Assemblage Dispersion Fields

Patrick Alexander Walkden

03/08/2021

PREDICTS Assembly

Assemblage dispersion fields

There are many decisions that need to be made when performing an analysis of trait based community assembly. The functional diversity metric to use, how to randomise the trait and community matrices to produce null models in the absence of a particular process, and, what we are concerned with here, how to define the species pool which species in this null community are drawn.

Defining the species pool correctly is integral in any analysis of community assembly and it can be thought of as the pool of species that could potentially disperse to and colonise a focal site in the absence of all other assembly processes, be it environmental filtering or biotic interactions such as competition or limiting similarity. The definition of the species pool affects the degree and thus the interpretation of subsequent results.

There have been a number of ways suggested to define the species pool. Most simply and commonly used is defining the species pool as all the species observed within a study across all sites. This works adequately when researchers have undertaken their studies with the explicit focus on testing community assembly processes and surveyed communities across an suitably broad environmental gradient. However, using data from PREDICTS studies rarely collect data across the entire land-use gradient therefore by using this method our species pool will always be restricted to those species that are able to persist in the land-use gradient surveyed rather than the entire species pool. Additionally, a lot of what is termed as "dark diversity" (rare species) can be missed.

Therefore another method proposed to define the species pool is by using biogeographic data such as species range maps to calculate an assemblage dispersion field (ADF; Graves & Rahbek., 2005) centered on the focal site. An ADF considers the species pool to be all species whose range overlaps the range of any species within the focal site. This can often produce excessively large species pools, especially when some species have expansive ranges, therefore a threshold can be introduced so cells which contain ~60% of the same species as the focal cell are included in the species pool thus creating a more restrictive pool. This has the benefit over the sampled species method as it is not restricted to only considering the species that have been observed within a study and can facilitate the testing of community assembly process on studies that survey a narrow environmental gradient in tandem with the many other studies within PREDICTS.

I will also be looking at the sensitivity of any results to the threshold of the ADF by having a spectrum of species pools for each site. The most restrictive being, only considering the species whose range maps overlaps with the focal site, then increasing the radius of the ADF to the immediate cells around the sites and a slowly decreasing threshold of the ADF to a minimum of 60% similarity.

Species-site overlap

Sooooo first things first calculate the first species pool, those whose ranges just overlap with the focal site.

```r
rm(list = ls())

library(tidyverse) ## for wrangling and general data handling
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(maptools) # mapping
```

```
## Warning: package 'maptools' was built under R version 4.0.5

## Loading required package: sp

## Warning: package 'sp' was built under R version 4.0.5

## Checking rgeos availability: TRUE
```

```r
library(rgdal) # mapping
```

```
## Warning: package 'rgdal' was built under R version 4.0.5

## rgdal: version: 1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.2.1, released 2020/12/29
## Path to GDAL shared files: C:/Users/patri/R/win-library/4.0/rgdal/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 7.2.1, January 1st, 2021, [PJ_VERSION: 721]
## Path to PROJ shared files: C:/Users/patri/R/win-library/4.0/rgdal/proj
## PROJ CDN enabled: FALSE
## Linking to sp version:1.4-5
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading rgdal.
## Overwritten PROJ_LIB was C:/Users/patri/R/win-library/4.0/rgdal/proj
```

```
library(sp) ## mapping
library(raster) ## mapping
```

```
## Warning: package 'raster' was built under R version 4.0.5
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(rgeos) ## mapping
```

```
## rgeos version: 0.5-5, (SVN revision 640)
##  GEOS runtime version: 3.8.0-CAPI-1.13.1
##  Linking to sp version: 1.4-5
##  Polygon checking: TRUE
```

```
require(sf) ## mapping
```

```
## Loading required package: sf
```

```
## Warning: package 'sf' was built under R version 4.0.5
```

```
## Linking to GEOS 3.9.0, GDAL 3.2.1, PROJ 7.2.1
```

```
require(doParallel) ## parallelisation
```

```
## Loading required package: doParallel
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loading required package: iterators

## Loading required package: parallel
```

```
require(foreach) ## parallelisation
require(fasterize)
```

```
## Loading required package: fasterize

## Warning: package 'fasterize' was built under R version 4.0.5

##
## Attaching package: 'fasterize'

## The following object is masked from 'package:graphics':
##
##     plot

## The following object is masked from 'package:base':
##
##     plot
```

```
### load in the refined PREDICTS dataset

PREDICTS <- readRDS("Outputs/refined_predicts.rds")
```

I am going to extract the coordinates from each of the study sites and using maps from Birdlife extract all those species that overlap

Now equipped with the species maps I am going to collate all the species whose ranges overlap with each site to generate the most restrictive species pool.

```
memory.limit(120000)
```

```
## [1] 120000
```

```
### register the 8 cores to use while looping

#  registerDoParallel(cores = 8)
#
#  site_spp_matrix <- foreach(range = species_files,  ## for each range file
#                             .packages = c("tidyverse","maptools","rgdal","sp","raster","rgeos","sf"),
#                             .combine = "cbind", ## how I want the output combined - each iteration wi
#                             .inorder = FALSE) %dopar% {   # FALSE speeds things up
#
#    load(range) ## load in range
```

```
#
#
#     #### filter data so that distribution polygons are those that represent extant, probably extant a
#                       #### native, reintroduced and introduced origin and those that are resident, br
#       sp_name <- as.character(data$binomial[1])
#
#     data <- data %>% dplyr::filter(presence %in% c(1,2,3), origin %in% c(1,2,3), seasonal %in% c(1,2,
#
#     if(nrow(data) == 0){
#        mat <- matrix(rep(NA,nrow(sites)),nrow = nrow(sites), ncol = 1)
#    colnames(mat) <- sp_name
#    rownames(mat) <- as.character(sites$SSBS)
#
#    return(mat)
#     } else {
#
#
#    ## Sometimes the polygons are in funny classes so coerce into MULTIPOLYGON class
#
#    if(any(class(data$Shape)[1] == "sfc_MULTISURFACE", class(data$Shape)[1] == "sfc_GEOMETRY")){
#      for(k in 1:NROW(data)){
#        data$Shape[[k]] <- st_cast(data$Shape[[k]], "MULTIPOLYGON")
#      }
#    }
#
#    #combine all polygons together
#
#    shape <- as_Spatial(st_combine(data$Shape))
#
#    ##create a blank matrix species as columns, sites as rows
#
#  mat <- matrix(rep(NA,nrow(sites)),nrow = nrow(sites), ncol = 1)
#    colnames(mat) <- as.character(data$binomial)[1]
#    rownames(mat) <- as.character(sites$SSBS)
#
#    ### iterate through sites
#
#    for(j in 1:nrow(sites)){
#  print(j)
#      ## calculate overlap this will produce a value of zero if the site lies within it's range
#
#      Overlap <- suppressWarnings(gDistance(site_coords[j],shape))
#
#      ### if Overlap = 0 add a 1 to the site indicating presence else a 0
#
#      mat[j,1] <- ifelse(Overlap == 0, 1, 0)
#
#    }
#
#  return(mat)
#     }
#  }
#
```

```r
#  ### close all conenctions
#
#  registerDoSEQ()
#  closeAllConnections()
#
#  ### save
#
# write_rds("Outputs/site_spp_matrix.rds",x =  site_spp_matrix)
site_spp_matrix <- readRDS("Outputs/site_spp_matrix.rds")

site_species_richness <-data.frame(Site = rownames(site_spp_matrix), spp_rich = rowSums(site_spp_matrix
species_occurences <- data.frame(Species = colnames(site_spp_matrix), occurrences = colSums(site_spp_ma


## the number of species just overlapping the site ranges from 59 in Sao Tome and 591 in Uganda - howeve

head(site_species_richness)
```

```
##                                             Site spp_rich
## AD1_2012__Yamaura 1  1 AD1_2012__Yamaura 1  1      164
## AD1_2012__Yamaura 1  2 AD1_2012__Yamaura 1  2      164
## AD1_2012__Yamaura 1  3 AD1_2012__Yamaura 1  3      163
## AD1_2012__Yamaura 1  4 AD1_2012__Yamaura 1  4      162
## AD1_2012__Yamaura 1  5 AD1_2012__Yamaura 1  5      163
## AD1_2012__Yamaura 1  6 AD1_2012__Yamaura 1  6      164
```

```r
## the most commonly occurring birds is Falco peregrinus almost ubiquitously distributed across all site

head(species_occurences)
```

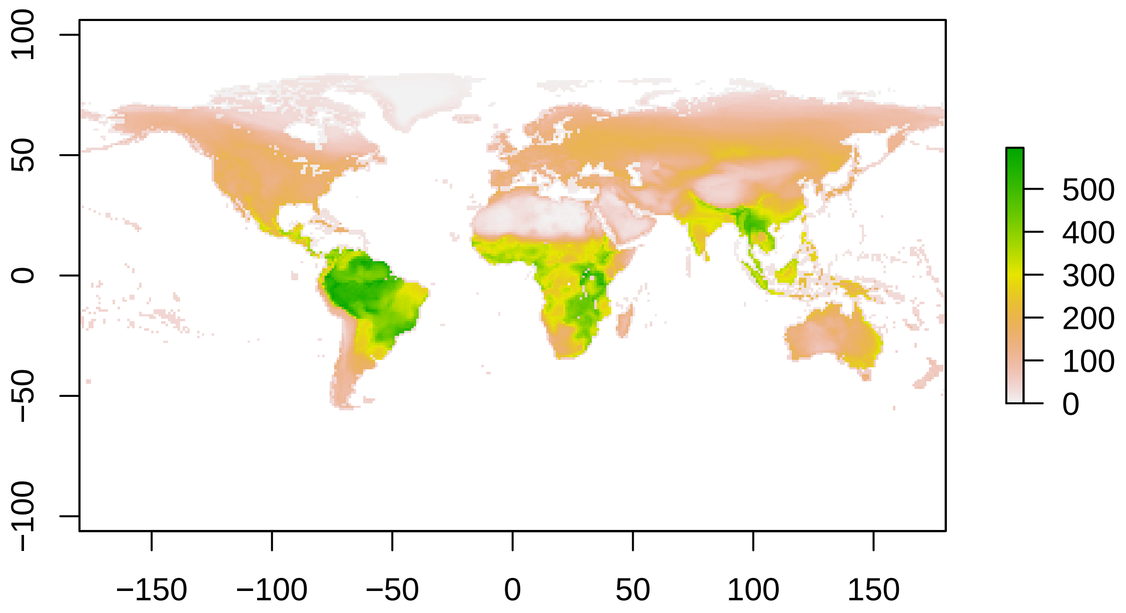
```
##                                              Species occurrences
## Abeillia abeillei               Abeillia abeillei           16
## Abroscopus albogularis     Abroscopus albogularis           32
## Abroscopus schisticeps     Abroscopus schisticeps           18
## Abroscopus superciliaris Abroscopus superciliaris          142
## Aburria aburri                     Aburria aburri            0
## Acanthagenys rufogularis Acanthagenys rufogularis           43
```

Assemblage Dispersion Fields

Assemblage Dispersion Fields (ADF) are typically calculated on a 1 degree resolution raster map Borregaard et al, 2020 so first I am going to create a global map of avian diversity by converting species distribution maps into presence absence cells globally. This can be done by "rasterising" the polygons and then summing all the species rasters.

The rationale for including species with an assemblage dispersion field in the potential species pool for a focal site is that if a good proportion of species from the focal site have been able to disperse to the site from a grid cell then the remainder of the species within the target cell should be able to disperse to the focal cell. This has some very obvious caveats that we will address a little later including differential dispersal capabilities of species and environmental affinities (Lessard et al, 2012).

```
#### load in a 1 deg raster map that can be converted into a blank raster

#### get a blank world map at a one degree resolution -- approx 100km2 at the equator

blank <- raster("../../Datasets/Environmental_Variables/gpw_v4_population_density_adjusted_to_2015_unwpp

### reclassify all values that aren't NA - and therefore terrestiral to 0

blank <- reclassify(blank, c(-Inf,Inf,0))

#### write blank raster map for future use

write_rds(x = blank, file = "Outputs/blank_map_1deg.rds")

#########################
#########################


# registerDoParallel(cores = 8)
#
# map_data <- foreach(range = species_files,
#                     .combine = "rbind",
#                     .packages = c("tidyverse", "maptools", "rgdal", "sp", "raster","rgeos","sf","fast
#                     .inorder = FALSE) %dopar% {
#
#
#                       load(range)
#
#                       #### filter data so that distirbution polygons are those that reprsent extant, 
#                       #### native, reintroduced and introduced origin and those that are resident, br
#
#
#                       data <- data %>% dplyr::filter(presence %in% c(1,2,3), origin %in% c(1,2,3), se
#
#
#                           if(nrow(data) == 0){
#                             load(range)
#       mat <- matrix(blank@data@values, nrow = 1, ncol = length(blank@data@values))
#       rownames(mat) <- data$binomial[1]
#       colnames(mat) <- 1:length(blank@data@values)
#       return(mat)
#     } else {
#
#
#
#                       ##### funny thing with some ranges so need to convert back to type MULTIPOLYGON
#
#                       if(any(class(data$Shape)[1] == "sfc_MULTISURFACE", class(data$Shape)[1] == "sfc
#                         for(k in 1:NROW(data)){
#                           data$Shape[[k]] <- st_cast(data$Shape[[k]], "MULTIPOLYGON")
#                         }
#                       }
#
#                         #combine all polygons together and convert to a sf class polygon
#
#                         shape <- as_Spatial(st_combine(data$Shape))
#                         shape <- st_as_sf(shape)
#
#                         ### convert polygon into a presence raster on the blank world map
#
#                         ras <- fasterize(sf = shape, raster = blank,fun = "count")
#
#                         #### extract cells which overlap with the polygon and exclude marine and oceani
#                         #### that has non-terrestrial cells as NA
#
#                         ras_data <- (blank + ras)
#
#
#                         #### create matrix where the row is the species and each column is a cell in th
#
#
#                             mat <- matrix(ras_data, nrow = 1, ncol = length(ras_data))
#                         colnames(mat) <- 1:length(ras_data)
#                         rownames(mat) <- data$binomial[1]
#
#                         return(mat)
# }
#                       }
#
# registerDoSEQ()
# closeAllConnections()
#
# ## save output map data that can then be put given to the blank raster
# write_rds(x= map_data, file = "Outputs/raster_species_matrix.rds")
map_data <- readRDS("Outputs/raster_species_matrix.rds")

spp_rich_df <- data.frame(Cell = colnames(map_data), spp_rich = colSums(map_data, na.rm = TRUE))

diversity <- blank

diversity@data@values <- (blank@data@values + spp_rich_df$spp_rich)

plot(diversity)
```



With this diversity matrix with the species as rows and cells as columns we can then can the similarity between any two cells. Using this principle I can then begin to calculate the assemblage dispersion fields for each of the sites within PREDICTS. First by locating which cell each site is in and then calculating the similarity of every other cell.

```
########################################################
###### PREDICTS SITES ASSEMBLAGE SIMILARITY MATRIX #####
########################################################

#### Which cell is each site located in

sites$cell <- raster::cellFromXY(blank, sites[,c("Longitude", "Latitude")])


## This loop is going to create a matrix with rows being the cell in which a site is located and column
## the value of each cell will the proportion of species shared between each cell

site_cells <- unique(sites$cell)



adf_sim_func <- function(cells){

                  mat <- matrix(rep(NA,ncol(map_data)), nrow = 1, ncol = ncol(map_data))
```

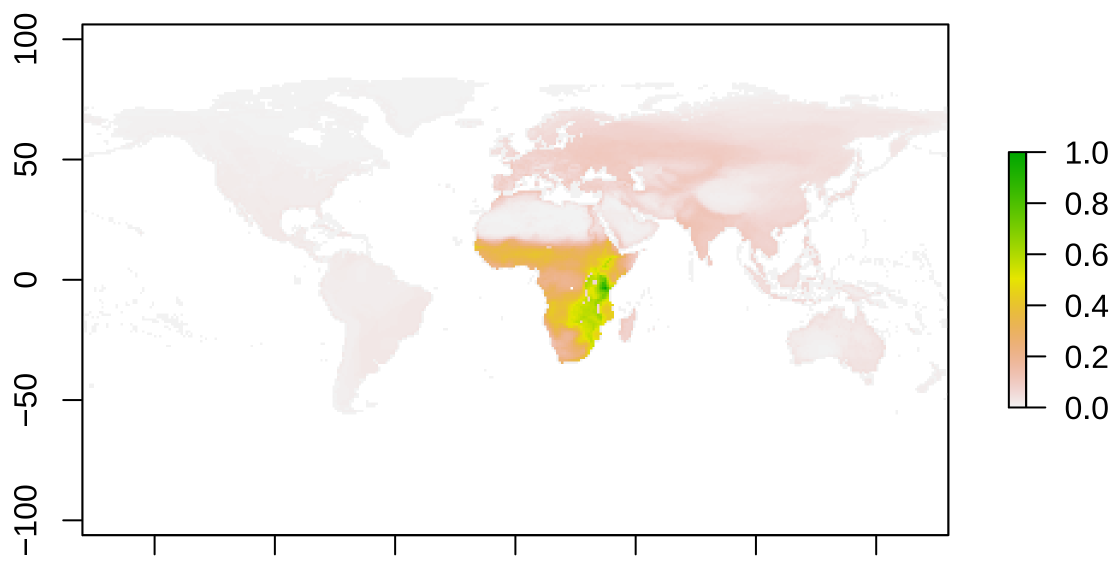
```r
  ### This bit is kinda redundant when looking just at sites but if we were
  ### the target cell had no species present within - this includes marine

  if(!any(!is.na(as.numeric(map_data[,cells])))){
    rownames(mat) <- cells
    colnames(mat) <- 1:ncol(map_data)
  } else {


    ## extract species that are present within a specified cell

    focal_spp <- names(which(map_data[,cells] == 1))

    ### get all other cells minus the focal cell

    other_cells <- c(1:(cells-1),(cells+1):ncol(map_data))

    ### create blank matrix

    colnames(mat) <- 1:ncol(map_data)

    ## cell shares all species with itself

    mat[,cells] <- 1
    rownames(mat) <- cells

    ### for all other cells extract species and calculate proportion of pse

    for(other in other_cells){

      ### if there are no species that aren't classified as NA in the map d

      if(!any(!is.na(as.numeric(map_data[,other])))){
        prop <- NA
      } else {

        ## extract the species with in the other cell

        other_spp <- names(which(map_data[,other] == 1))

        ### if the cell is terrestrial but has no species in it then the ot

        if(is_empty(other_spp)){
          other_spp <- "none"
        }

        ### how many species are shared between the two cells

        sim <- length(which(focal_spp %in% other_spp))

        # if none similarity is 0
```

```
                                    prop <- ifelse(sim != 0, sim/length(focal_spp), 0)


                                  }

                                  ### input the data into the matrix

                                  mat[,other] <- prop
                                }

                              }

                              return(mat)
               }

# registerDoParallel(cores = 5)
#
# site_cell_matrix <- foreach(cell = site_cells,
#                              .combine = "rbind",
#                              .packages = c("tidyverse", "raster", "maptools", "rgeos", "sf", "sp"),
#                              .inorder = FALSE) %dopar% {
#
#                 adf_sim <- adf_sim_func(cells = cell)
#
#
#                              }
#
# registerDoSEQ()
# closeAllConnections()
#

#  write_rds(file = "Outputs/predicts_site_cell_matrix.rds", x = site_cell_matrix)
 site_cell_matrix <- readRDS("Outputs/predicts_site_cell_matrix.rds")
```

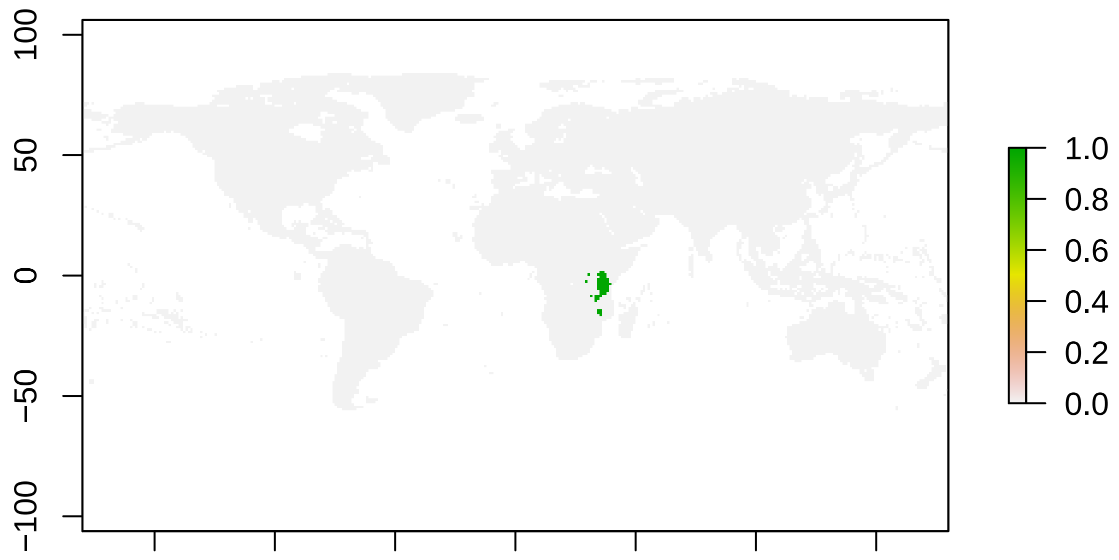
The resulting matrix will have the proportion of species shared between the focal cell. So now we can begin to visulaise the shapes of these dispersion fields and the area from which we are going to be extracting our species pool from.

```
#dir.create("assemblage_dispersion_fields")

for(i in 1:nrow(site_cell_matrix)){

  ### get the studies which have sites within this cell

  study <- sites %>% filter(cell == rownames(site_cell_matrix)[i]) %>% distinct(SS) %>% pull()
  study <- paste(study,collapse = "&")

  # create a directory

  dir <- paste("assemblage_dispersion_fields/",paste(study,i,sep = "_"), sep = "")

 #dir.create(dir)
```

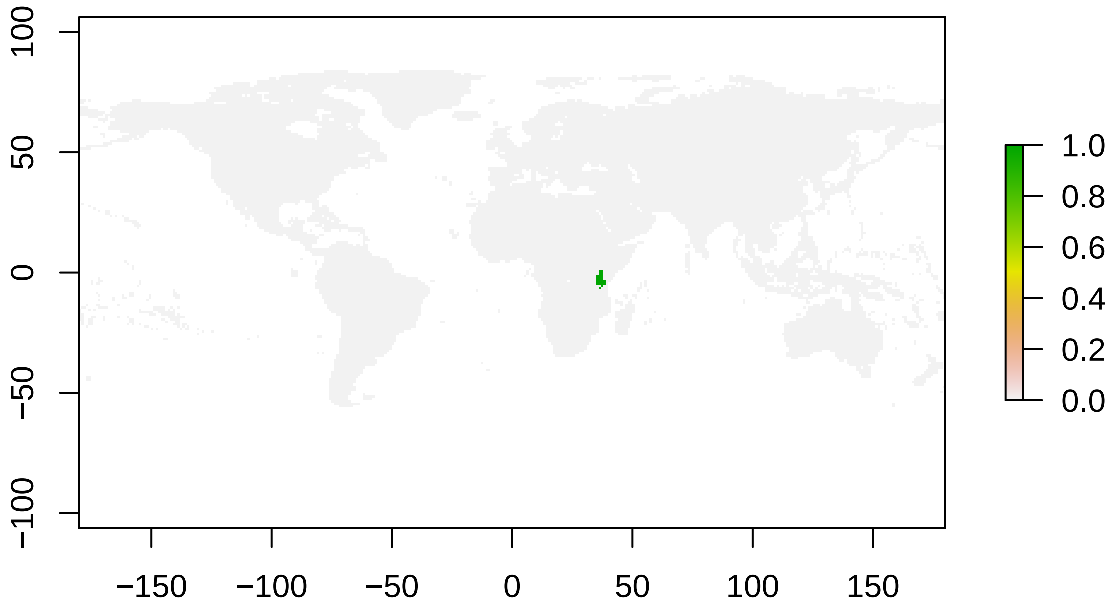
```
  ## get each of the cell similarity scores

  adf <- site_cell_matrix[i,]

  ## input into blank raster

adf_map <- blank

adf_map@data@values <- adf

### save the raw similarity scores

png(paste(dir,"/","adf.png",sep = ""))
plot(adf_map)
dev.off()

### which cells are above the threshold for similarity

for(sim in c(0.9,0.8,0.7,0.6)){

  ## save the resulting map indicating the cells whose species will be included in the species pool.

  png(paste(dir,"/",sim,".png",sep = ""))
  plot(adf_map > sim)
  dev.off()
}

}

plot(adf_map)
```



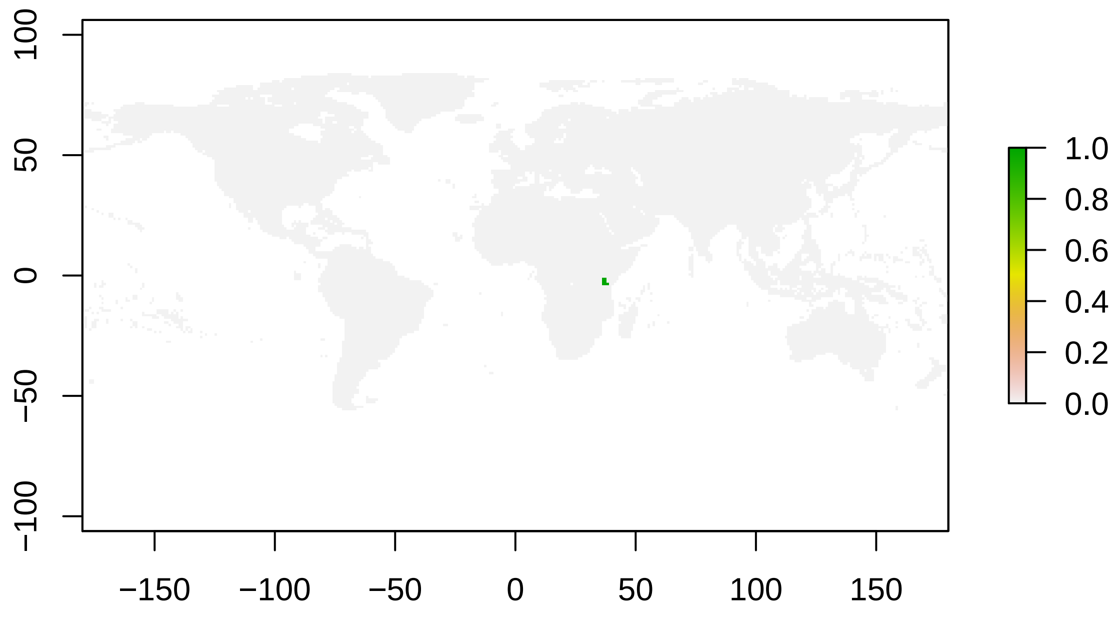
```
plot(adf_map > 0.6)
```



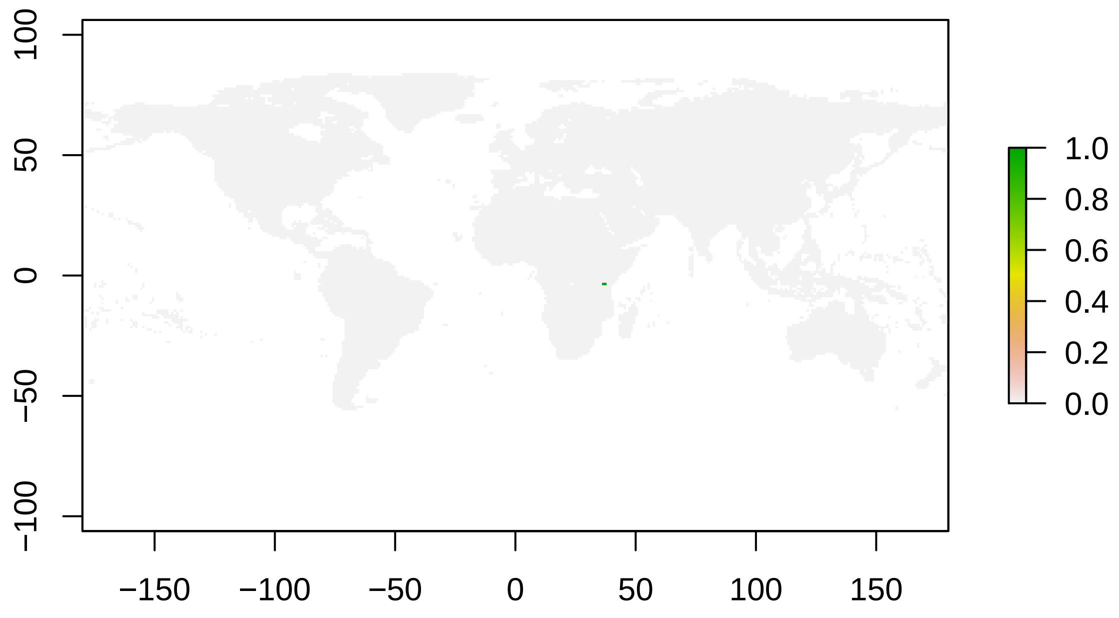
```
plot(adf_map > 0.7)
```



```r
plot(adf_map > 0.8)
```

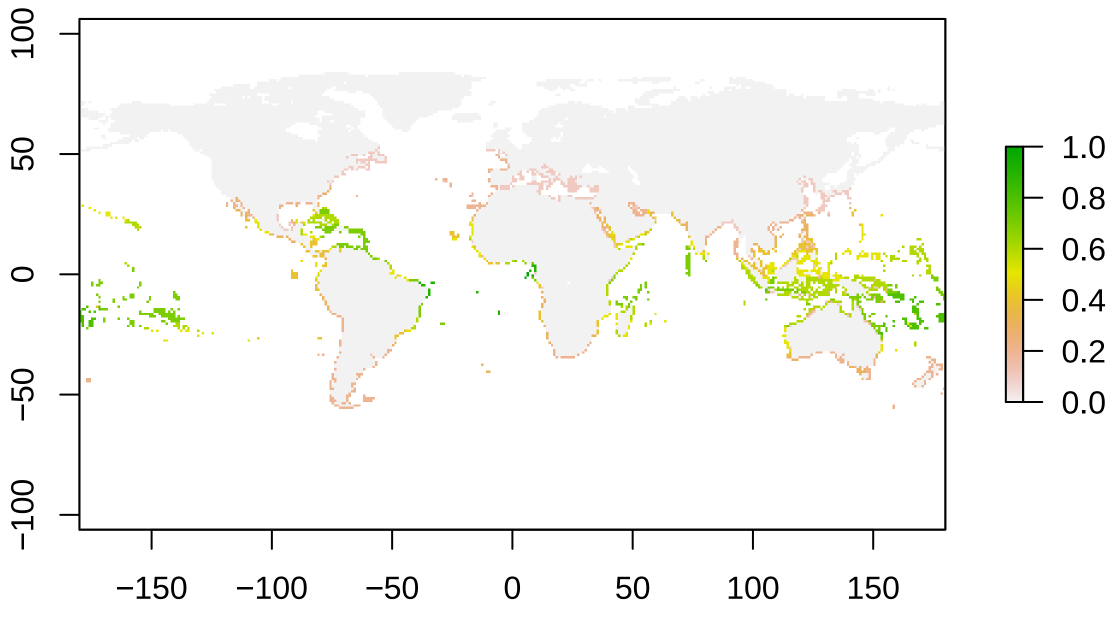


```
plot(adf_map > 0.9)
```



After this I would like to see how many species each of these thresholds generally include in there ADF, as obviously the lower the threshold the more species that are going to be included and this can influence the inferences of an anylsis as teh larger the species pool the more likely you are to find a difference between the observed and the null communities (Cornell & Harrison, 2014).

```
######## This function checks to see how many species are incorporate into the adf depending on the thr


adf_spp_rich <- function(cells, data){

  ## create a blank matrix with 11 rows (seq(0,1,0.1)) and the amount of columns of the cells to check

adf_nspp <- matrix(rep(NA,11*length(cells)), nrow = 11, ncol = length(cells))
rownames(adf_nspp) <- as.character(seq(0,1,0.1))
colnames(adf_nspp) <- as.character(cells)

### for each cell and proportion combination

for(cell in cells){

for(prop in seq(0,1,0.1)){

prop_cells <- which(data[as.character(cell),] > prop)

if(length(prop_cells) == 1){
  prop_species <- rownames(map_data)[which(map_data[,prop_cells] == 1)]
} else{
prop_species <- rownames(map_data)[which(rowSums(map_data[,prop_cells],na.rm = TRUE) > 0)]
}

#### how many species occupy the cells with that similarity

n_spp <- length(prop_species)

adf_nspp[as.character(prop),as.character(cell)] <- n_spp
}
}
return(adf_nspp)
}

#### run funciton for all our site focal cells

adf_richness <- adf_spp_rich(site_cells, site_cell_matrix)

adf <- blank

adf@data@values <- site_cell_matrix["32227",]


plot(adf)
```



Here we can see that some of the focal cells for some of our sites are predominately in the ocean so are picking up oceanic and palegic species in the similarity analysis, so I am going to identify the problem cells and reassign the focal cell to the surrounding cell which has the most similar species to those that directly overlap with the site coordinates.

The problem cells can be easily identified with the number of species that are present within the unrestricted adf.

```
## identify problem cells

problem_cells <- as.numeric(names(which(adf_richness[1,] < 9000)))

## data frame to contain the re-assigned cell

resolve <- data.frame(problem_c = problem_cells, resolve_cells = NA)

### for each of the problem cells

for(problem in problem_cells){

  ### what sites are within the probelm cell and pick one

prob_site <- sites %>% dplyr::filter(cell == problem) %>% dplyr::distinct(SSBS) %>% pull() %>% as.chara
prob_site <- prob_site[1]

## what are the species overlapping the site

prob_spp <- names(which(site_spp_matrix[prob_site,] == 1))

  ### get the surround cells

surround <- adjacent(blank, problem, directions = 8 )[,2]


### for each of the surrounding cells

sound_prob <- c()
for(s_cell in surround){

  ### what species are within

  sound_spp <-  names(which(map_data[,s_cell] == 1))

  ### calculate similarity between surrounding cell and site

  sound_sim <- length(which(prob_spp %in% sound_spp))/length(prob_spp)

  # store

  sound_prob <- c(sound_prob,sound_sim)

}

# which of the surrounding cells has teh greatest similarity

solve_cell <- surround[which.max(sound_prob)]
```

```
## reassign

resolve <- resolve %>% dplyr::mutate(resolve_cells = ifelse(problem_c == problem, as.numeric(solve_cell

}

sites$new_cell <- sites$cell


for(i in 1:nrow(resolve)){
sites <- sites %>% dplyr::mutate(new_cell = ifelse(cell == resolve[i,"problem_c"], resolve[i,"resolve_c
}
```

Soo hopefully now the problem cells have been resolved can re-do the species-threshold check

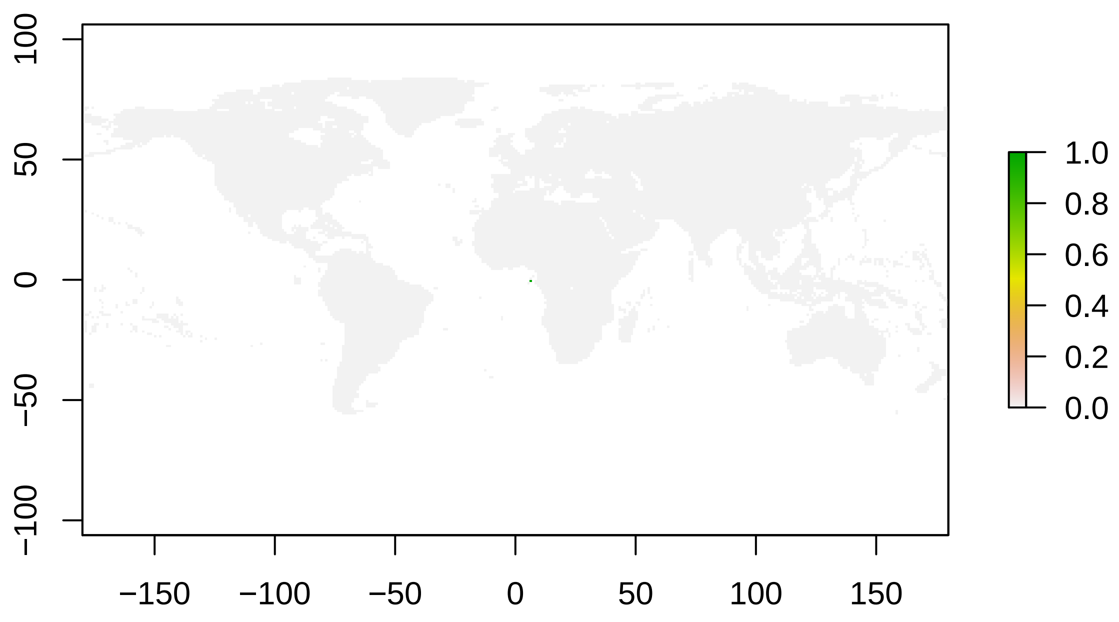
```
## new cells

new_cells <- unique(sites$new_cell)


#### add new cells to the site_cell_matrix

add <- new_cells[which(!(new_cells %in% site_cells))]




new_mat <- c()
for(a in add){
  m <- adf_sim_func(cells = a)
  new_mat <- rbind(new_mat,m)
}


site_cell_matrix <- rbind(site_cell_matrix,new_mat)


## run the cell richness function again

adf_richness <- adf_spp_rich(new_cells,site_cell_matrix)


##### the persistent problem cells are those of the studies on small islands, such a sao tome and princ

problem_cells <- as.numeric(names(which(adf_richness[1,] < 9000)))

####### so what I'm going to do is work out all the species that overlap with the island polygon and th

island_polys <- readRDS("Outputs/assembly_islands.rds")
island_spp <- readRDS("Outputs/assembly_island_spp.rds")


adf_sim_func_island <- function(cells){
```

```
         mat <- matrix(rep(NA,ncol(map_data)), nrow = 1, ncol = ncol(map_data))


               ### This bit is kinda redundant when looking just at sites but if we were
               ### the target cell had no species present within - this includes marine 

               if(!any(!is.na(as.numeric(map_data[,cells])))){
                 rownames(mat) <- cells
                 colnames(mat) <- 1:ncol(map_data)
               } else {



                 coords <- sites %>% dplyr::filter(new_cell == cells) %>% dplyr::distinc
                 coords <- coords[1,]
                 coords <- SpatialPoints(coords[,c("Longitude","Latitude")], proj4string


                 ## extract species that are present within a specified cell for the isl

                     for(island in names(island_polys)){
  if(suppressWarnings(gDistance(coords,island_polys[[island]])) < 0.01 ){
    focal_spp <- island_spp[[island]]
    break()
  }
}




                 ### get all other cells minus the focal cell

                 other_cells <- c(1:(cells-1),(cells+1):ncol(map_data))

                 ### create blank matrix

                 colnames(mat) <- 1:ncol(map_data)

                 ## cell shares all species with itself

                 mat[,cells] <- 1
                 rownames(mat) <- cells

                 ### for all other cells extract species and calculate proportion of pse

                 for(other in other_cells){

                   ### if there are no species that aren't classified as NA in the map d

                   if(!any(!is.na(as.numeric(map_data[,other])))){
                     prop <- NA
                   } else {

                     ## extract the species with in the other cell
```

```
                  other_spp <- names(which(map_data[,other] == 1))

                  ### if the cell is terrestrial but has no species in it then the ot

                  if(is_empty(other_spp)){
                    other_spp <- "none"
                  }

                  ### how many species are shared between the two cells

                  sim <- length(which(focal_spp %in% other_spp))

                  # if none similarity is 0


                  prop <- ifelse(sim != 0, sim/length(focal_spp), 0)


                 }

                 ### input the data into the matrix

                 mat[,other] <- prop
                }

               }

               return(mat)
       }


for(prob in problem_cells){
  m <- adf_sim_func_island(prob)
  site_cell_matrix[as.character(prob),] <- m
}



for(i in problem_cells){

  ### get the studies which have sites within this cell

  study <- sites %>% filter(new_cell == i) %>% distinct(SS) %>% pull()
  study <- paste(study,collapse = "&")

  # create a directory

  dir <- paste("assemblage_dispersion_fields/",paste(study,i,sep = "_"), sep = "")

 dir.create(dir)

  ## get each of the cell similarity scores
  adf <- site_cell_matrix[as.character(i),]

  ## input into blank raster

adf_map <- blank

adf_map@data@values <- adf

### save the raw similarity scores

png(paste(dir,"/","adf.png",sep = ""))
plot(adf_map)
dev.off()

### which cells are above the threshold for similarity

for(sim in c(0.9,0.8,0.7,0.6)){

  ## save the resulting map indicating the cells whose species will be included in the species pool.

  png(paste(dir,"/",sim,".png",sep = ""))
  plot(adf_map > sim)
  dev.off()
}

}
```

```
## Warning in dir.create(dir): 'assemblage_dispersion_fields\KS1_2009__SuarezRubio
## 1_25673' already exists

## Warning in dir.create(dir): 'assemblage_dispersion_fields\KS1_2009__SuarezRubio
## 1_25674' already exists

## Warning in dir.create(dir): 'assemblage_dispersion_fields\DI1_2013__deLima
## 1_32587' already exists

## Warning in dir.create(dir): 'assemblage_dispersion_fields\DL1_2012__Dallimer
## 1_32227' already exists

## Warning in dir.create(dir): 'assemblage_dispersion_fields\SC1_2005__Marsh
## 1_36945' already exists
```

```
adf <- blank

adf@data@values <- blank@data@values + site_cell_matrix["32587",]

plot(adf)
```



Excluding species from the species pools

Certain species are going to need to be excluded form the species pool depending on their life history because it is unlikely they would have been observed during the surveying or if they were it is unlikely that they are resident in habitat and therefore not going to be impacted by the land use. Excluded species include:

- Nocturnal species
- Marine species
- Coastal species
- swifts - aerial
- swallows - aerial
- martins - aerial
- raptors - aerial

```
## READ in AVONET and make sure some taxonomic discrepancies are resovled.

AVONET <- read.csv("../../Datasets/GBD/GBD_2021_BirdLife_Taxo_19 April.csv") %>%
  dplyr::mutate(Birdlife_Name = ifelse(Birdlife_Name == "Gorsachius magnificus", "Oroanassa magnifica",
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula krameri", "Alexandrinus krameri", pa
                Birdlife_Name = ifelse(Birdlife_Name == "Arachnothera hypogrammica", "Kurochkinegramma h
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula eupatria", "Palaeornis eupatria", pa
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula finschii", "Himalayapsitta finschii
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula roseata", "Himalayapsitta roseata",
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula longicauda", "Belocercus longicaudus
                Birdlife_Name = ifelse(Birdlife_Name == "Hapalocrex flaviventer", "Laterallus flavivent
                Birdlife_Name = ifelse(Birdlife_Name == "Hylocharis chrysura", "Amazilia chrysura", pas
                Birdlife_Name = ifelse(Birdlife_Name == "Porzana spiloptera", "Laterallus spilopterus",
                Birdlife_Name = ifelse(Birdlife_Name == "Hylocharis cyanus", "Amazilia cyanus", paste(Bi
                Birdlife_Name = ifelse(Birdlife_Name == "Chrysuronia oenone", "Amazilia oenone", paste(
                Birdlife_Name = ifelse(Birdlife_Name == "Hylocharis eliciae", "Amazilia eliciae", paste
                Birdlife_Name = ifelse(Birdlife_Name == "Juliamyia julie", "Amazilia julie", paste(Bird
                Birdlife_Name = ifelse(Birdlife_Name == "Lepidopyga goudoti", "Amazilia goudoti", paste
                Birdlife_Name = ifelse(Birdlife_Name == "Lepidopyga coeruleogularis", "Amazilia coerule
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula himalayana", "Himalayapsitta himala
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula columboides", "Nicopsitta columboid
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula cyanocephala", "Himalayapsitta cyan
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula calthrapae", "Nicopsitta calthrapae
                Birdlife_Name = ifelse(Birdlife_Name == "Psittacula eques", "Alexandrinus eques", paste
                Birdlife_Name = ifelse(Birdlife_Name == "Lepidopyga lilliae", "Amazilia lilliae", paste
                Birdlife_Name = ifelse(Birdlife_Name == "Heteroglaux blewitti", "Athene blewitti", past
                Birdlife_Name = ifelse(Birdlife_Name == "Atlantisia rogersi", "Laterallus rogersi", pas
                Birdlife_Name = ifelse(Birdlife_Name == "Psitteuteles iris", "Trichoglossus iris", past

## read in the Foraging dataset

Forage <- readRDS("../PREDICTS_Taxonomy/PREDICTS_imputed_BL_traits_forage.rds")

### extract all the species that have been flagged for exclusion
Nocturnal_spp <- AVONET %>% dplyr::filter(Nocturnal == 1) %>% pull(Birdlife_Name)
marine_spp <- AVONET %>% filter(Primary_Habitat_updated == "Marine") %>% pull(Birdlife_Name)
swift_spp <- AVONET %>% filter(Birdlife_family %in%  c("Apodidae","Hemiprocnidae")) %>% pull(Birdlife_Na
swallow_spp <- AVONET %>% filter(Birdlife_family %in% c("Hirundinidae","Artamidae")) %>% pull(Birdlife_
raptor_spp <- AVONET %>% filter(Birdlife_family %in% c("Accipitridae","Cathartidae")) %>% pull(Birdlife
coastal_spp <- AVONET %>% filter(Primary_Habitat_updated == "Coastal") %>% pull(Birdlife_Name)
Extinct_spp <- AVONET %>% filter(is.na(Extinct..Yes.1..No.0.)) %>% pull(Birdlife_Name)
Extinct_spp <- unique(c(Extinct_spp,"Pinguinus impennis","Conuropsis carolinensis","Ectopistes migrator

drop_spp <- unique(c(Nocturnal_spp,marine_spp,swift_spp,swallow_spp,raptor_spp,coastal_spp,Extinct_spp,

### excluding 1385 species from inclusion in the species pool.

write_rds(file = "Outputs/assembly_drop_spp.rds", drop_spp)
```

I am going to define a spectrum of species pools from the most restrictive to just the species whose ranges overlaps the site coordinates and then just those species that occur in the immediate surrounding cells. Then the rest are adf species rnging from a similarity threshold of 0.6 to 0.9.

```
### species poool function that has the input of the focal site


species_pool_func <- function(site){
```

```
### create a list on item for each of the species pools
spp_pool_list <- c(rep(list(NA),6))

### name them
names(spp_pool_list) <- c("0.6","0.7","0.8","0.9","surround","site_overlap")


### each species pool should always have the species that were observed in the study which after begi

# PREDICTS extraction

site_spp <- PREDICTS %>% dplyr::filter(SSBS == site, !pseudospp) %>% distinct(Birdlife_Name) %>% pull

## site overlap extraction

site_spp <- unique(c(names(which(site_spp_matrix[site,] == 1)),site_spp))

## drop species

site_spp <- site_spp[which(!(site_spp %in% drop_spp))]

### assign species

spp_pool_list[["site_overlap"]] <- site_spp


## now we need to pull the cell in which the site occurs

site_cell <- sites %>% dplyr::filter(SSBS == site) %>% dplyr::distinct(new_cell) %>% pull()

## if the cell is part of the problem cells which are on islands teh species pools need to be defined

if(site_cell %in% problem_cells){

  ## get the coordinates so that we know which island species need to be assigned

  coords <- sites %>% dplyr::filter(SSBS == site) %>% dplyr::distinct(Longitude,Latitude)
  coords <- SpatialPoints(coords[,c("Longitude","Latitude")], proj4string = CRS("+proj=longlat +datum=

  ### for the surrounding species it will just be those species which occur anywhere on the entire is

  for(island in names(island_polys)){

    ## find out which island
    if(suppressWarnings(gDistance(coords,island_polys[[island]])) < 0.01){
      surround_spp <- unique(c(site_spp,island_spp[[island]]))

      ## drop species

      surround_spp <- surround_spp[which(!(surround_spp %in% drop_spp))]

      #assign
```

```
        spp_pool_list[["surround"]] <- surround_spp

        ## stop looop once assigned
        break()
      }
    }

    ## for each of the threshold of similarity

      for(prop in c(0.6,0.7,0.8,0.9)){

        ### get the cells which surpass the threshold of similarity
  prop_cells <- which(site_cell_matrix[as.character(site_cell),] > prop)

if(length(prop_cells) < 2){

  ## if there are less that two cells so it would be cell the single focal cell assign all the species 
  prop_species <- surround_spp

} else{

  ### assign all the species on the island and those cells that surpass the threshold
  prop_species <- unique(c(site_spp,surround_spp,rownames(map_data)[which(rowSums(map_data[,prop_cells]
  prop_species <- prop_species[which(!(prop_species %in% drop_spp))]
}
  spp_pool_list[[as.character(prop)]] <- prop_species

  }


    ######################################## THIS IS FOR THE NON_PROBLEM CELLS
      ########################################

  } else {

    ## get all the cells which surround the focal cell AND the focal cell

  surround_cells <- unique(c(site_cell,adjacent(site_cell,x = blank,directions = 8 )[,2]))

  #the species that contained within those cells

  surround_spp <- unique(c(site_spp,rownames(map_data)[which(rowSums(map_data[,surround_cells],na.rm = 

  #drop species

  surround_spp <- surround_spp[which(!(surround_spp %in% drop_spp))]

  #assign

  spp_pool_list[["surround"]] <- surround_spp

  #for each similarity threshold
  for(prop in c(0.6,0.7,0.8,0.9)){

    ## the cells that surpass that threshold

  prop_cells <- which(site_cell_matrix[as.character(site_cell),] > prop)

if(length(prop_cells) < 2){

  # species that occur within a single cell - random that this needs to be done but rowSums doesn't wor
  prop_species <- rownames(map_data)[which(map_data[,prop_cells] == 1)]

  # drop

  prop_species <- prop_species[which(!(prop_species %in% drop_spp))]

} else{

  # species that occur in those cells
  prop_species <- unique(c(site_spp,rownames(map_data)[which(rowSums(map_data[,prop_cells],na.rm = TRUE

  #drop

  prop_species <- prop_species[which(!(prop_species %in% drop_spp))]
}

  #assign

  spp_pool_list[[as.character(prop)]] <- prop_species

  }
  }

return(spp_pool_list)

}


## the sites
sp_pool_sites <- as.character(unique(sites$SSBS))

# blank list

species_pools <- rep(list(NA),length(sp_pool_sites))
names(species_pools) <- sp_pool_sites

i <- 1
for(site in sp_pool_sites){

  dat <- species_pool_func(site = site)

  species_pools[i] <- list(dat)

  i <- i +1
```

```r
}

#save

write_rds(file = "Outputs/predicts_sites_species_pools.rds", species_pools)
```

Further refining the species pool

Lessard et al, 2012's paper "Inferring local ecological processes amid species pool influences" suggests various ways of defining the species pool to detect non-random processes of assembly and highlights the utility of making the definition ecologically explicit to impart as much realism as possible. Much of the idiosyncrasy in community ecology arises as a result of geographic variation in the evolutionary and historical processes that have shaped the structure of the species pool. As such, species pools need to take into account the ecological processes that may influence the structure of the species pool, such as species dispersal capabilities and environmental affinity for a specific location.

These ecological contingencies can be accounted for when performing null model analyses by weighting the probability of a species' traits being selected by species-specific dispersal and environmental probabilities.

Dispersal probability

Recently the Hand-Wing Index (HWI) has been adopted as a proxy for dispersal ability in birds Sheard et al, 2020 with larger HWI values being associated with a greater dispersal capability. Therefore a potential metric for dispersal probability could look like (HWI)/(distance to range edge).

```r
rm(list = ls())


species_pools <- readRDS("Outputs/predicts_sites_species_pools.rds")
species_files <- list.files("../../Datasets/Birdlife_Maps/Shapefiles/PREDICTS_BL/",full.names = TRUE)
PREDICTS <- readRDS("../PREDICTS_Assembly/Outputs/refined_predicts.rds")


assembly_species <- unique(c(unlist(unlist(species_pools))))


index_list_sp <- function(list){
  pool_sp <- c()
  for(i in 1:length(list)){
    pool_sp <- unique(c(pool_sp,list[[i]]))
  }
  return(pool_sp)
}




site_find <- function(sp){
  sp_site <- c()

  for(i in 1:length(species_pools)){
    pool_sp <- index_list_sp(species_pools[[i]])
    if(sp %in% pool_sp){
      sp_site <- c(sp_site,names(species_pools)[i])
    }
  }
  return(sp_site)
}




coords <- PREDICTS %>% distinct(SSBS,Longitude,Latitude)


# registerDoParallel(cores = 8)
#
# site_spp_dist <- foreach(sp = assembly_species,
#                       .combine = "cbind",
#                       .packages = c("tidyverse", "maptools", "rgdal", "sp", "raster","rgeos","sf","fast
#                       .inorder = FALSE) %dopar% {
#
#
#
#
# load(species_files[which(grepl(species_files, pattern = paste("/",sp,"Birdlife",sep = "")))])
#
#
#                         #### filter data so that distribution polygons are those that represent extant,
#                         #### native, reintroduced and introduced origin and those that are resident, br
#
#   mat <- matrix(rep(NA,length(species_pools)), ncol = 1)
#   colnames(mat) <- sp
#   rownames(mat) <- names(species_pools)
#
#
#                         data <- data %>% dplyr::filter(presence %in% c(1,2,3), origin %in% c(1,2,3), se
#
#
#
#
#                         ##### funny thing with some ranges so need to convert back to type MULTIPOLYGON
#
#                         if(any(class(data$Shape)[1] == "sfc_MULTISURFACE", class(data$Shape)[1] == "sfc
#                           for(k in 1:NROW(data)){
#                             data$Shape[[k]] <- st_cast(data$Shape[[k]], "MULTIPOLYGON")
#                           }
#                         }
#
#                         #combine all polygons together and convert to a sf class polygon
#
#                         shape <- as_Spatial(st_combine(data$Shape))
#
#
#                         sp_sites <- site_find(sp)
```

```
#
#
#                        for(sit in sp_sites){
#
#                            s_coords <- SpatialPoints(coords[which(coords$SSBS == sit),c("Longitude","L
#
#                            mat[sit,sp] <- suppressWarnings(gDistance(s_coords,shape))
#
#
#                        }
#
#                        return(mat)
# }
#
# registerDoSEQ()
# closeAllConnections()
#
# write_rds(site_spp_dist, file = "Outputs/site_sp_distance.rds")
site_spp_dist <- readRDS("Outputs/site_sp_distance.rds")



traits <- readRDS("../PREDICTS_Taxonomy/PREDICTS_imputed_BL_traits.rds") %>% dplyr::mutate(ScaleHWI = Ha



## higher the hand wing index the greater the probability of selection +
## the further away the range edge the lower the probability of selection


# sooooooooooooooooo if dist to range edge probability should be 1 and then the greater the distance the
###




dispersal_prob <- site_spp_dist/max(site_spp_dist,na.rm = TRUE)
for(sp in colnames(site_spp_dist)){

  HWI <- traits %>% dplyr::filter(uniqueIDS == sp) %>% dplyr::select(ScaleHWI) %>% pull()

  dispersal_prob[,sp] <- ifelse(dispersal_prob[,sp] == 0, 1, 1-(dispersal_prob[,sp]/HWI))
}

min(dispersal_prob, na.rm = TRUE)
```

```
## [1] -92.88103
```

```
## Hand_wing index scaled between 0 and 1
```

Environmental affinity

An ideal approach for estimating species specific environmental affinities would be to quantify directly habitat requirements and phsiological limits (potentially by performing species distribution modelling whose output is a probability of presence based on the environmental conditions at a cell). I think that these maps may be available from Stephen Willis in Durham so will need to ask Joe about these. Lessard cautions against using the same data to derive environmental affinities as used for delineating the species pool to avoid the risk of circular reasoning so getting these projections would be useful as they'll defintely be based on new data.