|  |  |  |  |
| --- | --- | --- | --- |
| Text  Description automatically generated | | logo_black.gif | |
|  | **201587763** | Logo  Description automatically generated | 1999 |
| Logo  Description automatically generated with medium confidence | GEOG5003: Programming |  |  |
| A picture containing text  Description automatically generated | Assignment #2 (Independent project) |
| Background pattern  Description automatically generated with low confidence | Andy Turner |

|  |
| --- |
|  |
| 1. |
| 2. |
| 3. |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

**INDEPENDENT PROJECT: SITE LOCATION PROBLEM**

# 1. Introduction

Geographic Information Systems (GIS) provides useful tools for land spatial analysis, overlay multi-layer data (Abou-Najem *et al.*, 2019), and make a multi-criteria decision-making analysis (Badi *et al.*, 2021) or also called a Multi-criteria evaluation (MCE). In GIS, MCE is used in land allocation based on a variety of characteristics of the selected study area (Eastman, 1999).

This is a complex process involving qualitative and quantitative criteria (Prieto-Amparán *et al.*, 2021). In addition to the combination of GIS and MCE, Python-based programming has been used in determining the suitable location for wind mills (Hossain, 2021), a fire station (Kazemi *et al.*, 2013), tower cranes (Irizarry and Karan, 2012), and others. Similarly, this approach is used for a site location problem of a factory in this report.

# 2. Intention of the software

The aim of the developed software is to provide an application in Python language to locate a factory that makes rock aggregate somewhere in the UK. The application should combine maps of the United Kingdom (UK) that represent important criteria to the factory company. Each map has a shading intensity representing the best areas based on different factors, in this case Geology, Transport, and Population factors.

# 3. Instructions to run the code

* To run the code, you should open the Spyder application, browse to the python file named “Site.py” and open it. After that, click the “run” button to start the application.
* A new window called “Suitability Analysis” appears showing all three factor maps used in the analysis: “Geology”, “Transport”, and “Population”. This window also displays three main menus to use: “Weights”, “Model”, and “Suitability”.
* The “Weights” menu provides the option “Assign weights” that gives the user the chance to assign a specific weight for each map factor. After clicking the button, a new window called “Weighting” appears where the user can assign a weight from 0 to 10 using sliders. After that, the user clicks on the “Assign” button and then the “OK” button to close the window.
* Now, the user clicks on the “Model” menu, then on “Show suitability”, and a new window appears with the suitability map having the same scale of the input factors, from 0 (less suitable) to 255 (most suitable) in a grey scale colour. Then, the user clicks on “Print outcome” button to create a “csv” file from the output called “Suitable areas” on the same location of the input files.
* Finally, as additional analysis, the user can click on the “Suitability” menu and click on the top 10%, 20%, and 30% suitable areas of the outcome to have a more specific visualisation. Each option shows a suitable map with the respective percentage. Also, a last option “Back to 100%” also shows the overall outcome again.

# 4. Development issues and solutions

One issue found was the combination of the three map factors. Each map factor must be multiplied by an integer (the weight assigned by the user) and then all be summed. However, each factor is expressed as a “list” file and the calculations are more complex in this way. To facilitate the calculation, each factor was converted into an “array” file.

Having made the calculations, another issue was the scale conversion. All inputs have values from 0 to 255 but the output might have another scale due to the calculations. To solve this problem, the scale of the output was re-calculated using the output minimum and maximum values. It means the lowest value set to 0 and the highest values set to 255.

Finally, the figures need better formatting using the matplotlib tools. Given the basics from the practicals, it was necessary to explore further elements such as title, colour ramp, and labels. To utilise these tools, the documentation of matplotlib in Python (Matplotlib, 2021) was reviewed and applied to the layout of the three map factors and to the output map.

# 5. General sources used

To produce the suitability map, several sources were used such as Python modules, packages, and libraries:

* The “csv” module implements classes to read and write tabular data in CSV (comma-separated values) format (Van Rossum and Drake, 2009). This module was used to convert each map factor into a “list” file to be analysed in the code. It was also used to convert back the final output into a “csv” file.
* The “numpy” library gives routines for fast operations including mathematical, logical, sorting, selecting, basic linear algebra, basic statistics, and others (Harris *et al.*, 2020). This library was used to make calculations among the inputs by converting “list” files into “array” files and then converting back the output into a “list” file.
* The “tkinter” package is a thin object-oriented layer which provides a platform independent windowing toolkit (Van Rossum and Drake, 2009). This package was used to generate a Graphical User Interface for our Python application so that the user can interact with the application.
* The “matpltlib” library is a comprehensive library for creating static, animated, and interactive visualisations in Python (Matplotlib, 2021). This library was used to create the figures of the inputs and the output. Plus, several tools from this library were used to enhance figure presentation of our application.

# 6. Data processing design

Based on the factory company requirements, the application is aimed to process several factors to locate the rock aggregate factory. These factors are three UK maps representing good geology areas, the importance of transport, and population workforce. Each map has raster values from 0 to 255, displaying from the worst to the best suitable areas.

First, the input data is converted from “csv” files to “list” files, and then to “array” files. Using the “csv” module, the raster data were converted into lists; and using the “numpy” library, the lists were converted into arrays.

Second, specific weights are given by the user from 0 to 10 and each weight is multiplied by each factor and then all are summed. Next, the output array file is converted back to list, and then re-scaled to the input’s scale (from 0 to 255) by the following expression: re-scale value = value \* 255 / maximum value.

Third, there is an additional option to select the top 10%, 20%, and 30% of the total suitable areas of the output. To obtain those, values greater than 230, 205, and 180 (given the 0-255 scale) were selected respectively. Then, new outputs are generated showing only the chosen percentages.

Finally, the output file is written as a “csv” file using the “csv” module. In this way, the user can produce and use later this file in any GIS-related software to present the results or continue with further analysis as required. The overall data process design is summarised in Figure 1.

Diagram

Description automatically generated

Figure 1. Overall data processing

# 7. Software development process

The application was developed using Spyder that is an open-source cross-platform integrated development environment (IDE) for scientific programming (Spyder, 2021). To build the code, Python 3.8.10 language was used to produce some codes organised as follow:

Lines 8 - 14

* To start, several modules, packages, and libraries were imported including “csv”, “numpy”, “tkinter”, and “matplotlib”. Each one has specific functions that were used in the development of the application.

Lines 17 - 53

* To upload the input data, the “open with” statement was used to extract values from “csv” files and then a “for loop” was used to create “list” files for each input.

Lines 56 – 76

* To show the map inputs, it was a created a figure with three subplots using some properties of the “matplotlib” library, including a title and a colour bar.

Lines 89 – 127

* To assign user’s specific weights, the “tkinter” package was used to create a slider for each input with values from 0 to 10. This code was then placed inside a function called “assign\_values” to be run by the respective command. Then, this code was placed inside another function called “assign\_weights” to be run by the command in the “Weights” menu.

Lines 134 – 173

* To make the calculations, the “numpy” library was used to easily multiplied each weight by each input. The calculations were made using “array” files and then converting the output into a “list” file again by using a for loop. To re-scale the output, the max() function was used to identify the maximum value. Finally, the “matplotlib” library was used to display the output with a 0-255 scale. All code was placed inside a function called “suitability”.

Lines 175 – 179

* To verify the correct output calculation, a specific point was tested. The final value should reflect the sum of the products of each factor and its weight, and then the re-calculated scale.

Lines 182 – 188

* To create the outcome as a “csv” file, the open statement and a for loop were used inside a function called “print\_outcome”

Lines 197 - 248

* To display the top 10%, 20%, and 30% of the suitable areas, three functions called “top\_10\_percent”, “top\_20\_percent”, and “top\_30\_percent” were created respectively. Within each function, a new list was created using a for loop and selecting only the values required using an “if” statement inside the for loop. Then, each outcome is presented as a figure map using the “matplotlib” library.

Lines 79 – 87, 129 – 132, 191 – 195, 251 – 259

* For the main interface, the main window and three menus were created: “Weights”, “Model”, and “Suitability” respectively, using the “tkinter” package.

Console products

* To test the correct values of the weights, the inputs, the scale, and the outputs; the code writes and presents these values on the console window as follow:
* User’s weights for Geology, Transport, and Population
* Maximum value obtained from calculations
* Confirmed maximum rescaled value
* Geology value
* Transport value
* Population value
* Output value example

# 8. Results

Given that the output depends on the user’s assigned weights; here, it is presented an example with all weights assigned to “1” to illustrate our application performance.

When running the application on Spyder, it appears the “Suitability analysis” window displaying the three input maps and their suitable areas for the factory (Figure 2).

Graphical user interface, website

Description automatically generated

Figure 2. Input maps for each factor: Geology, Transport, and Population

When the “Weights” menu is clicked, then the button “Assign weights”, a new window called “Weighting” appears to select the weights from 0 to 10 for each factor using sliders (Figure 3). Once the weights are chosen (to “1” in this example), clicking the button “Assign” assigns the weights and the button “OK” closes this window.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 3. Weighting window to assign weights

When the “Model” menu is clicked, then the button “Show suitability”, a new window appears showing the suitability map based on the selected weights (Figure 4). The map shows suitable areas from most suitable (black) to least suitable (white). If the user wants to create a “csv” file of the output, the button “Print outcome” from the same menu can be clicked.

Graphical user interface

Description automatically generated

Figure 4. Suitability map based on selected weights equals to one.

Additionally, when the “Suitability” menu is clicked, there are four options to display the top 10%, 20%, 30% or the total suitable areas again. A new window appears showing the selected suitability map, for example the top 10% (Figure 5).

Graphical user interface, application

Description automatically generated

Figure 5. Map of the top 10% suitable areas for the factory location.

Finally, the console products to test the code (in this example) are as follow:

* User’s weights for Geology, Transport, and Population = 1 , 1 , 1
* Maximum value obtained from calculations = 558.0
* Confirmed maximum rescaled value = 255.0
* Geology value = 215.0
* Transport value = 14.0
* Population value = 85.0
* Output value example = 143.5

GIS tools can analyse, combine, and map data for spatial planning (Günen, 2021). Therefore, scientific evidence can be generated based on specific criteria. Also, the combination of GIS with multi-criteria evaluation provides a framework for spatial planning problems (Saraswat *et al.*, 2021) besides the site-suitability location of a rock aggregate factory in this report. Further improvement may include more criteria and analysis from current results.

# 9. References

Abou-Najem, S. *et al.* (2019) ‘Land Capability for Agriculture, Hermel District, Lebanon’, *Journal of Maps*, 15(2), pp. 122–130. doi: 10.1080/17445647.2018.1561532.

Badi, I. *et al.* (2021) ‘Optimal site selection for sitting a solar park using a novel GIS- SWA’TEL model: A case study in Libya’, *International Journal of Green Energy*. Taylor & Francis, 18(4), pp. 336–350. doi: 10.1080/15435075.2020.1854264.

Eastman, J. R. (1999) ‘Multi-criteria evaluation and GIS’, *Philosophical Magazine Letters*, 74(3), pp. 189–194. doi: 10.1080/095008396180344.

Günen, M. A. (2021) ‘A comprehensive framework based on GIS-AHP for the installation of solar PV farms in Kahramanmaraş, Turkey’, *Renewable Energy*, pp. 212–225. doi: 10.1016/j.renene.2021.06.078.

Harris, C. R. *et al.* (2020) ‘Array programming with NumPy’, *Nature*. Springer US, 585(7825), pp. 357–362. doi: 10.1038/s41586-020-2649-2.

Hossain, M. R. (2021) ‘Python based location analysis of wind mills in urban environment’, *GeoJournal*. Springer Netherlands, 86(3), pp. 1355–1373. doi: 10.1007/s10708-020-10137-z.

Irizarry, J. and Karan, E. P. (2012) ‘Optimizing location of tower cranes on construction sites through GIS and BIM integration’, *Electronic Journal of Information Technology in Construction*, 17(March), pp. 361–366.

Kazemi, M. *et al.* (2013) ‘Optimization model for fire station location based on GIS and python: A case study in north cyprus’, *Applied Mechanics and Materials*, 330(June 2020), pp. 1059–1064. doi: 10.4028/www.scientific.net/AMM.330.1059.

Matplotlib (2021) *Matplotlib: Visualization with Python*. Available at: https://matplotlib.org (Accessed: 28 April 2020).

Prieto-Amparán, J. A. *et al.* (2021) ‘A regional gis-assisted multi-criteria evaluation of site-suitability for the development of solar farms’, *Land*, 10(2), pp. 1–19. doi: 10.3390/land10020217.

Saraswat, S. K. *et al.* (2021) ‘MCDM and GIS based modelling technique for assessment of solar and wind farm locations in India’, *Renewable Energy*. Elsevier Ltd, 169, pp. 865–884. doi: 10.1016/j.renene.2021.01.056.

Spyder (2021) *Spyder*. Available at: https://www.spyder-ide.org (Accessed: 28 April 2020).

Van Rossum, G. & Drake, F.L., 2009. Python 3 Reference Manual, Scotts Valley, CA: CreateSpace.