Visualize Your Data: Generating Datasets with Dissimilar Custom Shapes and Similar Statistical Properties

Algorithm Engineering 2022 Project Paper

Stephan Olbrich
Friedrich Schiller University Jena
Germany
stephan.olbrich@uni-jena.de

Mohamad Wael Kheshfeh
Friedrich Schiller University Jena
Germany
mohamad.wael.kheshfeh@uni-jena.de

ABSTRACT

The paper named "Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing" published by Justin Matejka and George Fitzmaurice [8] introduced a method to generate dissimilar datasets with similar statistical properties. Motivated by this we improved the existing method and implementation. The generated datasets have the goal to illustrate the importance of visualizing data and not reducing it just to statistical properties.

KEYWORDS

simulated annealing, distance matrix, statistics, data visualization

1 INTRODUCTION

The field of statistics is probably one of the most used tools to quantify and qualify data in modern sciences. This ranges from political and social science (e.g. observing and analyzing the development of society) to computer and natural science (e.g. machine learning). Due to the great importance of statistics it is even more important to have a perfect understanding of it. Some of the most well known statistical parameters are mean values, variance and correlation between datasets. Unfortunately a lot of people keep forgetting that those statistical quantities are additional tools when observing data and depending on the application field their importance differs. The statistical values of a dataset are by no means an identification of it. This can be made clear by visualizing the data and seeing that multiple datasets with the same properties are very different. In this project we improved the performance of an existing idea to modify datasets while maintaining certain statistical data.

1.1 Background

For the sake of completeness we repeat the definitions of the statistical quantities used in the project. Given two vectors $x, y \in \mathbb{R}^n$ with $n \in \mathbb{N}$ the mean value and the variance of x are defined as the following sums [9]:

$$\bar{x} := \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \sigma_x := \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

analogously for y. Further, Pearson's correlation between x and y is defined as follows [9]:

$$r_{xy} := \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

.

One further point that has to be mentioned is that through the program the method of simulated annealing [11] is used. The idea is to allow a certain errors in the beginning of an iteration and reduce the amount of errors while the amount of iterations increase.

1.2 Related Work

As mentioned in the abstract this project was motivated and is based on the work of Justin Matejka and George Fitzmaurice. Matejka and Fitzmaurice developed a Python program that manipulates two-dimensional datasets to a preferred target shape while obtaining the statistical values (mean, variance, Pearsons correlation) of the initial dataset. Those target shapes are predefined and hard-coded so that unfortunately there is no possibility for the user to use a custom target shape, unless they implement it by themselves. The program runs through a custom amount of iterations where one point of the dataset is manipulated in each iteration. The manipulation is carried out so that the dataset is directed to the target shape. The optimization strategy that is used is simulated annealing where in the first iterations a certain error is allowed, i.e. the change can lead away from the shape. The visualization of their results is done by generating both .csv and .png files. To present the change of points over time there is the possibility of generating a custom amount of frames during the iteration.

Since the work of Matejka and Fitzmaurice builds a base for our project, further related work is described in it. For detailed information and additional related work we highly recommend to look into the listing of their work. We will give a briefly introduction of them here.
It seems that one of the first approaches to generate different datasets with identical statistics was introduced in 1973 by Anscombe [1]. Anscombe introduced an example of four two-dimensional datasets each with obviously distinct shapes but all sharing the same mean value, standard deviation and correlation. In 2007 Chatterjee and Firat [2] introduced a genetic algorithm approach to generate different datasets with same statistics. Those datasets however did not have any particular shape. The goal was to achieve graphical differences between the sets.

One work worth mentioning is the datasauRus package for R [7]. It is also motivated by [8] and contains the ”awesome Datasaurus Dozen datasets”. This work allows a visualization of the results generated by Matejka and Fitzmaurice in R.

Finally, we want to also mention the work ”Same Stats, Different Graphs: Exploring the Space of Graphs in Terms of Graph Properties” [5]. This work is less about the methods of manipulating the data. Instead, it is more about a similar problem in the context of graph mining. The authors focus on graph properties e.g. amount of vertices, edges or triangles. Among other things they generate multiple random graphs to examine the distribution of graph properties. They also introduce a method of creating multiple different graphs with similar graph properties and statistics.

1.3 Our Contributions

As mentioned before the idea of our project was motivated by an already existing one. The existing program of Matejka and Fitzmaurice was coded in Python. Our program was developed in C++ code, which automatically increased the performance significantly. In terms of performance optimization we took one further step by discretizing the initial datasets. This allows us to do all complex calculations of calculating the distances to the target shape once in the beginning. After that the following iterations aren't that time consuming. By doing this discretization we also improved one major point regarding the target shapes. In the previous work those shapes where hard coded and therefore not modifiable. With our program the user can specify a custom target. It has to be said that this discretiazation can lead to small errors in the statistical values from the original dataset. A further addition to the existing program is the user interface through the command line. After calculating the the distances the user can modify the permutation parameters such as amount of iterations or the error allowed by the simulated annealing. This is being done without recalculating the distance matrix.

1.4 Outline

The structure of the paper is the following: In section 2 the method of our implementation is explained. This is being done rather abstract to avoid presenting bigger chunks of code. In section 3 we present two examples to illustrate our results. In section 4 a comparison is made with the already existing implementation in terms of visual and time results. Also a "bad" example is given to illustrate certain risks. This is then the transition to our conclusions and to points that could be improved.

2 THE ALGORITHM

The basic approach to generate a dataset different from the original but with the same properties is to modify single points of it step by step. The reason for that is that in general it is difficult to generate a dataset with particular statistic properties from scratch. It is way more easy to modify one existing set step by step and allowing only small changes in the statistics [8]. Therefore the program consists of a loop that runs a given amount of iterations. In each iteration one point is modified. After the modification the statistical values of the new set are checked. If the error is small enough the modification is accepted and taken into the new set.
To achieve in the end the target shape the modification in each step has to be biased to move the point towards the target shape. If no error is allowed by the simulated annealing a while-loop tries multiple random modifications until the concerned point satisfies one of the following conditions: the distance from the target shape is smaller than the previous distance or smaller than a given tolerance distance. In Figure 1 we can see a representation of possible movements during one iteration. According to the just explained in case of the left subplot the while-loop either accepts the movement (simulated annealing or distance tolerance is not exceeded) or tries another movement. In the case of the right subplot the movement is accepted.

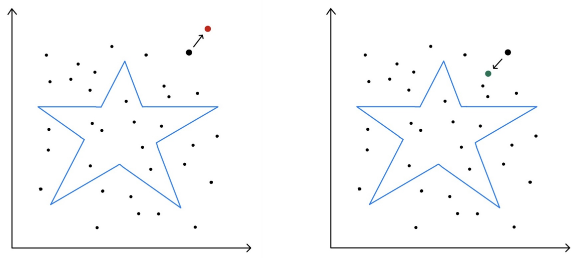

Figure 1: Representation of one iteration. A single point is manipulated and moved either away from the target shape (left) or towards it (right).

2.1 Discretization

During each iteration the distance of the point that is to be moved has to be calculated. The complexity of this calculation can vary depending on the target shape, e.g. the distance from a circle is calculated simply with the help of the radius while the distance from a polygon has to take into account all the edges and is therefore more time-consuming. To bypass this problem we discretized the initial dataset into a grid of the dimension 1000×1000. We generate an integer grid of those sizes and map the dataset on it by stretching, moving and rounding the points of it. Since we then have only finite possible places for the points to be, we can calculate the distances of each of those points to the target shape before starting the actual iterations and store it in a distance matrix. This is being done in parallel with OpenMP [10]. Because of the monotony of the square root function we only calculate the squared distances to avoid loosing performance to an expensive square root calculation.
The target shape is defined in a custom .txt file from the user. The file must contain 1000×1000 binary values with the ones representing the target shape.

3 EXAMPLE RESULTS

In the following we want to present some results provided by our program. All the used target shapes were drawn in the Paint program, then extracted as 1000×1000 bitmap files and later with the help of an image to binary converter tool [4] saved as binary .txt files. This is a simple procedure for any user of our program to use customized shapes. For the starting datasets that are to be manipulated we used the same that were used in [8].

3.1 Example 1: Angled Blob to Thunder

In this first example we start with a dataset named "angled blob" which consists of random points with a certain orientation, as seen in the upper left plot of figure 2, and try to modify it into the thunder shape. We want to keep constant the mean values and variances of x and y as well their Pearson's correlation. The initial values (after discretization and up to two decimal points) are $\bar{x} = 54.26$, $\bar{y} = 47.83$, $\sigma_x = 16.75$, $\sigma_y = 26.92$, $r_{x,y} = 0.82$. The change of the dataset after different amount of iterations is shown in figure 2. To get those

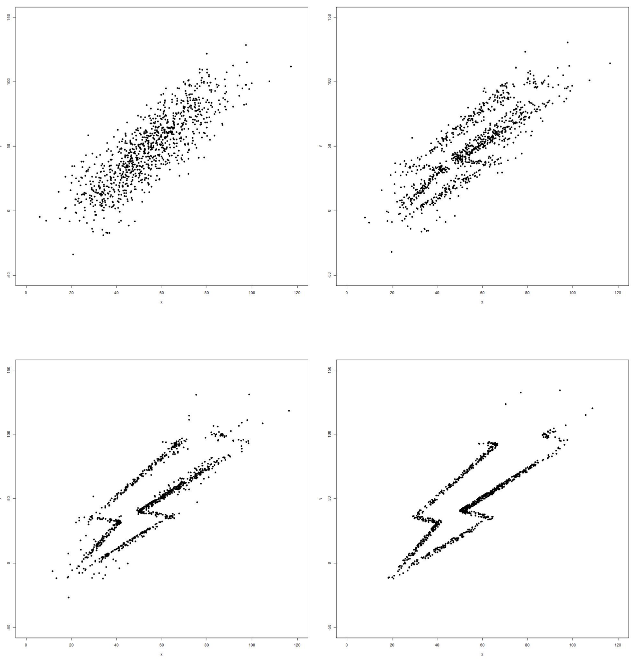

Figure 2: Progress of the modification from the "angled blob" dataset towards the thunder shape while remaining same statistical properties. Change after 0 (upper left), 20000 (upper right), 50000 (lower left) and 300000 (lower right) iterations.

results we calculated the distance matrix once and then did multiple simulations with different amount of iterations. The calculation of the distance matrix took on a laptop computer 2.96 seconds. For the final result after 300000 iterations the computer took 2.13 seconds. The statistical values of all datasets were identical up to the second decimal place.

3.2 Example 2: Slanted Less to Heart

In this example the original dataset has less structure than before. We are considering the dataset named "slanted less" and try to modify it into the shape of an heart. The statistical values in this example are $\bar{x} = 54.01$, $\bar{y} = 48.09$, $\sigma_x = 14.48$, $\sigma_y = 24.72$, $r_{x,y} = 0.32$. The changes after the same amount of iterations as in the previous example are shown in figure 3. The calculations for the distances matrix took 1.46 seconds and the calculations to get the final result after 300000 iterations took 1.16 seconds.

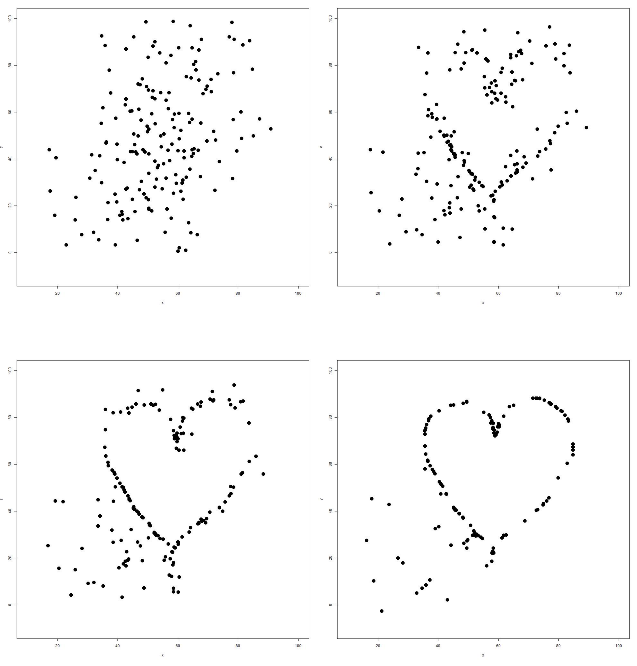

Figure 3: Progress of the modification from the "slanted less" dataset towards the heart shape while remaining same statistical properties. Change after 0 (upper left), 20000 (upper right), 50000 (lower left) and 300000 (lower right) iterations.

4 EXPERIMENTS

We want to try and compare our implementation with the previous one of Matejka and Fitzmaurice. For this we will compare the visual results after the same amount of iterations with the same dataset. Also, we want to compare the running times between the to implementations. We are aware that the comparison between a Python and a C++ implementation can be possibly unfair since C is closer to a low level programming language than Python. Nevertheless the comparison can be interesting from a user's point of view which is only interested in the visual results. Furthermore we present a "bad" example where the target shape does not fit the initial dataset and it's statistical properties.

4.1 Visual and Time Comparison

Starting from the well known dataset "Datasaurus" [8] we want to generate a dataset with the shape of a circle. In figure 4 the results are shown. It has to be additionally mentioned that for the modification with the C++ implementation we set the stretching factors during the discretization to be equal. This was done since otherwise the resulting circle has an oval shape. It is apparent that both results are good and the target shape was achieved. However table 1 clearly states the performance gain we obtained by our implementation. In the Python program we disabled the "frames" option so that only the last dataset is saved. This was done to not loose performance by saving the intermediate values. Also, we choose the circle shape because it was the less time consuming

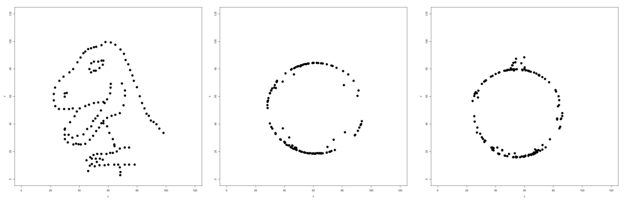
Figure 4: Modification of the "Datasaurus" dataset (left) to a circle. Result with C++ (middle) and Python (right) after 100000 iterations.

shape for the Python implementation. The calculation time for the C++ program is composed by the distance matrix calculation (2.49 seconds) and the actual iterations (0.28 seconds).

Table 1: Time comparison between Python and C++ implementation.

Language	Iterations	Running time (MM:SS)
Python	100000	04 : 48
C++	100000	~00 : 03

4.2 Unsuitable Shapes

In this experiment we want to make aware of the risk of choosing shapes that do not fit the original dataset. Therefore we start again with the "angled blob" dataset (figure 2 upper left) and try to convert it into the custom shape of a bow tie (figure 5 left). The target shape in the result is poorly recognizable compared to the previous examples. The reason for that is that the points of the "angled blob" aren't scattered enough. This could also happen with the circle shape. Especially such faulty results would occur when the initial dataset is not scattered enough but the target shape requires that.

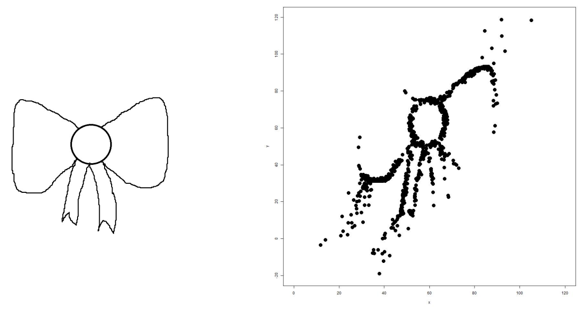
Figure 5: Attempt to modify the "angled blob" dataset into the shape of a bow tie (left). Result after 300000 iterations (right).

5 CONCLUSIONS AND FUTURE WORK

Motivated by an already existing program to generate visual dissimilar datasets with similar statistical properties we created a faster implementation that gives the user more flexibility. With our implementation it is now possible to generate custom shapes within seconds. However, during the experiments and the implementation a few points came up that could be improved. We came up with three major points that would improve the performance: First it should be useful to replace all the vectors in our program with arrays. This of course would make the memory handling riskier but it should give a small performance gain. Next, we could improve the calculation of the statistical values. Since they are being calculated in every single iteration it would be useful to reduce their calculation time. An approach would be to parallelize the calculations with the help of intrinsics [6]. One further major point that could improve the performance is the calculation of the distance matrix. At this moment the calculation takes longer the more points the shape has because every point in the matrix calculates the minimum distance to all the points of the shape. There are multiple ways to calculate such a matrix, as presented in [3]. A more efficient algorithm could be better especially for "bigger" shapes. In addition to that it would be even better to have the possibility to store for each point the direction in which the closest point of the target shape is. Therefore the biasing of the points towards the shape would happen much faster.

REFERENCES

[1] F. J. Anscombe. 1973. Graphs in Statistical Analysis, The American Statistician. 27, 1 (Feb 1973), 17–21.

[2] Sangit Chatterjee and Aykut Firat. 2007. Generating Data with Identical Statistics but Dissimilar Graphics: A Follow up to the Anscombe Dataset, The American Statistician. 61, 3 (Aug 2007), 248–254.

[3] Dabbling Badger LLC. 2020. Implementing Euclidean Distance Matrix Calculations From Scratch In Python. https://www.dabblingbadger.com/blog/2020/2/27/implementing-euclidean-distance-matrix-calculations-from-scratch-in-python [Online; accessed 28 Feb 2020].

[4] dCode. 2022. Image to Binary Converter — dCode. https://www.dcode.fr/binary-image [Online; accessed 28 Feb 2020].

[5] Yafeng Lu Vahan Huroyan Ross Maciejewski Hang Chen, Utkarsh Soni and Stephen Kobourov. 2013. Same Stats, Different Graphs: Exploring the Space of Graphs in Terms of Graph Properties. (Mar 2013). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8863985

[6] Intel Corporation. 2021. Intel Intrinsics Guide — Intel. https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html [Online; accessed 28 Feb 2020].

[7] lockedata. 2022. datasaurus:R — GitHub repository. https://github.com/jumpingrivers/datasauRus [Online; accessed 28 Feb 2020].

[8] Justin Matejka and George Fitzmaurice. 2017. Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. (May 2017). http://dx.doi.org/10.1145/3025453.3025912

[9] Studiflix. 2022. Deskriptive Statistik. https://studyflix.de/statistik/thema/deskriptive-statistik-9, last accessed on 28 Feb 2022.

[10] Wikipedia contributors. 2021. OpenMP — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/OpenMP [Online; accessed 28 Feb 2020].

[11] Wikipedia contributors. 2021. Simulated annealing — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Simulated_annealing [Online; accessed 28 Feb 2020].