Automated detection and repair of visual distortion in data graphics

Alex Kindel

December 14, 2012

Introduction

Computer-generated data graphics are a ubiquitous tool for communicating scientific findings to non-specialists. Visualization is a particularly appropriate mode of quantitative representation for communicating to non-specialists because it assumes only basic spatial and mathematical knowledge: the ability to identify discrete objects, estimate their relative sizes, and identify a simple quantitative comparison. Despite this widespread usage of data visualization as a tool for communicating information, however, there are a number of common distortions that can transform data graphics from intuitive to misleading. In this paper, I lay out three of the most common kinds of distortion (color misuse, nonlinear scaling, unnecessary dimensions) and demonstrate a tool for detecting and repairing distortion in bar charts, one of the most common forms of data visualization.

The learning/design problem

To be useful in a real-world context, the learning/design problem is posed in the following way. The tool is provided with a bar chart whose labels and axes have been removed, along with a vector containing the labels for each bar. The tool doesn't store knowledge from previous runs.[1] Ideally, the tool should run quickly to accommodate the average user's attention span and level of investment in testing data graphics. Given these challenges, I selected algorithms primarily based on how quickly they could the solve subproblems defined in this section.

[1] Storing this knowledge could make the problem easier: see Challenges section for discussion.

Misuse of color

Formally, colors are represented by points in the discrete, finite RGB-model subspace: $(r, g, b) \in S_{rgb} \subseteq \mathbb{R}^3$. Generally, the number of colors used in a nondistorted graphic should be, at most, the number of data points being represented (i.e. each representation has its own color) plus one color (usually white) for the background. Additional colors would be inappropriate because multiple colors in the background or bars would distract from the data being communicated. Fewer colors are possible in a number of cases, e.g. all bars being the same color to avoid any color distortion issues. Still, even in a graphic which uses the appropriate number of colors, the colors of individual representations may be unevenly distributed across the color spectrum. To repair this issue, colors can be initially assigned to nearby basic colors (RGB, CMY) at the corners of the color space, which guarantees maximum contrast between bars. These basic assignments can then be moved to the center of the cluster to approximate the average of the colors being reassigned, which minimizes the visibility of the repairs.

As it turns out, this subproblem can be quickly solved using a modified implementation of k-means clustering. Each pixel in the input image is mapped onto S_{rgb}, and centroids are initialized to colors on the boundary of the space (starting with corners). This runs relatively quickly because the nature of the color subspace limits the number of possible states the centroid can end up in. Additionally, the toler-

ance value can be set relatively high, since an absolute convergence isn't necessarily more desirable than a relatively less-exact assignment for each bar.

Detecting nonlinear scaling

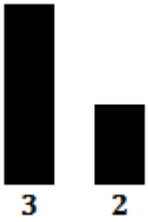
In data visualization, geometric shapes are used to represent relative quantities, usually numbers or percentages. In a nondistorted graphic, the size of a representation scales linearly with the data it represents. Formally, a scaling function can be represented as follows:

$$y_i = kx_i$$

Observe that the data point x_i and the area of its visual representation y_i are related only by k, the scaling factor. Importantly, linear data scaling preserves the observer's ability to correctly compare multiple pieces of quantitative data by examining their representations. The converse is also true: nonlinear data scaling can lead to errors resulting from visual comparison, especially if the graphic is unlabeled.

For example: say an observer sees a simplistic data graphic, as follows:

What do you estimate the proportion to be? (It seems fair to estimate that the larger bar is twice as large as the smaller bar, but feel free to make your own estimate.) Then, what if you later discovered the following labelings for each bar?



Was your estimate correct? Seems unlikely. Formally, the observed ratio between the bars is around

$$\hat{y}_1 : \hat{y}_2 :: 2 : 1$$

but the actual ratio of values for each bar is

$$x_1 : x_2 :: 3 : 2.$$

Note that the actual ratio for the bars is

$$y_1 : y_2 :: 9 : 4$$

If we fit a data scaling function to this graphic, we get $y = x^2$, i.e. the graphic has nonlinear (quadratic, in this case) scaling. This is a strong indicator of distortion. So, if we learn a linear data scaling function from the graphical representation that *doesn't fit* with the corresponding labels, the graphic is necessarily distorted.

To perform this repair step in practice, it turns out to be easier to calculate the area of each bar instead of its height. Area is a reliable proxy only if the width of each bar is constant, as was the case for test data in this project, but future implementations should use height data instead. This module used k-means clustering to quickly assign a label to each pixel based on its color and its adjacency to other pixels. This was necessary because the prior run of k-means didn't necessarily assign a unique color to each bar, so this run needed to take the location of the pixel in the image into account as well.

Challenges

Unnecessary dimensions, on-line learning, other chart types

As a general rule, the dimensionality of the graphic should not exceed the dimensionality of the data it

is representing. This is most frequently an issue with three dimensional bar or pie charts, which are often used for perceived aesthetic benefits rather than actual clarity of representation. To detect this flavor of distortion, we need to learn a representation of the *shape* of each geometric representation, then determine whether it is a proper rectangle. For this portion of the project, a reliable and general method of detecting shape wasn't found. It was possible to subdivide the graph into matrices for each bar, but this approach was computationally expensive and unreliable. Since the scope of this project was limited to bar charts with correct dimensionality, this wasn't an issue, but further implementations should take 3D charts into account.

A serious difficulty for this problem was the need to analyze each data graphic independently. This 'within-subjects' design for the tool and its underlying learning problem was more true-to-life because this tool would need to make a judgment on each data graphic as it came up. Additionally, each bar chart is relatively data-poor, in the sense that each has only a handful of bars/data labels to evaluate. This make the explicit computation of distortion very simple, but increases the complexity of the natural next-step question: is it possible for the tool to learn which graphical patterns are indicative of distortion without explicitly computing the distortion?

This would represent a 'between-subjects' design for the tool, as well as an underlying supervised learning meta-problem. The program would store an abstraction for the critical values in the data graphic, as well as a binary distortion label for each graphic. It would be interesting to see whether this approach increases the speed of the classification. Importantly, this approach doesn't allow repair of detected distortions, but if it is markedly faster than the explicit approach, it could be useful as a pre-processing step to improve the execution time of the tool.

Another source of difficulty came from considering other types of data graphic, especially pie charts. Bar charts were especially tractable for the scaling problem because they're ordered from left to right, which made comparisons with the labels straightforward. With pie charts, however, the polygons are arranged in a circle, so assigning labels to polygons becomes a nontrivial task. Other kinds of data graphic also present a challenge, especially when the organizing metaphor for the graphic varies from 'area-representing-number'. For example, how would one computationally detect distortion in line graphs?

Conclusion

Machine learning and interaction design

As it turned out, the design of the tool relied exclusively on k-means as the learning algorithm of choice. After trying PCA for the color module, it turned out that k-means was faster and resulted in fewer errors. This may have been a result of the data-impoverished nature of this particular problem: the limitation to a discretized, finite color subspace, as well as the relatively small size of each graphic, meant that running any algorithm was close in complexity to doing the same task by "brute force" (i.e. simply going through the image pixel by pixel). Additionally, during the design process, it became clear that the tool may be limiting to users with more skill at generating graphics. This highlighted a need for balance between skilled users, who would be able to design data graphics more flexibly while still maintaining good cognitive design principles (similar to how skilled writers can play with grammar), whereas beginners would need more scaffolding to create good graphics. The overarching question of 'what's the point' still remains: how can this tool best encourage behavior change in users? As a philosophical aside, the process this tool goes through to learn about distorted graphics closely mirrors how humans might learn to perform the same task, including the transfer step previously described. This project shows that simple learning algorithms are best for this kind of basic graphical distortion test, but how can the context surrounding that finding be best mobilized to support human learners?