**CARTOGRAMS IN PYTHON**

**BY LUKE HALES**

[Draft]

Python is a powerful tool in which many data processing operations can be undertaken, including on geospatial data. Despite this, there is no standardised way of creating cartograms using the platform. While this can theoretically be accomplished with the combination of geospatial and polygonal Python libraries such as Shapely, GeoPandas and Cartopy, this can become complex to the average user. This paper takes into consideration the process and algorithms of creating cartograms, the practical applications this library could provide, and the compatibility of relevant Python libraries in order to reach a conclusion of how best to design a viable cartogram library. There are a multitude of ways in which cartograms may be drawn, each of which has its own unique set of mathematics that comes with it. Therefore, it is imperative that appropriate methods are to be used in this library.

**1. Introduction**

Python is a straightforward programming language for beginners, making it easily accessible for the common user. Cartograms are used and studied by a plethora of different experts including geographers, cartographers, economists, social scientists, geometers and information visualisation researchers (Nusrat and Kobourov, 2016), therefore many of these users may not be able to understand a highly complex library. The library will have to be built with simplicity and ease in mind for the end users and thorough documentation must also be provided in order to aid those without intrinsic technical knowledge.

**2. Types of Cartograms**

A cartogram can take many different forms as there is no formal definition of how they should be presented (Nusrat and Kobourov, 2016). This means that several types of cartograms can be included into the library, with the user being able to select one based on their needs.

**2.1 Diffusion Cartograms**

The primary cartogram type to be included should be the ‘Diffusion Cartogram’ (Nusrat and Kobourov, 2016; Gastner and Newman, 2004). Gastner and Newman (2004) proposed an algorithm which scales regions of a map based on population. [Continue here once understand algorithm].

**2.2 Dorling Cartograms**

Another viable cartogram that can be represented in the Python library is the ‘Dorling Cartogram’ (Nusrat and Kobourov, 2016; Dorling, 1996). Dorling (1996) produces pseudo-code of a computational algorithm to apply to a map to create a ‘Dorling Cartogram’. This would make implementation of this style easier, however this does not fully eliminate all difficulties. The main challenge of implementing this would be the preservation of the map’s shape, as the map obviously must be recognisable to the end user. An algorithm must be created that keeps that map’s shape intact by positioning each circle next to each other and not allowing said circles to overlap each other.

**BIBLIOGRAPHY**

Nusrat, S. and Kobourov, S. (2016). The State of the Art in Cartograms. *Computer Graphics Forum*, 35(3), pp.619–642. doi:https://doi.org/10.1111/cgf.12932.

Gastner, M.T. and Newman, M.E.J. (2004). From The Cover: Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences*, 101(20), pp.7499–7504. doi:https://doi.org/10.1073/pnas.0400280101.

Dorling, D (1996). Concepts and Techniques in Modern Geography Area Cartograms: Their Use and Creation. Available at: https://www.dannydorling.org/wp-content/files/dannydorling\_publication\_id1448.pdf.

‌

‌

‌