**BSc Project Log Book**

**Student Name: Luke Hales**

**Project Name: Cartograms in Python**

**Internal Supervisor: Dr Rudy Arthur**

**External Supervisor: (if any)**

**Log of Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Key activities** | **Outcomes** | **Comments/Action plans** |
| 12/12 | Started designing of cartogram library structure | Library folder structure has been created. A main folder which will hold the object and a testing environment as well as a “pytest” folder has been created. | To create the object for the user to interact with. |
| 14/12 | - Attempted to create an object for the user to upload files to  - Created functions to ensure the parameters passed through will relay the correct information. This is tested using “pytest”. | - User can successfully upload GeoDataFrames and DataFrames. These parameters are verified to ensure they are of the correct type, or else the program will terminate to avoid any potential errors.  - The functions output the correct information about the respected data. This is verified using “pytest” | The data provided will need to be checked that there is a column in both sets of data that links the data, otherwise the data is unusable |
| 15/12 | Started to design a methodology in which the program decides whether the DataFrame and the GeoDataFrame are related in anyway | The creates two lists, one list holds the columns of the DataFrame and one list holds the columns of the GeoDataFrame. The programs then loops through one of the lists to check if the column name is present in the other list. This column is then used as the column that merges the DataFrame and GeoDataFrame into one, unless there are no values in the DataFrame then the program terminates. | This methodology does come with some flaws, as the loop terminates once a column is found, and this could be a column that has no values in common. Furthermore, there could be a column later on that houses the same values as another but goes unchecked. This must be updated later on to take into account these issues. However, this methodology is suitable for the time being due to the nature of the datasets being used. |
| 24/12 | Began to create a way of plotting a heatmap. A heatmap is a simple way of ensuring that the DataFrame and GeoDataFrame match and verifying the merging of the data has been done correctly. | The initialisation of the object has been changed; there is now an extra parameter to be passed through which is the name of the column that holds the statistics to be plotted. The heatmap function has been successfully created, however there are some issues. The merging process does not consider any missing data in the DataFrame, meaning that these show up blank on the heatmap. This must be addressed in some capacity as the cartogram will not be distorted correctly. | The average of the statistic column could be taken and put into the merged DataFrame for any missing values. This would require a recreation of the merging methodology. This is because the “merge” function in Pandas removes a column entry if there is no value in it and this cannot be easily rectified. This will mean the data will have to be combed through first, with data being inserted when necessary, before a merge can take place. |
| 27/12 | Attempted to redesign the merging methodology | Methodology has been partly remade. The function now goes through each value in the merging column in the DataFrame, and if a value present in the GeoDataFrame is not present in the DataFrame then this is added along with the average statistic of the statistic column passed through by the user. | This methodology still only uses the first column in common and does not handle data that may be present in the DataFrame and not the GeoDataFrame. This means that another rework of the function is in order. |
| 29/12 | - Attempted a fix on the current issues that is being faced in the merging function.  - Created functions that allow a user to choose the type of cartogram they wish to plot as well as plot it. | - Merge function now iterates through all columns, creating a list of all potential merge columns. If there are none, the program terminates. The function will then iterate through the list of columns and assess whether they are a suitable candidate for the merge column. This is done by copying the DataFrame, and dropping any values in the current column iteration of the copied DataFrame that isn’t present in the GeoDataFrame. If after all values have been checked and there are none left in the column, the process is repeated for the next column. If there are no viable columns, the program terminates. The function will then add any values not in the GeoDataFrame into it before merging the data together.  - The first function created allows the user to choose their preferred cartogram style. This required creating more class attributes. The other function is called when the user wishes to plot their cartogram. This currently just outputs a generic print statement. | - There may still be one potential shortcoming with the merge function, as even if there is only one value in the column, the column is still chosen. There must be a decision made as to the minimum number of values in a column for it to be deemed viable.  - A test algorithm should now be made to better understand border manipulation. This can be done more easily due to the functions created.  - A way of modifying the data uploaded could also be created, as this would allow for data in two different DataFrames that relate to the same map to be created without the need for another cartogram object. |
| 02/01 | Attempted to create an algorithm that manipulates a GeoDataFrame and outputs the result | Basic manipulation of points was attempted. The idea of the algorithm is to apply randomness to each point in the polygon to distort it. The algorithm failed to produce any meaningful results. This is because the algorithm does not take into account MultiPolygons. | An extra function could be created to deal with any MultiPolygons within the GeoDataFrame. Additional research will be required into the fundamentals and intricacies of MultiPolygons. |
| 05/01 | Rewrite the previously created algorithm to take into account MultiPolygons after research has been undertaken. | Created a new function that applies the original based on whether the geometry type is a Polygon or MultiPolygon. If the type is a MultiPolygon, the algorithm is applied to each individual Polygon. | This may need to be applied in the Diffusion Algorithm and Dorling Algorithm to deal with any MultiPolygons. |
| 09/01 | Began to create a basic Cython algorithm that is similar to the previous algorithm created to distort maps. This would optimise the process and create a basic understanding of Cython. After completion, the algorithm for the diffusion-based cartogram can then be built. | Encountered many issues when attempting to install Cython and subsequently use Cython. Cython failed to create a file required to run it. | Research will have to be done as to the cause of this issue. It seems to be a path issue despite all paths being correct and verified, so it is unclear as to why this issue exists on the surface. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |