

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

Michael T. Gastner^{a,1}, Vivien Seguy^b, and Pratyush More^a

^aYale-NUS College, Division of Science, 16 College Avenue West, #01-220 Singapore 138527; ^bGraduate School of Informatics, Kyoto University, 36-1 Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 Japan

This manuscript was compiled on February 22, 2018

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 the top half of Fig. 1 corresponds to a state that sent at least one Republican elector to the Electoral College. In the bottom half, 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 of Fig. 1 lacks the information where the states are located: neighboring bars do not necessarily correspond to

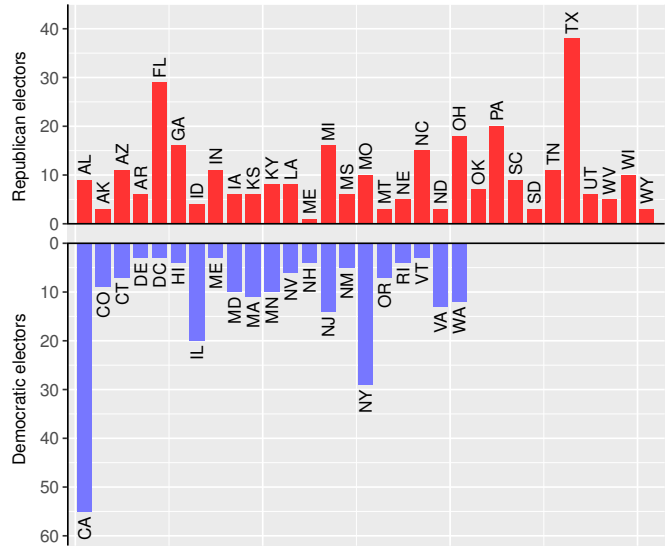


Fig. 1. A bar chart of the Electoral College vote for the US president in 2016. This diagram satisfies the area principle: the area of each bar is proportional to the number of electors. However, from this bar chart it is not clear where states are located geographically.

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

Significance Statement

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.

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

The authors declare no conflict of interest.

¹To whom correspondence should be addressed. E-mail: michael.gastner@yale-nus.edu.sg

*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 Fig. 7 of the SI Appendix.

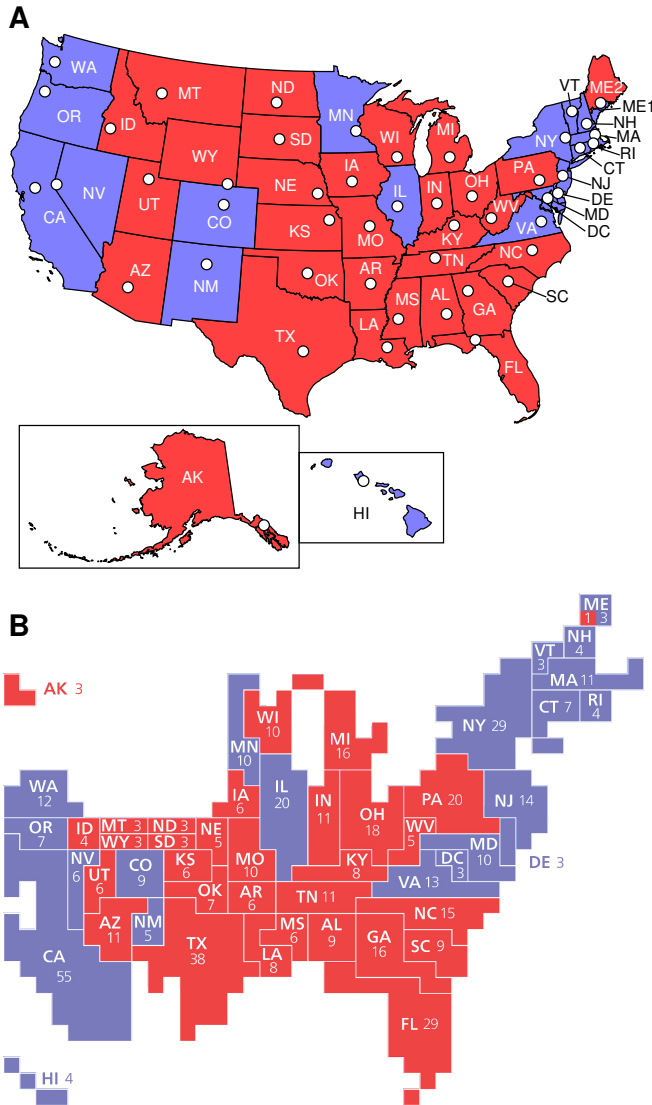


Fig. 2. (A) A conventional map projection (here an Albers projection) clearly shows the location of each state, but violates the area principle: states that occupy a large area do not necessarily have a large number of electors. (B) A cartogram of the 2016 Electoral College [adapted from Wikipedia (2)] 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 panel A).

Fig. 2A, 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 filled with either red or blue depending on the party affiliation of the electors.[†]

The map in Fig. 2A accurately shows the relative area and position of each state. However, it does not obey the area principle of statistics. For example, Montana (abbreviated by MT in Fig. 2A) covers more than 2000 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. 2A 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 (3, 4).

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 (5). The technique proposed in this article solves a completely different set of equations so that the computation can finish within a fraction of the previously needed time. We benchmark our algorithm with data from the USA, 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. 2B which we adapted from Wikipedia (2). Similar cartograms have been shown in the news media (6, 7). 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. 2A. Second, the set of neighboring states in Fig. 2A and Fig. 2B should be the same. Satisfying both objectives is not trivial. A careful comparison with Fig. 2A shows that, for example, Arizona (AZ) and Texas (TX) incorrectly appear as neighbors in Fig. 2B. On the other hand, the geographic neighbors Colorado (CO) and Nebraska (NE) have been separated in Fig. 2B to make space for other states in the vicinity.

For certain applications, it is perfectly acceptable that neighboring states are split apart. So long as the areas of the states are proportional to the number of electors, such representations are called noncontiguous cartograms (8). Dorling's circular cartograms are good examples of noncontiguous cartograms that, while not strictly maintaining the topology, indicate where the represented regions are located (9). 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 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 2nd congressional district (abbreviated as ME2 in Fig. 2A.)

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 (10–18). Each region is represented by one or multiple polygons. The input to these algorithms are a finite number of polygon corners $(x_1, y_1), \dots, (x_n, y_n)$. 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 $\mathbf{T}(x_1, y_1), \dots, \mathbf{T}(x_n, y_n)$. 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. 2B) 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 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 \mathbf{T} for the entire continuous set of longitudes and latitudes on the input map, including coordinates that are not on a boundary (5, 19–23). We refer to this group as “all-coordinates” algorithms. Generating the map projection \mathbf{T} for all longitudes and latitudes can be computationally more demanding than only shifting 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 (13, 14, 17). 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 (24) or individual addresses], the all-coordinates algorithms usually outpace their boundaries-only counterparts.

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

stant 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 two-dimensional 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 y . Such a transformation \mathbf{T} is called a density-equalizing projection. Taking the limits $dx \rightarrow 0$ and $dy \rightarrow 0$ and assuming that \mathbf{T} is differentiable, we obtain the condition (5, 19),

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

which is called a prescribed Jacobian equation (25, 26). For convenience, we choose the constant $\bar{\rho}$ to be the spatially averaged density so that the total mapped area is preserved.

Equation 1 alone does not uniquely specify \mathbf{T} because it is only one single equation for the two unknowns T_x and T_y (21). As a consequence, there are infinitely many different strategies to obtain a density-equalizing projection \mathbf{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 (20, 23). 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 (10, 12). 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 (27). 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 is 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 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

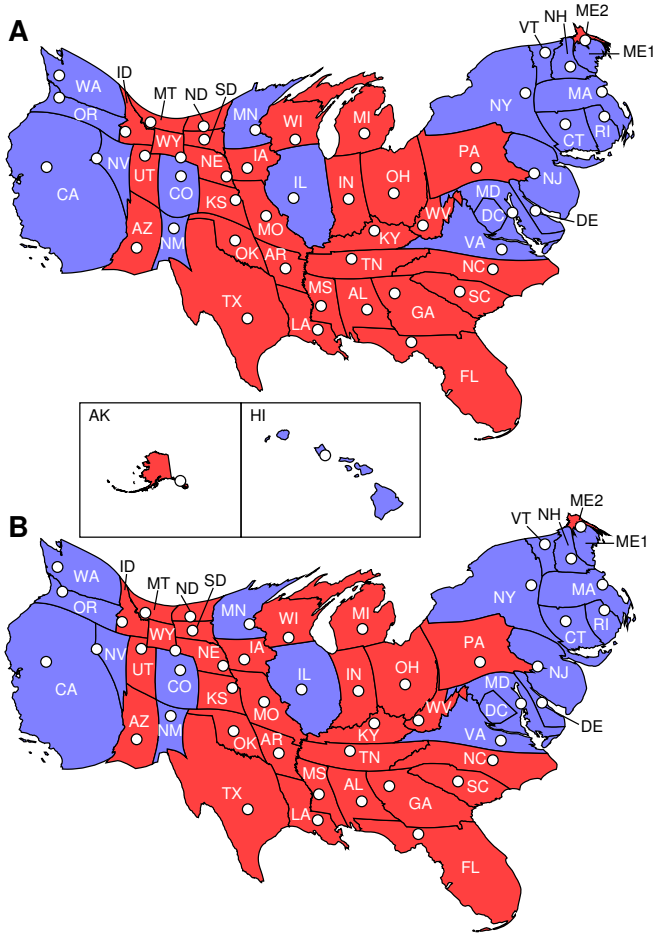


Fig. 3. The 2016 US Electoral College vote represented on cartograms generated with (A) the diffusion algorithm of Ref. (5) and (B) the alternative flow-based algorithm based on Eq. 4–7. The insets for Hawaii and Alaska apply to both (A) and (B) as these regions’ areas match both cartograms. All areas 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.

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 (5, 26). An example of a diffusion cartogram is Fig. 3A where we show the US Electoral College results. The diffusion algorithm guarantees that, unlike in Fig. 2B, 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 (4). 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: $\lim_{t \rightarrow \infty} \rho(x, y, t) = \bar{\rho}$ 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,

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

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 $\mathbf{r}(0)$,

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

The projection \mathbf{T} is the function that shifts $\mathbf{r}(0)$ to $\lim_{t \rightarrow \infty} \mathbf{r}(t)$. In the 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 (5), but Fickian diffusion is only one special case among 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 (see section 2 of the SI Appendix) 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,

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

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 (25). We derive the concrete formulas for \mathbf{v} in the SI Appendix (section 2) and only give 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 $t \leq 1$ can be expressed in terms of sine and

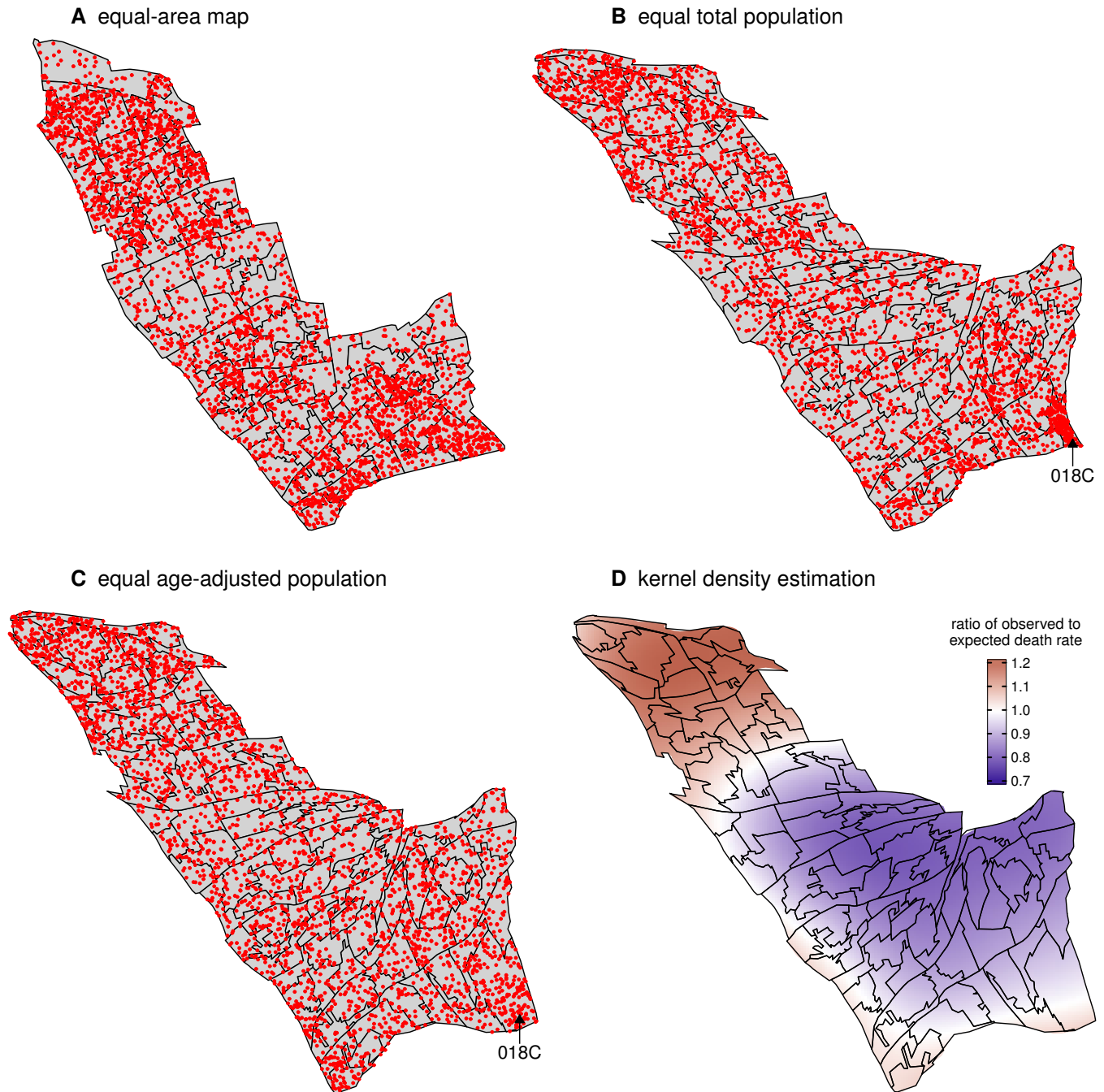


Fig. 4. Maps with scatter plots of death cases in Kensington and Chelsea between 2011 and 2014 on (A) an equal-area map and on cartograms equalizing (B) the total population in each Lower Layer Super Output Area (LSOA) and (C) age-adjusted population (i.e., the expected number of deaths given the age 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 panel (D) indicates an increasing trend in the age-adjusted death rate from the southeast to the northwest.

cosine Fourier transforms,

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

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

with

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

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

Equations 5–7 look superficially similar to the corresponding equations in the diffusion-based cartogram (5), but there are two important differences. First, neither the sums in Eq. 5, 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 only require 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 (see section 2 of the SI Appendix). 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 (28).

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 $L_x L_y$ coordinates, we compute the sums and integrals in Eq. 5–7 at the start of the calculation with the fast Fourier transform algorithm (29). We have found that the time needed for this one-time procedure is a negligible fraction of the total run time. After storing the $L_x L_y$ 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 non-grid 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 $L_x L_y$ distinct integrands to different processing units. In practice, given the wide availability of multi-core processors nowadays, this parallelization enormously boosts the speed of the calculation.

Benchmarking the algorithm with data for the USA, India, and China

We have implemented the algorithm based on Eq. 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. 3B), 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 two-dimensional space. As described above, we embed the resulting map (Fig. 2A, 5A and 6A, 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 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 \mathbf{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 than the frames shown in Fig. 5 and Fig. 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 in the USA; Daman and Diu in India (abbreviated by DD in Fig. 5); and Macao in China (MO in Fig. 6). Officially, these regions have neither the status of a state nor 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 runtime) during the initial integration. We expedite the convergence by applying a Gaussian blur of moderate width to the initial density prior to 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

[‡]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.

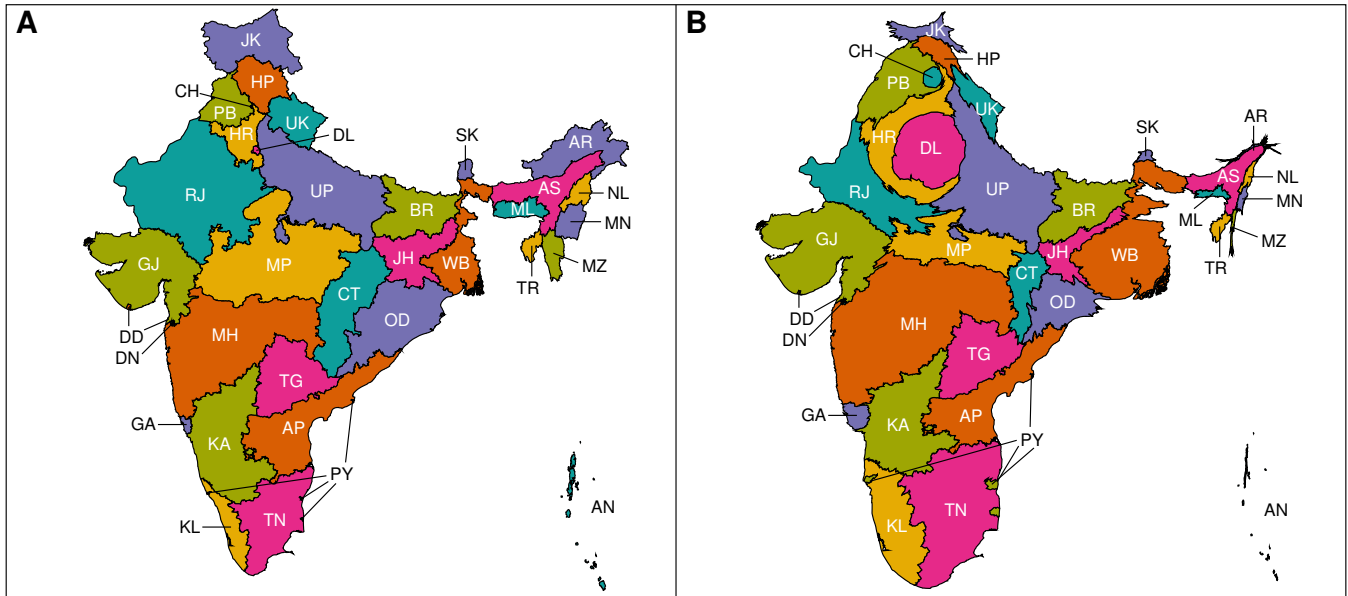


Fig. 5. The states and union territories of India on (A) an equal-area map, (B) a cartogram where the area of each region is proportional to GDP (data from Statistics Times (30)). The two largest states by area, Rajasthan (RJ) and Madhya Pradesh (MP), shrink on the cartogram because they only rank 7th 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 only needs 2.6 seconds to construct the cartogram. AN, Andaman and Nicobar Islands; AP, Andhra Pradesh; AS, Assam; BR, Bihar; CH, Chandigarh; CT, Chhattisgarh; DN, Dadra and Nagar Haveli; DD, Daman and Diu; GA, Goa; GJ, Gujarat; HR, Haryana; HP, Himachal Pradesh; JK, Jammu and Kashmir; JH, Jharkhand; KA, Karnataka; KL, Kerala; MN, Manipur; ML, Meghalaya; MZ, Mizoram; NL, Nagaland; OD, Odisha; PY, Puducherry; PB, Punjab; RJ, Rajasthan; SK, Sikkim; TN, Tamil Nadu; TG, Telangana; TR, Tripura; UP, Uttar Pradesh; UK, Uttarakhand; WB, West Bengal.

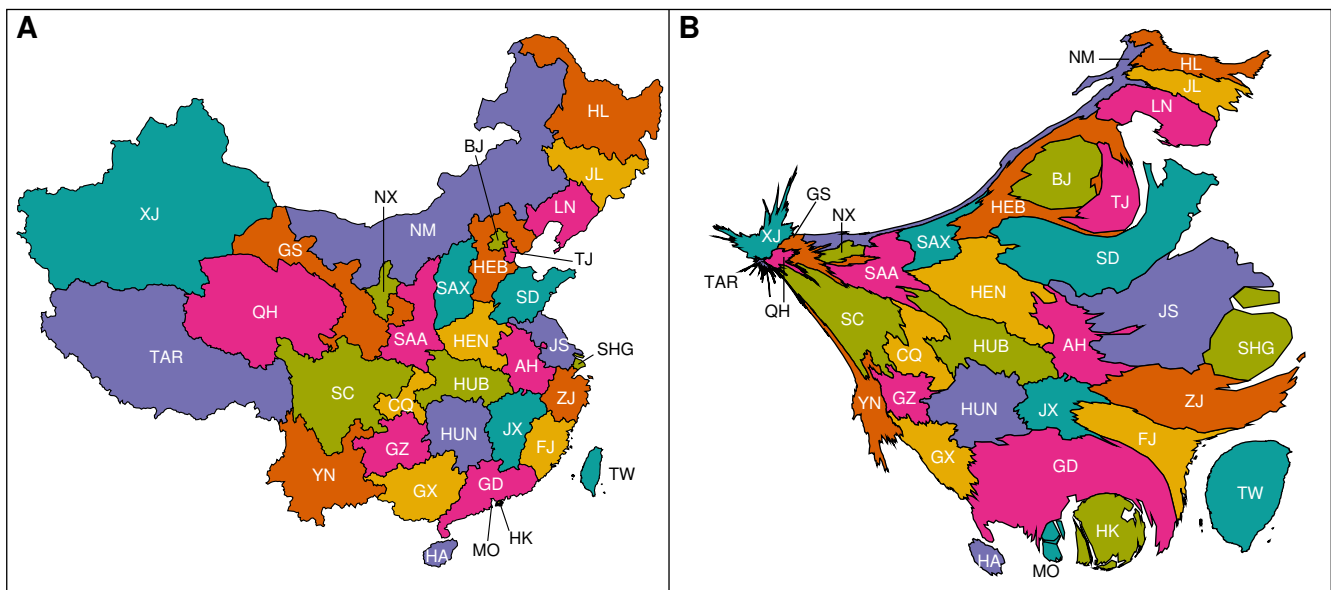


Fig. 6. Provincial-level administrative divisions of mainland China and Taiwan on (A) an equal-area map, (B) a cartogram where areas are proportional to GDP (data from Wikipedia (31)). Some coastal cities such as Shanghai (SHG) and Hong Kong (HK) increase remarkably on the cartogram. By contrast, western states such as Xinjiang (XJ) and the Tibet Autonomous Region (TAR) shrink dramatically. Despite the substantial deformations, our algorithm only needs 2.7 seconds to construct the cartogram. AH, Anhui; BJ, Beijing; CQ, Chongqing; FJ, Fujian; GS, Gansu; GD, Guangdong; GX, Guangxi; GZ, Guizhou; HA, Hainan; HEB, Hebei; HL, Heilongjiang; HEN, Henan; HUB, Hubei; HUN, Hunan; NM, Inner Mongolia; JS, Jiangsu; JX, Jiangxi; JL, Jilin; LN, Liaoning; MO, Macao; NX, Ningxia; QH, Qinghai; SAA, Shaanxi; SD, Shandong; SAX, Shanxi; SC, Sichuan; TW, Taiwan; TJ, Tianjin; YN, Yunnan; ZJ, Zhejiang.

areas with arbitrary precision. For the contiguous 48 states of the USA, 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 twelve times and for China six times. The maximum differences between target and objective area are then 0.72% for the Andaman and Nicobar Islands (AN in Fig. 5B) and 0.83% for Tibet (TAR in Fig 6B), 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 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 seconds for the US Electoral College cartogram (Fig. 3B), 2.6 seconds for the India GDP cartogram (Fig. 5B), and 2.7 seconds for the China GDP cartogram (Fig. 6B). Compared with the diffusion algorithm, which needs 59.5 seconds to generate the US cartogram (Fig. 3A) with equal accuracy, this is a speedup by roughly a factor 40. Among other all-coordinates cartogram algorithms, only the rubbersheet method Carto3F (23) can achieve comparable speed, but not for all types of input. For a cartogram of Chinese provinces, Carto3F needs 8 minutes 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 \mathbf{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 3197 death cases in the London borough of Kensington and Chelsea between the years 2011 and 2014. The database from the UK’s Office for National Statistics (ONS) (32) lists the number of deaths in each of London’s 4835 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. 4A. 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. 4B).

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. 4B.

Data from the ONS (32, 33) include population size and death cases in the following age groups for each LSOA: 0 years old, 1-4, 5-9, ..., 85-89, and ≥ 90 years old, with each age group divided into men and women. For each 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, 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 i -th 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 (34). 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. 4C, 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-adjustment. We can quantify the deviation from spatial homogeneity, for example, with the Hopkins statistic H (35), 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. 4C, indicating that the data are inconsistent with a homogeneous Poisson process.

We show a kernel density estimate of the underlying probability distribution in Fig. 4D. We use a bivariate normal kernel with a bandwidth chosen according to Ref. (36). The figure 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 (37). A choropleth map of deprivation in Kensington and Chelsea (38) does indeed follow a strikingly similar regional pattern as the death rate in Fig. 4D.

The flow-based method of Eq. 4–7 calculates the cartograms in Fig. 4B and C in 1.6 and 1.9 seconds respectively. To avoid boundary effects in Fig. 4D, 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 and 99.5 seconds respectively.

Measures of distortion

Our algorithm is not only accurate and fast, but also generates cartograms whose visual appearance is on par with previous methods. In Fig. 3 we directly compare the diffusion cartogram of the USA (panel A) with the faster method based on Eq. 4–7 (panel B). The border between Illinois (IL) and Indiana (IN) is straighter in Fig. 3A than in Fig. 3B and thus more similar to the input map (Fig. 2A). On the other hand, the border between New Mexico (NM) and Colorado (CO) is straighter and Oklahoma’s (OK) panhandle less bent in Fig. 3B. 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) . Figure 8 in the SI Appendix shows concrete examples of Tissot indicatrices for our benchmarking examples. We denote the semi-major and -minor axes of the Tissot indicatrix by $a(x, y)$ and $b(x, y)$ respectively. Two measures of the local distortion error are (39)

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

and (40)

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

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 the SI Appendix (section 3), except in a few special cases there cannot be a conformal density-equalizing projection (21). 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,

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

where Ω is the spatial domain of the input map. In our comparison of the diffusion and fast flow-based algorithm 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}_{∞} .

When computing e and \tilde{e} , we need to know \mathbf{T} at each coordinate (x, y) so that these measures can only be applied 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. (41): the average aspect ratio α , the Hamming distance δ and the relative position error θ . We provide details of their definition in the 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 (42). 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 area. 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 (43).

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 method. 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 (44–46) in order to improve health services (47). 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 multi-core web 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 number FP7-PEOPLE-2012-IEF 6-4564/2013).

1. R. D. de Veaux, P. F. Velleman, and D. E. Bock. *Stats: Data and Models*. Pearson, Harlow, 4th edition, 2016.
2. K. Ma. 2012 Electoral Vote. https://commons.wikimedia.org/wiki/File:Cartogram%E2%80%94942012_Electoral_Vote.svg, 2012. Accessed 09 Feb 2018.
3. W. Tobler. Thirty five years of computer cartograms. *Annals of the Association of American Geographers*, 94(1):58–73, 2004.
4. S. Nusrat and S. Kobourov. The state of the art in cartograms. *Computer Graphics Forum*, 35(3):619–642, 2016.
5. M. T. Gastner and M. E. J. Newman. Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences of the United States of America*, 101(20):7499–7504, 2004. .
6. L. Garnio. Election maps are telling you big lies about small things. *The Washington Post*, 1 Nov 2016, <https://www.washingtonpost.com/graphics/politics/2016-election/how-election-maps-lie/>, 2016. Accessed 25 Apr 2017.

Table 1. Measures of distortion applied to the diffusion algorithm and the flow-based algorithm using Eq. 4–7. Smaller values are highlighted in bold.

Map	Algorithm	e_a	e_∞	\tilde{e}_a	\tilde{e}_∞	α	δ	θ	run time (seconds)
USA	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 & Chelsea (age adj.)	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

7. A. Parlapiano. There are many ways to map election results. We've tried most of them. *The New York Times*, 1 Nov 2016, <https://www.nytimes.com/interactive/2016/11/01/upshot/many-ways-to-map-election-results.html>, 2016. Accessed 25 Apr 2017.

8. J. M. Olson. Noncontiguous area cartograms. *The Professional Geographer*, 28(4):371–380, 1976.

9. D. Dorling. *Area cartograms: their use and creation*. Concepts and Techniques in Modern Geography, Environmental Publications, Norwich, 1996.

10. J. A. Dougenik, N. R. Chrisman, and D. R. Niemeyer. An algorithm to construct continuous area cartograms. *Professional Geographer*, 37(1):75–81, 1985.

11. D. W. Merrill, S. Selvin, and M. S. Mohr. Density equalizing map projections: A new algorithm. Technical Report LBL-31984, Lawrence Berkeley Laboratory, Feb 1992.

12. D. H. House and C. J. Kocmoud. Continuous cartogram construction. In *Proceedings of the IEEE Conference on Visualization*, pages 197–204, Oct 1998.

13. D. A. Keim, S. C. North, and C. Panse. Cartodraw: a fast algorithm for generating contiguous cartograms. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):95–110, 2004.

14. D. A. Keim, C. Panse, and S. C. North. Medial-axis-based cartograms. *IEEE Computer Graphics and Applications*, 25(3):60–68, 2005.

15. R. Inoue and E. Shimizu. A new algorithm for continuous area cartogram construction with triangulation of regions and restriction on bearing changes of edges. *Cartography and Geographic Information Science*, 33(2):115–125, 2006.

16. J. H. Kämper, S. G. Kobourov, and M. Nöllenburg. Circular-arc cartograms. In *2013 IEEE Pacific Visualization Symposium (PacificVis)*, pages 1–8, Feb 2013.

17. R. G. Cano, K. Buchin, T. Castermans, A. Pieterse, W. Sonke, and B. Speckmann. Mosaic drawings and cartograms. *Computer Graphics Forum*, 34(3):361–370, 2015.

18. B. S. Daya Sagar. Cartograms via mathematical morphology. *Information Visualization*, 13(1):42–58, 2014.

19. W. R. Tobler. A continuous transformation useful for districting. *Annals of the New York Academy of Sciences*, 219(1):215–220, 1973.

20. C. Cauvin and C. Schneider. Cartographic transformations and the piezopleth maps method. *The Cartographic Journal*, 26(2):96–104, 1989.

21. S. M. Gusein-Zade and V. S. Tikunov. A new technique for constructing continuous cartograms. *Cartography and Geographic Information Systems*, 20(3):167–173, 1993.

22. H. Edelsbrunner and R. Waupoltsch. A combinatorial approach to cartograms. *Computational Geometry*, 7(5):343–360, 1997.

23. S. Sun. A fast, free-form rubber-sheet algorithm for contiguous area cartograms. *International Journal of Geographical Information Science*, 27(3):567–593, 2013.

24. B. Hennig. *Rediscovering the World*. Springer, Berlin, 2013.

25. B. Dacorogna and J. Moser. On a partial differential equation involving the Jacobian determinant. *Annales de l'Institut Henri Poincaré*, 7(1):1–26, 1990.

26. A. Avinyó, J. Solà-Morales, and M. València. On maps with given Jacobians involving the heat equation. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, 54(6):919–936, 2003.

27. R. Henriques, F. Bação, and V. Lobo. Carto-SOM: Cartogram creation using self-organizing maps. *International Journal of Geographical Information Science*, 23(4):483–511, 2009.

28. R. P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In J. F. Traub, editor, *Analytic Computational Complexity*, pages 151–176. Academic Press, New York, 1975.

29. M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

30. Statistics Times. Indian states by GDP. <http://statisticstimes.com/economy/gdp-of-indian-states.php>, 2017. Accessed 3 May 2017.

31. Wikipedia. List of Chinese administrative divisions by GDP. https://en.wikipedia.org/wiki/List_of_Chinese_administrative_divisions_by_GDP, 2017. Accessed 3 May 2017.

32. Office for National Statistics. Number of deaths from all causes, by sex, age and LSOA 2001 and 2011, England and Wales, deaths registered 2001 to 2014. <https://www.ons.gov.uk/peoplepopulationandcommunity/healthandsocialcare/causesofdeath/>, 2016. Accessed 25 Apr 2017.

33. Office for National Statistics. Land area and population density for MSOA and LSOA, 2015. Accessed 25 Apr 2017.

34. D. E. Lilienfeld and P. D. Stolley. *Foundations of Epidemiology*. Oxford University Press, New York, 3rd edition, 1994.

35. B. Hopkins and J. G. Skellam. A new method for determining the type of distribution of plant individuals. *Annals of Botany*, 18(2):213–227, 1954.

36. W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, 2002.

37. D. Dorling. *The 32 Stops: The Central Line*. Penguin Books, London, 2013.

38. The Economist. Kensington and Chelsea: a wealthy but deeply divided borough. 24 June 2017, <https://www.economist.com/news/britain/21723839-grenfell-tower-fire-has-become-stark-reminder-glaring-gap-between-rich-and-poor-even>, 2017. Accessed 23 Dec 2017.

39. A. Papadopoulos. Quasiconformal mappings, from Ptolemy's geography to the work of Teichmüller. arXiv:1702.03756, 2017.

40. J. P. Snyder. Map projections – a working manual. Technical Report 1395, U.S. Government Printing Office, Washington, 1987.

41. M. J. Alam, S. G. Kobourov, and S. Veeramoni. Quantitative measures for cartogram generation techniques. *Computer Graphics Forum*, 34(3):351–360, 2015. ISSN 1467-8659.

42. S. S. Skiena. *The Algorithm Design Manual*. Springer, London, 2008.

43. R. Heilmann, D. A. Keim, C. Panse, and M. Sips. Recmap: Rectangular map approximations. In *IEEE Symposium on Information Visualization*, pages 33–40, 2004.

44. S. Selvin, D. Merrill, S. Sacks, L. Wong, L. Bedell, and J. Schulman. Transformations of maps to investigate clusters of disease. Technical Report LBL-18550, Lawrence Berkeley Laboratory, Oct 1984.

45. D. Dorling. Mapping disease patterns. In *Encyclopedia of Biostatistics*. John Wiley & Sons, Hoboken, 2005.

46. S. C. Wieland, J. S. Brownstein, B. Berger, and K. D. Mandl. Density-equalizing Euclidean minimum spanning trees for the detection of all disease cluster shapes. *Proceedings of the National Academy of Sciences of the United States of America*, 104(22):9404–9409, 2007.

47. D. A. Lovett, A. J. Poots, J. T. C. Clements, S. A. Green, E. Samarasinghe, and D. Bell. Using geographical information systems and cartograms as a health service quality improvement tool. *Spatial and spatio-temporal epidemiology*, 10:67–74, 2014.

48. M. T. Gastner, C. R. Shalizi, and M. E. J. Newman. Maps and cartograms of the 2004 US presidential election results. *Advances in Complex Systems*, 8(1):117–123, 2005.

49. J. D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, New York, 2nd edition, 1991.

50. L. Ambrosio, N. Gigli, and G. Savaré. *Gradient flows in metric spaces and in the space of probability measures*. Birkhäuser, Basel, 2008.

51. J. Moser. On the volume elements on a manifold. *Transactions of the American Mathematical Society*, 120(2):286–294, 1965.

52. D. A. Keim, S. C. North, C. Panse, and J. Schneidewind. Visualizing geographic information: VisualPoints vs CartoDraw. *Information Visualization*, 2(1):58–67, 2003.

SI Appendix

1. Cartogram of the popular vote in the 2016 US presidential election

US presidential elections are indirect: voters do not directly elect the president, but instead choose electors for their state who represent a presidential candidate in the Electoral College. The candidate with most votes in the Electoral College becomes the next president. 48 out of 50 states and Washington DC apply a winner-takes-all rule: the presidential candidate with the largest number of votes cast by the population in the state wins all of the state’s electoral votes. The only exceptions are Maine and Nebraska. These two states apply the congressional district method: besides two electors for the state’s aggregate winner, each congressional district chooses one elector for the candidate with most votes in this district.

The composition of the Electoral College does not need to be an accurate representation of the nationwide popular vote. The predominant winner-takes-all rule gives an advantage to a candidate who wins many states with narrow margins even if the opponent may have won more votes in the population as a whole. Furthermore, the number of votes in the Electoral College is not strictly proportional to state populations. There is a small bias in favor of less populated states by guaranteeing every state a minimum of three electors.

Historically, the winner of the nationwide popular vote has usually also won the Electoral College. However, the 2016 election was one of the exceptions. Hillary Clinton gained 48.2% of the popular vote, Donald Trump only 46.1%. Nevertheless, Trump won the Electoral College by 304 to 227 votes.

We visualize the popular vote on a cartogram (Fig. 7) by making each state’s area proportional to the number of combined votes cast for Trump or Clinton in this state. We indicate the result with a color between blue (100% for Clinton) and red (100% for Trump). The shade of purple indicates how votes were split in each state. Cartograms with the same color scheme have been shown for previous US presidential elections (48).

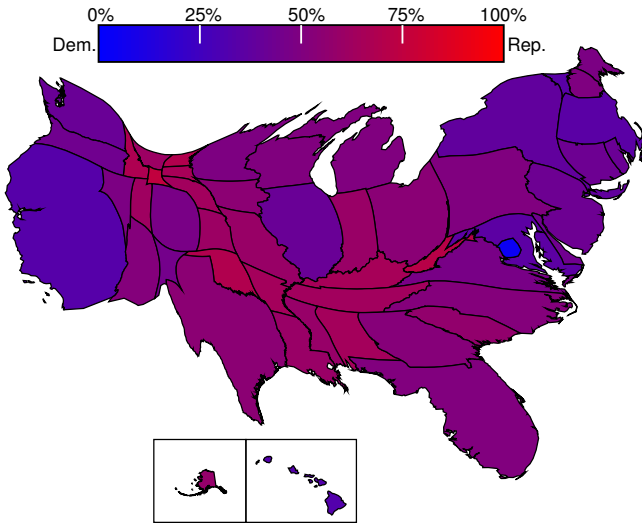


Fig. 7. The popular vote in the 2016 US presidential election on a cartogram made with the fast flow-based algorithm described in the main text.

2. Motivating the equations used by the algorithm

Flow-based density-equalizing projections. Suppose we are given a population density $\rho_0(\mathbf{r})$ for every point $\mathbf{r} = (x, y)$ in a rectangle defined by $0 \leq x \leq L_x$ and $0 \leq y \leq L_y$. Our objective is to map the rectangle onto itself with a density-equalizing projection \mathbf{T} . That is, assuming \mathbf{T} is differentiable, it must satisfy

$$\det(\nabla \mathbf{T}(\mathbf{r})) = \frac{\rho_0(\mathbf{r})}{\bar{\rho}} \quad [8]$$

for every point \mathbf{r} in the rectangle. The left-hand side is the Jacobian determinant

$$\det(\nabla \mathbf{T}(\mathbf{r})) = \frac{\partial T_x}{\partial x} \frac{\partial T_y}{\partial y} - \frac{\partial T_x}{\partial y} \frac{\partial T_y}{\partial x}$$

and the denominator in Eq. 8 is the spatially averaged density

$$\bar{\rho} = \frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} \rho_0(x, y) dx dy.$$

Loosely speaking, $\det(\nabla \mathbf{T}(\mathbf{r}))$ is the factor by which a small area element near \mathbf{r} is rescaled after applying the transformation \mathbf{T} . The general form of Eq. 8 is called a “prescribed Jacobian equation”.

The idea behind flow-based methods to find a solution \mathbf{T} is to define a sequence of densities $\rho(x, y, t)$, where the nonnegative variable t represents time. We start from the given population density,

$$\rho(x, y, 0) = \rho_0(x, y), \quad [9]$$

and demand that ρ approaches in the long run the spatially averaged density,

$$\lim_{t \rightarrow \infty} \rho(x, y, t) = \bar{\rho}. \quad [10]$$

For constructing a flow-based cartogram, we also need a two-dimensional velocity field $\mathbf{v} = (v_x, v_y)$ for all x, y , and t . We define the map projection \mathbf{T}_t of a point that is initially at $\mathbf{r} = (x, y)$ by

$$\mathbf{T}_t(\mathbf{r}) = \mathbf{r} + \int_0^t \mathbf{v}(\mathbf{T}_{t'}(\mathbf{r})) dt'. \quad [11]$$

We now argue that in the limit of infinite time, \mathbf{T}_∞ is a density-equalizing projection (i.e., it satisfies Eq. 8) if the combination of ρ and \mathbf{v} satisfies the continuity equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{J}. \quad [12]$$

Here

$$\mathbf{J} = \rho \mathbf{v}$$

is the flux (i.e., the population that flows per unit time through a line of unit length perpendicular to \mathbf{J}) and $\nabla \cdot \mathbf{J} = \frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y}$ is the so-called divergence of \mathbf{J} .

An intuitive explanation why Eq. 9–12 imply Eq. 8 for \mathbf{T}_∞ is as follows. Suppose a small, simply connected region \mathcal{R} with area A contains the point \mathbf{r} . Because of the definition of the population density ρ_0 , the initial population contained inside \mathcal{R} is approximately equal to $A\rho_0(\mathbf{r})$. After the boundary has drifted with the flow and reached its final position, \mathcal{R} has been mapped to a new region \mathcal{S} with area $\approx A \cdot \det(\nabla \mathbf{T}_\infty(\mathbf{r}))$. As a consequence of Eq. 10, the population contained in \mathcal{S} is

approximately $\bar{\rho}A \cdot \det(\nabla \mathbf{T}_\infty(\mathbf{r}))$. The continuity equation 12 guarantees that the population inside any closed boundary is preserved while the boundary is drifting with the velocity field (49). Therefore, $A\rho_0(\mathbf{r}) = \bar{\rho}A \cdot \det(\nabla \mathbf{T}_\infty(\mathbf{r}))$. After canceling the common factor A on both sides and comparing with Eq. 8, we conclude that \mathbf{T}_∞ is indeed a density-equalizing projection.

For a rigorous proof that Eq. 8 is a consequence of Eq. 9–12, we would have to impose several demands on the continuity and integrability of ρ and \mathbf{v} so that the solution of Eq. 11 is guaranteed to exist. The technical details are beyond the scope of this article. In general, we can safely assume that the conditions are valid in our case. The interested reader may consult Ambrosio et al. (50) for details, especially their Proposition 8.1.8.

The general solution for vortex-free flow. Equations 9–12 are, under mild assumptions, sufficient to ensure that \mathbf{T}_∞ is a density-equalizing projection. However, the equations have multiple solutions and many of them are in practice unsuitable for producing cartograms. In particular, solutions with vortices in the flux field \mathbf{J} create severe local distortions in the vicinity of each vortex. We therefore add one more demand to Eq. 9–12,

$$\frac{\partial J_x}{\partial y} = \frac{\partial J_y}{\partial x}, \quad [13]$$

which guarantees that there are no vortices (49).

Can we construct concrete pairs of a density $\rho(x, y, t)$ and a velocity $\mathbf{v}(x, y, t)$ that satisfy Eq. 9–13? Let us assume that $\rho(x, y, t)$ is a piecewise continuous function. At all points (x, y) where $\rho(x, y, t)$ is continuous, the cosine Fourier series of ρ converges pointwise to ρ . Thus, at these points we have

$$\rho(x, y, t) = \frac{1}{L_x L_y} \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \tilde{\rho}_{mn} f_{mn}(t) \cos\left(\frac{m\pi x}{L_x}\right) \cos\left(\frac{n\pi y}{L_y}\right), \quad [14]$$

where

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

is the backward cosine Fourier transform of the initial density, δ_{m0} is the Kronecker symbol

$$\delta_{m0} = \begin{cases} 1 & \text{if } m = 0, \\ 0 & \text{otherwise,} \end{cases}$$

and $f_{mn}(t)$ is a function that must be consistent with the constraints expressed by Eq. 9–13.

The functions $\cos\left(\frac{m\pi x}{L_x}\right)$ with $m = 0, 1, \dots$ are mutually orthogonal so that $f_{mn}(t)$ on the right-hand side of Eq. 14 is uniquely determined by $\rho(x, y, t)$ on the left-hand side. From this observation and Eq. 9, it follows that

$$f_{mn}(0) = 1 \quad \text{for all } m \text{ and } n. \quad [15]$$

Because of $\tilde{\rho}_{00} = \bar{\rho}L_x L_y$ and Eq. 10, we must have

$$\lim_{t \rightarrow \infty} f_{mn}(t) = \begin{cases} 1 & \text{if } m = n = 0, \\ 0 & \text{otherwise.} \end{cases} \quad [16]$$

To interpret the remaining constraints (i.e., Eq. 12 and Eq. 13) we must specify the boundary conditions of the flux \mathbf{J} . We assume that there is no flow through the edges of the rectangular box $[0, L_x] \times [0, L_y]$. Then it must be possible to express the x - and y -coordinates of the two-dimensional function \mathbf{J} in terms of the following mixed sine and cosine Fourier transforms at all points (x, y) where \mathbf{J} is continuous,

$$J_x(x, y, t) = \frac{1}{L_x L_y} \sum_{m=1}^{\infty} \sum_{n=0}^{\infty} \tilde{J}_{x,mn}(t) \sin\left(\frac{m\pi x}{L_x}\right) \cos\left(\frac{n\pi y}{L_y}\right), \quad [17]$$

$$J_y(x, y, t) = \frac{1}{L_x L_y} \sum_{m=0}^{\infty} \sum_{n=1}^{\infty} \tilde{J}_{y,mn}(t) \cos\left(\frac{m\pi x}{L_x}\right) \sin\left(\frac{n\pi y}{L_y}\right). \quad [18]$$

We insert Eq. 14, 17, and 18 into Eq. 12, interchange differentiation and summation, and finally compare each term in the series on the left- and right-hand side. The result is

$$\tilde{\rho}_{mn} f'_{mn}(t) = -\frac{m\pi}{L_x} \tilde{J}_{x,mn}(t) - \frac{n\pi}{L_y} \tilde{J}_{y,mn}(t). \quad [19]$$

For $m = n = 0$, the right-hand side is 0 so that $f'_{00}(t) = 0$. From this result and Eq. 15, we can deduce that

$$f_{00}(t) = 1 \quad \text{for all } t. \quad [20]$$

Similarly, we obtain, after inserting Eq. 17 and 18 into Eq. 13,

$$\frac{n}{L_y} \tilde{J}_{x,mn}(t) = \frac{m}{L_x} \tilde{J}_{y,mn}(t). \quad [21]$$

Combining Eq. 19 and Eq. 21, we can solve for the Fourier coefficients of the flux,

$$\tilde{J}_{x,mn}(t) = -\frac{mL_x L_y^2}{\pi(m^2 L_y^2 + n^2 L_x^2)} \tilde{\rho}_{mn} f'_{mn}(t), \quad [22]$$

$$\tilde{J}_{y,mn}(t) = -\frac{nL_x^2 L_y}{\pi(m^2 L_y^2 + n^2 L_x^2)} \tilde{\rho}_{mn} f'_{mn}(t). \quad [23]$$

In summary, a flow-based density-equalizing projection is vortex-free if and only if $f_{mn}(t)$ satisfies Eq. 15, 16, 20 and the Fourier coefficients of the flux obey Eq. 22 and 23.

Equations 4–7 in the main text as a special density-equalizing projection with vortex-free flow. There are many possible choices of f_{mn} consistent with Eq. 15, 16 and 20. The diffusion-based method of Ref. (5) corresponds to the choice

$$f_{mn,\text{diff}}(t) = \exp\left[-\left(\frac{m^2}{L_x^2} + \frac{n^2}{L_y^2}\right)t\right]. \quad [24]$$

According to Eq. 22 and 23, the Fourier coefficients of the flux are then given by

$$\tilde{J}_{x,mn,\text{diff}}(t) = \frac{m}{\pi L_x} \tilde{\rho}_{mn} \exp\left[-\left(\frac{m^2}{L_x^2} + \frac{n^2}{L_y^2}\right)t\right], \quad [25]$$

$$\tilde{J}_{y,mn,\text{diff}}(t) = \frac{n}{\pi L_y} \tilde{\rho}_{mn} \exp\left[-\left(\frac{m^2}{L_x^2} + \frac{n^2}{L_y^2}\right)t\right]. \quad [26]$$

It is computationally disadvantageous that t appears in the argument of the exponential function in Eq. 24–26. Whenever the numerical integration of Eq. 11 must advance the time t by

a small increment, the Fourier coefficients, including the exponential function, must be computed again. Although modern computers can evaluate the exponential function relatively quickly, it is still slower than the four basic arithmetic operations (i.e., addition, subtraction, multiplication, and division). Even more time-consuming than the exponential function are the backward Fourier transforms to $\rho(x, y, t)$ and $\mathbf{J}(x, y, t)$, which we need in order to evaluate \mathbf{v} appearing in Eq. 11.

The alternative approach that we explore in this article is based on the choice

$$f_{mn}(t) = \begin{cases} 1 & \text{if } m = n = 0, \\ 1 - t & \text{if } (m, n) \neq (0, 0) \text{ and } 0 \leq t \leq 1, \\ 0 & \text{otherwise.} \end{cases} \quad [27]$$

instead of Eq. 24. Performing the backward transform in Eq. 14 shows that the density is

$$\rho(x, y, t) = \begin{cases} (1 - t) \rho(x, y, 0) + t \bar{\rho} & \text{if } 0 \leq t \leq 1, \\ \bar{\rho} & \text{if } t > 1. \end{cases} \quad [28]$$

Although the physical interpretation of the resulting flow is now less intuitive than for the diffusion-based method, the mathematical literature has explored solutions of the prescribed Jacobian equation 8 based on Eq. 28 ((25, 51)).

The Fourier coefficients of the flux follow from Eq. 22 and 23,

$$\tilde{J}_{x,mn}(t) = \begin{cases} \frac{mL_x L_y^2}{\pi(m^2 L_y^2 + n^2 L_x^2)} \tilde{\rho}_{mn} & \text{if } 0 \leq t \leq 1, \\ 0 & \text{otherwise.} \end{cases} \quad [29]$$

$$\tilde{J}_{y,mn}(t) = \begin{cases} \frac{nL_x L_y}{\pi(m^2 L_y^2 + n^2 L_x^2)} \tilde{\rho}_{mn} & \text{if } 0 \leq t \leq 1, \\ 0 & \text{otherwise.} \end{cases} \quad [30]$$

Upon inserting Eq. 29 and 30 into Eq. 17 and 18, we obtain the flux

$$J_x(x, y, t) = -\frac{L_y}{\pi} \sum_{m=1}^{\infty} \sum_{n=0}^{\infty} \left[\frac{m}{m^2 L_y^2 + n^2 L_x^2} \tilde{\rho}_{mn} \times \sin\left(\frac{m\pi x}{L_x}\right) \cos\left(\frac{n\pi y}{L_y}\right) \right], \quad [31]$$

$$J_y(x, y, t) = -\frac{L_x}{\pi} \sum_{m=0}^{\infty} \sum_{n=1}^{\infty} \left[\frac{n}{m^2 L_y^2 + n^2 L_x^2} \tilde{\rho}_{mn} \times \cos\left(\frac{m\pi x}{L_x}\right) \sin\left(\frac{n\pi y}{L_y}\right) \right] \quad [32]$$

for $0 \leq t \leq 1$. For $t > 1$, we simply get $J_x = J_y = 0$. When we divide J_x and J_y by the density ρ in Eq. 28, we obtain equations 5 and 6 in the main text.

There are multiple advantages when choosing Eq. 27 instead of Eq. 24.

- As we have just derived from Eq. 27, the flux is zero after $t = 1$. It follows that $\mathbf{T}_1 = \mathbf{T}_\infty$. Hence, there is no need to take the limit $t \rightarrow \infty$ when we perform the integral in Eq. 11. In practice, we no longer need to apply heuristics to test whether the integrand at time t is small enough to terminate the numerical integration. Instead, we integrate until the fixed upper integration limit $t = 1$, which is easier to implement.

- Unlike in the diffusion-based method, we can calculate the density $\rho(x, y, t)$ in Eq. 28 without Fourier transforms.
- In the diffusion-based method, the Fourier coefficients of the flux (Eq. 25 and 26) are time-dependent. Therefore, at every new time step during the numerical integration of Eq. 11, we must carry out a new backward Fourier transform. By contrast, the right-hand sides of Eq. 31 and 32 do not depend on t . It suffices to perform the summations once at the start of the algorithm, most efficiently with the fast Fourier transform technique (29). If we store the result in memory, we do not need any more Fourier transforms at all during the integration.
- After computing the sums in Eq. 31 and 32 at the beginning of the code, we only need addition, subtraction, multiplication, and division. In particular, we never need to evaluate the exponential function that appears in Eq. 24 of the diffusion-based method.

The overall effect is remarkably fast computer code. For typical runs, we find that a serial implementation of the algorithm based on Eq. 27 only takes around 18% of the time needed for the diffusion-based method. By parallelizing the integrator, it is possible to speed up the code even further. With a 12-core processor, we were able to reduce the time needed by the new algorithm to only around 3% of the run-time for the diffusion-based code.

3. Tissot ellipses and angular-distortion metrics

In cartography, the Tissot indicatrix is a visual and numerical concept to analyze the distortions generated by a map projection. Introduced by Nicolas Auguste Tissot in the nineteenth century, the Tissot indicatrix has become an important tool, especially when characterizing projections of the Earth's (nearly) spherical surface onto a two-dimensional plane. The framework of our article is different: we are transforming a two-dimensional map (the cartogram input) to another two-dimensional map (the cartogram). Still, we can use Tissot indicatrices to measure the magnitude of the distortions produced by different cartogram algorithms.

Tissot ellipses. Consider an infinitesimally small circle centered at (x, y) on the input map. Locally, a smooth map projection \mathbf{T} is approximately equal to the affine transformation

$$\mathbf{T}(x + \delta_x, y + \delta_y) \approx \mathbf{T}(x, y) + \nabla \mathbf{T}(x, y) \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix}, \quad [33]$$

so long as δ_x and δ_y are sufficiently small. Here $\nabla \mathbf{T}$ is the Jacobian matrix. It can be shown that any affine transformation applied to a circle results in an ellipse. The Tissot indicatrix of (x, y) under the projection \mathbf{T} is the ellipse generated by the affine transformation on the right-hand side of Eq. 33 when applied to the infinitesimal circle at (x, y) . In the left-hand column of Fig. 8, we show several circles placed at regularly spaced locations on the input maps of our benchmarking examples (USA by electors, India and China by GDP). We use a finite radius to make the circles visible. In the middle and right-hand columns, we show the corresponding Tissot indicatrices centered at locations $\mathbf{T}(x, y)$ on diffusion and fast-flow based cartograms, respectively.

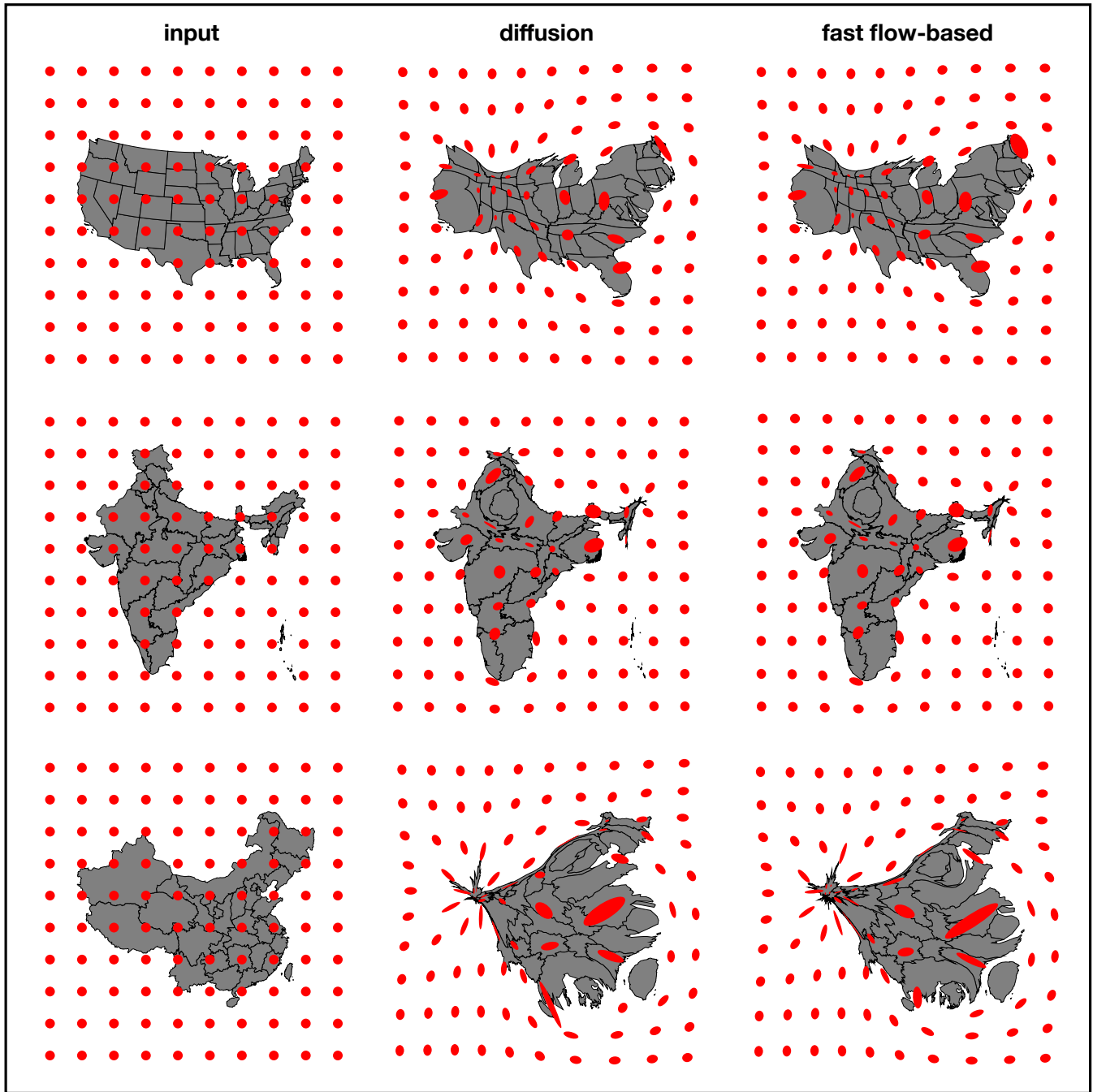


Fig. 8. Tissot indicatrices obtained for the diffusion-based algorithm (middle column) and the fast flow-based algorithm proposed in the main text (right column). The unprojected circles are displayed in the left column. The cartograms for the USA (top row), India (middle row), mainland China and Taiwan (bottom row) are based on the same data as Fig. 3, 4, and 5 of the main text.

Angular-distortion metrics. It is desirable for a density-equalizing projection \mathbf{T} to preserve shapes as much as possible so that each area is easily recognizable by the reader of the cartogram. One way to interpret shape preservation is to demand that angles remain locally unchanged by the transformation \mathbf{T} . This property is referred to as *conformality*. Equivalently, a conformal transformation must satisfy both Cauchy-Riemann equations

$$\frac{\partial T_x}{\partial x} = \frac{\partial T_y}{\partial y}, \quad \frac{\partial T_x}{\partial y} = -\frac{\partial T_y}{\partial x}.$$

Together with the prescribed Jacobian equation (1) in the main text, a conformal density-equalizing projection would have to satisfy three equations. In general, two functions T_x and T_y cannot satisfy three independent constraints so that a perfectly conformal density-equalizing solution is infeasible (21). Yet, a visually pleasing cartogram should deviate from conformality as little as possible. Although our present paper focuses on building a fast cartogram algorithm rather than achieving small conformality error, we compute several conformality metrics to verify that our proposed algorithm produces cartograms that are in this respect as good as the state-of-the-art diffusion algorithm.

Angular distortion metrics can be derived from the properties of Tissot ellipses. Consider the Tissot ellipse that is the image of the unit-radius circle centered at (x, y) after applying the affine transformation of Eq. 33. We denote the length of the ellipse’s semi-major and semi-minor axis by $a(x, y)$ and $b(x, y)$, respectively. In the case of a conformal projection \mathbf{T} , we would have $a(x, y) = b(x, y)$. That is, the Tissot ellipse would be a circle whose radius can be smaller or bigger than 1. Hence, we can define a measure of the angle distortion at (x, y) by

$$e(x, y) = \ln \left(\frac{a(x, y)}{b(x, y)} \right), \quad [34]$$

as described for example in Ref. (39). We choose the average

$$e_a = \frac{1}{|\Omega|} \int_{\Omega} \ln \left(\frac{a(x, y)}{b(x, y)} \right) dx dy,$$

and the largest value

$$e_{\infty} = \sup_{\mathbf{x} \in \Omega} \ln \left(\frac{a(x, y)}{b(x, y)} \right)$$

as two global measures for the distortion error, where Ω is the total area of the cartogram. Here we choose Ω as the $L_x \times L_y$ bounding rectangle described in the section “Benchmarking the algorithm with data for the USA, India, and China” in the main text. Another angular-distortion metric can be computed from the local maximum angular-value change (see derivation in (40)),

$$\tilde{e}(x, y) = 2 \arcsin \left(\frac{a(x, y) - b(x, y)}{a(x, y) + b(x, y)} \right), \quad [35]$$

which also provides two global angular-distortion metrics \tilde{e}_a and \tilde{e}_{∞} . We display these errors for both the diffusion-based and our new proposed algorithm in Table 1 of the main text.

4. Polygon-level distortions

The metrics e and \tilde{e} defined in Eq. 34 and 35 are local: they can be computed only for “*all-coordinates*” cartograms for which we know the transformation \mathbf{T} at every location (x, y) . In order to obtain metrics that are well-defined for general cartograms, one has to measure distortions at the level of polygons instead of all coordinates. Such metrics have been introduced in several previous articles (41, 43, 52). According to some metrics, the fast flow-based algorithm defined in the main text and several other contiguous methods, including the diffusion cartogram, are already optimal. For example, both the diffusion and fast flow-based algorithm succeed in rescaling the regions to their objective areas and preserve the adjacency between polygons. Three metrics that meaningfully compare the diffusion and fast-flow based algorithm are (1) the average aspect ratio α , (2) the total Hamming distance δ , and (3) the relative position error θ . We describe them below and display the results for each cartogram in Table 1 of the main text.

Average aspect ratio. Cartograms in which polygons become thin and elongated are difficult to read. It is also difficult to place labels inside such polygons. The aspect ratio of a polygon quantifies how stretched it appears. For the i -th polygon on the cartogram, we define $l_i(\phi)$ and $s_i(\phi)$ as the longer and shorter side length, respectively, of the bounding rectangle whose edges are at angles ϕ and $\phi + 90^\circ$ with respect to the x -axis.

We define $\phi_{\min, i}$ as the angle at which the bounding rectangle for the i -th polygon has the minimum area,

$$\phi_{\min, i} = \arg \min_{\phi \in [0^\circ, 90^\circ]} [l_i(\phi) \cdot s_i(\phi)].$$

The aspect ratio of the i -th polygon is the ratio of the larger to the smaller side length of the bounding rectangle of minimum area for that polygon,

$$\alpha_i = \frac{l_i(\phi_{\min, i})}{s_i(\phi_{\min, i})}.$$

If there are p polygons on the cartogram, we define α as the mean aspect ratio,

$$\alpha = \frac{1}{p} \sum_{i=1}^p \alpha_i.$$

Total Hamming distance. The Hamming distance h measures the difference in the shapes of two polygons. It is computed by superimposing one polygon on top of another and measuring the fraction of area that lies in only one, but not both polygons,

$$h = \frac{\text{area in exactly one polygon}}{\text{sum of areas of individual polygons}}.$$

In our application, one of the polygons is from the input map, the other is the corresponding polygon from the cartogram. We rescale the cartogram polygon so that it has the same area as the polygon before the cartogram projection. Otherwise we would unfairly penalize cartograms that correctly changed the polygon areas to their objective values. To make the measure translation invariant, we define δ_i as the minimum Hamming distance of all possible translations of the rescaled i -th cartogram polygon with respect to the i -th unprojected polygon. The total Hamming distance δ is obtained by summing the Hamming distances of all polygons.

Relative position error. We can quantify changes in the relative position of two polygons i and j between input map and cartogram by measuring the angle ϕ_{ij} between the lines connecting the centroids before and after the projection. If $\mathbf{c}_i, \mathbf{c}_j$ are the centroids on the input map and $\mathbf{d}_i, \mathbf{d}_j$ on the cartogram, then

$$\phi_{ij} = \arccos \left(\frac{(\mathbf{c}_i - \mathbf{c}_j) \cdot (\mathbf{d}_i - \mathbf{d}_j)}{|\mathbf{c}_i - \mathbf{c}_j| \cdot |\mathbf{d}_i - \mathbf{d}_j|} \right).$$

We define the relative position error θ as the average of ϕ_{ij} over all possible pairs of polygons. We also divide by π ,

$$\theta = \frac{2}{p(p-1)\pi} \sum_{i=1}^{p-1} \sum_{j=i+1}^p \phi_{ij},$$

so that $\theta \in [0, 1]$ (43).