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*)

___Shaolin___

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 (validdate ("2023-02-20 12:00 AM")) #prints True print (validdate ("2023-02-10 2:00 AM")) #prints True print (validdate ('2/26/2009 3:00 PM')) # prints False, must have hyphens (-) not slashes (/) print (validdate ('21-11-06 2:00 AM')) # prints False, must have 2021 not 21 print (validdate ('2023-02-20 12:00:00 AM')) # prints False, has extra seconds in string print (validdate ("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