Introduction

Background

Our project, "The Planning Project" is part of the beginning steps in conservation and land planning for the Nunavut Planning Commission (NPC). Our client contact is Adrian Gerhartz, a Planner / Geographic Information Systems for the NPC working out of the Iqaluit office. He is an expert in conservation planning and is working to find ways to use the available knowledge and data to identify key areas that need to be protected. Since July 2015 the NPC has been the "gatekeeper" for all project proposals, that is, all activities proposed in Nunavut have to first be submitted and reviewed by them. One of the mandates of the NPC is to "determine whether a project proposal is in conformity with a land use plan." Our project will be to develop a tool that will automate the process of preparing data that will be fed into the conservation planning software, Marxan, that is used as part of this process.

Requirements and Priorities

The requirement will be to provide a script to automate the data preparation for the conservation planning software. This will involve generating a customizable planning unit (hexagonal) grid, loading Conservation Feature (CF) layers, determining the planning unit and CF area overlap based on user input about the project and features of concern, and outputting files in the proper format for use in Marxan.

Purpose

The purpose of this script is to automate the process of preparing data for use in the conservation planning software Marxan. This software displays conservation planning data with a hexagonal grid and requires a csv input that is difficult to create. Manual preparation is possible but may be tedious and prone to errors if significant manual data manipulation is required. As well it may require the use of several softwares to get the data into the required format. Building all of the functionality into a script will reduce manual intervention, human error, and produce clean data while only working with one interface. To accomplish this data preparation the script will create a hexagonal grid, allow the user to input various conservation features, and then find the area of intersection between these two layers. The output of this operation will be a csv file in the exact format that Marxan requires for its input.

User input

- target coordinate reference system that will be used for all functions
- shapefile input to define boundary and position of grid
- option to input hexagon size, grid size and position instead of a shapefile
- input on which planning unit(s) the proponent project will occupy
- selection of planning layers to load
- user query of relevant CF planning features from planning layers

Output

- file ingestible by Marxan
  - csv containing 3 columns (species, pu, amount)
  - species is the conservation feature id
  - pu is the hexagonal cell id
  - amount is the area of overlap between the two features
- planning unit hexagonal grid
- filtered conservation features

Scope

The following lists the overall, high-level features that were developed for this project. They generally correspond with the menu options provided to the user when running the script.

- Create Planning Unit Grid
  - Create Grid from Shape File extents
  - Create Grid from Shape File extents and clip to shape
  - Load existing Grid from File
  - Create Grid from User Input
- Load Conservation Features Files
  - Allow user to load in multiple files of Conservation features
- Filter Conservation Features
  - Allow user to filter the loaded Conservation Features so calculations only include particular features
- View Layers
  - Provide basic plotting to visual confirmation of data
- Calculate Overlap
  - Calculate the area of overlap with each cell in the Planning Unit Grid for each conservation feature
- Save Results
  - Output GeoDataFrames used in the calculation process including the Planning Unit Grid and the filter conservation features
  - Format the data returned from the calculate Overlap step to be in the format required for use with Marxan and save as a .csv

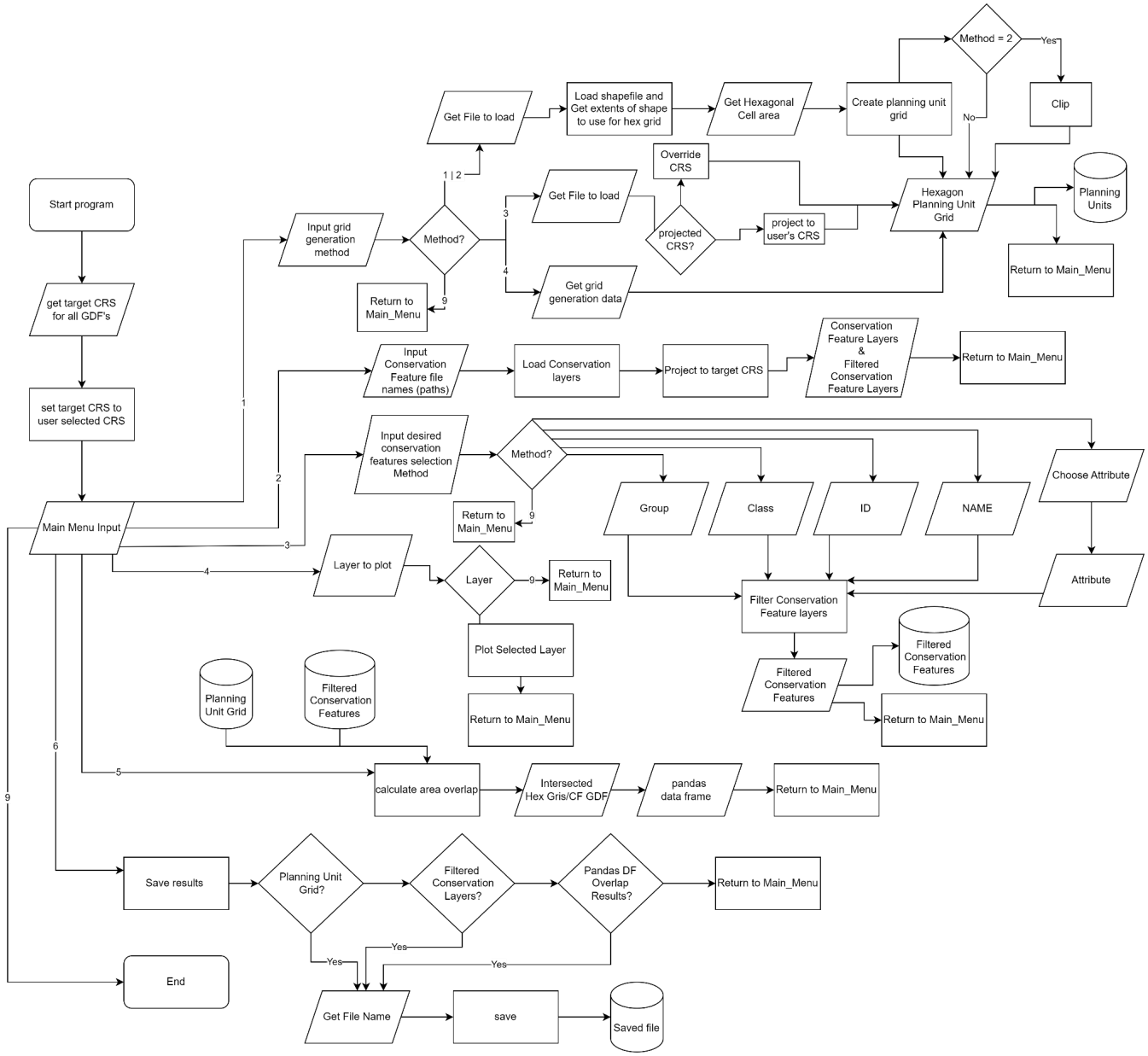
Workflow

- Once the program has begun the user will first be prompted to enter the Coordinate Reference System (CRS) that they will be using for their analysis. Throughout the code this will be used as the target crs to ensure any data that is loaded or created is within the same projections.
- The user will then go to the main menu, here they can chose which of the steps they would like to proceed with first, including creating the planning unit grid, loading

conservation features, filters conservation features, visualizing the current data on a map, intersecting the features, and finally saving the various outputs that had been created.

- When a user selects input grid generation method they will then have to choose one of the three grid generation methods that are available.
- The user can create a grid that is defined by a shapefile of their choice. The entered shapefile will have its bounds used to define the bounds of the grid after it has been reprojected to the target crs. The user will also be prompted to enter the area of each individual hexagonal cell. A unique planning unit id is assigned to each hexagon that is created so they can be identified during the intersection.
- The user is also able to load a grid from a file, this could be done to ensure consistency with past research that has been done, as well as speeding up the loading time of the code if the grid has a fine resolution and a large extent.
- Lastly a grid can be created entirely with manual inputs, with the user entering the central x and y coordinate, the length and width of the grid, and the area of the hexagon cells. The coordinate and x/y coordinates will be entered in the same units as the coordinate reference system, with the area being in those units squared.
- After the grid is created it can then be clipped to the exact dimensions of the shapefile rather than its bounds, this speeds up the processing time as it eliminates cells that will not intersect with any conservation features during the overlay process.
- Once the grid is generated the user will be taken back to the main menu where they can choose to load their conservation features. These features are input using a path provided by the user or by utilizing a pop up window that will allow them to select the file directly. The file is then projected to the target crs and the user then returns back to the main menu.
- If the conservation features that are loaded contain features the user is not interested in examining they are then able to filter it based on the various attributes of the file. This could be fields such as group, id, name, class or another attribute if other files are used.
- If the user selects Layer to Plot from the main menu these layers can then be plotted to allow the user to visualize them.
- The layers will now be ready for the most important step in the process which is the intersection between the two planning unit grid and conservation features. The user can select Calculate Area Overlap, the two layers will be overlaid and intersected with one another. This creates a csv that contains three columns, a species column for the unique ID associated with the conservation features, a pu column that has the ID for each individual hexagon in the planning unit that intersects with a conservation feature, and the area of overlap between the two features.
- Finally in the main menu the user can select Save Results, this will allow them to save the planning unit grid and filtered conservation features so they can save time on potential future work. They will also save the final csv that will be used input into Marxan.

Flowchart



Documentation

Dependencies

Hardware:

- The script was tested on up to date versions of Windows 10 and 11. It will likely work on other platforms such as Mac or Linux but this has not been tested or verified.
- Since the overlay operation can be time intensive the workload is split between the number of logical cores available on the computer. This means you will likely achieve faster computations with more cores even if they have lower clock speeds.

Software:

- Python >= 3.10
  Python is a high-level, interpreted programming language that is used for a wide range of applications. It was used in this project, and the project was tested with 3.10 version.

- Geopandas - geospatial data handling
  Geopandas is a Python library that provides geospatial data handling capabilities.It is built on top of the pandas library, and adds support for spatial data types and spatial operations. It was used in this research to handle geospatial data.

- Shapely - geometry creation
  Shapely is a Python library that provides tools for creating and manipulating geometric shapes, such as points, lines, and polygons. It is built on top of the GEOS library, and provides a high-level interface to the GEOS operations. It was used to create and compute geometry.

- pandas - data manipulation, csv output
  Pandas is a Python library that provides data manipulation and analysis tools. It provides data structures like DataFrames and Series, and includes functions for reading and writing various file formats, such as CSV, Excel, and SQL databases. It was used for producing the output csv.

- matplotlib - visualizations
  Matplotlib is a Python library that provides data visualization tools.It provides functions for creating various types of plots, such as line charts, scatter plots, and bar charts. This library handled all forms of visualizations used in this project

- tkinter - GUI toolkit
  Tkinter is a Python library that provides a GUI toolkit for creating graphical user interfaces.It is built on top of the Tk GUI toolkit, and provides widgets like buttons, labels, and text boxes.

- psutil - process and system utilities
  Psutil is a Python library that provides system and process utilities. It provides functions for getting information about system resources like CPU usage, memory usage, and disk usage, as well as information about running processes. It was used to get the number of logical cores available in order to spread the workload across them appropriately.

Installation/ setup
https://github.com/GEOM4009/planningproj/blob/db1c0951d7ba0e8a13548b6da9c03987b9d10184/README.md

The following steps assume you have an installation of Anaconda (or one of its variants)
- https://www.anaconda.com/products/distribution (full GUI available)
- https://docs.conda.io/en/latest/miniconda.html (lightweight CLI only)
- https://anaconda.org/conda-forge/mamba (for faster environment solving)

Download or clone this repository, navigate to it in an Anaconda Prompt, and type
- conda env create -f planningproj_env.yml

Once this has completed activate the environment with
- conda activate planningproj_env

You can then run the script with
- python planning.py

User Guide
User Guide file in repository
https://github.com/GEOM4009/planningproj/blob/df9260001bb56564250086cf1a139249a68d4849/User%20Guide.md

Troubleshooting / FAQ
Troubleshooting/FAQ file in repository
https://github.com/GEOM4009/planningproj/blob/df9260001bb56564250086cf1a139249a68d4849/Troubleshooting_FAQ.md

Why is the processing time so long?
- There are many factors that can contribute to long processing times with this code
- Creating a grid with a resolution that is very small or covers a large extent can result in long load times
- Intersecting these large grids with the conservation features can also be a time intensive process and may take more than an hour
- Using a multiple overlapping conservation features could also result in longer processing time during overlaps
- As long as you can see the scrolling dots indicating something is happening the process is still working and you should wait for it to complete

I entered the incorrect file or variable input during one of the steps, how do I fix this?
- The step can be repeated by returning to the main menu and doing the same process again
- Some steps can be aborted with ‘crtl+c’ (this will be indicated in the console) if you do not want to wait for the operation to complete. For example an incorrect unit was entered and resulted in a very small hexagon size that will take a long time to generate.

- Note: that it may be a better option to start the script over depending on the situation, for example, a new target CRS cannot be entered after the beginning.

What units are the variable inputs for the manual grid generation?

- The units of any input directly related to a CRS will be in the units of the CRS. However some inputs can be given in different units if the correct suffix is entered as indicated during the input prompt. The default is meters, and all calculations that occur will be performed in meters.

Can this tool be applied to regions outside of Nunavut?

- Any coordinate reference system that can be input into the code can have its area studied
- Polygon shapefiles that don't have the same fields as the Nunavut conservation features can also be used as the code allows for the selection of any attributes within the feature, simply use the “Choose Attribute” options to avoid errors. It is important to note that if the naming of columns is not consistent across loaded Conservation Feature files then filtering them will be ineffective as all files will be filtered by a single attribute. The script relies on the Conservation feature files having the correct column name for the “id” so that it can access it as expected. The input data should either be updated to be correct or the “ID” column name could be updated in the defs.py file.

What coordinate reference systems can be used with this application?

- Any CRS used for this function must be a projected coordinate system and not a geographic projection systems
- Any coordinate system with an EPSG code can be entered if it is not the default option provided (Albers Equal Area)
- If the CRS does not have an EPSG code it can be loaded from a file that has the desired CRS

Can I use a pre-existing Planning Unit Grid?

- Yes, simply select the option "Load existing Grid from File" to load a grid from file
- However, If you use a pre-existing grid it must contain a column "GRID_ID" with a unique value for each cell, the column name can be updated in the def.py file if desired.
- Note: Loading a pre-existing planning unit grid will (if it is projected) override the target CRS to avoid distortion, this CRS will then be used for all files

Sphinx Output

Sphinx output is included in the Github Repository under docs/Build/HTML/Index, the index file can be opened in Chrome to view the auto generated documentation.
Sphinx is a Python documentation generator that is used to build high-quality documentation for Python projects. Sphinx can be installed using ‘conda install -c anaconda sphinx’, if document regeneration is desired.

Discussion

Challenges

We encountered some challenges with understanding the Marxan filetype, running the query by conservation layers. For the querying, it was challenging as we were essentially trying to query a list of geodataframes, which is not something we've encountered before. Eventually we figured it out but it took a lot of digging and trial and error. There were also many issues that were encountered in the process of writing the grid generating function. The code used to generate the hexagons did not create hexagons that were the correct size given the area that the user input, a function then had to be written that could correctly take in the area of the hexagon and then convert that to the edge length of the individual cell for the function to utilize. The shapefile to grid process also ran into some problems early on as the grid was being created in the same coordinate reference system as the shapefile, but this did not align with the target crs, leading to distortion in the grid. This was corrected by reprojecting the loaded shapefile before any work was done with it.

Limitations

One of the limitations of the program is runtime. If the user decides to run a high density area with small hexagon cells it can take a long time to run. Generally, the program takes anywhere between no time (instant run) and hours, depending on the area. Another limitation was the save file for the GeoPackage. This caused errors because it was not able to view the file as an Albers Equal Area projection in ArcGIS Pro.

Client interactions

We communicated with Adrian when we ran into questions and issues, and met with him two times. We received guidance on what he was looking for in the project, and specifics like the CRS to use, how he plans on using our script later, and what interactive components are interesting to him. In our second meeting with Adrian he specified that instead of entering the coordinates and extent of the grid manually, he prefered to use a shapefile input to define the area of the hexagonal grid. This allowed us to focus more on that method of input, instead of developing other input methods such as drawing a polygon on a generated map, which he was not interested in utilizing. Adrian also informed us that this tool could be used for other regions outside of Nunavut, so we ensured that the code took user input on the CRS, this way the grid could be generated for any projection. Further discussion with Adrian provided us with more detail on the format of the files that Marxan uses, such as the order of columns, the amount of decimal places, the area units required, and the need for a grid to be loaded beforehand as well.

Future work

If future work is done to build on our project, it may be worth looking into building in more interactive components. This could include selecting the area of interest on a map. This was something we thought about at the start, but ended up being outside of the scope of Adrian's interest for this project. However, it may be of use if the code was used for different applications. Additionally, it could be worth looking into ways to speed up the process of intersecting the

planning unit grid and the conservation features, as it can take a very long time. Moving the script towards a full GUI application to help improve usability and also packaging the script as a standalone application (executable) would significantly reduce the hassle of dealing with setting up the proper environment for it to work.

Conclusion

Our project was successful, with limited obstacles encountered. We wrote a Python program that generates a hexagonal planning grid, overlays it with various conservation features, and determines the area of the feature within each cell. It then formats the output for use in Marxan, as requested by the client. This program will be used by Adrian in conservation planning tasks. The project was overall interesting, challenging and a rewarding experience.

Acknowledgements

Thank you to Adrian Gerhartz for his assistance in understanding Marxan filetypes and the scope of the project, and for being our client for this project. As well, thank you to Professor Derek Mueller for running the course, finding us a client, and facilitating this project.

Sources

Hex grid generation - Source:
https://gis.stackexchange.com/questions/341218/creating-a-hexagonal-grid-of-regular-hexagons-of-definite-area-anywhere-on-the-g
Author: Kadir Şahbaz

Repository Files

https://github.com/GEOM4009/planningproj