Determining DBSCAN-Entrpopy Hybrid Algorithm Parameters for Converting GPS Points to Activities

true true

2021-12-15

Abstract

DBSCAN Entropy algorithms appear to be the most accurate way to process the GPS data into trips/activities. This algorithm takes four parameters to work: minimum number of points (minpts), radius for those points (eps), entropy threshold (entr_t), and a time difference threshold (delta_t). While there are lots of studies that explain ways to determine minpts, eps, and entr_t, information lacks deciding on an accurate delta_t parameter. This paper explains one method of how to choose an adequate delta_t parameter: implement a DBSCAN Entropy algorithm in R alongside raw GPS data maps to use for comparison.

Question

Global Positioning System (GPS) surveys have become a more accurate and reputable alternative to previous travel survey methods that collect activity-travel patterns. Despite GPS devices' ability to record time and positional characteristics, further processing is required to determine the number of trips.

A DBSCAN algorithm (Density-Based Spatial Clustering of Applications with Noise) is one of the most accurate ways to process GPS data because it "overcome[s] some inherent limitations of partitioning and hierarchical algorithms"(Kefanjin 2019). In fact, one experiment (Luo et al. 2017) using just a DBSCAN cluster-based algorithms proved to be 92% precise which is significantly better than the typical 43% to 61% reported from using rule-based algorithms (Shen and Stopher 2014).

If one wishes to remove error even further, entropy (the chaotic movement betwen points in a cluster) can be added to the DBSCAN algorithm as done by Gong et al. in 2018 ((Gong L. 2018)). Including entropy removes inaccuracies that come from situations like a stoplight being mistaken as its own trip.

For a DBSCAN-Entropy Algorithm to work, four parameters are required: minimum number of points to be considered a cluster (minpts), radius in which those minpts can occur in (eps), the entropy threshold (entr_t), and the time threshold in which one trip is considered two trips at the same location (delta_t).

One common way to select the minpts and radius (eps) thresholds is to arbitrarily pick the minpts based on how large the data set is (with a minimum of three) and then set k = minPts in a k-distance plot (Kassambara 2018) to find eps. Unfortunately, this method only works in a pure DBSCAN algorithm where only minpts and eps are accounted for, not entr_t and delta_t. Gong et al. (2018) was able to determine an accurate entr_t threshold: . However, delta_t is still arbitrarily chosen and not as much research or experiments have been done to determine an accurate delta_t.

Hence, the purpose of this paper is to explore a method of how to select an accurate delta_t parameter to use in a DBSCAN-Entropy Algorithm while keeping the other three parameters constant.

Methods

Data

The GPS data used to determine the most accurate delta_t parameter came from one volunteer in the Utah County area and was taken over a period of six months. For the purposes of this question, only 10 days worth of GPS data will be processed.

Models

Ultimately, to choose the most accurate delta_t parameter, the number of clusters calculated by the algorithm (algorithm clusters) was compared to the number of clusters it appeared there should be for that day (manual clusters). The method for how those clusters were created are described below.

Algorithm Clusters

Once the data was cleaned and properly formatted, it was run through a DBSCAN-entropy hybrid algorithm largely based on the method created by Gong et al. in 2018 (Gong L. 2018). The concepts from Gong's DBSCAN-Entropy algorithm were taken and written in R using the *dbscan* package (Hahsler 2021) and a new *gpsactivs* package written by Dr. Gregory Macfarlane that has yet to be published to CRAN for public use.

For this algorithm, the four parameters of minpts, eps, entr_t, delta_t are required. Based on the current literature, the three constant parameters were set as follows:

- mintpts = 3
- eps = 25 meters
- entr_t = 1.0

To compare the accuracies of different delta_t parameters, 20 draws were done for each of the 10 days. Each draw kept the same constant parameters as listed above, and the delta_t parameter was randomly selected from a range of 1 to 400 seconds. By the end of running this algorithm, each of the 10 days had 20 outputs for the number of clusters as determined by the randomly selected delta_t parameter.

Manual Clusters

To get the number of "manual clusters" per day, maps of the raw GPS data were created using the *sf* package in R (Pebesma et al. 2021). Those maps were then referenced as the researchers made their own GeoJSON files that stored the geometric locations of where potential trips looked like they were happening. One GeoJSON file was created for each day.

Finally those GeoJSON files were read into R and appended onto the table including the algorithm's calculated number of clusters. For each of the 10 days, the 20 algorithm possibilities for number of clusters was compared to the number of manual clusters picked in the GeoJSON files. From this, the percent error was calculated and the delta_t parameter that consistently gave the lowest error across all 10 days was deemed to be the most accurate delta_t parameter to use in this DBSCAN-Entropy Algorithm.

Findings

The error between the two numbers of clusters per day was calculated by taking the difference in algorithm clusters and manual clusters. Then, the percent error was found by dividing the integer error by the number of manual clusters, since that was treated as the "goal" for the algorithm to match. The percent error is indicated by the "pctError" column.

```
##    eps minpts  delta_t entr_t       date    manual  clusters error  pctError
## 1:  40     20 29.18139      1 2020-02-16 <sf[3x2]> <sf[1x6]>     2 0.6666667
## 2:  40     20 29.18139      1 2020-02-17 <sf[3x2]> <sf[2x6]>     1 0.3333333
## 3:  40      20 29.18139      1 2020-04-15 <sf[4x2]> <sf[2x6]>     2       0.5
## 4:  40      20 29.18139      1 2020-04-16 <sf[6x2]> <sf[6x6]>     0         0
## 5:  40      20 29.18139      1 2020-04-17 <sf[5x2]> <sf[2x6]>     3       0.6
## 6:  40      20 29.18139      1 2020-05-14 <sf[5x2]> <sf[1x6]>     4       0.8
```

Figure 3.1 visualizes the percent error between algorithm clusters and manual clusters for all 10 dates. The black line represents the overall trend line.

(#fig:showErrorPlot)delta_t versus percent error by date

Most of the dates appear to follow the same trend with a decrease in error when delta_t is equal to 10.78 seconds. However, this is when the error is the largest for February 17th and May 27th, so is not the best option. The percent error is close to 0 for all dates when delta_t is equal to 106.3 seconds and 144.7 seconds because delta_t defines how long the minpts must be in eps radius for something to count as a trip. It is a lot more likely that a trip/activity is occurring if someone stays put for over a minute than only 10 seconds. For example, someone could be at a red light for 10 seconds, but that should not count as its own separate trip. Having delta_t be closer to 100 seconds removes potential error from situations such as that.

The trend line also shows that the larger delta_t gets, the larger the percent error gets. However, this is likely due to the outliers of February 17th and May 27th. Without those dates, the trend line would likely not start as low. It is also important to consider the possible gaps in this analysis: falsely identifying manual clusters in the GeoJSON software and the constant parameters. Manual clusters would be more accurate if confirmed by the respondent via a journal. Also, the algorithm would potentially predict a different number of clusters for each delta_t if the three constant parameters were different. Therefore, more rounds of this study should be conducted to confirm these results. Further applications of this method should be done with more dates, confirmed correct manual clusters, more than 20 draws for delta_t, and different options for the constant parameters.

Acknowledgements

References

Gong L., Morikawa T, Yamamoto T. 2018. "Identification of Activity Stop Locations in GPS Trajectories by DBSCAN-TE Method Combined with Support Vector Machines." *Transportation Research Procedia*. https://doi.org/10.1016/J.TRPRO.2018.10.028.

Hahsler, Michale. 2021. *Dbscan: Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms.*

Kassambara, Alboukadel. 2018. "DBSCAN: Density-Based Clustering Essentials." *DataNovia*. https://www.datanovia.com/en/lessons/dbscan-density-based-clustering-essentials/.

Kefanjin. 2019. "DBSCAN: Density-Based Clustering." *DataScience/MachineLearning*. https://ospinaforerolab.home.blog/2019/07/28/chapter-3-dbscan-density-based-clustering/.

Luo, Ting, Xinwei Zheng, Guangluan Xu, Kun Fu, and Wenjuan Ren. 2017. "An Improved DBSCAN Algorithm to Detect Stops in Individual Trajectories." *ISPRS International Journal of Geo-Information* 6 (3). https://doi.org/10.3390/ijgi6030063.

Pebesma, Edzer, Roger Bivand, Jeroen Ooms, and Dewey Dunnington. 2021. *Sf: Simple Features for r*.

Shen, Li, and Peter R. Stopher. 2014. "Review of GPS Travel Survey and GPS Data-Processing Methods." *Transport Reviews* 34 (3): 316–34. https://doi.org/10.1080/01441647.2014.903530.