A Solution for Sourcing Fine-Grained Mobility Patterns within Large-Scale GPS Datasets

A Geospatial Analysis by Sophie Kolston (GISCI343 Project)

3951 words

Problem

What is the problem you are trying to solve?

The ubiquity of Global Positioning System (GPS) technology in the modern world is undeniable. It is depended upon for navigation of our daily environments in a way that was never possible before its conception, potentially offering the extraction of insights into human mobility that will be significantly more useful and accessible than traditional sources of movement data (Siła-Nowicka & Fotheringham, 2019).

Individual-level human mobility is extremely important to understand for urban planning, traffic forecasting, tracking virus spread and so on (Gonzalez, Hidalgo, & Barabasi, 2008). The problem is that user-friendly tools that exist for analyzing mobility often only see a dataset as a whole, not able to examine an individual agent. As existing models approach a well-defined limit in predictability of human mobility (Lu, Wetter, Bharti, Tatem, & Bengtsson, 2013), a tool that can focus on the mobility of a single agent could provide insights that allow models to be built with higher levels of accuracy. This tool would be able to load a large dataset of human trajectories and cluster movements. Patterns of the individual derived from this proposed tool could be used to pseudo-randomize agent parameters, building on model accuracy.

The purpose of this report is to outline the development of such a tool in order to solve this problem. It will be basic, but will prove the concept of a user-friendly way to produce fine-grained analysis of large datasets. These datasets will be basic but large GPS trajectories, which are readily available (Zheng, Zhang, Xie, & Ma, 2009).

Who is the intended user for your solution?

The intended user for the solution will be anyone who is interested in mobility data, but does not have programming experience. To this end, it would likely be a GIS analyst who is not interested in programming, but who wants to perform analysis out of the range of traditional 'desktop' GIS tools. It could be a model builder wanting to see fine-grained patterns to tune in agent parameters, or even a planner who is interested in clusters of individual-level movements to design infrastructure around.

The solution will be a minimally viable product (MVP) to achieve basic clustering analysis on the individual, meaning it could be built upon by those desiring further functionality. For the purposes of the MVP, we can conceive an intended user as an individual who wants to see single-user data of a large dataset without any prior programming knowledge.

How will your software solve the problem for this user?

This is a broad and rather ambiguous definition for an intended user. Instead of precisely defining an individual, we can focus in on the key attributes that the software will have in order to solve the problem for this user.

The software will solve the problem for this user by:

- Allowing for the loading of a large dataset, either directly into the memory of the users' computer or onto a PostGIS server.
- Displaying the data within this dataset in a visually compelling and useful way.
- Running through the steps of the analysis if required so that the user can make modifications where necessary.
- Splitting the dataset into that of the individual agent, then running a clustering classification algorithm.
- Doing all of these previous tasks with a user-friendly interface that requires no programming knowledge. It will have the option to be web-based to make it even easier to access.

By adhering to these key attributes, the software should be an extremely effective tool for the intended users. The software will thus solve the problem of easily sourcing fine-grained mobility patterns within large datasets.

Literature Review

Human mobility is a revived research area thanks to the abundance of volunteered and commercially-obtained telemetry data. Previously data collection in human mobility was time-consuming and expensive. Geospatial sensors on smartphones and other devices have drastically reduced the difficulty in acquiring detailed data in human mobility (Siła-Nowicka et al., 2016). These sensor data, in the form of GPS trajectories, are often used to make judgements and further claims about a population, especially in urban areas with huge amounts of data. We can look into existing studies on fine-grained GPS trajectories to see what existing solutions exist to this problem and how we might improve upon them.

Jinjun et al. (2015) used GPS trajectories from taxi services in Harbin City, North-east China. The researchers looked at these taxi trips and performed statistical analysis on average speed, occupancy, travel time and more. Introduced is the Density-based spatial clustering of applications with noise (DBSCAN) algorithm (Tang, Liu, Wang, & Wang, 2015). They expanded on this clustering algorithm significantly with attractiveness of points within the calculation, but on its own it provided a good baseline clustering result. Distribution of taxi trips showed a significant between occupied and non-occupied trips, as well as similar trends with other variables that impacted the fitting parameters of the models employed. Analyses used in this paper were not approachable nor user-friendly. Interesting conclusions were made, but a solution is needed that can be approached by anyone.

A potential use case for the proposed solution was to collect data to be used for agent-based modelling. Literature supports using GPS trajectories to tune parameters for an agent-based model, such as an agent representing a fire emergency vehicle based on real-time driving data in Allahabad City, India (Bandyopadhyay & Singh, 2016). Researchers combined these trajectories with road network, population density and landuse data to build a fire emergency vehicle agent that could be executed to determine response time. They found that the behaviour of the agent, or at least the key output variable of response time, matched that of the real-world, indicating that the GPS trajectories were good source data for such a use case. The GPS trajectories used were pre-processed into a shapefile and were not analyzed individually. If the proposed tool was employed, the researchers may have had additional insights into emergency vehicle movement and been able to do the processing themselves (ie. remove noise), strengthening their research output.

Bringing in geospatial data for real-time transport monitoring is a constant area of study. In an older era of GIS, GPS technology installed on laptops was used to analyze day-to-day variations of travel times within an urban environment (Ohmori, Muromachi, Harata, & Ohta, 2002). This is relevant to the proposed system as it shows the difficulty in handling large GPS datasets on desktop GIS platforms which remains today. Huge amounts of processing were required to load on desktop software. It may be necessary for the proposed solution to have a form of simplification on the data in order to be processed on an computer accessible by the potential user.

Trip identification is an ever-increasing analysis on GPS data, made even more effective with the aid of modern machine learning and other big data techniques (Montini, Rieser-Schüssler, Horni, & Axhausen, 2014). For example, Lee, Yu and Kim (2021) used GPS data and defined points-of-interest to classify the purpose of trips employing a public bike sharing program. A trip 'inference model' was built using machine learning algorithms run on this data and could be used to understand the reason behind human mobility within cities (Lee, Yu, & Kim, 2021). Another study, using the GeoLife dataset, inferred transportation modes using a novel iteration of a random forest classification algorithm (Li et al., 2021). These methods will extend beyond the scope of the proposed tool, but should be made possible to be implemented by a knowledgeable user. Even identifying the particulars within a classified trip mode can be employed, such as detecting the points at which a vehicle has stopped during traffic using GPS vehicle trajectories (Rehrl, Gröchenig, & Kranzinger, 2020). The clustering techniques of the proposed tool should provide data to make inferences into travel mode, or at the very least will provide a framework to run further big data analysis using more complicated methods by an experienced user. To this end, the proposed tool should be able to operate on datasets that do not have pre-classified trip modes.

Other literature shows how the proposed tool could be used for capturing information on human mobility within strictly defined areas or events. Participatory data collection was employed at a festival in Switzerland to capture the dynamics of crowd movement (Blanke, Tröster, Franke, & Lukowicz, 2014). The proposed tool could perform basic versions of this research in an easily accessible way, to be used as a starting point for further research by the right user. We can clearly see the potential of an accessible way to produce this data, and how it is not strictly limited to urban mobility and transportation. Any dataset with GPS trajectories, urban or otherwise, should be accessible within the proposed solution.

The extensive depth and amount of research into GPS trajectories is clear. We can see that tools that exist to perform analysis are often lacking in approachability. The proposed tool will aim to solve this problem, while also providing a framework to perform more complex analysis on if the user desires.

Data

The dataset that this analysis will focus on is the GeoLife GPS trajectory dataset. This was collected over a four year period by Microsoft Research Asia and consists of GPS trajectories, along with some labelled travel modes, from smartphones of 181 users (Zheng, Fu, Xie, Ma, & Li, 2011). Most of the dataset does not contain labels for travel mode. Each user has a set of trajectories associated with them, with the entire dataset containing over 24 million rows of data. These data are ideal to test the proposed data analysis system as they are fine-grained at the scale of an individual agent. GeoLife has many studies based on it, such as using a neural network to classify transportation modes of the trajectories (Wang, Liu, Duan, & Zhang, 2017).

I wrote a class that extracted this dataset and uploaded it to a PostGreSQL server. Table 1 shows a randomly selected set of rows from this dataset. We can see that it has been uploaded in a format of a unique identifier (gid), a timestamp (time), location (geom), travel mode (label) and the user it belongs to (user_id).

Table 1: Five Random Data Points from the GeoLife Dataset

gid integer	time timestamp without time zone	geom geometry	label integer	user_id integer
20709707	2009-04-24 10:09:40	0101000020E610000093E...	0	153
22688720	2008-06-20 07:59:19	0101000020E6100000921...	1	163
6185106	2009-02-07 04:44:51	0101000020E6100000208...	0	25
9471275	2009-03-30 03:19:25	0101000020E6100000E7B...	0	41
21159765	2010-02-16 05:31:43	0101000020E61000006E1...	0	153

We will focus on Beijing, China, as this is where the study focussed and is where the vast majority of data originates from. The application and validation of existing models in Beijing to be extrapolated to the whole of China (Long & Shen, 2015) means that this area is a powerful barometer of urban human mobility. That being said, while it will not be a priority of the MVP, the solution will have the framework to be developed to run on other GPS trajectory datasets.

Implementation

Designing the Solution - Pseudocode

Before writing code to implement the solution, we need to design exactly how the code will work in order to solve this problem of accessible, fine-grained human mobility data analysis. This will be done using high-level pseudocode, an effective way of outlining the logic behind algorithms (Oda et al., 2015), where a series of steps will be listed to denote future solution functionality.

The solution should be executed in a way that chronologically achieves the following functions (pseudocode):

1. Let the user point to the location on their local computer the GeoLife dataset is saved. Load the data into the program's memory. See Figure 1.
2. Upload the dataset to a PostGreSQL server (with PostGIS).
3. Display a map of the area of interest (Beijing, China) showing the urban and natural environments. See Figure 2.
4. Display a heatmap of the area of interest (see Figure 3). The amount of data shown will depend on the limitation the user has set (see Figure 4), as the entire dataset can be too difficult to handle.
5. Let the user select a user within a list of all unique user identifications within the dataset (see Figure 5). This will be the user of interest, where gaining fine-grained trends of their mobility is the main purpose of the solution.
6. Run and validate a simplification algorithm on the data of the chosen user:
    - As with the heatmap limitation, this is because the dataset is resource-intensive, and reducing processing demand will increase the accessibility of the application.
    - To validate, the raw and simplified datasets will first be plotted next to one another for visual inspection (see Figure 6).
    - Finally, the datasets will be plotted atop one another (see Figure 7) to see that the simplification completely matches the mobility of the raw data.
7. Run a clustering algorithm on this user to see patterns of mobility (see Figure 8, 9).
8. Display a map of the output of the clustering algorithm, showing fine-grained patterns of movement (see Figure 10).
9. Allow the user to select other users to repeat these steps (see Figure 10, 11).

While a drastic simplification, this gives the overall impression of the steps required for the program to meet the solution.

Figure 1: UI showing User Selection of GeoLife Dataset and Loading into Local Memory

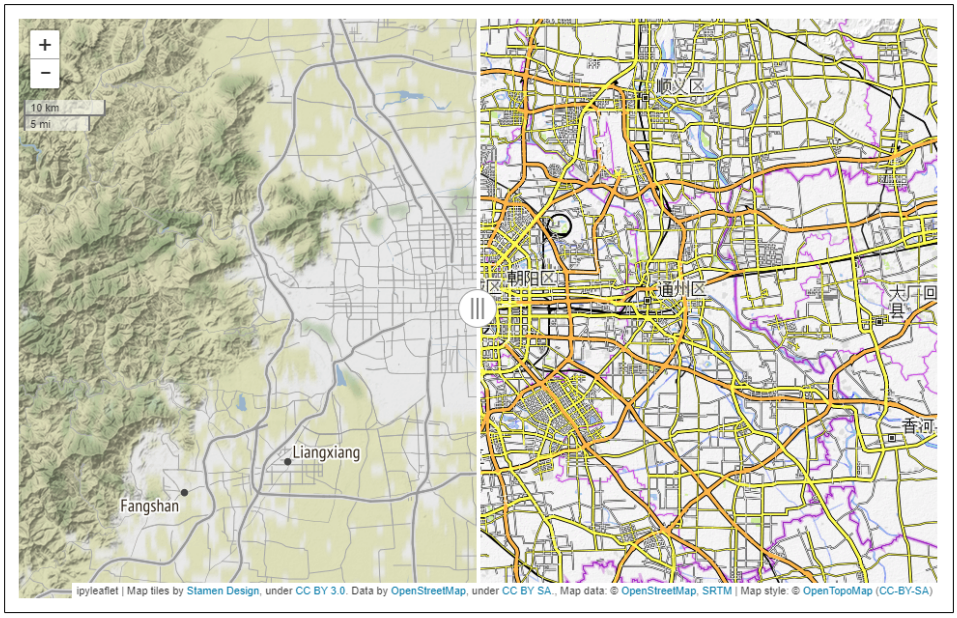

Figure 2: Map showing Beijing, China, with Stamen Terrain (natural, left) and OpenStreenMap (urban, right) Basemaps

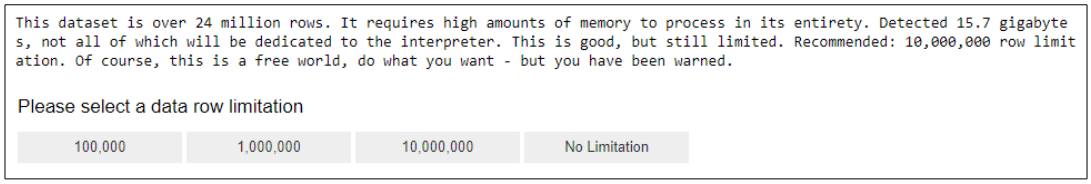

Figure 3: UI showing User Selection of Data Limitation with Memory-Specific Recommendation

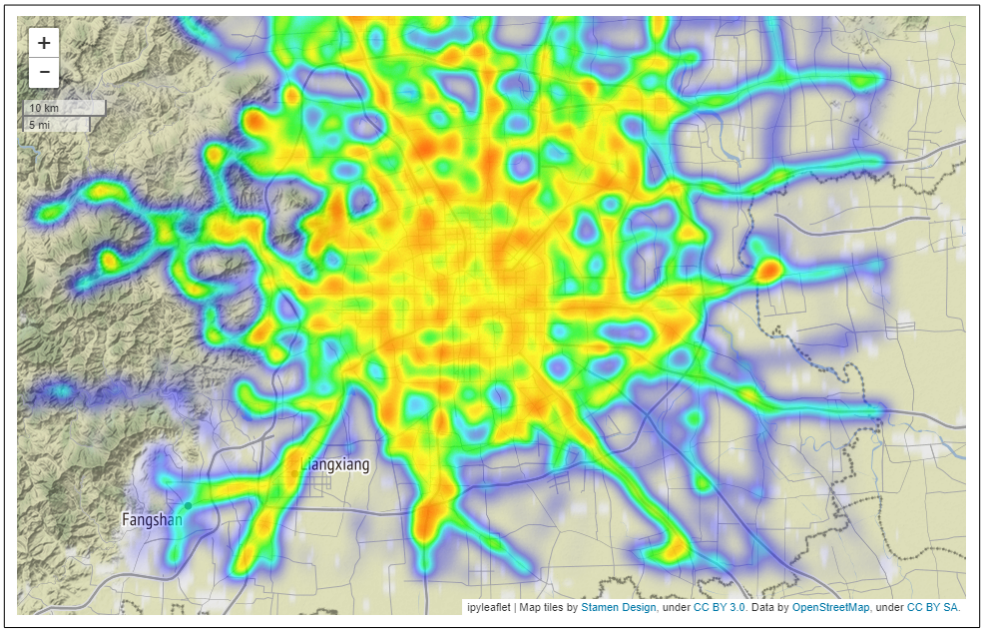

Figure 4: Heatmap of Human Mobility within Beijing, China, based on GeoLife Dataset

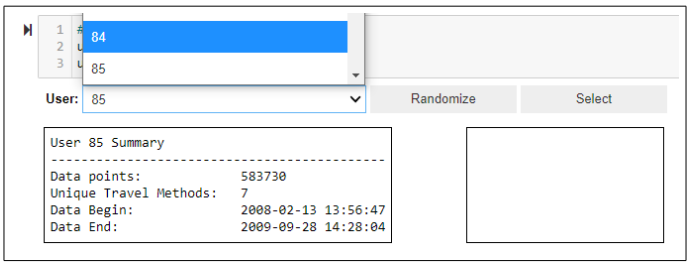

Figure 5: UI showing User Selection of Individual 'Agent' within GeoLife Dataset, with Summary Information

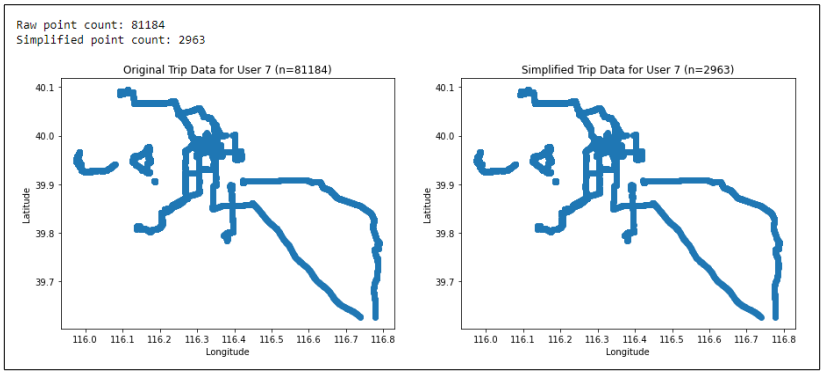

Figure 6: Plots of Raw and Simplified Datasets of User 7

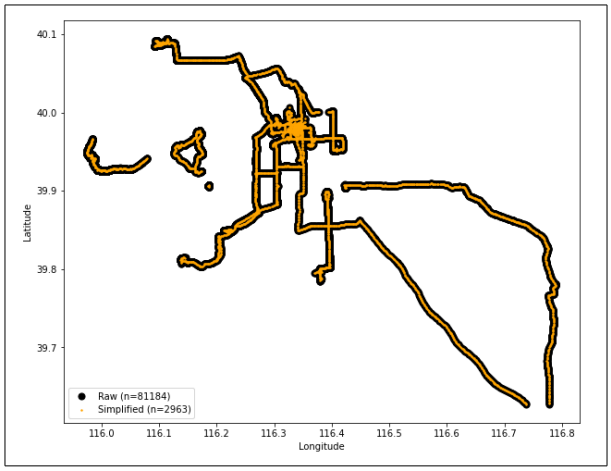

Figure 7: Plot of Raw and Simplified Datasets of User 7, showing Simplified Dataset is Contained by Raw Dataset

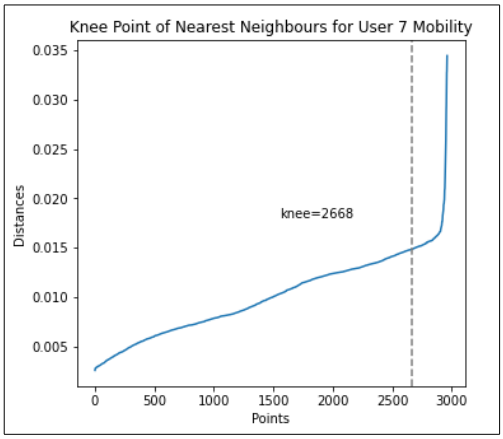

Figure 8: Plot of Nearest Neighbours Output for User 7, with Knee Point

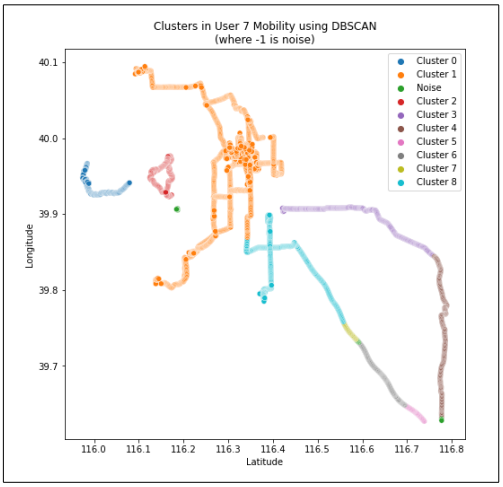

Figure 9: Plot of DBSCAN Clusters for User 7

Implementing the Dashboard

GIS dashboards have become an increasingly powerful tool to display geographic information in an approachable way, especially in the decision-making surrounding the Covid-19 pandemic (Mollalo, Vahedi, & Rivera, 2020). The solution needs to be approachable, and panning through a Jupyter Notebook, while incredibly powerful for data scientists (Perkel, 2018), can be confusing to those unfamiliar with programming. To meet this key attribute of approachability for the MVP, we can consolidate all of the functions run through the Notebook into an individual cell. We can combine all of the important graphical and analytical components into one easy-to-deploy section. This will herein be referred to as the 'dashboard' component of the solution.

Progressing to a dashboard, we can have a look at each object that comprises the solution, and how they interact with each other at a lower level. Paper prototypes are important to the agile methodology, and I used them in the design of the user interface of the dashboard. The product of the design process can be seen on Figure 10, where I have created a dashboard interface. This dashboard focuses around a folium map, with matplotlib plots and input/outputs straddling this main feature. The design should appeal to the potential user as it is user-friendly - the main attention is given to the map, with more features there if needed. Paper prototyping is typically used to extract the features required by a stakeholder, to then be implemented by software developers (Vijayan & Raju, 2011). In this case, I have used it as both a developer and representative of a stakeholder, but have resulted in a final design that will aid in meeting the solution for the potential user nonetheless. It will achieve this by offering a straightforward, visually appealing interface with more complicated information available for more advanced users.

Code Walkthrough

We have now established that a series of algorithms, as explained in the pseudocode, execute in order to create an approachable solution to analyzing fine-grained human mobility data. What are these algorithms exactly? How do they work? Here we will go through each object (class), explaining how they work and what they achieve.

GeoLifeHandler: Interacts with the raw GeoLife dataset. This class was based heavily on that written by Here Technologies for their Point Processing ToolKit (*Point Processing Toolkit (PPTK) Documentation*, 2018), with a few key usability improvements. I added the ability to select a file location graphically (see Figure 1), as well as implementing a progress bar for the loading of the dataset. The users chooses the file location, clicks select, then waits as the data is loaded as a GeoPandas GeoDataFrame. The travel mode label (where applicable) is replaced with a numeric identifier to reduce redundant data.

PostGISUploader: Uploads raw GeoLife data to a PostGreSQL Server. Is also operated through a Graphical User Interface (GUI). I opted to iterate through the GeoDataFrame and insert values row-by-row as this offers the ability to show progress, and is less prone to crashing. This class also handles the limiting of data to Beijing (see Figure 3) Note that this class was omitted from the final cell as it is assumed that a functioning PostGIS server is present, although this is easily remedied.

FoliumCreator: Gets and sets variables around folium map objects. Optimizes map production by automating the gathering of map objects, with manual settings if needed. Output can be seen on Figure 2, 4.

UserObject: Serves as a container for individual user information, as well as probing for other user information. Runs queries based on new user selections, displays appropriate widgets, sets and gets the user of interest's identifier, and so on. Also handles the AntPath folium map for mobility. Probably the most extensive class, for good reason - the fine-grained information is what is most important to this project.

DBSCANner: Handles the analysis and plotting of DBSCAN clustering. Starts by getting the distances for Nearest Neighbours. Then calculates the knee point (see Figure 8), using this as the epsilon value in the DBSCAN algorithm. Applies labels to data based on DBSCAN clustering output, as well as plotting the output (see Figure 9).

Other, non-object-oriented code: Some code exists independently of objects. This includes the definition of constants (variables that do not change regardless of input, eg. Coordinate Reference System), the method that executes queries to the server, the initialization of the GUI and so on. Important to mention is the importing of modules, which provide functionality that can be considered as prerequisites for the solution to operate. An example of this is matplotlib, which plots data into graphs, tables and other visualisations.

The code I have written is accessible on GitHub, and is freely available under the MIT license (Kolston, 2021). For more information, you can look through the annotated Notebook by cloning the repository.

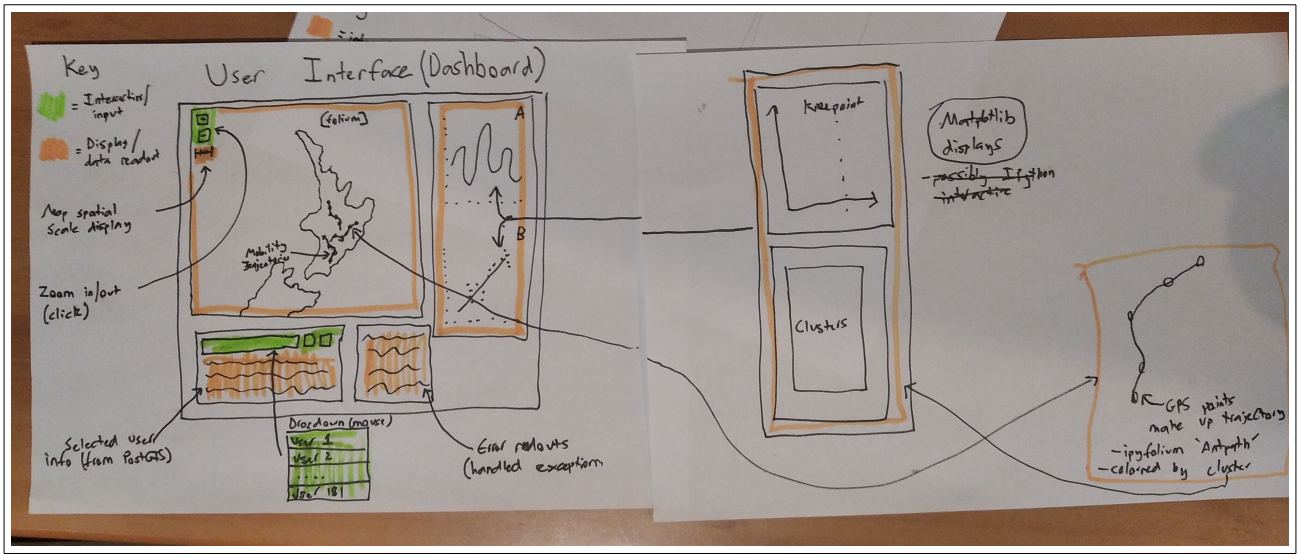

Figure 10: Paper Prototype of the Dashboard User Interface

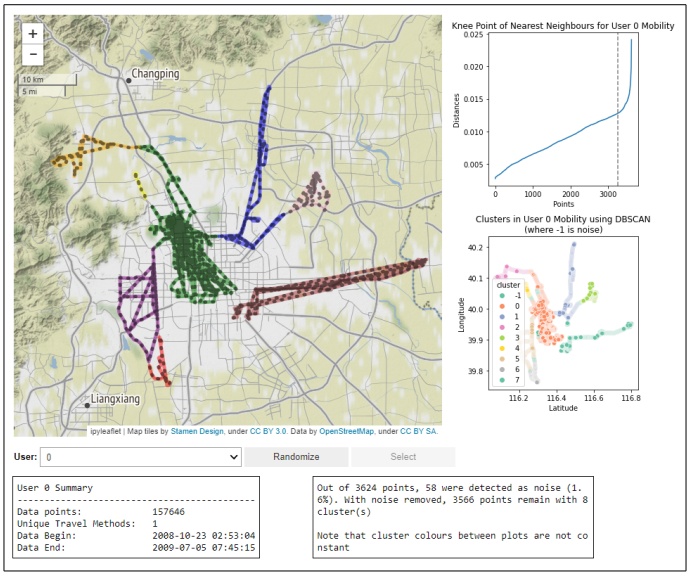

Figure 11: Dashboard Solution, Focussing on User 0

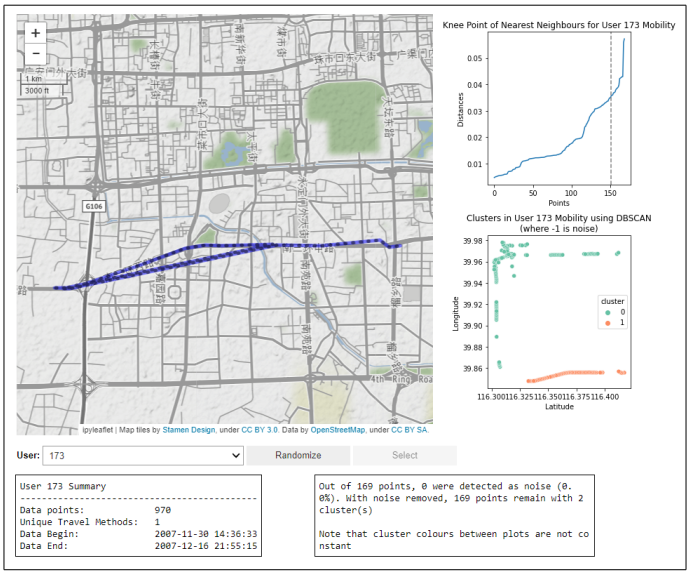

Figure 12: Dashboard Solution, Focussing on a Section of User 173's Mobility

Post Mortem

Process of Creating the Program - Reflection

What was easier to accomplish? I did not find any of the development to be particularly easy, developing and implementing the solution proved to be quite the challenge. That being said, the DBSCAN component was easier than I anticipated. Scikit-Learn have fantastic documentation and functions automatically tune the appropriate parameters. Once the initial learning curve of understanding how the clustering works, implementing this third-party package was straightforward and effective. Working with objects in general was also easier to accomplish than I imagined. Once I got the FoliumCreator object up and running, I found myself naturally writing classes where it made sense - as opposed to just writing them for the sake of the assignment which I naively thought I would end up doing. The versatility and efficiency objects offer was obvious, and Python makes it really easy to work with them.

What proved to be much harder? There were a few parts of the development process which stand out to be as being particularly difficult. Dealing with a dataset this large introduced lots of hurdles to deal with, especially when introducing PostGreSQL into the equation. It took several days for me to figure out just how to go about uploading the dataset to memory, and then to the local server. Luckily, once the data is on the server PostGIS makes it very easy to work with, even when you are dealing with tens of millions of data points. Choosing a user interface method proved to be much harder than I anticipated. QT is great, but hard to work with on a Notebook (the same goes for TKinter). Ipywidgets are great, but event handling is less-than-intuitive. I ended up settling with this, as I like the idea of it all being web-based for approachability. Creating the dashboard as a whole was very hard. Surprisingly, there is no decent open-source package for 'dashboarding' matplotlib, folium, ipywidgets etc. together. I had to mess around with custom HTML solutions, horrendous CSS spacing settings and so on. This was much harder than it needed to be, and I can definitely see an opportunity for an open-source GIS dashboard package. A final difficult development process worth mentioning was querying for user information when selecting new IDs under the UserObject. SQL cursors within Python is poorly documented and took a very long time and effort to figure out. Implementing an efficient information display for user selection thus was much harder than anticipated. All that being said, I am very pleased with the solution and any undue difficulty involved ended up being a brilliant learning experience.

Did your pseudocode match the final solution? I believe that the processes outlined in the pseudocode matched what the final solution achieved. Learning from my past experiences as a part of the development process, I wrote the pseudocode vaguely but deliberately - it reads as a readme file for a programming project. In my experience, this method is far more effective and aligns with the agile methodology that I have adopted in the background of development. As per the pseudocode, the solution handled the data, plotted and mapped it, all the while presenting the output in an approachable and meaningful way. As such, I can confidently say that the pseudocode matched the final solution, albeit in a more fleshed-out and feature-rich way.

Did the final features of the solution match what you thought it might have? The final solution featured an interactive UI, user selection, clustering and displaying of human mobility. This mobility was fine-grained, focussing on a particular user. These features were exactly what I envisioned in the design period, so I can confidently conclude that the final features of the solution matched what I thought it might have. That is not to say that there is not room for improvement.

What would you improve on in the future? The main improvement I would like to make would be the ability to run the solution on other datasets. Theoretically this would not be too difficult to implement, but would be time-consuming and not necessary for the brief of the solution I wrote. Another improvement would be user control for plot placement. Currently the plot positioning is static. User-resizeable and moveable widgets would be a great approachability feature that would improve on the solution in the future. Finally, I would like to see consistent colour-coding for clusters between the matplotlib plot and the folium map. I spent a very long time attempting to make this work, but requires a series of conditionals due to some datasets having noise and some not.

Conclusion

Overall, the development of a solution to the sourcing and presentation of fine-grained human mobility was an informative and useful endeavour. There now exists an accessible, visually-appealing and effective data presentation tool for human mobility on the scale of the individual. It has the potential to be run as a standalone web-based Notebook, or cloned onto a local computer for more advanced users to build on the framework I have produced. The main takeaway I have from this process is the need for an open-source alternative to the ArcGIS Dashboard software. There is the opportunity for the development of a Python package that can handle the presentation of spatial big data. That being said, what was developed for this project is effective and matches what was intended.

References

Bandyopadhyay, M., & Singh, V. (2016). Development of agent based model for predicting emergency response time. *Perspectives in Science*, *8*, 138–141.

Blanke, U., Tröster, G., Franke, T., & Lukowicz, P. (2014). Capturing crowd dynamics at large scale events using participatory gps-localization. In *2014 ieee ninth international conference on intelligent sensors, sensor networks and information processing (issnip)* (pp. 1–7).

Gonzalez, M. C., Hidalgo, C. A., & Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *nature*, *453*(7196), 779–782.

Kolston, S. (2021). *Geolife dashboard.* `https://github.com/Yozpoz64/geolife-dashboard`. GitHub.

Lee, J., Yu, K., & Kim, J. (2021). Public bike trip purpose inference using point-of-interest data. *ISPRS International Journal of Geo-Information*, *10*(5), 352.

Li, J., Pei, X., Wang, X., Yao, D., Zhang, Y., & Yue, Y. (2021). Transportation mode identification with gps trajectory data and gis information. *Tsinghua Science and Technology*, *26*(4), 403–416.

Long, Y., & Shen, Z. (2015). Big models: from beijing to the whole china. In *Geospatial analysis to support urban planning in beijing* (pp. 255–272). Springer.

Lu, X., Wetter, E., Bharti, N., Tatem, A. J., & Bengtsson, L. (2013). Approaching the limit of predictability in human mobility. *Scientific reports*, *3*(1), 1–9.

Mollalo, A., Vahedi, B., & Rivera, K. M. (2020). Gis-based spatial modeling of covid-19 incidence rate in the continental united states. *Science of the total environment*, *728*, 138884.

Montini, L., Rieser-Schüssler, N., Horni, A., & Axhausen, K. W. (2014). Trip purpose identification from gps tracks. *Transportation Research Record*, *2405*(1), 16–23.

Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., & Nakamura, S. (2015). Learning to generate pseudo-code from source code using statistical machine translation (t). In *2015 30th ieee/acm international conference on automated software engineering (ase)* (pp. 574–584).

Ohmori, N., Muromachi, Y., Harata, N., & Ohta, K. (2002). Analysis of day-to-day variations of travel time using gps and gis. In *Traffic and transportation studies (2002)* (pp. 1306–1313).

Perkel, J. M. (2018). Why jupyter is data scientists' computational notebook of choice. *Nature*, *563*(7732), 145–147.

Point processing toolkit (pptk) documentation. (2018). HERE Technologies. Retrieved from `https://heremaps.github.io/pptk/`

Rehrl, K., Gröchenig, S., & Kranzinger, S. (2020). Why did a vehicle stop? a methodology for detection and classification of stops in vehicle trajectories. *International Journal of Geographical Information Science*, *34*(10), 1953–1979.

Siła-Nowicka, K., & Fotheringham, A. S. (2019). Calibrating spatial interaction models from gps tracking data: an example of retail behaviour. *Computers, Environment and Urban Systems*, *74*, 136–150.

Siła-Nowicka, K., Vandrol, J., Oshan, T., Long, J. A., Demšar, U., & Fotheringham, A. S. (2016). Analysis of human mobility patterns from gps trajectories and contextual information. *International Journal of Geographical Information Science*, *30*(5), 881–906.

Tang, J., Liu, F., Wang, Y., & Wang, H. (2015). Uncovering urban human mobility from large scale taxi gps data. *Physica A: Statistical Mechanics and its Applications*, *438*, 140–153.

Vijayan, J., & Raju, G. (2011). A new approach to requirements elicitation using paper prototype. *International Journal of Advanced Science and Technology*, *28*, 9–16.

Wang, H., Liu, G., Duan, J., & Zhang, L. (2017). Detecting transportation modes using deep neural network. *IEICE TRANSACTIONS on Information and Systems*, *100*(5), 1132–1135.

Zheng, Y., Fu, H., Xie, X., Ma, W.-Y., & Li, Q. (2011, July). Geolife gps trajectory dataset - user guide [Computer software manual].

Zheng, Y., Zhang, L., Xie, X., & Ma, W.-Y. (2009). Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on world wide web* (pp. 791–800).