Rule-based algorithm to identify GPS location clusters from location data

After some (internal) data transformation, the comprehensive function called *find_cluster* searches within *dist_thres* (function argument measuring the distance in km between locations) and *time_thres* (function argument measuring the time separating two locations, in number of days) of the first chronological location to identify associated points; for instance, all the points corresponding to dist_thres < 0.5 km and time_thres < 0.75 days (18 hours) are regarded by the user as suitable candidates. The first two points meeting the time-space constraints produce a seed cluster from which the geometric center is calculated. The program then sequentially adds points occurring within space and time constraints of the geometric center. The program recalculates the geometric center after each additional point and the process is repeated until no more points can be added. Clusters are allowed to persist beyond the temporal screen provided that the difference between the last point and the next new point is within *time_thres*. The process is repeated, i.e. new clusters are tentatively identified, until all the locations have been used.

After calculations are completed, the function returns two different tables which are also exported as text files in the user's working directory (remember to close them after reading if re-running the program, as these files are being overwritten): 1) a table called *data_and_cluster* containing the original data and some additional information, including to which cluster the location has been ascribed to; and 2) a table called *cluster_center* containing, for each identified cluster, the geometric center (*lat* and *long*), number of locations (*n*) and date and time of the first (*i*) and last (*f*) location of each cluster. Finally, a map is also produced showing on a google map background each individual location (different colors for different clusters) and the location of each cluster's geometric center (as identified on the map by the cluster's number).

After loading in R the libraries, sample data (i.e., bobcat_1) and *find_cluster* function (all the R code is in the 'find_cluster.r' file), you can run a couple examples such as:

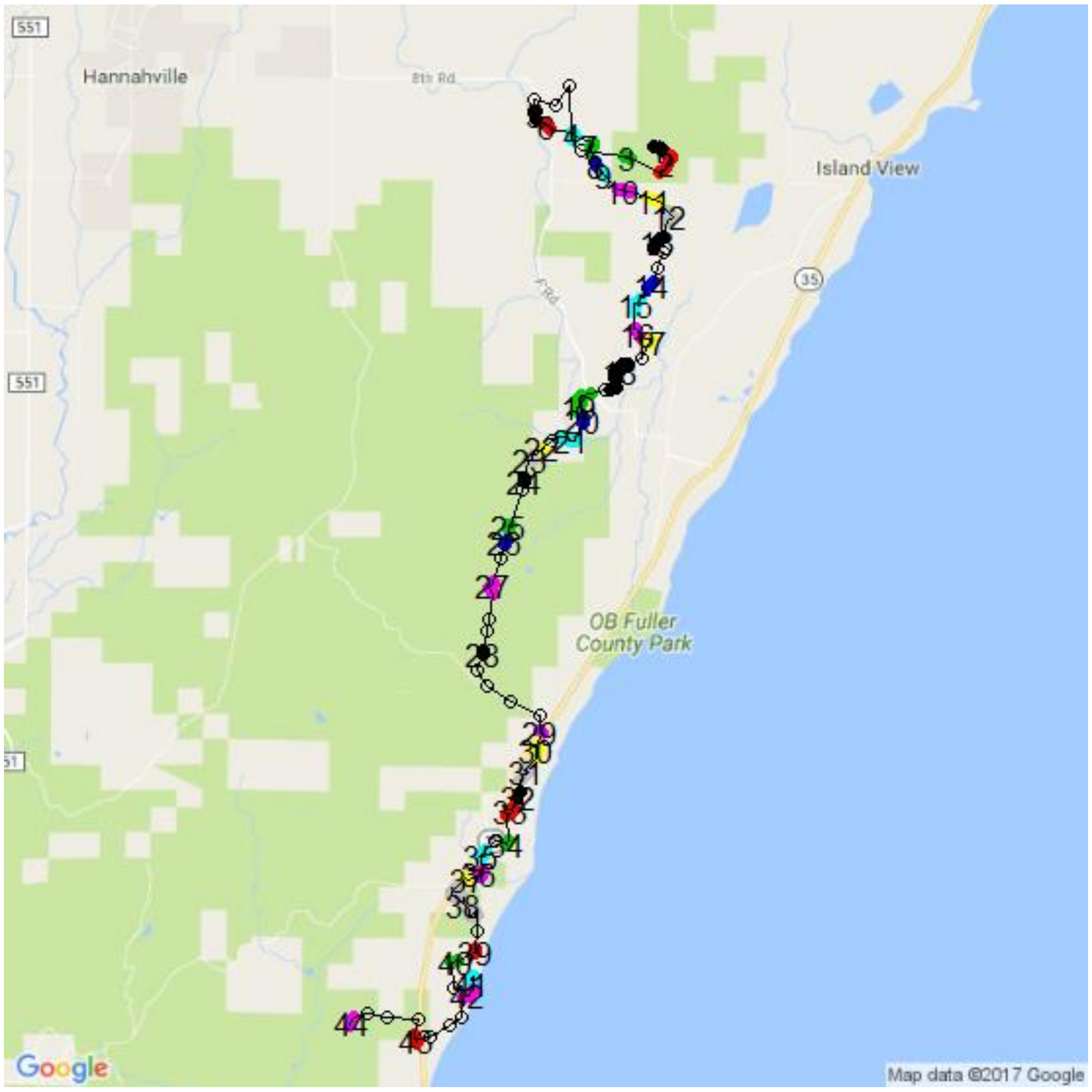
```
test.1 = find_cluster(data = bobcat_1, time_thres = 1, dist_thres = 0.15); test.1
```

The first 3 clusters in the *cluster_center* text file of your working directory (also accessed in R using the command `test.1$cluster_center[1:3,]`) should look like this:

cluster	lat	long	n	Date_i	Time_i	Date_f	Time_f
1	45.64198658	-87.23955508	4	5/16/2009	12:32:22	5/17/2009	8:02:21
2	45.6399599	-87.23805193	7	5/17/2009	16:01:17	5/19/2009	16:01:24
3	45.64079527	-87.24606227	3	5/20/2009	0:00:55	5/20/2009	16:01:49

In addition, you should obtain in R the Figure 1 below showing, overlaid on a Google map, the locations of clusters, i.e. potential kill sites of carnivores (in this case, a bobcat) to be searched for evidence by field crew.

Fig. 1


Hannahville
8th Rd
Island View
OB Fuller County Park
551
35
Google
Map data ©2017 Google