**CalcVitesseVerticale\_final.m (Matlab code by Katell Guizien)**

**Summary**

This code takes each of the downloaded HYCOM netCDF files and calculates vertical velocities using horizontal velocities (northward and eastward velocity) from each of the available depths based on the principle of mass conservation.

**Input**

netCDF files for each 3-hourly time-step downloaded from HYCOM. Missing time-steps were filled by averaging data from the previous and next time-step (outside of this code). The data consists of filenames in the format nest\_1\_yyyymmddhhmmss.nc

**Details**

Nothing specific apart from what is mentioned in the summary. Only that vertical velocity is calculated only for depths upto 400 m (until the 27th depth level from the surface) – beyond this there would be anomalies in the estimates.

**Output**

A new file with the vertical velocity is created separately (with the suffix ‘\_w’), along with a file with other variables as in the input file but with only data for the first 27 depth levels.

**code0\_releaseFile – Create release file for CMS input (R code by Bharti Dharapuram)**

**Summary**

CMS needs an input file which has information on the date and time of release of a particle, the depth of release and the coordinates from which the particle is released. For our analysis, we released a particle from each of the release locations every three hours at 1 m depth. The particle was carried out in this manner for a period of ten days, and the time for which the released particles were tracked was for 120 days.

There is some code in the end to check whether the release file has been created corresctly by checking the number of rows of data, the last run date and the number of unique time steps of release.

**Input**

1. release\_wgs\_final.csv – it is a list of coordinates of release locations. Release locations were created at intervals of 5 km along the coastline and were enclosed within polygons, such that each polygon had roughly 4 release locations within it.

**Details**

Functions from the package lubridate are used to generate the regular time series needed in the CMS release file.

For each of the 10-day long release files, the code needs to be initiated with the first date of release.

**Output**

1. Text file named releaseFile\_dd\_mm\_yyyy, used as input to run CMS simulations

**code1\_extract\_pld – Extract particle trajectory information for PLD of interest (R code by Bharti Dharapuram)**

**Summary**

The CMS output is a netCDF format file (with the extension .nc), which is a data format very commonly used for oceanographic data or heavy geospatial data. In this code we use the particle trajectory information obtained as output from the CMS simulations and extract the position of a particle at different time points since release, corresponding to different pelagic larval durations we are interested in. The netCDF output has various kinds of information listed, of which we extract the depth, latitude, longitude, exit status (relevant when the trajectory was stopped for some reason – refer to CMS manual for more information), release location id and the (cumulative) distance travelled by a particle.

**Input**

Trajectory file named in the format traj\_file\_yyyymmdd.nc

**Details**

The PLD values for which information needs to be extracted has to be specified in the pld.day argument. Apart from the arguments extracted here, additionally salinity and temperature information can also be extracted if the ocean data used to run the simulations contains the data.

**Output**

1. csv data corresponding to each trajectory file named man\_op\_traj\_file\_yyyymmdd.csv

2. yyyy\_mon\_man\_dat.all.rds (where monsoon is the season; ne: north-east monsoon, sw: south-west monsoon): combines all the above files corresponding to different trajectory outputs within a year and a monsoon as a single R data file (extension .rds), so that the data is easy to access for downstream analysis.

**code2\_calculate\_transfer\_probability – Create connectivity matrices from particle trajectories, build connectivity networks and calculate network indices (R code by Bharti Dharapuram)**

**Summary**

This code takes the particle trajectory output (subset for the PLD of interest in the above code) and creates connectivity matrices by overlaying points over coastal polygons to detect successful dispersal (when a particle is present within any of the pre-defined coastal polygons).

The centrality indices calculated using each connectivity matrix are strength, degree, clustering coefficient and bridging coefficient.

**Input**

Subset trajectory data for a certain year and monsoon stored as yyyy\_mon\_man\_dat.all.rds

**Details**

For particles which are still active or moving (not moved outside of model domain or have reached land), the identity of the source polygon from where the particle originated and the identity of the destination polygon where it is present at the end of the PLD are used to build the connectivity matrices. Particles that are present outside of the coastal polygons are considered lost.

Counts of connectivity between polygons in the matrix are converted into transfer probabilities by dividing it by the area of the recipient polygon and the total number of particles released by the source polygon (to account for differences in the number of particles released from each polygon and differences in the area between polygons).

The connectivity matrices across the PLD range (2-50 days) are combined into four PLD classes – 2-4, 6-12, 14-20 and 22-50 days, based on the distribution of PLD among coastal marine invertebrates found along the Indian coastline/Indian Ocean (see Supplementary Material). Each of these PLD-classes consists of an array of multiple connectivity matrices corresponding to PLDs within the range of interest in intervals of every 2 days. For example, the PLD-class 6-12 days consists of connectivity matrices corresponding to the PLDs – 6, 8, 10 and 12 days. Element-wise mean is calculated across the array to give rise to a single connectivity matrix for each PLD class. Connectance is calculated for each connectivity matrix.

The PLD-class connectivity matrix is used for community detection using the Infomap algorithm and a coherence ratio is calculated for each community to measure the proportion of particles released from a community that are retained within it for that PLD-class. Summary of the connectivity network and centrality measures (strength, degree, bridging centrality, closeness centrality, clustering coefficient and bridging centrality are calculated.

**Output**

1. tp\_yyyy\_mon\_sim\_xx.csv (where xx is the PLD in days and sim is simulation number ranging from 1-12 corresponding to different spawning periods): connectivity matrix corresponding to each PLD class for each year, season and simulation (there are 12 simulations corresponding to 12 different spawning/particle release times within an year).

2. sum\_infomap\_yyyy\_mon.csv: a dataframe which consists of the summary of community detection across all simulations and PLD-classes within a certain year and monsoonal season. It consists of columns year, season, sim (simulation number – corresponding to a spawning period), pld.class (each of the four classes – 2-4 days, 6-12 days, 14-20 days and 22-50 days, clusters (total number of Infomap communities detected), clusters2 (number of Infomap communities with >1 node (coastal polygon) membership and columns numbered for 1-61, 68-534 (these are coastal polygon ids), con (connectance), con.prop (proportion connectance). The entries corresponding to columns representing coastal polygon ids are the community membership code given to each coastal polygon.

3. close\_cen\_in\_yyyy\_mon.csv: Closeness centrality in

4. close\_cen\_out\_yyyy\_mon.csv: Closeness centrality out

5. str\_cen\_in\_yyyy\_mon.csv: In-strength in

6. str\_cen\_out\_yyyy\_mon.csv: Out-strength

7. deg\_cen\_in\_yyyy\_mon.csv: In-degree

8. deg\_cen\_out\_yyyy\_mon.csv: Out-degree

9. trn\_cen\_yyyy\_mon.csv: Clustering coefficient

10. brid\_cen\_yyyy\_mon.csv: Bridging centrality

11. All the above centrality measures and network indices representing different years and monsoons are put together into a single dataframe and saved with the ending ‘\_raw.csv’ for ease of access

**Note:** Betweenness centrality and Closeness centrality (for weak and strong subgraphs) was calculated using the Matlab script betweenness\_w.m and calc\_subgraphs (\_weak.m and \_strong.m) respectively using Costa’s Tattoo toolbox. The R code for the other centrality measures described here was adapted based on the code from the Tattoo toolbox. Results from running the Matlab and R scripts were compared to make sure the same results were obtained.

**code3\_community\_boundaries – Intersects spatial extents of Infomap communities with the coastlibe to detect boundaries of these communities (R code by Bharti Dharapuram)**

**Summary**

**Input**

1. sum\_infomap\_all.csv – dataframe where community membership details of each coastal polygon is saved (output of the code above)

2. all\_buf\_line\_wgs\_final\_26Jul19 – coastline shapefile (this was also used to create the release locations)

**Details**

Creating a single spatial polygon (single polygon with adjacent boundaries merged) using an input shapefile where each polygon is listed as a separate element. This is first intersected with the coastline and any boundary points recovered are saved. This is done, as there are gaps between some polygons, which would lead to the generation of spurious boundary points when intersected with the coastline.

We only use communities for which the rho value (refer details in the previous code) is greater than 0.5. For each row of community membership data from sum.infomap.all, communities with rho>0.5 are subsetted. For each community, polygons sharing membership are merged to generate a single large polygon. This large polygon is intersected with the coastline to obtain coastline fragments – the end points of each of these fragments are saved. Since each edge is shared by two polygons, taking unique points into consideration

**Output**

1. infomap\_yyyy\_mon\_pldclass.csv (where PLD class refers to the PLD-class for which Infomap results were obtained)

2. All the above results are combined as a single file names infomap\_cuts\_all.csv