**CLIMATE STUFF**

**Apologies for these might be a little TECHNICAL!**

1. Identify the main working objective: CLIMATE RISK and deduce a framework for its calculations. We used the modified IPCC risk framework (HAZARD \* EXPOSURE \* VULNERABILITY).
2. Determine climate hazard metrics, scenarios, variables, and periods, carefully considering local-level relevance.
3. Identify agroecological exposures (8 variables): crop cover, mangrove cover, seagrass, coral cover, and four functional diversity metrics (Steph’s Paper).
4. Consider the STUDY EXTENT and SPATIAL RESOLUTION at which data will be harmonised. Here, all data is harmonised at a ~25 km grid along the coast (Note that all rasters were re-grided at this resolution beforehand). A global coastline shapefile was downloaded from *NaturalEarth* and cropped to WIO extent using the bounding box of the WIO countries (use the country\_shape.shp by Maina et al. dropbox). Create a 25km buffer at either side of the coastline shapefile. Using this output, generate a Grid Index Feature. Keep the PageName as IDs throughout the analysis. This spatial analysis was conducted in ArcGIS Pro, but codes for doing the same analysis are archived in the work folder.
5. For each grid feature, summarise the crop cover, mangroves, seagrass, and corals. The simplest approach is to convert these covers into raster using ArcGIS’ “Polygon to Raster” functions. For this step, set the preferred resolution for processing. I used a conservative 100m resolution to potentially obtain smooth edges of the resulting rasters. Next, using the Spatial Analyst Tool, “Tabulate Area’, calculate the coverage of each system within a defined grid. The Tabulate Area functions will return a spreadsheet, so join the back to the Grid Feature using the “Join and Relate” function.
6. Time to link the four functional metrics. Steph’s outputs (“functional data”) are available as RDS, which can only be read in R. Import both functional data and Grid Feature into R. The functional data contains xy coordinates, which means we can convert these into a spatial object using the ***sf*** functions. Afterwards, intersect the spatial object and the grid feature. Next, group by the “PageName” and find the median of key metrics, including FEve, FRic, Nb\_sp, and FDiv. Merge back to the Grid Feature and keep backup on the local drive.
7. Import the NetCDF file for each climate variable (here, Majambo had processed the multimodal ensembles).
8. Calculate metrics using already developed key functions developed, including slope, tx9p, R10p, txInt, normalise, etc.
9. From our conceptualisation, some variables are land variables only, while others are ocean variables only. This means extracting raster values to Grid Feature will generate NAs. So, we use the knnLookup functions of the SearchTree library to fill near-shore values for the land variables or near-land values for the sea variables.
10. For each scenario and period, we then apply the quantile-transform functions to rescale between 0 and 1. We did the same for the exposures.
11. Estimate the mean and standard deviation for (1) Hazards (i.e., across the 14 standardised scores of the climate metrics for each scenario and period) and (2) Exposure (i.e., across the eight standardised scores of exposures).
12. Generate inverse variance function weights for both domains.
13. Estimate the climate change impacts for each grid as (HAZARD\*w1) \* (EXPOSURE\*w2) divided by w1+w2.
14. Think about the village-level data (at this stage, the methods had already been implemented, and village-level aggregate is available).
15. Import into R and convert a spatial object. Intersect both and assign grid IDs to villages.
16. To estimate climate impacts at the village level, use the “inverse distance weighting methods” (see the main document for justification).
17. To achieve this, create a matrix that depicts all grids as neighbours (use the grid IDs; i.e., imp.matrix = expand.grid(source=grid$ID, neighbour=grid$ID). Merge the xy coordinates to the source and neighbour grids. Now calculate Euclidean Distances between source and neighbours using the **sf** functions. Use parallel processing to facilitate these. Remove self-intersections (i.e., those with distance == 0 meters) and keep copy on local drive.