EGM722 – programming for GIS and Remote Sensing
Assignment part 1: program instruction manual

GitHub repository for assignment part 2:
https://github.com/TheBigRJM/assignment

1. Introduction

This code has been developed to automate ecological data searches used in the formulation of preliminary ecological appraisals and environmental impact assessments. To be able to determine the impact of landscape changes, a thorough evidence base is required from which to determine impact. The information generally required as part of these searches are:

- Designated nature conservation sites (i.e. sites which are known to support high biodiversity via the habitats and/or species they support)
- Protected, notable or Biodiversity Action Plan (BAP) species. This may be all species or a subset of particular species groups depending on what is being assessed.
- Invasive species which may have a detrimental effect on native species or habitats.

Ecological data searches are systematic and repeatable, therefore an ideal process to automate, the additional modules and packages developed for Python make it possible to carry out these data searches without the reliance on any specific GIS packages, aside from the creation of the input datasets. Using Python to automate this process it will also make it possible to host this application on the web.

In simple terms the program is designed to take spatial inputs from the user (e.g. grid reference and a buffer) allow the user to select from a list of search parameters:

- Protected species
- Great Crested Newts ONLY
- Bats ONLY
- Invasive species
- Nature conservation sites
- Nature conservation sites and protected species

The program processes these inputs and returns all the feature from the selected parameters which are within the buffer area specified by the user, saved as an excel spreadsheet. It also produces a JPEG map of the search area with the features plotted.

2. Setup and installation

The source code for this program can be found on GitHub in the following repository: https://github.com/TheBigRJM/assignment

Instructions for installing the necessary software and setting up the environment from which to run the code on your local system can be found in the README file within the repository.

The main dependencies that the script relies on are in the table below and can also be found in the environment.yml file provided in the GitHub repository.

```
name:
EGM722_assignment
                    channels:
                      - conda-forge
                      - defaults
                    dependencies:
                      - python=3.8.8
                      - geopandas
                      - cartopy=0.18.0
                      - notebook
                      - rasterio
                      - numpy
                      - pandas
                      - matplotlib
                      - shapely
                      - pysimplegui
                      - pip
                      - pip:
                        - bng
                        - matplotlib-scalebar
```

Specific test data has been supplied in the GitHub repository for use with the tool (https://github.com/TheBigRJM/assignment/tree/main/SampleData). If the code has been installed as per the instructions in the instructions in the README file, then the file paths within the code should be able to automatically locate and load the files. If the user has saved the data files in a local directory different to the default when cloning the repository; the code will need to be edited to locate the specific files. Section 4 of the README provides guidance on which lines of code need to be edited to do this.

2.1 Search limitations

For testing purposes, the user must limit their search area inputs to within specific geographic bounds. Section 3 of the README in the GitHub repository specifies the bounds to which searches must be limited.

An example shapefile input has been provided for demonstrate the tools capability of searching from a file: https://github.com/TheBigRJM/assignment/tree/main/SampleData/example%20search%20area, but the user is also free to use their own file input, adhering to the geographic limitations. The Coordinate Reference System (CRS) of any inputs must be set to EPSG:27700 and they must fall within the specified search area otherwise this will return no results or generate a terminal error.

2.2 Running and using the tool

To run the tool, first the user must ensure that the IDE interpreter is working in the correct environment for this project (this should have been setup in step 2 of the README). The tool can then be loaded using the terminal in your IDE. Using the terminal, navigate to your local repository save location and type: “ipython RM_AssignmentScript.py”, this should open a separate python window called 'Data Search Enquiry' (figure 1). The tool has been designed so that all the user interaction is done via a GUI as it the tool would mainly be used by people with little or no experience of programming and is therefore more user friendly.

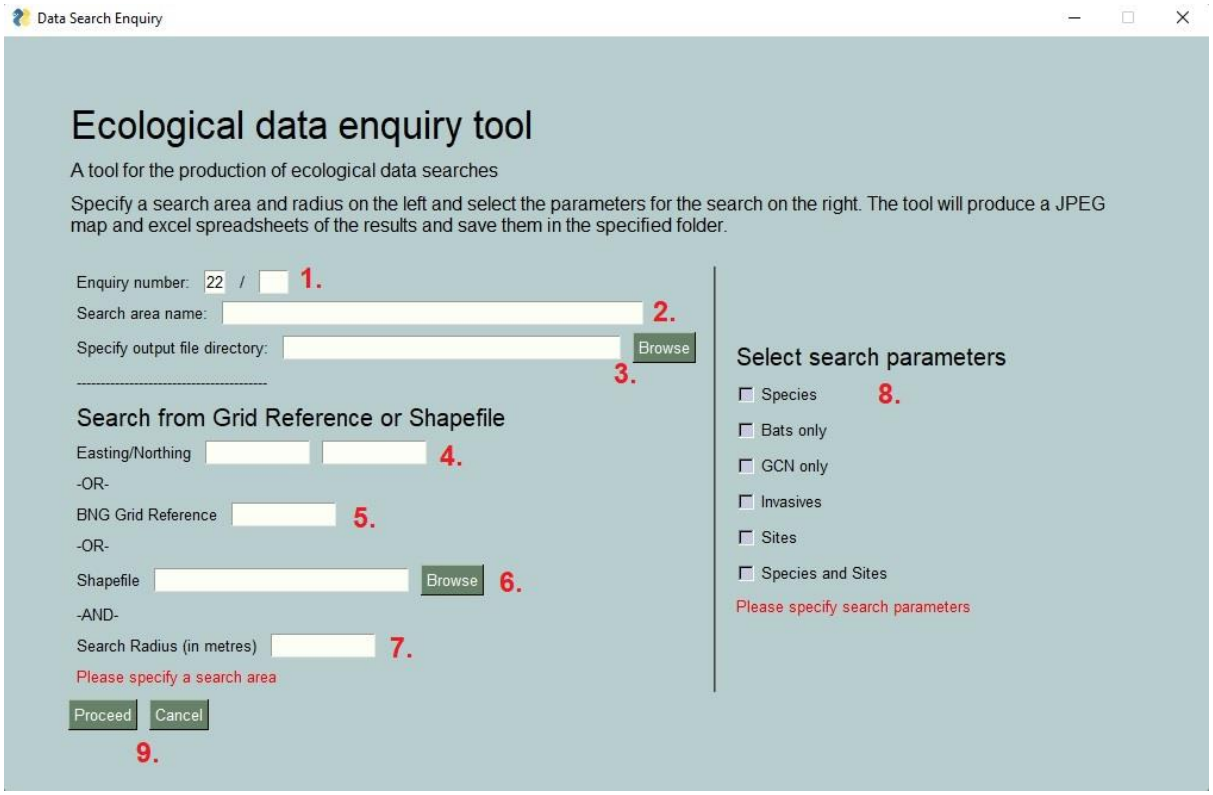

Figure 1 - The graphical user interface which should display if the code has run successfully. Parts of the tool where the user can interact have been annotated.

To use the tool the user must complete certain input fields to be able to click proceed and generate the outputs. The annotations on figure 1 highlight the user input fields. A guide to the requirements for each field is as follows:

1. The number of the enquiry. This is for administration purposes to keep track of enquiries, information entered here will be used to name the saved file outputs.

2. The name of the search area. Generally, this is supplied by a client, but it will also be used to title the map.
3. Choose a save location for the output files.
4. Easting and Northing from which to carry out the search from, this can be input as an integer or floating-point number. This will act as the central point of the search area.
   OR
5. British National Grid reference formatted correctly e.g. SK000000 prefixed by two letters. This will only accept grid references with an even length. This will act as the central point of the search area.
   OR
6. Allows the user to browse for a file (The browse field is currently locked to only search for SHP and TAB files) from which to carry out the data search. The feature(s) in the file will act as the central area of the search. If using this option, the file **MUST have a CRS of EPSG:27700.**
7. A buffer radius in metres, this can be 0 if the user only requires records from their site boundary but cannot be left blank
8. Checkboxes to choose what parameters the user wishes to search. This will accept multiple selections but may cause issues with saving the outputs (particularly the map title for example) if several are selected.
9. Proceed and cancel buttons. Proceed will run the search once the required fields have been completed. Cancel exits the window.

Please only select **ONE OPTION** of either 4, 5 or 6 otherwise this will cause errors.

3. Methods

This methods section runs through the elements of the code in order as they have been written in the program itself (https://github.com/TheBigRJM/assignment/blob/main/RM_AssignmentScript.py) to attempt to simplify to the user what the code is doing throughout the process.

(lines "x-x") refers to the line numbers in the program code where the specific element being described can be found. The **modules** being used at each specific point have been written in bold.

3.1 Module import (lines 1- 16)

Firstly the necessary additional modules are imported in to the script to enable specific steps to work. The key modules required in this program are:

- numpy
- pandas
- rasterio
- geopandas
- cartopy
- matplotlib
- matplotlib-scalebar
- shapely
- PySimpleGUI

- bng
- pathlib
- datetime

3.2 Get current date (lines 21 & 22)

These two lines of code get the current year using the **datetime** module and turn this into a two-digit year e.g., 2022 > 22. This is later populated as a default input when prompting the user for an enquiry number.

3.3 GUI layout and build (lines 24-68)

Firstly, the overall layout of the GUI must be defined and sent to a window first so that processes can be carried out based on the user inputs, selected values, and events. The **PySimpleGUI** module tools were used to produce the GUI.

The GUI layout has been arranged into two columns; column 1 containing all the user inputs, and column 2 which contains checkboxes allowing the user to select their search parameters. Two dialogue messages instructing the user to provide inputs and select parameters are also defined at the bottom of the respective columns. A window title, subtitle, instructions and two buttons (proceed and cancel) are also added to the layout outside of the columns structure.

Most of the elements within the GUI are assigned a key so that their specific values can be found and used during the main part of the program function.

Line 68 calls the defined layout to a window so that the user can interact with the program outside of the interpreter.

3.4 Define functions within GUI (lines 75 – 523)

The function definitions are called after the GUI layout has been created and called so that the functions are working from within the GUI itself.

More information on the values returned can be found in the docstring of each function.

3.4 .1 Raster mosaic function (lines 73 -110)

Returns a mosaic of inputted raster tiles to form the basemap for the map plot.

3.4.2 Area search from point (lines 113 – 154)

This function takes a user inputted easting/northing value and buffer radius, produces a buffer around the specified point, and outputs both the point and buffer as **geopandas** GeoDataframes and **shapely** geometries.

3.4.3 Area search from polygon (lines 157 – 196)

This function takes a user inputted geometry file (e.g. shapefile) and buffer radius, produces a buffer around the specified file, and outputs both the point and buffer as **geopandas** GeoDataframes and **shapely** geometries.

3.4.4 Species search (lines 199 – 233)

This function carries out a protected species search on two separate species files (one with points with a grid reference accuracy of 6 fig or more, and one with an accuracy of 4 fig, i.e. 100m+ precision or 1km precision) by intersecting these datasets with the buffer created from the users' inputs using **geopandas**. The results of the 6 figure search are exported as a geodataframe to plot on the map. The results of the two searches are concatenated together and output as a **pandas** dataframe.

3.4.5 Species style (lines 236 – 320)

This function styles and plots the results of the protected species search on a map axis using **matplotlib.lines**, it then creates label handles matching the styles of the plotted which can be used for a legend.

3.4.6 Bats search (lines 323 – 364)

This function carries out the same search as the species search in 3.4.4, but first filters both the 6 figure and 4 figure grid reference species datasets where the "informal group" column in the geodataframe identifies the record as a bat record. The function also outputs the handle for the bat records.

3.4.7 Great Crested Newt search (lines 367 – 408)

This function carries out the same search as the species search in 3.4.4, but first filters both the 6 figure and 4 figure grid reference species datasets where the "species" column in the geodataframe identifies the record as a Great Crested Newt (GCN) record. The function also outputs the handle for the GCN records.

3.4.8 Invasive species search (lines 411 – 442)

This function carries out the same search as the species search in 3.4.4 but using the invasive species datasets instead of the protected species dataset.

3.4.9 Sites search (lines 445 – 494)

This function carries out a search for nature conservation sites within the user specified buffer by carrying out an intersection of Sites of Biological Importance, Biodiversity Alert Sites with the buffer geometry. The results of the two intersections are outputted as **geopandas** geodataframes. The results of the intersection of the two datasets are then concatenated two form one **pandas** dataframe.

Label plots are calculated for both geodataframe outputs and plotted onto the centroid of each site that falls within the buffer search area.

Finally, the function uses **matplotlib.mpatches** to create handles containing the style information for each of the geodataframes.

3.4.10 Load basemap (lines 497 – 523)

This function uses **rasterio** to read the basemap file and get the bounds of the image, convert the image to a **numpy** array, transpose the array to display correctly on the plot, and define a variable which holds the image display information.

3.5 Define the Coordinate Reference System (CRS) (line 528)

Using **cartopy** to assign EPSG:27700 to a variable to use when plotting geodataframe data.

3.6 Load data layers (lines 531 – 542)

The vector data which is being searched is loaded into the program using **geopandas.** The basemap is also loaded using the function described in 3.4.10.

3.7 GUI event loop (lines 548 – 762)

This is where the main body of the program runs. As the program is working within a **PySimpleGUI**, it begins with an infinite event loop which is ended if the user exits the GUI window (lines 548 – 555). The GUI window will remain open unless the user exits the window or presses cancel.

Lines 569 – 761 of the program execute several IF statements within the main event loop which check the events and values based on the users' interaction with the GUI. These are carried out in order within the loop and produce outputs based on whether the user has fulfilled the necessary criteria to satisfy one or some of the IF statements. As these IF statements are within the infinite loop of the GUI window it is possible to run multiple data search enquiries without closing the GUI window.

One of the first IF statements produce a blank **matplotlib** figure and axis using the CRS defined in 3.5, adds a scalebar, basemap and gridlines ready to plot the results of the search(es).

There are several processes and parameter choices which rely on user inputs, each of which will call a different function and ultimately end up producing a different output. Figure 2 shows this in a simplified flow chart of what is happening when the user interacts with the tool. Note that this does not include any mechanisms for handling errors or calling dialogue message boxes to prompt the user.

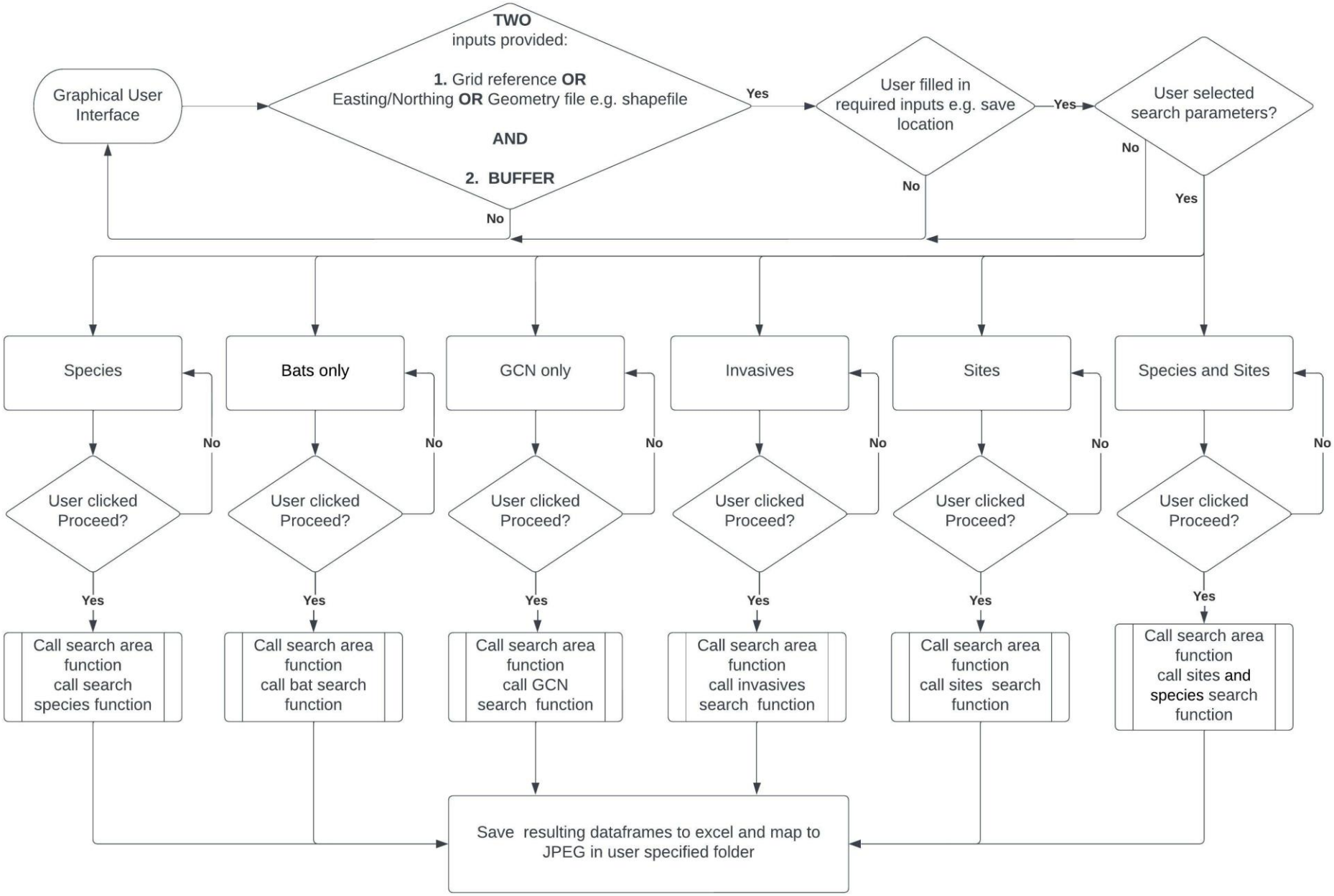

Figure 2 Simplified flow chart showing the GUI event loop based on the users' interactions with the window.

4. Results of running the code

The output of the code will depend on the users' inputs and parameter selection using the GUI. At a minimum the output produced would be an excel spreadsheet containing a list of species records. In most cases the script will return a JPEG map of the results along with one or more excel spreadsheets depending on the specific parameters the user chooses, which will be saved in the user specified folder.

5. Troubleshooting

The tool is coded to handle most user input errors and inform the user via use of popup dialogue boxes; however, some unhandled errors may still occur.

If you have any error or problems running the code raise an issue in the GitHub repository: https://github.com/TheBigRJM/assignment.