A constraint-based layout approach to data visualization

Joey Huchette and Hadley Wickham, *Rice University*

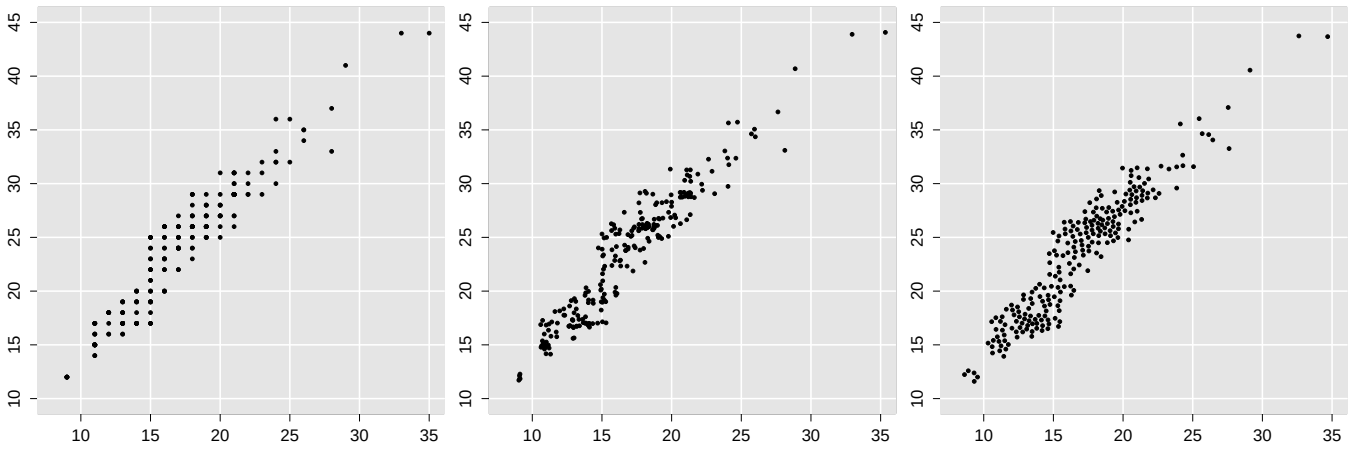


Fig. 1. A scatterplot with extreme overplotting (left), with random jitter (middle) and the result from our constraint-based layout framework (right).

Abstract—This work explores the connection between statistical data visualization problems and simulated physical systems. Applying techniques for constraint-based layout, data points are treated as physical objects under the influence of forces such as gravity and electrostatic repulsion. Many common statistical data visualization problems can be viewed through this lens; we consider scatterplot overplotting and the Dorling cartogram as our motivating problems. The reference implementation is straightforward and capable of producing instructive visualizations. Additionally, we present a suite of tools useful for tuning the system parameters that determine the quality of the produced visualization.

Index Terms—Constraint-based layout, visualization, statistical graphics

1 Introduction

Approaching a statistical visualization problem is inherently a balancing act, a trade-off between fidelity to the data and the need for visual clarity. A canonical example would be a scatterplot with extreme overplotting; that is, with many data points that are extremely close together. Oftentimes it is beneficial to slightly perturb clusters of points, to show the density of the points while maintaining the original spatial layout as much as possible. These alterations can be performed in an ad-hoc way, but it makes sense to try to formalize the manipulations. As such, a useful analogy is to treat the data points as elements in a physical system, subject to certain prescribed forces that are set to best construct the desired image. The visualization problem can then be viewed as a graph constraint-based layout problem, with spatial objects (such as data points) as nodes and edges encoding a force relationship between these spatial objects. The problem is then to allow the system to evolve and, hopefully, converge to a useful visualization.

The rest of the paper is structured as follows. Section 2 covers the existing literature in constraint-based layout and graph visualization. Section 3 briefly presents the two motivating problems used as examples in the text. Section 4 describes the physical system framework used in detail (particularly the forces we found most useful), and presents useful information and tools gleaned from our reference implementation, the source code of which is available at `http://github.com/hadley/imvisoned`. Section 5 displays and discusses the results of the approach on the motivating problems and describes the challenges that arose in generalizing the approach.

- *Joey Huchette is a student at Rice University. E-mail: joehuchette@gmail.com.*
- *Hadley Wickham is Chief Scientist at RStudio. E-mail: h.wickham@gmail.com.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 27 September 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Related Work

We attempt to apply approaches gleaned from these graph-drawing literature to the context of statistical data visualization problems. In particular, we attempt to apply these methods to relatively small data sets, specifically applied to problems arising in 2D scatterplots and bubbleplots. Certain specific visualizations have been implemented in a way that use similar approaches; see, for example, beeswarm [14] and dot plots [27].

`D3` [4], a data visualization software package written in JavaScript, includes a physical system framework that served as a primary inspiration for this work. In particular, a talk given to VisualizeMyData [3] hints at the potential for this approach as applied to statistical data visualization.

More broadly, graph drawing and constraint-based layout have been studied formally in recent years [7, 6]. Force-directed methods exist in a number of contexts: [8] discusses a topological force-directed drawing algorithm, [15] presents a curvilinear drawing approach, and [2] attempts to preserve edge-crossing properties in the original drawing. The MagnetViz software [23] presents a related approach for general graph layout problems, using "magnets" placed on the graph to induce desirable properties in the graph drawing. Additionally, literature exists to apply unique constraint types to standard graph drawing problems [5, 12]. These approaches are often tailored to a specific problem arising in a certain application; for instance, visualizing link bundles in dense networks [21].

Graph layout problems have been studied formally in with application domains as far afield as archeology [17] and circuit design [24]; significant progress has been made in developing algorithms for specific applications. In particular, these problems have been tackled by those interested in their relationship to computer graphics [20, 10]. Due to the particular needs of the industry, these works focus on algorithms that are fast [11], efficient [16], and simple to implement [19] for problems typically encountered in the scope of computer graphics.

3 Motivating Problems

Throughout the text, two constraint problems will be discussed as motivation for the various topics addressed: a traditional overplotting problem arising in scatterplot construction, and the Dorling cartogram problem.

3.1 Overplotting

Overplotting is often encountered in the construction of scatterplots: how can we best to depict the distribution of the data when the points overlap to the resolution of the plot? A prototypical example is shown Figure 1; since the data points take discrete values, it is impossible to estimate the density of the points in this particular visualization. Also shown is a typical fix: random noise added to the data in an attempt to convey the density of the data. A formulation of the overplotting problem in this context would be to find an optimal spatial layout of the graph: one that accurately depicts the distribution of the data, while also preserving the original spatial layout as much as possible.

3.2 Dorling Cartogram

A Dorling (a.k.a. circular) cartogram [9] is a visualization that displays geographic objects (e.g. states, countries, etc.) as circles whose size corresponds to some meaningful data value (e.g. population, crime rate, etc.). The intent is to preserve the geographic structure as much as possible, while displaying this additional information. In order to best impart information, a strict non-overlapping constraint is imposed on the data points; that is, the distance between any two nodes cannot be less than the sum of their radii. The Dorling cartogram can be viewed as a bubble plot with added geographic constraints added. A prototypical Dorling cartogram is shown in Figure 2.

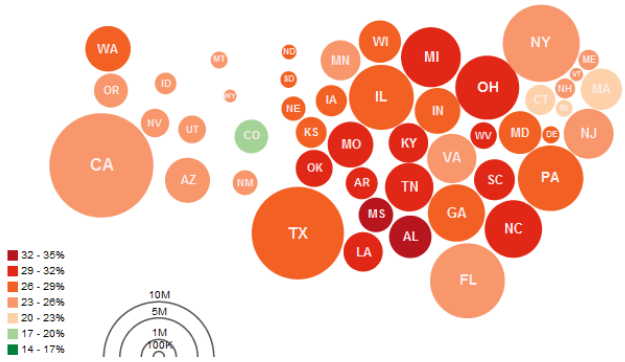


Fig. 2. A prototypical Dorling cartogram. The plot shows each U.S. state, with size corresponding to population. Image sourced from [18].

4 Framework Development

4.1 Overarching Structure

This work deals with data sets represented as graphs; that is, collections of nodes, with edges that encode connectivities between these nodes. Nodes most often represent the spatial location of a data point, whether it is the starting position or the current position after a series of iterations. Nodes and edges can carry additional information that determine the relative strength of the forces. For instance, in the Dorling cartogram, the radii of each node is stored, and an edge also encodes the sum of the radii of the two endpoints, used to enforce the non-overlap constraint.

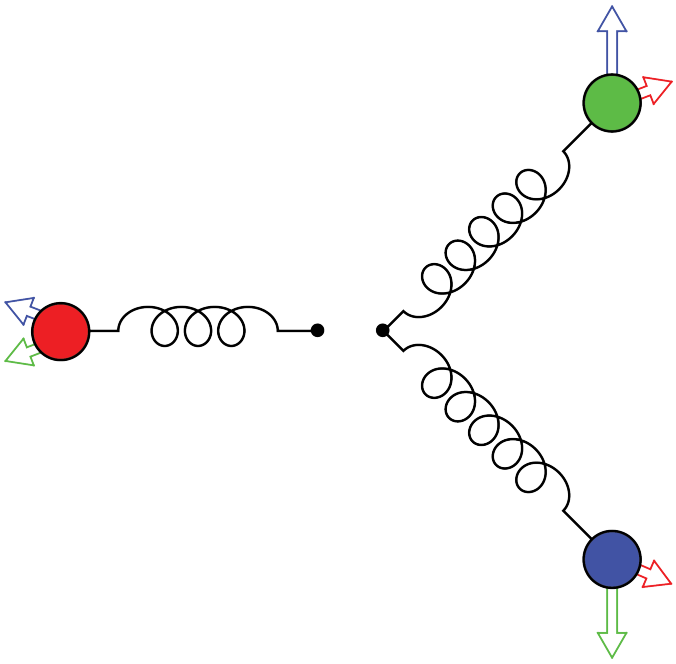


Fig. 3. A simple system with three nodes and homing, repulsive, and non-overlap forces. Each node take a different color and is tethered by a spring to its initial position. Transparent arrows indicate the direction and strength of the repulsive force between pairs of nodes. The nodes have nonzero area, and the non-overlap force simply requires that one node does not lie on top of another. Not drawn to scale.

The forces applied in will be applied discretely through time, through an approach informed by Verlet integration for numerically solving equations of motion [25]. An iterative process is used, calculating the instantaneous displacement for each force on the nodes separately, applying all forces cumulatively, and then repeating the process until some convergence criterion is met. Per [4] and [10], an artificial "cooling" factor is introduced, decreasing the strength of the forces at every iteration, inducing convergence of the system to some equilibrium in a finite (and hopefully tractable) number of iterations.

An instructive way to conceptualize the model is depicted in Figure 3. The data points are tethered to their initial positions with springs, and there is an electrostatic repulsion between these points. Additionally, in the Dorling cartogram instance the points have nonzero area, and so two data points cannot occupy the same space in the plot. This model incorporates the three main forces of interest: homing, repulsion, and non-overlapping. Variations of these basic forces can take a number of forms, but we present a generic form that will be instructive of the difference among these different force classes.

In the algorithms in the coming sections, $nodes \in \mathbb{R}^{n\times 2}$ is the array of spatial data for each of the n nodes in the system. Additionally, β is used as a generic "strength" factor; in general, this factor will be different for each force in a given iteration, and is dependent on the cooling factor. All the methods return $disp \in \mathbb{R}^{n\times 2}$, which contains the change in position for each of the nodes.

4.1.1 Repulsion

The repulsion force models Coulombs Law describing the force between charged particles. As such, it scales with the inverse square of the displacement between particles. A simple version is shown in Algorithm 1.

4.1.2 Homing

The homing force pulls the nodes back to their original position on the plot. This is particularly useful for maintaining the original layout of the data set as much possible: for instance, to counteract the

Algorithm 1 A repulsion force

procedure REPULSE($nodes$, β)
  for $i \leftarrow 1,\ldots,n$ **do**
    for $j \leftarrow i+1,\ldots,n$ **do**
      $dx \leftarrow nodes(i,1) - nodes(j,1)$
      $dy \leftarrow nodes(i,2) - nodes(j,2)$
      $\ell \leftarrow \sqrt{dx^2 + dy^2}$
      $disp(i,1) \leftarrow disp(i,1) - \frac{\beta \cdot dx}{\ell^2}$
      $disp(i,2) \leftarrow disp(i,2) - \frac{\beta \cdot dx}{\ell^2}$
      $disp(j,1) \leftarrow disp(j,1) + \frac{\beta \cdot dx}{\ell^2}$
      $disp(j,1) \leftarrow disp(j,2) + \frac{\beta \cdot dx}{\ell^2}$
    end for
  end for
  return $disp$
end procedure

repulsion force applied to alleviate overplotting. The homing force as implemented is analogous to Hooke's Law for describing a spring force, which scales with the displacement between nodes. A simple homing force is described in Algorithm 2, where $start \in \mathbb{R}^{n\times 2}$ is the array of starting locations for each of the n nodes..

Algorithm 2 A homing force

procedure HOMING($start$, $nodes$, β)
  for $i \leftarrow 1,\ldots,n$ **do**
    $dx \leftarrow nodes(i,1) - start(i,1)$
    $dy \leftarrow nodes(i,2) - start(i,2)$
    $\ell \leftarrow \sqrt{dx^2 + dy^2}$
    $disp(i,1) \leftarrow disp(i,1) - \beta \cdot dx \cdot \ell$
    $disp(i,2) \leftarrow disp(i,2) - \beta \cdot dy \cdot \ell$
  end for
  return $disp$
end procedure

4.1.3 Non-overlap

The non-overlap force is meant to counteract any overlapping between adjacent nodes; for example, it displaces nodes if the distance between any two nodes is less than the sum of their prescribed radii (or some other, more general, metric). There is no simple analogy to a physical force here, and since it is applied for all violations at once, it often needs to be applied repeatedly for total compliance. The non-overlap force is particularly useful for the Dorling cartogram problem. Algorithm 3 depicts a simple non-overlapping force, where $radii \in \mathbb{R}^n$ contains the radius of each node in the system.

4.1.4 Gravitational

Taking a cue from [4], we also incorporate a simple gravitational force. Rather than attempting to emulate Newton's gravitational law (which in a sense would be equivalent to our repulsion force), "gravity" here merely refers to a force pulling all nodes towards a single point, with the strength proportional to the displacement. We found this particularly helpful in helping to counteract repulsive and non-overlap forces. The general method is shown in Algorithm 4, where $center \in \mathbb{R}^2$ is the center of attraction for all nodes in the plot.

4.2 Reference Implementation

The implementation of the various forces discussed is relatively straightforward; they can be expressed very concisely, as they boil down to either a single loop through either the nodes or the edges (which most commonly means a doubly-nested loop through the nodes). We used the R statistical programming language for the reference implementation, since it lends itself nicely to these problems in data manipulation and visualization. However, since many of the

Algorithm 3 A non-overlapping force

procedure NONOVERLAP($radii$, $nodes$)
  for $i \leftarrow 1,\ldots,n$ **do**
    $c_i = radii(i)$
    for $j \leftarrow i+1,\ldots,n$ **do**
      $dx \leftarrow nodes(i,1) - nodes(j,1)$
      $dy \leftarrow nodes(i,2) - nodes(j,2)$
      $\ell \leftarrow \sqrt{dx^2 + dy^2}$
      $c_j \leftarrow radii(j)$
      $\delta \leftarrow c_i + c_j$
      if $\ell < \delta$ **then**
        $disp(i,1) \leftarrow disp(i,1) + \frac{dx \cdot c_i}{\ell^2}$
        $disp(i,2) \leftarrow disp(i,2) + \frac{dy \cdot c_i}{\ell^2}$
        $disp(j,1) \leftarrow disp(j,1) - \frac{dx \cdot c_j}{\ell^2}$
        $disp(j,2) \leftarrow disp(j,2) - \frac{dy \cdot c_j}{\ell^2}$
      end if
    end for
  end for
  return $disp$
end procedure

Algorithm 4 A gravitational force

procedure GRAVITY($center$, $nodes$, β)
  for $i \leftarrow 1,\ldots,n$ **do**
    $disp(i,1) \leftarrow \beta \cdot (center(1) - nodes(i,1))$
    $disp(i,2) \leftarrow \beta \cdot (center(2) - nodes(i,2))$
  end for
  return $disp$
end procedure

forces are applied only within some threshold interval, introducing a conditional statement, it is not straightforward to "vectorize" many of these forces in an attempt to get the best performance in the naive R implementation. As such, we used the packages `inline` [22] and `Rcpp` [13] interface R for higher-level coding and C++ for these large loops. These packages allowed a straightforward implementation of the forces without sacrificing performance.

More complex data structures can be introduced to the implementation to further enhance performance. The quadtree is a natural choice for (2-dimensional) forces applied to nodes within a certain threshold radius; the structure provides an efficient look-up mechanism for these closest neighbors. The `SearchTrees` package [1] provides a useful R implementation of the quadtree data structure.

In summary, the naive implementation of the forces in R is very straightforward, and with a little effort, the efficiency of the code can be increased by a significant factor. The source code is available at `https://github.com/hadley/imvisoned`.

4.3 Construction Tools

To help tune system parameters effectively—often a difficult task—we developed a suite of tools: iteration movies, force line drawings, and a parameter-setting widget.

4.3.1 Iteration Slideshow

The most basic tool we developed is a simple slideshow with each frame corresponding to an iteration. This tool helps visualize the entire evolution process of the system, rather than just a final result which may be dismayingly unhelpful if the starting force parameters were chosen poorly. Notably, a system pause is needed for smaller problems, otherwise the iterations will pass imperceptibly fast.

4.3.2 Force Lines

To help gain an intuitive understanding of the relative strength of the forces being applied, force lines were added to the iteration movie. At

each iteration, line segments were plotted from the starting position to the ending position for each node.

The force lines helped immensely in tuning parameters; they explicitly show the strength and orientation each force is applying to the points in the system. Figure 4 and 5 show selected frames from the iteration slideshow for the data from Figure 1, with different initial conditions: a slight perturbation of the starting positions (Figure 4) and a uniform random position within the original frame (Figure 5).

4.3.3 Widget

The most elaborate tool we developed was a GUI widget to tweak parameter values and observe the resulting iterations in real-time. Built on top of the `gWidgets` R package [26], which facilitates GUI creation in R via a higher-level API, the widget offers sliders for each of the parameters. An iteration movie is created, allowing the user to track the effect of parameter changes on the output. Additionally, the cooling factor was restarted for each parameter change, which helped show more completely the changes the parameter tweaking affected. We found it useful to map the slider values logarithmically to the parameter values, since parameter values can span orders of magnitude. Finally, a randomizing button was added, sending the points to random spots in the plot's domain; seeing the forces attempting to create some sort of equilibrium after such a drastic change was often instructive. Figure 6 includes a screenshot of our GUI in use.

5 Results

5.1 Overplotting

For the overplotting problem, we again return to the test case shown in Figure 1. We found a repulsive force and a homing force were sufficient for inducing the desired results, once properly calibrated. Our results are shown in Figure 1, alongside the original plot and the "jittered" plot. The plot manages to maintain the spatial orientation of the data points as much as possible, while also showing the density of the data points at each discrete value. The plot has the outline as the original one, but manages to convey the point density much more clearly.

Referring back to Figures 4 and 5, we note that the randomized starting locations used in the second experiment has a faster convergence. It was also considerably more robust, because if the parameters were not tuned appropriately, the strong repulsive forces seen in the first iteration could push the nodes far enough from their initial position that they were not drawn back into the frame before the cooling factor forced convergence. As such, we found the random initial starting point to be an effective approach for ensuring the resulting plot was not unduly sensitive to the calibration of the homing and repulsive forces.

5.2 Dorling Cartogram

Our second motivating problem, the Dorling cartogram, is more subtle. We found that a combination of gravitational, homing, and non-overlapping forces were most useful for the visualization. Most notably, compliance with the non-overlapping constraint on each point often drives the data points undesirably far from their original locations. To alleviate this problem, we applied the forces in three stages:

1. just non-overlap forces, applied until all nodes are compliant
2. all forces applied together, until the cooling point was reached
3. just non-overlap forces, applied until all nodes are compliant

We found this approach maintains the original layout of the plot to a suitable degree for the test problems we considered. In particular, it would immediately fix overlap violations, then attempt to revert to the original layout as much as possible (while accommodating for overlap violations), and then finally fix any remaining overlapping, which was usually minimal at this point.

Our chief test problem is similar to the one displayed in Figure 2: the U.S. states, with radius proportion to the states total area. The resulting plots for two such functions are depicted in Figure 7, alongside the starting configurations. Notably, the method is capable of producing valid (and useful) Dorling cartogram layouts for two different radius functions—one that produced few large nodes, one that produced roughly equally-sized nodes—with the same parameter tuning.

5.3 Challenges

Despite the relative ease with which the forces can be implemented, constructing a useful physical system turns out to often be a much more subtle issue than initially anticipated. Our initial goal—producing a "black box", which, when fed an arbitrary data set, produced an instructive visualization—proved a subtle and nontrivial endeavor. The complication arose from the relative strength of the various forces applied to the system: values that were useful for one data set were in general not transferable to another dataset or visualization type. The most common problem was scaling the parameters relative to each other such that one did not overpower the others. This can be seen in Figure 4; if the red repulsive forces are tuned relatively high, the first iteration would easily send the nodes extremely far off the frame—much farther than a relatively weak homing force could hope to right before the cooling factor artificially forces convergence. The tools discussed in Section 4.3 helped greatly in setting these parameters for a particular visualization.

6 Conclusion

The physical system analogy proved to be a fruitful one, capable of tackling constraint-based graph layout problems and producing instructive plots. Moreover, a handful of tools we developed made tweaking parameters in the physical system relatively straightforward. However, since the physical system is such an artificial construct, system parameters are difficult to determine in any generality, leaving them non-portable between projects. As such, the most pressing future work is a further analysis of the system parameter tuning. Ideally, there would be some way to automatically tune the physical system to a given data set and visualization, producing a useful plot in a single run. Additionally, user studies to empirically show the efficacy of the constructed visualizations are critical for the further development of this approach.

References

[1] G. Becker. *SearchTrees: Spatial Search Trees*, 2012. R package version 0.5.2. `http://cran.r-project.org/web/packages/SearchTrees/`.

[2] F. Bertault. A Force-Directed Algorithm that Preserves Edge Crossing Properties. In J. Kratochvyl, editor, *Graph Drawing*, volume 1731 of *Lecture Notes in Computer Science*, pages 351–358. Springer Berlin Heidelberg, 1999.

[3] M. Bostock. d3.j: Data-Driven Documents, 2012. Data Visualization Meetup.

[4] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[5] R. Chernobelskiy, K. I. Cunningham, M. T. Goodrich, S. G. Kobourov, and L. Trott. Force-Directed Lombardi-style Graph Drawing. In *Proceedings of the 19th international conference on Graph Drawing*, GD'11, pages 320–331, Berlin, Heidelberg, 2012. Springer-Verlag.

[6] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for Drawing Graphs: an Annotated Bibliography. *Comput. Geom. Theory Appl.*, 4(5):235–282, Oct. 1994.

[7] J. Díaz, J. Petit, and M. Serna. A Survey of Graph Layout Problems. *ACM Comput. Surv.*, 34(3):313–356, Sept. 2002.

[8] W. Didimo, G. Liotta, and S. Romeo. Topology-Driven Force-Directed Algorithms. In U. Brandes and S. Cornelsen, editors, *Graph Drawing*, volume 6502 of *Lecture Notes in Computer Science*, pages 165–176. Springer Berlin Heidelberg, 2011.

[9] D. Dorling. Area Cartograms: Their Use and Creation. Concepts and Techniques in Modern Geography 59, Quantitative Methods Research Group of the Royal Geographical Society, 1996.

[10] T. Dwyer. Scalable, Versatile and Simple Constrained Graph Layout. *Computer Graphics Forum*, 28(3):991–998, 2009.

[11] T. Dwyer, K. Marriott, and P. J. Stuckey. Fast Node Overlap Removal. In *In: Proc. 13th Int. Symp. on Graph Drawing (GD'05). Volume 3843 of LNCS. (2006) 153–164*, pages 153–164. Springer, 2005.

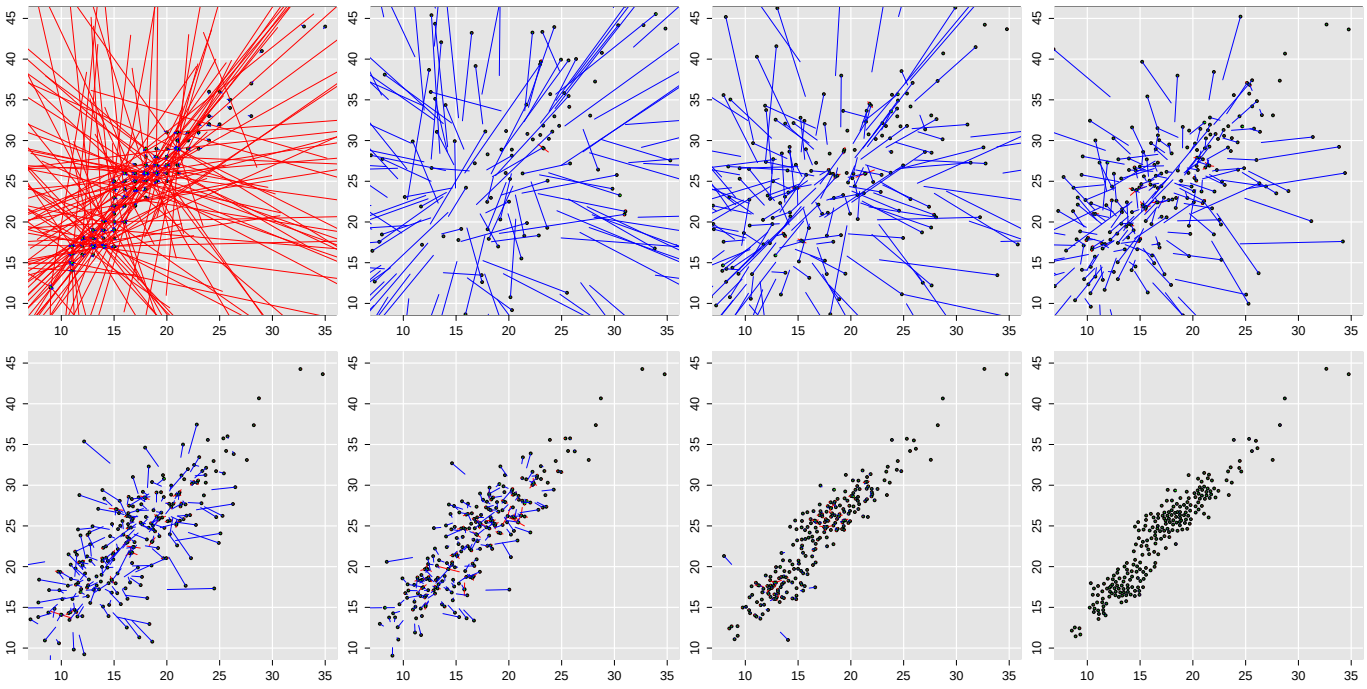


Fig. 4. Snapshots of the iterations used on the dataset in Figure 1 with added force lines: red for repulsion and blue for homing. Iterations 1, 2, 3, 4, 5, 6, 10, 20 are shown. The initial positions are slight perturbations of the raw data; as such, the first generation sees massive repulsion forces as the overplotting is corrected, while the next iterations see large homing forces as the points are drawn to their original positions. Due to the convergence of the system (aided by the cooling factor), later generations see little change.

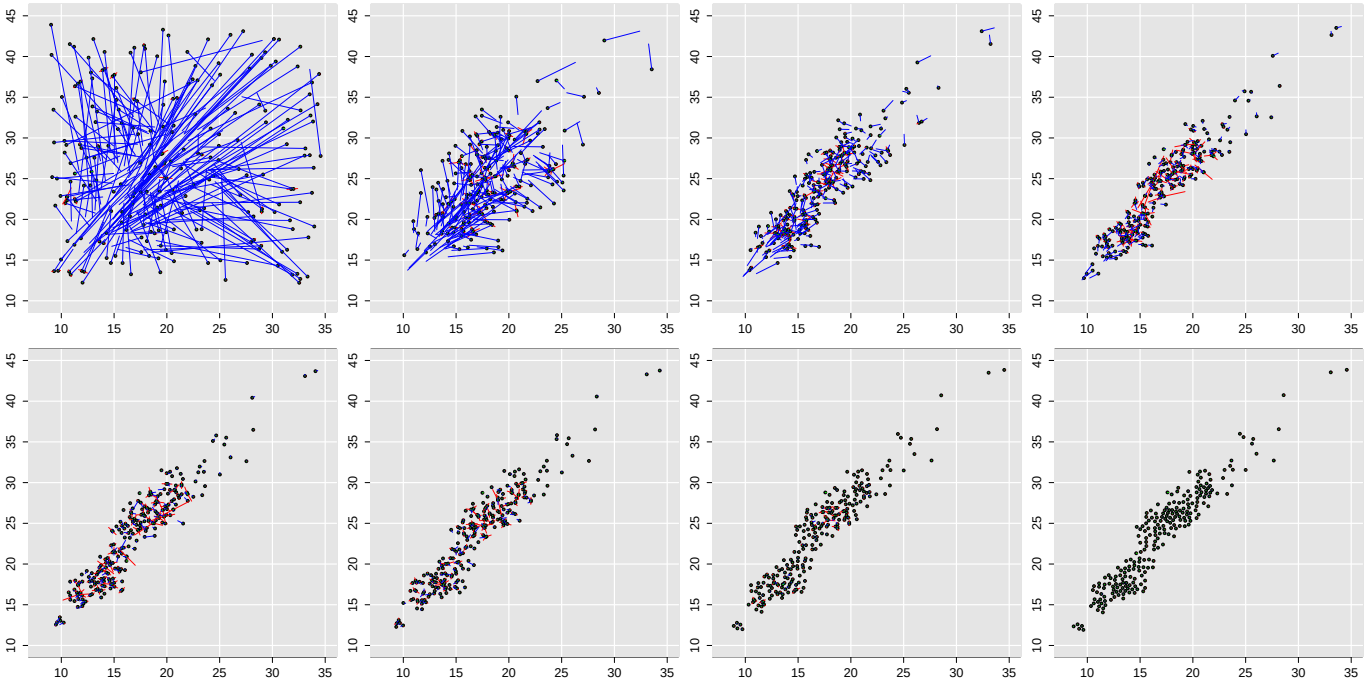


Fig. 5. Iteration snapshots similar to those in Figure 4, but with uniform random initial positions within the original frame of the plot. Iterations 1, 2, 3, 4, 5, 6, 10, 20 shown. The data points enjoy a much quicker convergence to their final positions; for this reason, this randomized approach is preferable for its speed and robustness.

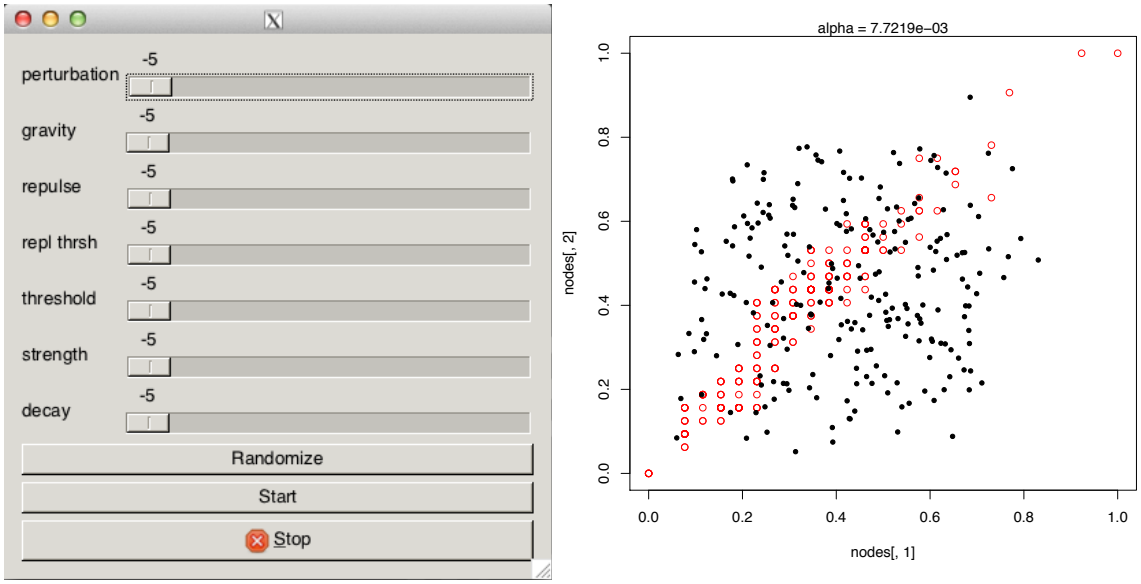
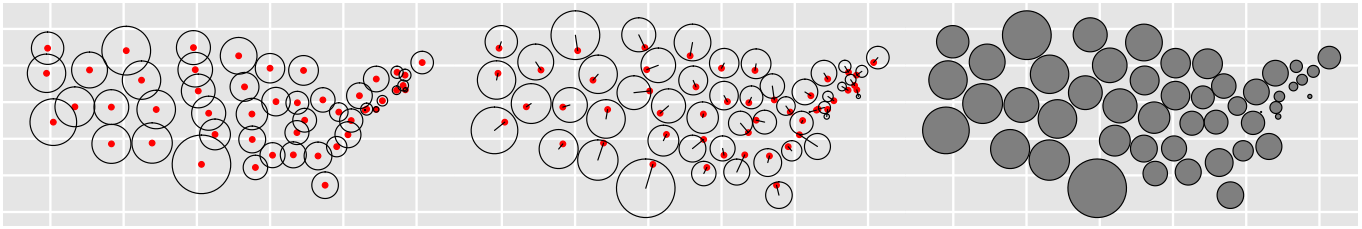
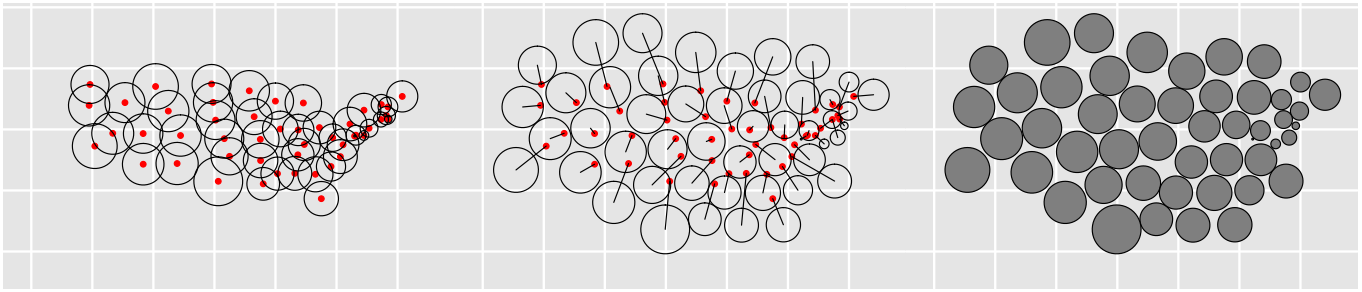


Fig. 6. (Left) Starting configuration for the widget. (Right) Sample output for the widget. Red circles show starting position, black dots show current position.



Radius function: $r \propto A$



Radius function: $r \propto \log(A+1)$

Fig. 7. Plots of the U.S. states, with two circle radius determinations (r is drawn circle radius, A is state area). (Left) The original plot, (Center) original and displaced circles overlaid, and (Right) just the displaced circles. Red circles indicate the center of the state's original location, while the black lines show the movement from start to final state. Note that the significant overlapping is alleviated, while still maintaining a contained layout.

[12] Dwyer, Tim and Robertson, George. Layout with Circular and Other Non-linear Constraints Using Procrustes Projection. In D. Eppstein and E. Gansner, editors, *Graph Drawing*, volume 5849 of *Lecture Notes in Computer Science*, pages 393–404. Springer Berlin Heidelberg, 2010.

[13] D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011.

[14] A. Eklund. *Beeswarm: The Bee Swarm Plot, an Alternative to Stripchart*, 2012. R package version 0.1.5. `http://cran.r-project.org/web/packages/beeswarm`.

[15] B. Finkel and R. Tamassia. Curvilinear Graph Drawing Using the Force-Directed Method. In J. Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 448–453. Springer Berlin Heidelberg, 2005.

[16] Y. Hu. Efficient and High Quality Force-Directed Graph Drawing. Technical report, Wolfram Research Inc, 100 Trade Center Drive, Champaign, IL 61820 USA, 2005.

[17] C. Hundack, P. Mutzel, I. Pouchkarev, and S. Thome. ArchE: A Graph Drawing System for Archaeology, 1997.

[18] V. O. Jeffrey Heer, Michael Bostock. A Tour Through the Visualization Zoo.

[19] U. Lauther. Multipole-Based Force Approximation Revisited - a Simple but Fast Implementation Using a Dynamized Enclosing-Circle-Enhanced k-d-Tree, 2007. 10.1007/978-3-540-70904-6_4.

[20] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position Based Dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109 – 118, 2007.

[21] N. H. Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the Design Space of Interactive Link Curvature in Network Diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 506–513, New York, NY, USA, 2012. ACM.

[22] O. Sklyar, D. Murdoch, M. Smith, D. Eddelbuettel, and R. Francois. Package 'inline'. 2012. `http://cran.r-project.org/web/packages/inline/inline.pdf`.

[23] A. Spritzer and C. Dal Sasso Freitas. Design and Evaluation of MagnetViz—A Graph Visualization Tool. *Visualization and Computer Graphics, IEEE Transactions on*, 18(5):822–835, 2012.

[24] C. D. Thompson. Area-Time Complexity for VLSI. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, STOC '79, pages 81–88, New York, NY, USA, 1979. ACM.

[25] L. Verlet. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.*, 159:98–103, Jul 1967.

[26] J. Verzani. gWidgetsWWW: Creating Interactive Web Pages within R. *Journal of Statistical Software*, 49(10):1–12, 2012.

[27] L. Wilkinson. Dot Plots. *The American Statistician*, 53:276–281, 1999.