TagTravel

Verson 0.1

Developed by Brice X. Semmens, NRC Research Fellow

Northwest Fisheries Science Center, NOAA

Funded in part by the Lenfest Ocean Program, Pew Foundation

Made available through the Greenboxes code sharing network for biologists:
http://www.ecologybox.org

Introduction:

This program animates the acoustic tagging data based on a system of passive listening stations . The animation can include any number of tags. For instance, you may have conducted a study with 5 tagged fish and an array of 10 listening stations (e.g. VR2 hydrophones). This software will allow you to animate the paths of your 5 tagged fish across 10 listening stations simultaneously. This software shows paths in real-time sped up. In other words, the time-line is preserved, but the user has the option to increase the display speed of that time line.

Each fish included in the data file is randomly assigned a unique color. When there are many fish, these colors can be difficult to tell apart (author's note: this can be fixed, and is something to work on in the future). In order to show the position of multiple fish at the same hydrophone and to show time movement (the screen update at each time step), the program jitters the fish position around the hydrophone location. When more than one fish is at a given hydrophone, their locations can't be simply displayed as the lat/long of the hydrophone position, since they would all precisely overlap and you would only see only one dot. To avoid this problem, the program adds 'random noise' into position of the fish on each display up date. This makes the fish look as if it is bouncing around a hydrophone location, rather than sitting on it. This is also useful as it provides a sense of time passed because the viewer is able see time steps with each jittery update.

In addition to the movement paths of animals, the program 1) approximates day light by lightening and darkening the plot coincident with the time of day, and 2) gives the daily lunar phase by plotting a red line over the lunar phase plot (above the animation plot) to indicate the fullness of the moon.

Time format used:

The software uses the Matlab date format, which is basically a Julian Date. Matlab date numbers represent the number of days that have passed since January 1, 0000. For instance, May 15, 1996, is represented as 729160 in the Matlab software. The difference in dates between the Excel software and the MATLAB software is a constant, 693960 (729160 minus 35200). Hours, minutes, and seconds are all represented as proportions of days.

Data Provided in Text Files:

The main data used in this analysis includes the tagging information by date and hydrophone location. The columns in order are [ID, Lat, Long, Time]. The ID field includes a unique numerical value associated with each tag. The Lat and Long fields should use a consistent coordinate system with the rest of the data inputs (the example uses decimal degrees), and the time field is in Matlab date format. The resolution of the time field (fraction of day) should be at least as small as the time window used in the analysis (see below for a description of the time window). Generally, go as small as possible on the time

resolution; keep in mind, however, that the size of this data file will affect model performance, as the whole thing is read and processed by the program before animation. The data should look as follows:

85	19.686	-80.0075	732687.0506
101	19.68983	-80.0707	732687.0508
18	19.68983	-80.0707	732687.051
95	19.686	-80.0075	732687.0511
91	19.65643	-80.0637	732687.0512
101	19.68983	-80.0707	732687.052
95	19.686	-80.0075	732687.0523
101	19.68983	-80.0707	732687.0536
101	19.68983	-80.0707	732687.0563

Lunar data (moondat.txt):

You will need to provide a table that includes fraction of the moon illuminated (2nd column) each day of the animation time frame (1st column). You can download this information from the following US Navy site: http://aa.usno.navy.mil/data/docs/MoonFraction.php. The data table should be saved as a space delimited file in the directory that the software is in. The data should look as follows:

731949	0.83
731950	0.9
731951	0.95
731952	0.98
731953	1
731954	1

You must provide the program with lunar data from dates at least 10 days before and 10 days after the period of time you intend to animate (this is a requirement in order to show perspective in the lunar cycle—in other words, it is simply an aesthetic requirement).

Map Data:

When you check 'include map?' the program will look for a text file (map.txt) in the program directory (where the .exe file is) to use in making the map. To use this feature you must first create a space delimited text file with latitude in first column and longitude in second. Coordinates system should be identical to the one used to georeference your hydrophones. Each separate polygon should be separated from others in the columns by NaN values (literally, include the text string NaN rather than a number). This forces the plotting program to put line breaks while drawing (e.g. if you want to draw two separate islands). The world vector shoreline database is a great place to find such data- http://rimmer.ngdc.noaa.gov/mgg/coast/wvs.html. The data should look as follows:

-80	19.707
-80.002	19.706
-80.005	19.706
-80.006	19.705
-80.01	19.705

Author's note: This could be a lot better, (e.g. including Sat. images and other data from Arc software) but I do not currently have the Matlab Mapping Toolbox.

GUI Model Inputs:

Start Date (Matlab Format):

You must provide the program with the data you wish to begin the animation on. Obviously, this date must be included in the data you provide the program. This data must be a round day (no decimal places), and must be in Matlab date format. See the introduction section for details on how to turn dates into Matlab date format.

Animate how many days?

Tell the program how many days you want it to run over. For instance, if you enter 10, the program will animate 10 days worth of data starting with the start date specified.

Sliding Window Size:

In order to put all tagged individuals in the datafile into a common time currency for animation, the program processes the data into time bins before animation. The size of these bins is user defined. For instance, if you used a Time Window of 5 hours, the program creates a time line with 5 hour bins across the entire defined time frame, and then assigns tags to the hydrophones they are heard at (if any) during each of the time bins. If a tag is not heard during a time bin, the tag will not displayed during that time step of the animation. **If a tag is heard at more than one hydrophone during that time bin, the animation will draw arrows between these hydrophones in order to depict movement direction**. The size of the time window, entered in the Time Window field, basically adjusts t the length of arrow trails as a function of time. Thus, the size of the time window is critical to showing movement, and care should be taken to adjust this value in order to highlight the most interesting characteristics of movement. The size of the time window is also vital for showing coincident movement (eg-are tagged animals moving together?).

Refresh Rate:

You must provide amount of time you want each video frame to show on the screen before moving on to the next. I have found that 0.5 second updates is a good rate of speed. Note that at faster and faster speeds (smaller update times) the program bumps up against computing power. In the case of my laptop, I can't get the program to refresh the screen more than about 0.25 seconds. I am not sure why you would want to go much faster than this anyway.

Time Step Size Between Updates:

The time step size defines the jump size of the moving window at each time step. For instance, if we are using a 2 hour step size and window of 4 hours, the program will display all the tag information heard during a 4 hour period, and then at the next refresh, will shift the window forward 2 hours, and again display the 4 hours of data at that new time step. The size of the time step will obviously speed up the the rate at which the program moves through chronological data. Be careful, however, as steps too large (particularly in relation to the window size) can result in skipped information.

Future things:

This program is REALLY rough. There are virtually no error messages or catches programmed into it right now. Which, basically, means that if you aren't doing it right, it won't go and it won't tell you why. This is not ideal. I need to fix this. Also, as mentioned previously, I don' t have the Matlab mapping toolbox, which would allow for much more sophisticated spatial data incorporation. If someone buys it for me, I'll include it.

There are an endless number of ways to extend this program; lots of work for the future.