial distribution of deaths in the London borough of Kensington and Chelsea between 2011 and 2014.

cartography | data visualization | statistical analysis | computer graphics

A guideline for displaying statistical data in a diagram is the "area principle": Each part of the diagram should have an area in proportion to the number it represents (1). For many categorical data, a bar chart is a simple visualization method that satisfies the area principle. For example, if our data are the electors that voted for the US president in December 2016, we can categorize the electors by US state. Every bar in Fig. 1, *Top* corresponds to a state that sent at least one Republican elector to the Electoral College. In Fig. 1, *Bottom*, the bars show the states with Democratic electors. The colors chosen for the bars are the traditional red for Republicans and blue for Democrats. Because the bar chart satisfies the area principle, the election is won by the color that occupies more area, which is evidently red in this example.*

Although a bar chart is often a suitable visualization tool, it cannot reveal the spatial pattern behind the data. The bar chart in Fig. 1 lacks the information where the states are located: Neighboring bars do not necessarily correspond to states that are geographic neighbors. If we want to visualize how the states fit together in real space, we need a different approach. The common alternative is to show a map such as Fig. 2*A*, where we use an Albers equal-area conic projection for the contiguous United States to produce their familiar geographic outline. We add Alaska and Hawaii, suitably rescaled, below the map of the contiguous United States. Each state is colored with either red or blue, depending on the party affiliation of the electors.†

The map in Fig. 2*A* accurately shows the relative area and position of each state. However, it does not obey the area principle of statistics. For example, Montana (abbreviated MT in Fig. 2*A*) covers more than 2,000 times the area of Washington, DC, but both regions have the same number of electors. On aggregate, Republican electors won 74% of the US area in square kilometers, but had only 57% of the vote share in the Electoral College. So, Fig. 2*A* has the opposite problem of Fig. 1 where we satisfied the statistical area principle, but conveyed no information about the states' locations. One might suspect that showing the locations and simultaneously satisfying the area principle are as impossible as squaring the circle. Fortunately, however, there is a visualization method, known as a cartogram, that can tackle this challenge (2, 3).

After a brief review and classification of cartograms, we introduce a technique that produces cartograms of a quality comparable to the most popular technique currently in use: the diffusion cartogram (4). The technique proposed in this article solves a completely different set of equations so that the computation

Significance

Geographic maps are a popular means to visualize spatial statistics. Conventionally, each map region is displayed with an area proportional to the actual land area. But equal-area maps can grossly misrepresent demographic data: Densely populated cities should be given more prominence than large, but sparsely populated territories. Cartograms solve this problem by rescaling map regions in proportion to, for example, population or gross domestic products. Until now, it has generally been cumbersome or slow to calculate map projections for contiguous cartograms. Here we describe and benchmark a fast flow-based algorithm that computes cartograms in a matter of seconds, yet maintains the strengths of previous methods—a development which may lead to a more widespread adoption of cartograms.

Author contributions: M.T.G. designed research; M.T.G., V.S., and P.M. performed research; M.T.G. analyzed data; and M.T.G., V.S., and P.M. wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission.

[1]To whom correspondence should be addressed. Email: michael.gastner@yale-nus.edu.sg.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1712674115/-/DCSupplemental.

*Because of peculiarities in the US electoral system, the Electoral College is not an exact representation of the proportion of votes cast by the US population at large. The Republican candidate Donald Trump was elected as US president despite losing the popular vote to the Democratic candidate Hillary Clinton. We show a cartogram of the popular vote distribution in *SI Appendix*, section 1.

†The only exception is Maine which applies the "congressional district method": Although the majority in Maine voted for the Democratic candidate Hillary Clinton, the Republican candidate Donald Trump still gained one electoral vote for winning the second congressional district (abbreviated as ME2 in Fig. 2*A*).

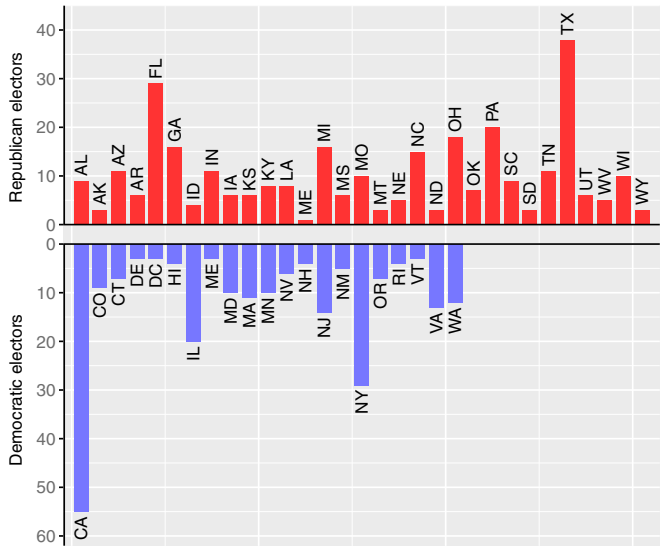

Fig. 1. A bar chart of the Electoral College vote for the US president in 2016. This diagram satisfies the area principle: The area of each bar is proportional to the number of electors. However, from this bar chart it is not clear where states are located geographically.

can finish within a fraction of the previously needed time. We benchmark our algorithm with data from the United States, India, China, and the London borough of Kensington and Chelsea to demonstrate that our method accurately satisfies the area principle and generates visually pleasing cartograms.

Classification of Cartogram Methods

In a cartogram, regions are deformed such that their areas are equal to statistical data such as population, votes in an election, or gross domestic product. An example, showing the Electoral College on a cartogram, is the diagram in Fig. 2*B* which we adapted from Wikipedia (https://commons.wikimedia.org/wiki/File:Cartogram%E2%80%942012_Electoral_Vote.svg). Similar cartograms have been shown in the news media (5, 6). Here each elector is represented by a small square. The squares are then positioned with two objectives in mind. First, the shapes on the cartogram should resemble those on the map in Fig. 2*A*. Second, the set of neighboring states in Fig. 2 *A* and *B* should be the same. Satisfying both objectives is not trivial. A careful comparison with Fig. 2*A* shows that, for example, Arizona (AZ) and Texas (TX) incorrectly appear as neighbors in Fig. 2*B*. On the other hand, the geographic neighbors Colorado (CO) and Nebraska (NE) have been separated in Fig. 2*B* to make space for other states in the vicinity.

For certain applications, it is perfectly acceptable that neighboring states are split apart. As long as the areas of the states are proportional to the number of electors, such representations are called noncontiguous cartograms (7). Dorling's circular cartograms are good examples of noncontiguous cartograms that, while not strictly maintaining the topology, indicate where the represented regions are located (8). Contiguous cartograms, by contrast, not only rescale the regions, but also keep the topology intact (i.e., neighbors on the map are neighbors on the cartogram and vice versa).

The methods that have been proposed for generating contiguous cartograms fall into two distinct categories. The first group consists of algorithms that operate only on the boundaries of regions (9–17). Each region is represented by one or multiple polygons. The input to these algorithms is a finite number of polygon corners $(x_1, y_1), \ldots, (x_n, y_n)$. Here (x_i, y_i) is a projection of the longitude and latitude, usually obtained from a conventional projection (e.g., plate carrée or an equal-area projection). The algorithm generates transformed polygon coordinates $\mathbf{T}(x_1, y_1), \ldots, \mathbf{T}(x_n, y_n)$. For the first group of algorithms, these n points are in fact the only output and, hence, we refer to them as "boundaries-only" algorithms. In other words, boundaries-only methods do not transform points that are in the interior of a polygon. For example, on a US state cartogram (such as Fig. 2*B*) we would not be able to uniquely locate a state capital such as Austin, TX, because it is far from any state border. One might symbolically place all capitals at the centroid of the corresponding polygon, but some centroids might be outside the polygon if it is concave or contains holes (e.g., lakes or enclaves). The situation is even more complicated if we want to represent multiple distinct points or lines (e.g., rivers or

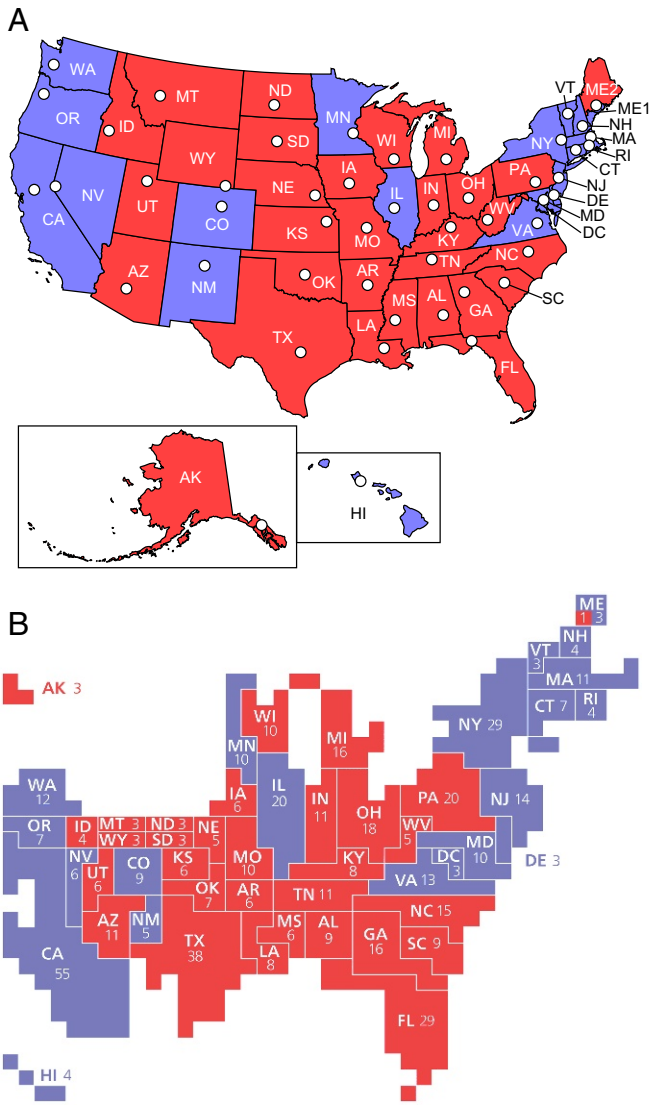

Fig. 2. (*A*) A conventional map projection (here an Albers projection) clearly shows the location of each state, but violates the area principle: States that occupy a large area do not necessarily have a large number of electors. (*B*) A cartogram of the 2016 Electoral College satisfies the area principle. Each elector is represented by a small square at the approximate location of the elector's home state. Cartograms such as these are popular in the media, but are not map projections in a strict sense: There is no continuous mathematical function that transforms coordinates of longitude and latitude to coordinates on the cartogram. For example, in *B* it is not possible to identify the location of the state capitals (indicated by white circles in *A*).

roads) inside a state as distinct objects on a boundaries-only cartogram.

The second group of contiguous cartogram algorithms approaches the problem from a different point of view by producing a continuous transformation **T** for the entire continuous set of longitudes and latitudes on the input map, including coordinates that are not on a boundary (4, 18–22). We refer to this group as "all-coordinates" algorithms. Generating the map projection **T** for all longitudes and latitudes can be computationally more demanding than shifting only the boundary coordinates. In fact, for applications where only the boundaries are of interest—as is the case for the US election map—the boundaries-only algorithms can give adequate results. However, the run time of these discrete algorithms typically increases steeply with the number of corners. As a result, they often rely on coarse-grained input to gain speed, for example by removing Michigan's Upper Peninsula from the US map (12, 13, 16). If we wish to show data that are resolved at a scale much finer than the polygons to be displayed [e.g., graticules for a fine, spatially regular grid (23) or individual addresses], the all-coordinates algorithms usually outpace their boundaries-only counterparts.

In this article, we describe an all-coordinates algorithm that needs only a few seconds to produce the complete projection **T** for realistic input. Knowing **T** will allow us to show the positions of all US state capitals with respect to the states' boundaries (Fig. 3*B*) and the coordinates of individual death cases in London (Fig. 4 *B* and *C*).

Previous All-Coordinates Methods to Produce a Cartogram Projection

For the sake of concreteness, let us assume that we want to make a cartogram whose areas are proportional to the population. We define the population density as the function $\rho(x, y)$ such that a small rectangular area element with the corners $(x \pm dx/2, y \pm dy/2)$ contains the population $\rho(x, y)\, dx\, dy$. Some data allow us to model $\rho(x, y)$ with variations on fine spatial scales. (Our application below to the mortality statistics of Kensington and Chelsea belongs to this category.) In other cases, it is more natural to model $\rho(x, y)$ as a piecewise constant function. For example, California's 55 electors can be represented by a constant density in this state equal to the number of electors divided by the state's geographic area.

An accurate cartogram must project the rectangle $(x \pm dx/2, y \pm dy/2)$ onto a quadrilateral $\mathbf{T}(x \pm dx/2, y \pm dy/2)$ in such a way that the area of the quadrilateral is proportional to $\rho\, dx\, dy$. In other words, we are looking for a 2D function $\mathbf{T} = (T_x, T_y)$ such that $\rho(x, y)\, dx\, dy = \bar{\rho}\, dT_x\, dT_y$, where $\bar{\rho}$ depends neither on x nor on y. Such a transformation **T** is called a density-equalizing projection. Taking the limits $dx \to 0$ and $dy \to 0$ and assuming that **T** is differentiable, we obtain the condition (4, 18)

$$\frac{\partial T_x}{\partial x}\frac{\partial T_y}{\partial y} - \frac{\partial T_x}{\partial y}\frac{\partial T_y}{\partial x} = \frac{\rho(x, y)}{\bar{\rho}}, \qquad [1]$$

which is called a prescribed Jacobian equation (24, 25). For convenience, we choose the constant $\bar{\rho}$ to be the spatially averaged density so that the total mapped area is preserved.

Eq. **1** alone does not uniquely specify **T** because it is only one single equation for the two unknowns T_x and T_y (20). As a consequence, there are infinitely many different strategies to obtain a density-equalizing projection **T**. In practice, however, only a few methods are computationally efficient, produce attractive graphics, and are independent of the choice of coordinate axes. Most of the methods that have been proposed in the literature are based on physical analogies. A common metaphor is to view the undistorted input map as a rubber sheet. Forces or stresses act on the rubber sheet such that the points move toward equilibrium positions that satisfy Eq. **1** (19, 22). Although such mechanical metaphors make intuitive sense, there is no direct physical connection between force and area. Therefore, it is not immediately obvious how the forces should be chosen as functions of $\rho(x, y)$ to ensure that Eq. **1** is valid. Some methods treat the term "force" in a less literal sense so that the area constraints are more explicitly part of the equations (9, 11). However, these algorithms must take special care to avoid topological errors (e.g., regions that are flipped or boundaries that intersect themselves) during the relaxation of the forces. Another method, based on neural networks, starts by placing sample points on a regular grid (26). During the training of the network, the samples are attracted toward regions of high density to mimic the population distribution. Their final positions define a mapping which can produce a cartogram by considering its inverse. However, a large number of sample points are necessary to produce smooth boundaries.

An alternative physical metaphor is to view the process that generates the cartogram as the flow of a fluid. In this analogy, we think of the map as a Petri dish covered with a thin layer of water. In an experiment, we would model the population density $\rho(x, y)$ by injecting small particles with spatially varying concentrations into the water layer. The particles then diffuse across the

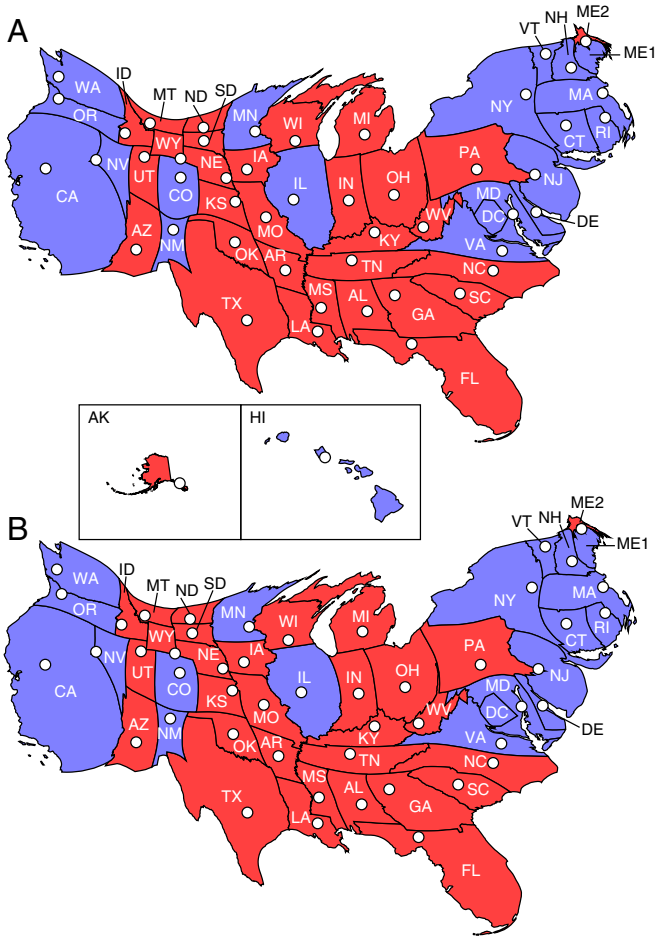

Fig. 3. (*A* and *B*) The 2016 US Electoral College vote represented on cartograms generated with (*A*) the diffusion algorithm of ref. 4 and (*B*) the alternative flow-based algorithm based on Eqs. **4**–**7**. *Insets* for Hawaii and Alaska apply to both *A* and *B* as these regions' areas match both cartograms. All areas differ by <1% from their target values (i.e., the proportion of votes in the Electoral College). Cartograms *A* and *B* differ in detail, but appear remarkably similar, considering that generating *B* needs only 2.5% of the time required by the diffusion algorithm. The white circles indicate the positions of the state capitals.

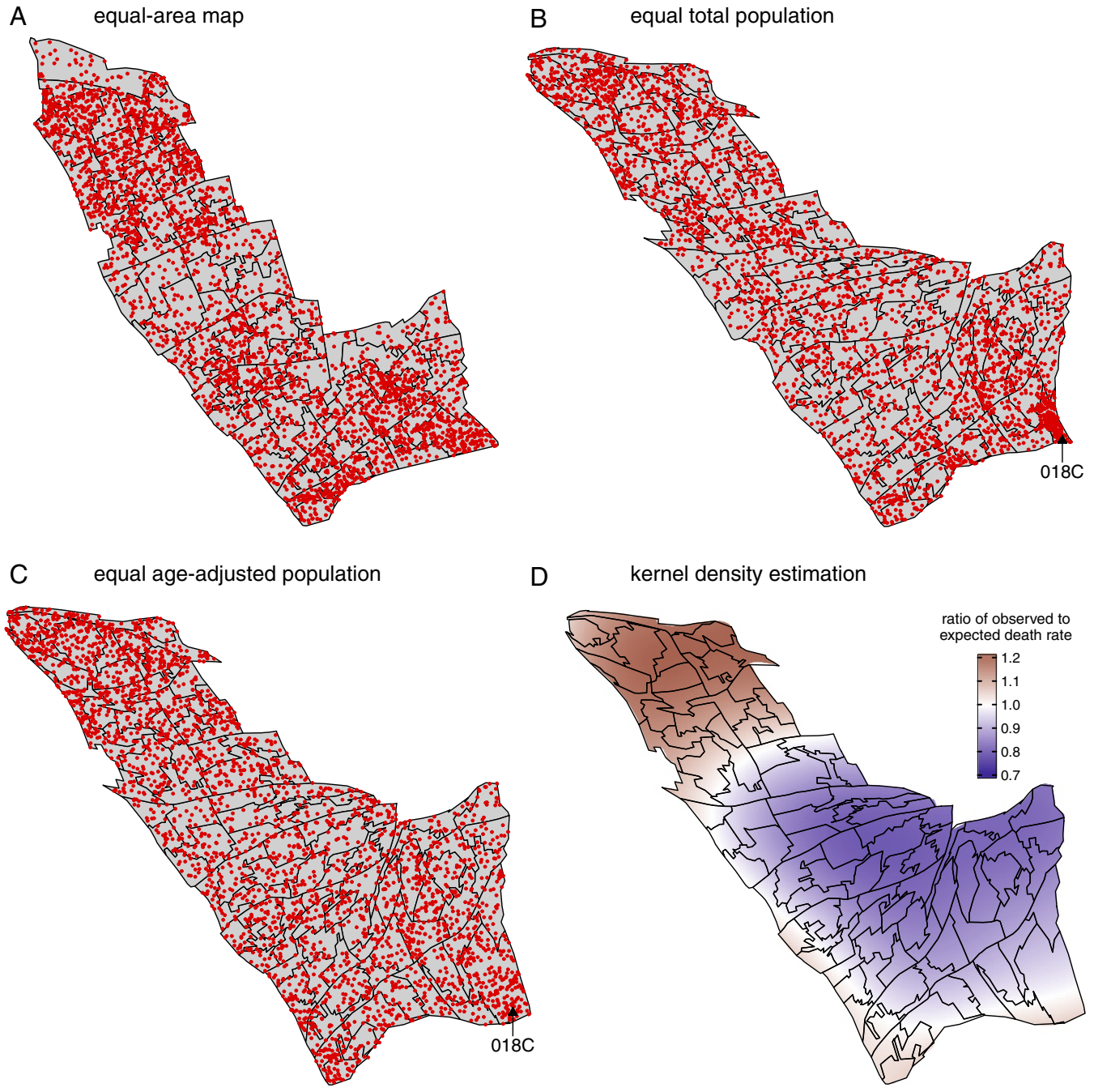

Fig. 4. (*A*–*C*) Maps with scatter plots of death cases in Kensington and Chelsea between 2011 and 2014 on (*A*) an equal-area map, (*B*) cartograms equalizing the total population in each LSOA and (*C*) age-adjusted population (i.e., the expected number of deaths given the age and gender composition of the LSOA). Cartogram *B* reveals a high per-capita mortality in LSOA 018C in the southeast of the borough caused by a nursing home located inside this polygon. When accounting for the heterogeneous age distribution across the borough in *C*, LSOA 018C has approximately the expected number of death cases. In other LSOAs, however, the expected and observed numbers differ. A kernel density estimate in *D* indicates an increasing trend in the age-adjusted death rate from the southeast to the northwest.

entire Petri dish. In the long run, the probability density function of finding a particle becomes a constant everywhere inside the dish. We can make a cartogram by translating this simple physical model of density equalization into a geographic map projection.

The most familiar process that equilibrates the density is Brownian motion. On a macroscopic scale, the Fokker–Planck equation that describes Brownian motion is Fick's second law, $\partial\rho/\partial t = D\nabla^2\rho$. Here t stands for time, D is the diffusivity, and ∇^2 is the Laplace differential operator. This equation, known as the diffusion or heat equation, is at the heart of the "diffusion cartogram" method (4, 25). An example of a diffusion cartogram is Fig. 3*A*, where we show the US Electoral College results. The diffusion algorithm guarantees that, unlike in Fig. 2*B*, each state keeps its neighbors while still reaching the target areas to any desired level of accuracy. The diffusion cartogram distorts the shapes of the states, which is inevitable for any contiguous cartogram method. The shapes are, however, still recognizable; this is one of the reasons why diffusion cartograms have become

popular in the past decade (3). Another reason is that, despite the apparent complexity of the equations, they can be computed relatively efficiently.

However, Fickian diffusion is only one of many types of fluid dynamic rules that make particle densities equal everywhere. As we argue now, there is an alternative that is computationally more efficient while producing cartograms of comparable quality.

Flow-Based Cartogram with Linear Equalization

In a flow-based cartogram, the population density ρ is treated not only as a function of position $\mathbf{r}=(x,y)$, but also as a function of time t. For a density-equalizing projection, the density must approach its mean in the long run: $\lim_{t\to\infty}\rho(x,y,t)=\bar{\rho}$ for all x and y. That is, the particles must flow in such a way that all initial differences in their density are completely leveled out over time. This condition alone, however, does not yet define the projection $\mathbf{T}$. We must also know the velocity $\mathbf{v}(x,y,t)$ with which a point at (x,y) is dragged along by the flow at time t. Because there are no sinks or sources in the flow, $\mathbf{v}$ must satisfy the mass conservation equation, also known as the continuity equation,

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\mathbf{v})=0. \qquad [2]$$

If we know $\mathbf{v}(x,y,t)$ for all x, y, and t, we can compute the position $\mathbf{r}(t)$ for a point that is initially at $\mathbf{r}(0)$,

$$\mathbf{r}(t)=\mathbf{r}(0)+\int_0^t\mathbf{v}(\mathbf{r}(t'),t')\,dt'. \qquad [3]$$

The projection $\mathbf{T}$ is the function that shifts $\mathbf{r}(0)$ to $\lim_{t\to\infty}\mathbf{r}(t)$. In *SI Appendix*, section 2, we explain in more detail why $\mathbf{T}$ is density equalizing.

We can satisfy the continuity equation while simultaneously demanding Fick's law $\mathbf{v}=-D(\nabla\rho)/\rho$. Substituting Fick's law into Eq. **2** shows that the evolution of ρ is then governed by the heat equation $\partial\rho/\partial t=D\nabla^2\rho$. This is the key motivation behind the diffusion cartogram method (4), but Fickian diffusion is only one special case within a large class of processes in which ρ relaxes to its mean density while satisfying the continuity equation for some velocity field $\mathbf{v}$. One advantage of Fickian diffusion is that the corresponding flow is guaranteed to be free of vortices that could cause severe local distortions in $\mathbf{T}$. However, Fickian diffusion is not unique in this respect (*SI Appendix*, section 2) so that one is left wondering whether other vortex-free, mass-conserving processes might also be suitable for generating cartograms. As we now argue, if we replace the heat equation by a linear equalization of the density toward the mean,

$$\rho(x,y,t)=\begin{cases}(1-t)\,\rho_0(x,y)+t\bar{\rho} & \text{if } t\le 1,\\ \bar{\rho} & \text{if } t>1,\end{cases} \qquad [4]$$

we can indeed compute $\mathbf{T}$ significantly faster. It has been shown that there exists a velocity field $\mathbf{v}$ for Eq. **4** so that the resulting transformation $\mathbf{T}$ satisfies Eq. **1** (24). We derive the concrete formulas for $\mathbf{v}$ in *SI Appendix*, section 2, and give only a brief summary here.

After an affine transformation of all coordinates, we place the mapped area inside a rectangular box with bounding coordinates $x_{\min}=0$, $x_{\max}=L_x$, $y_{\min}=0$, $y_{\max}=L_y$. (For later convenience, we choose L_x and L_y to be integers.) If we demand that there is no flow through the edges of the box, the velocity for $t\le 1$ can be expressed in terms of sine and cosine Fourier transforms,

$$v_x(x,y,t)=-\frac{L_y}{\pi\rho(x,y,t)}\sum_{m=1}^{\infty}\sum_{n=0}^{\infty}\left[\frac{m}{m^2L_y^2+n^2L_x^2}\tilde{\rho}_{mn}\times\sin\left(\frac{m\pi x}{L_x}\right)\cos\left(\frac{n\pi y}{L_y}\right)\right], \qquad [5]$$

$$v_y(x,y,t)=-\frac{L_x}{\pi\rho(x,y,t)}\sum_{m=0}^{\infty}\sum_{n=1}^{\infty}\left[\frac{n}{m^2L_y^2+n^2L_x^2}\tilde{\rho}_{mn}\times\cos\left(\frac{m\pi x}{L_x}\right)\sin\left(\frac{n\pi y}{L_y}\right)\right] \qquad [6]$$

with

$$\tilde{\rho}_{mn}=\frac{4}{(\delta_{m0}+1)(\delta_{n0}+1)}\times\int_0^{L_x}\int_0^{L_y}\rho(x',y',0)\cos\left(\frac{m\pi x'}{L_x}\right)\cos\left(\frac{n\pi y'}{L_y}\right)dx'\,dy'. \qquad [7]$$

Here $\delta_{00}=1$ and $\delta_{m0}=0$ if $m\neq 0$. For $t>1$, we simply obtain $v_x(x,y,t)=v_y(x,y,t)=0$.

Eqs. **5**–**7** look superficially similar to the corresponding equations in the diffusion-based cartogram (4), but there are two important differences. First, neither the sums in Eqs. **5** and **6** nor the integral in Eq. **7** depend on t so that the Fourier transforms need to be computed only once at the beginning of the calculation. Second, after we have computed the Fourier transforms, here we require only quick arithmetic operations: addition, subtraction, multiplication, and division. For a diffusion cartogram, by contrast, we must repeatedly calculate time-dependent Fourier transforms and evaluate the exponential function during the integration of Eq. **3** (*SI Appendix*, section 2). The speed of computing the exponential function depends on details of the implementation and hardware, but is in general much slower than addition, subtraction, multiplication, or division (27).

These mathematical differences alone already cut the time needed per integration step by more than half. Another simplification compared with the diffusion cartogram is that we need to integrate Eq. **3** only until $t=1$ instead of $t=\infty$. The benefit is that we no longer need to check whether the improper integral over the velocity has sufficiently converged. Most importantly, however, the integrals from different starting points $\mathbf{r}(0)$ can be performed in parallel as we now explain.

We overlay the map with an $L_x\times L_y$ square grid. For these L_xL_y coordinates, we compute the sums and integrals in Eqs. **5**–**7** at the start of the calculation with the fast Fourier transform algorithm (28). We have found that the time needed for this one-time procedure is a negligible fraction of the total run time. After storing the L_xL_y Fourier transforms in memory, we obtain $\mathbf{v}(\mathbf{r},t)$ at each grid point $\mathbf{r}$ with basic arithmetic. Subsequently, we find the integrand in Eq. **3** for nongrid positions $\mathbf{r}$ by interpolating between the grid points. We numerically approximate the integral using a predictor–corrector method that automatically adapts the size of the next time step. During each step, we distribute the integration of the L_xL_y distinct integrands to different processing units. In practice, given the wide availability of multicore processors nowadays, this parallelization enormously boosts the speed of the calculation.

Benchmarking the Algorithm with Data for the United States, India, and China

We have implemented the algorithm based on Eqs. **4**–**7** as a C program. In this section, we illustrate its performance with three case studies: the 2016 vote in the US Electoral College (Fig. 3*B*), the distribution of India's gross domestic product (GDP) by state (Fig. 5), and mainland China's and Taiwan's GDP by province (Fig. 6).

In each case, we first project the longitudes and latitudes of the territorial borders with an Albers equal-area conic projection onto a flat 2D space. As described above, we embed the resulting map (Figs. 2*A*, 5*A*, and 6*A*, respectively) inside an $L_x\times L_y$ rectangle whose edges act as reflecting boundaries for the flow. The rectangular box should, on one hand, be chosen large enough so

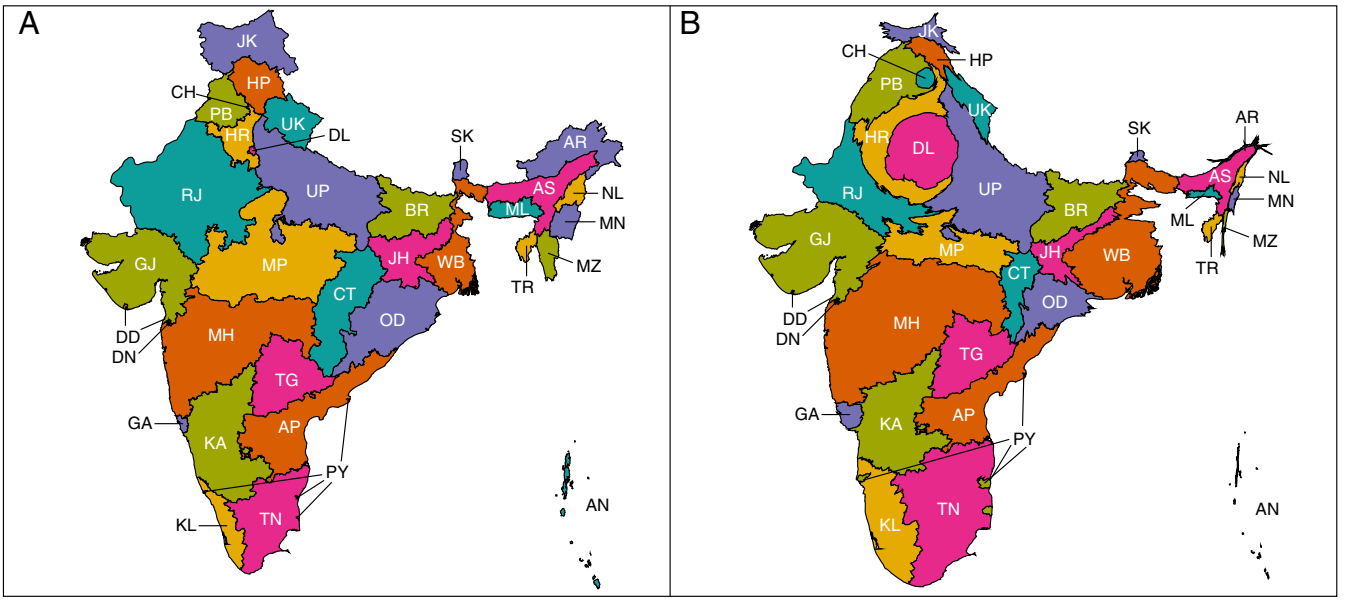

Fig. 5. (*A* and *B*) The states and union territories of India on (*A*) an equal-area map and (*B*) a cartogram where the area of each region is proportional to GDP [data from *Statistics Times* (29)]. The two largest states by area, Rajasthan (RJ) and Madhya Pradesh (MP), shrink on the cartogram because they rank only 7th and 10th in GDP, respectively. Maharashtra (MH), the state with the highest GDP, slightly grows on the cartogram. Even more striking is the increase of Delhi (DL): Although small in area, the capital city has a higher GDP than many larger states. The opposite happens for Arunachal Pradesh (AR) and several other northeastern states because they rank low in GDP. Our algorithm needs only 2.6 s to construct the cartogram. AN, Andaman and Nicobar Islands; AP, Andhra Pradesh; AS, Assam; BR, Bihar; CH, Chandigarh; CT, Chhattisgarh; DD, Daman and Diu; DN, Dadra and Nagar Haveli; GA, Goa; GJ, Gujarat; HP, Himachal Pradesh; HR, Haryana; JH, Jharkhand; JK, Jammu and Kashmir; KA, Karnataka; KL, Kerala; ML, Meghalaya; MN, Manipur; MZ, Mizoram; NL, Nagaland; OD, Odisha; PB, Punjab; PY, Puducherry; SK, Sikkim; TG, Telangana; TN, Tamil Nadu; TR, Tripura; UK, Uttarakhand; UP, Uttar Pradesh; WB, West Bengal.

that the cartogram is independent of the boundary conditions. On the other hand, it should not be so large that we spend the bulk of the run time on computing the projection **T** far from the region of interest. As a compromise, we have chosen the side length equal to 1.5 times the maximum of the countries' north–south and east–west extent. (These rectangular boxes are larger

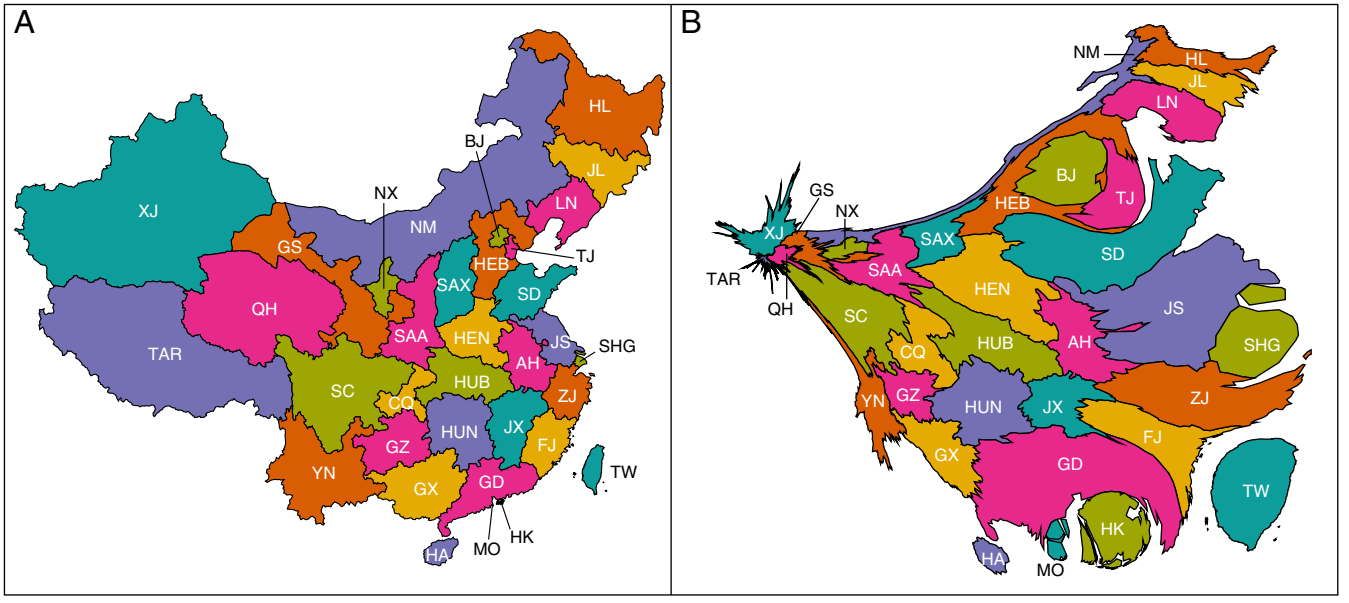

Fig. 6. (*A* and *B*) Provincial-level administrative divisions of mainland China and Taiwan on (*A*) an equal-area map and (*B*) a cartogram where areas are proportional to GDP [data from Wikipedia (https://en.wikipedia.org/wiki/List_of_Chinese_administrative_divisions_by_GDP)]. Some coastal cities such as Shanghai (SHG) and Hong Kong (HK) increase remarkably on the cartogram. By contrast, western states such as Xinjiang (XJ) and the Tibet Autonomous Region (TAR) shrink dramatically. Despite the substantial deformations, our algorithm needs only 2.7 s to construct the cartogram. AH, Anhui; BJ, Beijing; CQ, Chongqing; FJ, Fujian; GD, Guangdong; GS, Gansu; GX, Guangxi; GZ, Guizhou; HA, Hainan; HEB, Hebei; HEN, Henan; HL, Heilongjiang; HUB, Hubei; HUN, Hunan; JL, Jilin; JS, Jiangsu; JX, Jiangxi; LN, Liaoning; MO, Macao; NM, Inner Mongolia; NX, Ningxia; QH, Qinghai; SAA, Shaanxi; SAX, Shanxi; SC, Sichuan; SD, Shandong; TJ, Tianjin; TW, Taiwan; YN, Yunnan; ZJ, Zhejiang.

than the frames shown in Figs. 5 and 6 whose purpose is purely to visually separate the different panels in the figure.) The space between the country and the edges of the box is filled with the mean density $\bar{\rho}$. Other choices are conceivable and may improve shape preservation (e.g., by more faithfully retaining the outer boundaries of the map), but they would result in more complex computer code.

For the discrete Fourier transforms, we divide the large rectangular box into a grid of $L_x \times L_y$ smaller squares (in our examples $L_x = L_y = 512$, but the number can be adjusted if necessary) whose sizes are just fine enough to discern the smallest geographic regions on each map: Washington, DC; Daman and Diu in India (abbreviated DD in Fig. 5); and Macao in China (MO in Fig. 6). Officially, these regions have neither the status of a state nor that of a province: Washington, DC is a district, Daman and Diu a union territory, and Macao a Special Administrative Region. We still include these regions on the cartograms because they are typically included on maps showing the states and provinces of their respective countries.[‡]

When numerically integrating Eq. **3**, the choice of time steps determines how accurately we estimate $\mathbf{r}(1)$. One possible strategy for achieving a highly accurate cartogram is to take a large number of small steps. After some experiments, we have decided to use a different strategy that achieves quicker run times and ultimately also comes arbitrarily close to a perfectly density-equalizing map. We use only a moderate number of adaptive time steps (≈ 100 in a typical run; the exact number is determined at run time) during the initial integration. We expedite the convergence by applying a Gaussian blur of moderate width to the initial density before starting the integration. After one round of integration, the areas do not yet perfectly match their targets. For example, Washington, DC still needs to grow by a factor ≈ 50. The key feature is to use the output of the first integration as input to another round of integration, which then usually comes closer to the objective areas. By repeating the integration sufficiently often, we have in all test cases observed that we can reach the objective areas with arbitrary precision. For the contiguous 48 states of the United States, we perform five iterations. Afterward, even for the extreme case of Washington, DC, the smallest region in land area, the cartogram area differs by only 0.31% from the objective area. For India we iterate the integration 12 times and for China 6 times. The maximum differences between target and objective area are then 0.72% for the Andaman and Nicobar Islands (AN in Fig. 5*B*) and 0.83% for Tibet (TAR in Fig. 6*B*), respectively. These differences are certainly so small that they cannot be detected by eye. We generally set a maximum relative area error of $<1\%$, defined as

$$\text{relative area error} = \frac{\text{target area} - \text{objective area}}{\text{objective area}},$$

as the stopping criterion for the algorithm.

This level of accuracy is all the more remarkable when considering the speed of our implementation. On a Dell Precision T7810 workstation with a 12-core Intel Xeon E5-2680V3 processor and an Ubuntu 16.04.2 operating system, we need 1.5 s for the US Electoral College cartogram (Fig. 3*B*), 2.6 s for the India GDP cartogram (Fig. 5*B*), and 2.7 s for the China GDP cartogram (Fig. 6*B*). Compared with the diffusion algorithm, which needs 59.5 s to generate the US cartogram (Fig. 3*A*) with equal accuracy, this is a speedup by roughly a factor 40. Among other all-coordinates cartogram algorithms, only the rubber-sheet method Carto3F (22) can achieve comparable speed, but not for all types of input. For a cartogram of Chinese provinces, Carto3F needs 8 min of computer time. Our fast flow-based method achieves smaller area errors in a fraction of this time.

Benchmarking with Data for Mortality in Kensington and Chelsea (London) 2011–2014

As noted above, cartogram algorithms that generate the complete density-equalizing projection **T** are particularly advantageous when displaying demographic data that are individual points on a map. We now demonstrate how our algorithm can be applied to such input and how we can use it to compare different statistical models. The data also serve as another benchmark for the speed of our method. Our example involves the locations of all 3,197 death cases in the London borough of Kensington and Chelsea between the years 2011 and 2014. The database from the United Kingdom's Office for National Statistics (ONS) (30) lists the number of deaths in each of London's 4,835 Lower Layer Super Output Areas (LSOAs). A total of 103 LSOAs are located in Kensington and Chelsea. We show the density of death cases in this borough on an equal-area map in Fig. 4*A*. Each death corresponds to one point on the map placed at a random position inside the LSOA where it occurred.

The point pattern on the equal-area map is spatially heterogeneous with two bands of high density, one in the south and another in the north, separated by a band of lower density in the middle. However, it remains unclear from the equal-area map whether the differences in the spatial distribution of death cases are caused by differences in per-capita mortality or by a heterogeneous population density. We can distinguish between these two effects by projecting the death cases to a cartogram where each LSOA area is proportional to the number of inhabitants (Fig. 4*B*).

The most striking feature on this cartogram is the high per-capita mortality in the southeast corner of the borough. The reason for the high number of death cases in the LSOA with the ONS code "Kensington and Chelsea 018C" is a large proportion of elderly, most likely because of the St. Wilfrid's nursing home located in this LSOA. Because mortality increases markedly as a person becomes elderly, total population is too crude a measure to predict death rates. We now show how to improve the prediction by using each LSOA's age-adjusted mortality as the basis of a cartogram instead of the simple per-capita mortality displayed in Fig. 4*B*.

Data from the ONS (30, 31) include population size and death cases in the following age groups for each LSOA: 0 y old, 1–4 y old, 5–9 y old, ..., 85–89 y old, and ≥ 90 y old, with each age group divided into men and women. For each one of these 40 demographic subgroups, we can compute its total mortality in western central London (i.e., Kensington and Chelsea as well as the adjacent boroughs Brent, Westminster, Wandsworth, and Hammersmith and Fulham). We denote by p_j the size of the population that lives in this part of London and belongs, because of its gender and age, to the demographic group j. If there were d_j deaths in this subpopulation, its region-wide per-capita mortality is $m_j = d_j/p_j$. The expected number of deaths in the ith LSOA is thus $e_i = \sum_j p_{ij} m_j$, where p_{ij} is the population that lives in LSOA i and belongs to the demographic group j. This approach is known in the public health literature as age adjustment (32). Unlike the unadjusted population size $\sum_j p_{ij}$, the expected value e_i makes a fair comparison between, for example, an LSOA mostly inhabited by a younger population and an LSOA with a large proportion of elderly inhabitants such as 018C.

In Fig. 4*C*, we show a cartogram with LSOA areas proportional to e_i. On this cartogram, the density of points in 018C is near the average in the borough, visualizing that age is indeed an important predictor for local death rates. Across the borough, however, differences between death rates still remain despite age

[‡]We exclude the island territory of Lakshadweep from the maps of India because it is so small that it is neither visible on an equal-area map nor on a GDP cartogram.

Table 1. Measures of distortion applied to the diffusion algorithm and the flow-based algorithm using Eqs. 4–7

Map	Algorithm	e_a	e_∞	$\tilde{e}_a$	$\tilde{e}_\infty$	α	δ	θ	run time, s
United States	Diffusion	**0.278**	**6.85**	**0.273**	**3.01**	**2.01**	**17.1**	**0.0388**	59.5
	Fast flow-based	0.285	7.06	0.280	3.02	2.04	17.4	0.0435	**1.5**
India	Diffusion	**0.190**	3.95	**0.185**	2.59	2.45	**39.0**	**0.0281**	113.0
	Fast flow-based	0.191	**3.18**	0.187	**2.34**	**2.40**	39.7	0.0290	**2.6**
Mainland China and Taiwan	Diffusion	0.590	**5.07**	0.553	**2.83**	2.33	**18.6**	**0.0849**	178.5
	Fast flow-based	**0.570**	8.16	**0.530**	3.07	**2.19**	20.6	0.103	**2.7**
Kensington and Chelsea, age adjusted	Diffusion	**0.161**	**6.86**	**0.154**	**3.01**	**2.03**	**22.1**	**0.0589**	99.5
	Fast flow-based	0.163	7.08	0.156	3.03	2.20	24.1	0.0615	**1.9**

Smaller values are highlighted in boldface type.

adjustment. We can quantify the deviation from spatial homogeneity, for example, with the Hopkins statistic H (33), which is a number between 0 and 1. If a point pattern is caused by a homogeneous Poisson process (i.e., deaths are independent and equally likely everywhere), then the expected value of H equals 0.5. The more clustered the points are, the larger H is. We find $H = 0.524$ (95% confidence interval $[0.518, 0.530]$) in Fig. 4*C*, indicating that the data are inconsistent with a homogeneous Poisson process.

We show a kernel density estimate of the underlying probability distribution in Fig. 4*D*. We use a bivariate normal kernel with a bandwidth chosen according to ref. 34. Fig. 4*D* reveals a minimum in the age-adjusted death rate in the east of the borough and a maximum in the north. Previous studies have argued that indicators of health (e.g., life expectancy) in different parts of London are positively correlated to average household income (35). A choropleth map of deprivation in Kensington and Chelsea (36) does indeed follow a strikingly similar regional pattern to that of the death rate in Fig. 4*D*.

The flow-based method of Eqs. **4**–**7** calculates the cartograms in Fig. 4 *B* and *C* in 1.6 s and 1.9 s, respectively. To avoid boundary effects in Fig. 4*D*, we also include data for Kensington and Chelsea's neighboring boroughs when computing the cartograms and the kernel density estimate. The equivalent calculations with the diffusion-based method take 69.9 s and 99.5 s, respectively.

Measures of Distortion

Our algorithm not only is accurate and fast, but also generates cartograms whose visual appearance is on par with those of previous methods. In Fig. 3 we directly compare the diffusion cartogram of the United States (Fig. 3*A*) with the faster method based on Eqs. **4**–**7** (Fig. 3*B*). The border between Illinois (IL) and Indiana (IN) is straighter in Fig. 3*A* than in Fig. 3*B* and thus more similar to the input map (Fig. 2*A*). On the other hand, the border between New Mexico (NM) and Colorado (CO) is straighter and Oklahoma's (OK) panhandle less bent in Fig. 3*B*. Overall, however, the differences between both cartograms are only subtle.

Because visual appearance is not a fully satisfactory criterion, we now turn to quantitative measures of distortion. One way to compare the local distortion of different projections is by analyzing the Tissot indicatrix that is constructed as follows. Suppose we draw an infinitesimal circle at the coordinates (x, y) on the input map. Locally, the effect of the projection $\mathbf{T}$ is to deform the circle into an ellipse, called the Tissot indicatrix of $\mathbf{T}$ at (x, y). In *SI Appendix*, Fig. S2, we show concrete examples of Tissot indicatrices for our benchmarking examples. We denote the semimajor and -minor axes of the Tissot indicatrix by $a(x, y)$ and $b(x, y)$, respectively. Two measures of the local distortion error are (37)

$$e(x, y) = \ln\left(\frac{a(x, y)}{b(x, y)}\right)$$

and (38)

$$\tilde{e}(x, y) = 2 \arcsin\left(\frac{a(x, y) - b(x, y)}{a(x, y) + b(x, y)}\right).$$

For a conformal (i.e., angle-preserving) map, we would have $a = b$ for all (x, y) and thus $e = \tilde{e} = 0$. This scenario would be ideal, but, as we review in *SI Appendix*, section 3, except in a few special cases there cannot be a conformal density-equalizing projection (20). As a global measure for the deviation of a cartogram from conformality, we can use, for example, either the spatially averaged or the maximum local distortion error,

$$e_a = \frac{1}{|\Omega|}\int_\Omega e(x, y)\,dx\,dy, \qquad e_\infty = \sup_{(x,y)\in\Omega} e(x, y),$$

where Ω is the spatial domain of the input map. In our comparison of the diffusion and fast flow-based algorithms in Table 1, we choose Ω to be the rectangular $L_x \times L_y$ bounding box that contains the area to be mapped as described above. By replacing e with $\tilde{e}$, we obtain similar measures $\tilde{e}_a$ and $\tilde{e}_\infty$.

When computing e and $\tilde{e}$, we need to know $\mathbf{T}$ at each coordinate (x, y) so that these measures can be applied only to all-coordinates cartograms. Measures that aim to quantify the distortions also for other types of cartograms must instead rely on the polygons defining each region. In Table 1, we include three such measures from ref. 39: the average aspect ratio α, the Hamming distance δ, and the relative position error θ. We provide details of their definition in *SI Appendix*, section 4. Briefly, the aspect ratio of a region is the ratio of the larger to the smaller side length of the bounding rectangle with minimum area, minimized over all possible rotations with respect to the coordinate axes. The Hamming distance between two polygons is the area lying within exactly one of them (40). For the measurement in Table 1, we rescale each polygon on the input map and the corresponding polygon on the cartogram so that they have equal areas. We then calculate the minimum Hamming distance between these two polygons by shifting one polygon with respect to the other. We define δ as the sum of the minimum Hamming distances, where the summation is over all corresponding pairs of polygons. For the relative position error, we compute the angle between the line connecting the centroids of two polygons on the input map and the line that connects the centroids of the corresponding two polygons on the cartogram. We obtain θ by averaging over all pairs of polygons (41).

Most measures listed in Table 1 exhibit only small relative differences in the range of a few percent between the diffusion and fast flow-based methods. Diffusion performs a little better in the majority of examples and measures, but there are also cases where the fast flow-based method produces a smaller error. Considering the vastly different run times, the fast flow-based method is the better solution as a general-purpose algorithm for interactive applets.

Conclusion

The scientific value of cartograms can go far beyond providing mere entertainment, shock, or amusement. As "isodemographic maps" they have been used for mapping diseases and mortality for several decades (42–44) to improve health services (45). Arguably, the technical challenge of computing the map projection has so far prevented more widespread use. We accompany this article with C code available at https://github.com/Flow-Based-Cartograms/go_cart to alleviate some of the challenge. The code optionally produces the graticule of the inverse transformation so that features found on the cartogram can be identified in the original domain. We reconstruct the original positions by first approximating $\mathbf{T}$ as a piecewise linear function and then computing its inverse. The speed of the cartogram algorithm depends on the number of processing units available to the user. If the calculation runs on a multicore server, users will be able to take full advantage of the parallelized code at no cost. We hope that in this form the algorithm will be accessible to a wider audience.

ACKNOWLEDGMENTS. This research was supported by the European Commission (Project FP7-PEOPLE-2012-IEF 6-4564/2013).

1. de Veaux RD, Velleman PF, Bock DE (2016) *Stats: Data and Models* (Pearson, Harlow, UK), 4th Ed.
2. Tobler W (2004) Thirty five years of computer cartograms. *Ann Assoc Am Geogr* 94:58–73.
3. Nusrat S, Kobourov S (2016) The state of the art in cartograms. *Computer Graphics Forum* 35:619–642.
4. Gastner MT, Newman MEJ (2004) Diffusion-based method for producing density-equalizing maps. *Proc Natl Acad Sci USA* 101:7499–7504.
5. Gamio L (2016) Election maps are telling you big lies about small things. *The Washington Post*, Nov 1, 2016. Available at https://www.washingtonpost.com/graphics/politics/2016-election/how-election-maps-lie/. Accessed April 25, 2017.
6. Parlapiano A (2016) There are many ways to map election results. We've tried most of them. *The New York Times*, Nov 1, 2016. Available at https://www.nytimes.com/interactive/2016/11/01/upshot/many-ways-to-map-election-results.html. Accessed April 25, 2017.
7. Olson JM (1976) Noncontiguous area cartograms. *Prof Geogr* 28:371–380.
8. Dorling D (1996) *Area Cartograms: Their Use and Creation, Concepts and Teschniques in Modern Geography* (Environmental Publications, Norwich, UK).
9. Dougenik JA, Chrisman NR, Niemeyer DR (1985) An algorithm to construct continuous area cartograms. *Prof Geogr* 37:75–81.
10. Merrill DW, Selvin S, Mohr MS (1992) Density equalizing map projections: A new algorithm (Lawrence Berkeley Laboratory, Berkeley, CA), Technical Report LBL-31984.
11. House DH, Kocmoud CJ (1998) Continuous cartogram construction. *Proceedings of the IEEE Conference on Visualization* (The Association for Computing Machinery, New York), pp 197–204.
12. Keim DA, North SC, Panse C (2004) Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Trans Vis Comput Graph* 10:95–110.
13. Keim DA, Panse C, North SC (2005) Medial-axis-based cartograms. *IEEE Comput Graph Appl* 25:60–68.
14. Inoue R, Shimizu E (2006) A new algorithm for continuous area cartogram construction with triangulation of regions and restriction on bearing changes of edges. *Cartogr Geogr Inf Sci* 33:115–125.
15. Kämper JH, Kobourov SG, Nöllenburg M (**2013**) Circular-arc cartograms. *arXiv*: 1112.4626.
16. Cano RG, et al. (2015) Mosaic drawings and cartograms. *Computer Graphics Forum* 34:361–370.
17. Daya Sagar BS (2014) Cartograms via mathematical morphology. *Inf Vis* 13:42–58.
18. Tobler WR (1973) A continuous transformation useful for districting. *Ann N Y Acad Sci* 219:215–220.
19. Cauvin C, Schneider C (1989) Cartographic transformations and the piezopleth maps method. *Cartogr J* 26:96–104.
20. Gusein-Zade SM, Tikunov VS (1993) A new technique for constructing continuous cartograms. *Cartogr Geogr Inf Syst* 20:167–173.
21. Edelsbrunner H, Waupotitsch R (1997) A combinatorial approach to cartograms. *Comput Geometry* 7:343–360.
22. Sun S (2013) A fast, free-form rubber-sheet algorithm for contiguous area cartograms. *Int J Geogr Inf Sci* 27:567–593.
23. Hennig B (2013) *Rediscovering the World* (Springer, Berlin).
24. Dacorogna B, Moser J (1990) On a partial differential equation involving the Jacobian determinant. *Ann de l'IHP Analyse non Linéaire* 7:1–26.
25. Avinyó A, Solà-Morales J, València M (2003) On maps with given Jacobians involving the heat equation. *Z für Angew Mathematik Physik ZAMP* 54:919–936.
26. Henriques R, Bação F, Lobo V (2009) Carto-SOM: Cartogram creation using self-organizing maps. *Int J Geogr Inf Sci* 23:483–511.
27. Brent RP (1975) Multiple-precision zero-finding methods and the complexity of elementary function evaluation. *Analytic Computational Complexity*, ed Traub JF (Academic, New York), pp 151–176.
28. Frigo M, Johnson SG (2005) The design and implementation of FFTW3. *Proc IEEE* 93:216–231.
29. Statistics Times (2017) Indian states by GDP. Available at statisticstimes.com/economy/gdp-of-indian-states.php. Accessed May 3, 2017.
30. Office for National Statistics (2016) Number of deaths from all causes, by sex, age and LSOA 2001 and 2011, England and Wales, deaths registered 2001 to 2014. Available at https://www.ons.gov.uk/peoplepopulationandcommunity/healthandsocialcare/causesofdeath/. Accessed April 25, 2017.
31. Office for National Statistics (2015) Land area and population density for MSOA and LSOA. Available at https://files.datapress.com/london/dataset/super-output-area-population-lsoa-msoa-london/2015-11-26T15:41:48/land-area-population-density-lsoa11-msoa11.xlsx. Accessed April 25, 2017.
32. Lilienfeld DE, Stolley PD (1994) *Foundations of Epidemiology* (Oxford Univ Press, New York), 3rd Ed.
33. Hopkins B, Skellam JG (1954) A new method for determining the type of distribution of plant individuals. *Ann Bot* 18:213–227.
34. Venables WN, Ripley BD (2002) *Modern Applied Statistics with S* (Springer, New York).
35. Dorling D (2013) *The 32 Stops: The Central Line* (Penguin Books, London).
36. The Economist (2017) Kensington and Chelsea: A wealthy but deeply divided borough. *The Economist*, June 24, 2017. Available at https://www.economist.com/news/britain/21723839-grenfell-tower-fire-has-become-stark-reminder-glaring-gap-between-rich-and-poor-even). Accessed December 23, 2017.
37. Papadopoulos A (2017) Quasiconformal mappings, from Ptolemy's geography to the work of Teichmüller. arXiv:1702.03756.
38. Snyder JP (1987) Map projections – A working manual (US Government Printing Office, Washington, DC), Technical Report 1395.
39. Alam MJ, Kobourov SG, Veeramoni S (2015) Quantitative measures for cartogram generation techniques. *Computer Graphics Forum* 34:351–360.
40. Skiena SS (2008) *The Algorithm Design Manual* (Springer, London).
41. Heilmann R, Keim DA, Panse C, Sips M (2004) Recmap: Rectangular map approximations. *IEEE Symp Inf Vis*, 33–40.
42. Selvin S, et al. (1984) Transformations of maps to investigate clusters of disease (Lawrence Berkeley Laboratory, Berkeley, CA), Technical Report LBL-18550.
43. Dorling D (1998) Mapping disease patterns. *Encyclopedia of Biostatistics*, eds Armitage P, Colton T (Wiley, Chichester, UK), pp 2391–2401.
44. Wieland SC, Brownstein JS, Berger B, Mandl KD (2007) Density-equalizing Euclidean minimum spanning trees for the detection of all disease cluster shapes. *Proc Natl Acad Sci USA* 104:9404–9409.
45. Lovett DA, et al. (2014) Using geographical information systems and cartograms as a health service quality improvement tool. *Spat Spatio-Temporal Epidemiol* 10: 67–74.