W200 Project 1 Write Up:
GPS Summary & Steepness Calculator

Cameron Kennedy, Fall 2017

Program Overview

This program analyzes GPS data, summarizing information that seems difficult to obtain from existing software (steepness bands on a single event or grouping and comparing multiple events).

Main Menu

This menu is displayed at startup and redisplayed after every completed action.

1. **Load Single GPX File.** Loads a file to be analyzed. Prompts user to specify file location (relative or absolute). Replaces any previously loaded file(s).
2. **Load Multiple GPX Files.** Loads files to be analyzed and groups them. Prompts user to specify folder location (relative or absolute). It groups near-identical events (i.e., files that record the same route), and then prompts the user to choose the event grouping to analyze. Files must be in the same folder. Replaces any previously loaded file(s).
3. **Analyze Data.** Performs analysis of the loaded file(s).
    a. If a single GPX files is loaded (i.e., a single hike), in addition to standard summary statistics such as duration, distance, speed, and time of day, it will also calculate and return steepness statistics.
    b. If multiple GPX files are loaded, it provides stats for those events (e.g., median time), lists events chronologically, and compares events to the median and fastest times in that group.
4. **Quit.** Quits the program.

For those unfamiliar, a GPX file (GPS Exchange Format) is an XML file generated from GPS data – often from fitness devices such as watches, phones, or dedicated devices (e.g., a cycling GPS) – when a person records a journey (typically an athletic event such as running, cycling, or hiking). Most devices either store their data natively in GPX format, or can export to GPX.

Key Challenges Faced

In writing this program, I faced several challenges:

- The inherent imprecise nature of GPS data (especially with altitude data) proved particularly challenging, requiring many hours spent tweaking smoothing and clustering functions
- Using the clustering algorithm (DBSCAN, from scikit-learn) used to group similar, but not exact, events was challenging, though fun to learn!
- Learned fixed scaling was preferable to automatic scaling (to a standard deviation) for distance
- Learning the gpxpy library was challenging, though invaluable to this project
- Classes / object-oriented programming concepts are new to me, and were therefore challenging
- Learning to work with the file system was challenging, also because I've not done it before

How to Test

GPX files will be needed to test the program. Several sample GPX files have been included for testing, and they can be located in folders d1 and d2 under the program's main directory. Alternatively, if testers happen to have their own GPS device or know someone who does, they're welcome to test these files too.

Suggested Tests

Test single GPS files. Select option 1, then enter the filename "d1/a.gpx". Then select option 3 to analyze it. Repeat this for other files in that folder if you desire (b.gpx, c.gpx, etc.). Files in the d2 folder are fair game to test too, though their filenames are longer so they'll be mildly more annoying to test.

Test multiple GPS files. Select option 2 from the main menu, and then select folder "d1". This should only have 2 events that cluster in a group. Select this group (enter "1"). Then select option 3 to analyze it. Repeat this for folder d2, which has over 300 GPX files and will result in several groups. Select a group with numerous rides to see their statistics.

Please review the Design Document for more details about expected functionality. Also, if interested, the Jupyter Notebook GPS_Analyzer.ipynb has some additional considerations after the main code cell.

There is much more going on in the code that was too detailed to include in this summary document. Please contact me if you have questions about testing or the project and I'm happy to explain!

Reflections

It was great to learn so many concepts, both big and little, on this project. Here are my reflections:

- The primary factors for choosing DBSCAN vs. other algorithms were 1) its lack of need to specify the number of clusters as an input parameter, and 2) its speed.
- Calculating steepness (grade) between two points is often errant. To correct for this, I used a minimum distance between points to calculate grade, and then further cut the top and bottom 5% of grades). While admittedly not a perfect approach, it fit with empirical evidence.
- Complexity. This program's biggest bottleneck is likely in the clustering algorithm, which is $O(N^2)$. However, practically, this limitation is minor, because a single athlete can only have a finite number of events.
- I learned quite a bit about using another library (gpxpy). I didn't find it well documented, but sufficiently intuitive.
- I have a greater appreciation for why software is released iteratively. The wish list builds quickly! I feel like I could never stop making enhancements to add new functionality.
- Much code / time was dedicated to edge cases, formatting, error handling, etc., with less to core functionality.
- Several small but important learnings:
    - Sorting a list of objects by one of their attributes
    - The Conversions class I used, and passing values into it
    - Working with the file system, and using relative and absolute file / folder names
    - How *list returns list elements without the list (e.g., for arguments of itemgetter)