GEOM 4009 Team Report

Shaolin-Rose Gawat, Grace Thompson, Julien Belair, Graham Scott
GEOM4009 - April 2023

Introduction

- **Background on the client**

As part of the GEOM4009 group project, we worked alongside Iryna Borshchova, Research Officer for the National Research Council of Canada (NRC). The NRC's objective is to develop knowledge, apply advanced technologies, and collaborate with other innovators to find sustainable solutions to Canada's present and future economic, social, and environmental issues. They are currently working with Transport Canada (TC), whose goal is to create a safe, secure, efficient, and environmentally responsible transportation system in Canada. Their goal for the project is to increase the operational scope of Remotely Piloted Aircraft Systems (RPAS) within Canada, while enabling the safe and effective integration of RPAS into the Canadian airspace. RPAS still proves to be a challenge for airspace navigation as manned aircrafts are currently better suited for the aviation industry. To overcome this issue, TC has implemented the Specific Operations Risk Assessment (SORA) process. SORA is a metric that helps RPAS and authorities evaluate the risks associated with specific operations. In the SORA process, the Ground Risk Class (GRC) assessment involves steps 2 and 3, which are crucial for ensuring that an operation can be carried out safely.

- **Client requirements and priorities**

The client requires a tool to aid in the GRC assessment which determines the ground population density of a specific area where an aircraft will be operating. They need a dynamic view of population density on the ground while being able to account for temporal fluctuations that may be present in the area of interest. The tool must be capable of accommodating population data from both Statistics Canada and TELUS. It should also have the ability to handle several flight paths and multiple areas of interest (AOIs).

- **Purpose**

Iryna mentioned that the tool we developed could be used to measure the safety of a drone flight that would transport organs from one hospital to another within the Toronto area in the event of an emergency. Our tool should be able to help the user determine the safest flight path in accordance with SORA based on population density at ground level during the time of flight.

- **Scope (what scope did you work to) [please reuse/build-on past work, as appropriate]**

  This was the original scope of the project, as proposed by the client:

  1. Help select several locations to study (high density and low density gradient) in Toronto

2. Develop a tool to determine population density in an area of interest (such as a proposed RPAS flight path or area) input by the user (with a kml or entered interactively) at a specific time of day
3. Compare population density estimates from Stats Canada data and Telus (or similar) data, so the difference can be easily evaluated
4. Allow for the possibility of sampling of multiple AOIs and multiple times of day, so this analysis can be done across space and time
5. In anticipation of continuous high resolution real-time Telus data, mock-up a least-cost path tool to route flights around high density areas at specific times of day based on an acceptable risk criterion

As a team, we managed to get our tool to complete tasks one through four. We held multiple meetings with Iryna and concluded that it was not feasible to get task five to work. Instead, on top of being able to see a table output of the population density data in regards to the relative AOIs, we made the tool be able to export said table, export the AOIs, and export the flight paths as shapefiles and geopackages accordingly. We also allow the user to display the flight paths, AOIs and population density information on an interactive web map using the folium python library.

Workflow

- **How your script works in a bulleted list**
    - To run the tool, the user must enter three pieces of data after typing “run popFinder”: the shapefile with the dissemination areas (polygons), a StatsCan csv containing the population data, and a TELUS csv containing population data.
    - The code checks if the data is valid and processes the data into the codes internal data structure
    - After the user has entered their desired data, they can decide to do up to 7 things by typing different numbers:
        1) Add flight paths (as kml files)
            a) Code checks if the data is valid
        2) Add a 1 hour time frame (%Y-%m-%d %I:%M %p Or YYYY-MM-DD HH:MM P (where p is AM or PM)
            a) Code checks if the data is valid
        3) Calculate the data
            a) The code calculates the buffer around kml
            b) Calculate intersection of the buffer and the dissemination area polygon
            c) Calculate weight (intersected area/polygon area)
            d) Multiply weight by static population
            e) Add all weighted populations together
            f) Calculate static population density (weighted pop/ total intersected area
            g) Add All intersected area together

h) Create geopandas data frame with static population, static population density, geometry of the line and geometry of the buffer
    i) For each datetime
        (1) Multiply weight by datetime population
        (2) Add All weighted population together
        (3) Calculate datetime population density (weighted pop/ total intersected area
        (4) Append to geopanadas dataframe
4) Print the calculated data in the console
    a) Print main (calculated in 3)) geopandas data frame
5) Export the calculated data as a csv (called FlightPath.csv)
6) Export the calculated data as a geopackage
7) View the map of all the dissemination areas and flight paths (this makes an html file in the CompletedData folder)

- Once you enter done, the script ends the loop. Note: once the user enters "done", all the data the program calculated will be lost.

- **UML diagram or flowchart to illustrate [please reuse/build-on past work, as appropriate]**
  Here is the link to our flowchart:
https://app.diagrams.net/#G1O84YORygSKoJLPQwJZ-0SowOFm9A996F

Documentation

- **Dependencies (hardware and software); you may refer to a conda environment as defined in the appendix, but name the key packages here.**
  For our python tool to work efficiently, there are some key dependencies that are needed in order for it to run. The user must have the common programming language Python downloaded. With Python, you can have multiple libraries which will enhance the programming capabilities of the language. The required libraries to run the PopFinder tool are GeoPandas, Fiona, Folium, Shapely.
- **Installation/set up (for your script(s); see below for conda environment**)

In order to install our script, the user must first have Anaconda Prompt. Inside the Anaconda Prompt, the user must direct themselves so as to be in the same directory as the environment.yml file. Once the user is in the correct directory, they need to create the environment by typing this command:

```
conda env create -f environment.yml
```

This may take a few minutes, but once the environment is created, the user can enter this command to activate the environment:

```
conda activate popFinder_env
```

If you run into any problems please consult this website:
https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

- **User guide**

To run the tool, the user must enter three things after typing “run popFinder”: the shapefile with the dissemination areas (polygons), a StatsCan csv containing the population data, and a TELUS csv containing population data. This is an example of what can be entered: *run popFinder "./TestData/ExportedArea - Shapefile/ExportedAreas.shp" "./TestData/stat_can_data.csv" "./TestData/mock_telus_data.csv"*.

After the user has entered their desired data, they can decide to do up to 7 things by typing different numbers:

1) Add flight paths (as kml files)
2) Add a 1 hour time frame (%Y-%m-%d %I:%M %p Or YYYY-MM-DD HH:MM P (where p is AM or PM)
3) Calculate the data
4) Print the calculated data in the console
5) Export the calculated data as a csv (called FlightPath.csv)
6) Export the calculated data as a geopackage
7) View the map of all the dissemination areas and flight paths (this makes an html file in the CompletedData folder)

Once you enter done, the script ends the loop. Note: once the user enters “done”, all the data the program calculated will be lost.

- **Troubleshooting/FAQ**

If an error occurs in part 3, when calculating the data: a hard copy of your data will be saved up to that point. The program is still usable, however, at this point if you want to add more data, it would be easiest to restart the program. Note that all your progress will be lost.
If the program will not start: check the column names and make sure they match the required column heading names.
What happened to my files? Check in the directory of your program. Note that the files will share names. If you want a hard copy of a specific point, it is best to manually save and rename it to what you want.
How should my kml file be formatted? The program works best with lines. Although points and polygons will allow the program to run, lines work best.
My files are being overwritten. Note that the exported files will share the same name. If you want multiple files with different names, it is best to manually rename your files throughout your progress.
How do I see the Sphinx documentation? This is locatable in our GitHub server.
How do I see the finished map? Open the exported map file called ‘map.html’.
How do I see the population for the line? Click anywhere within the buffer of the line in the exported ‘map.html’ file.
How large is the buffer? The buffer for the kml line is set at 500 meters.
What is the projection of the file? The projection of the file is in CRS = "ESRI:102002".

How do I see one line at a time on the map? Open the content plain in the corner of the map. You can select the specific map layer you want to see from there.
My recently added data isn't showing up in prints or exports. Make sure to recalculate your data before any prints or exports, the new population data will be slow.
How is the population density being calculated? It is calculated by locating the intersected area of the buffer with the selected polygons. Note that these numbers are estimated.
Missing dissemination areas. The tool will only calculate data for the areas where dynamic and static data are both present. If there are missing dissemination areas (polygons), this is because there is missing dynamic or static data.
Will the tool run without dynamic data? Dynamic data needs to be provided in a csv file for the tool to run. If the csv file is empty but still has column names, the tool will work until you add a time.
Will the tool run without static data? The tool will not run without static data.
Will the tool without a shapefile? The tool will not run without a shapefile. The shapefile is what we use to visually represent the map areas.
Why do I not see the buffer in the gpkg? The gpkg exports all the data with the geometry of the line.

- **Sphinx output of the autogenerated documentation from source code docstrings [makes plain any assumptions regarding system setup, user behaviour or input data] [use of screenshots as appropriate]**

Population Project

Navigation

Quick search

Go

Welcome to Population Project's documentation!

Created on Wed Mar 15 20:54:35 2023. Last updated on Thurs Mar 30, 2023.

@authors: Grace, Graham, Julien and Shaolin

This program will allow the user to calculate an estimated population that a flight path (and its buffer) intercepts. The population is calculated from static population data (StatsCan table) and dynamic population data (TELUS table). Once the user ends the program, all data that was not exported is lost and will need to be recalculated.

This is a user lead program and the user can decide to do 8 things

1. Add flight paths
2. Subset a 1 hour time frame
3. Calculate the data
4. Print the calculated data in the console
5. Export the calculated data as a csv
6. Export the calculated data as a geo packages
7. View the map of all the dissemination areas and flight paths

This program takes in 3 files

1. A shapefile of dissemination areas with a DAUID and a DGUID columns
2. A dynamic (TELUS) csv with a DGUID, timeframe_bucket (%Y-%m-%d %I:%M %p), and a count columns
3. A static (StatsCan) csv with a DAUID, POP_2016, POP_DESNITY and LANDAREA columns

If these files are not provided the tool will not work Here is an example : run popFinder "./TestData/ExportedArea - Shapefile/ExportedAreas.shp" "./TestData/stat_can_data.csv" "./TestData/mock_telus_data.csv"

Known Problems:

ARCGIS/ESRI shapefile cuts off the full column name for times.

`popFinder.calculate_stat(kml, shapefile, times)`

____Grace____ Calculate the stats according to one kml file.n

Parameters:
- **kml** (*String*) – String to a valid kml
- **shapefile** (*geo pandas df*) – A GDF with population data - note that we are assuming that this shapefile has DAUID, POP_2016, and column names that match times * **In the future perhaps this should be an object! This way we know that it is in the proper format and we can add stuff to not recalculate the same data** *
- **times** (*List of datetimes*) – List of datetimes to calculate

Returns: **temp_df** – A gdf with the name of the kml, affected area (km^2), stat_pop, stat_density, and times + population density for the specified times

Return type: a geopandas df

popFinder.**checkHeading**(*filepath*)

____Julien____ This function checks if a csv file has actually been inputted, if not, it raises an error. If it is a csv, then it checks to see if it contains the expected headings. Returns a dataframe if it does or raises an exception if it does not. Tested using different valid and invalid csvs, works appropriately.

Parameters: **filepath** (*String*) – A file path to a csv file
Returns: **df** – The csv read into a pandas dataframe
Return type: pandas dataframe

popFinder.**choropleth_map**(*shp_df, aois, times, user_df*)

____Grace & Julien____ This creates a map of calculated lines and the dissemination areas The dissemination areas appear as a choropleth map The lines appear as orange lines with their buffers

Parameters:
- **shp_df** (*a geopandas df*) – The calculated shapefile that is created at the start of the tool
- **aois** (*list of strings*) – List of all AOIs the user has inputted -> this should match up with the recalculate function (will be solved with the recalculate)
- **times** (*list of datetimes*) – List of all times the user has inputted -> same as above
- **user_df** (*geopandas df*) – The calculated shapefile that is done by the recalculate function

Returns: **m** – A folium map with the dissemination areas and lines Please note that this only returns something for exporting -> which is something that is removed right now. If exporting is to stay removed then this should return nothing
Return type: folium.folium.Map

popFinder.**validAOI**(*filepath*)

____Graham____ This function will check if the file is a valid kml. This function does not check whether the kml file contains a linestring, point, or polygon, because all three will work either way However, this function also does not check whether or not the kml intersects a dissemination area because it is possible for the buffer to intersect the area but not the line. Return True if valid Return False if not

The function was tested by running in the command line with different file paths. A path to two different kml files was tested, all returned True. A gpkg file was tested for reference, it returned False.

Parameters:	**filepath** (*String*) – A filepath to a kml
Returns:	If the kml is valid or not
Return type:	bool

popFinder.**validDate**(*date*)

This function will check if the date is in the proper format of YYYY-MM-DD HH:MM P

P in this case is either AM or PM

This function was tested with the below code and worked: print (valid_date ("2023-02-20 12:00 AM")) #prints True print (valid_date ("2023-02-10 2:00 AM")) #prints True print (valid_date('2/26/2009 3:00 PM')) # prints False, must have hyphens (-) not slashes (/) print (valid_date ('21-11-06 2:00 AM')) # prints False, must have 2021 not 21 print (valid_date ('2023-02-20 12:00:00 AM')) # prints False, has extra seconds in string print (valid_date ("2023-02-10 20:00 AM"))# prints False the hour must be in a 12-hour clock as a zero-padded decimal number (i.e. 01, 02, ..., 12).

Return True if valid Return false if not

Parameters: **date** (*String*) – The input or desired date.
Returns: If the date is valid or not.
Return type: bool

Indices and tables

- Index
- Module Index
- Search Page

Discussion

- **Challenges [briefly describe any issues that you overcame]**

A challenge we overcame was the way we wanted to export our map. We were interested in using Folium to create an interactive map. But it was difficult to add multiple elements to the map, but we were able to create an html file with a pop-up in the buffer to display the data that we wanted.

Another challenge was the design of the tool. We had to go through some remodels to ensure that it was formatted correctly and all group members were doing the same thing, for example, the format of each individual's code.

Finally, a challenge we had to overcome was GitHub. Before this course, only one of our group members was fully familiar with the platform. It took some getting used to for our group to navigate through our branches, but we were able to figure it out and use it thoroughly.

- **Limitations [briefly describe any issues you could not overcome]**

The biggest limitation we faced was time. If we had longer than one short term to work on the tool, it could have been developed further to enhance the way the tool works.

Another limitation was the re-calculating feature in part 3 of running the code in the command line. It is re-calculating data that has already been calculated, and it is not very fast.

Finally, we did not have real TELUS data which meant all of our data was hypothetical.

- **Client interactions [briefly state how often you communicated, what guidance you received]**

Throughout the course of the term project working with Iryna, we met every Monday at 2:30 over Microsoft Teams to show our weekly progress, as well as hear her suggestions and feedback. We continued to meet as a team over Discord twice a week, Thursday mornings and Sunday evenings, to keep in contact with one another about our current progress on our tasks.

- **Future work [what would your recommendations be?]**

Future work for this tool would involve the recalculate functionality so it can understand not to recalculate everything. It would also be nice to create a legend in the Folium html exported map for the lines. Figuring out the warning you are prompted with when you recalculate (this is a new error as of March 30, 2023).

Work to be done in the distant future could be adding a csv or gpkg from a previous run and restart from that. It would also be good to fix the problem where the band data is added with no future recalculation work needed. It would be good to recalculate the data before each action. It would also be a good idea to have the ability to remove times or kml's, or add a batch of kml's and times. It would be ideal to have a fully customizable Folium map, which would take a lot of time to figure out. Also, it would be good to be able to change the CRS.

Conclusion

- **Wrap-up statements [a short paragraph will do]**

To sum up, it was a pleasure to work alongside Iryna as our client. She provided some much needed clarifications and suggestions to make our python script even better than it was. At the end of the day, we think that our product is a good outcome of our interactions with a real client, and a good example of what a real life work interaction would be in a geomatics setting. It was definitely the first time for all of us to get a meaningful experience in a work setting and it was beneficial for all of us. Overall, we had a really good team chemistry which helped smoothing out all the tasks we had to do. We hope that this tool will be helpful in the real life example given to us by Iryna of transporting organs from one airport to the other.

Acknowledgements/Sources

- **Give credit/thanks to those who assisted you**

We would like to thank Iryna for being the best client we could have asked for. The weekly meetings were a huge plus for us as a group to get feedback and make sure we were on the right track.

We would also like to thank Derek for being a great professor, and for greeting the first draft of the TELUS data for us, as we were unable to get the actual data from them.

- **Give links to websites, books, etc., where you got the ideas for your code**

The links to the websites have been provided in the documentation of each function in the python script. All resources can be found there.