Code for generating presence absence from presence-only data

`paSampling` performs a two-step procedure for uniformly sampling pseudo-absences within the environmental space.

1. Subset environmental rasters and collect them in a single dimension (sort of like principal component analysis)
2. Match suitable environmental indicators for presences and use the opposite to predict absences

More info on R-vignette (https://rdrr.io/github/danddr/UEsampling/f/README.md)

This is the example code for generating pseudo absences for Z. brevicauda species using this GBIF data (https://www.gbif.org/species/2439344)

Steps

1. Read environmental raster data
2. read presence only data and convert it to a lat-lon point data frame
3. run the paSampling grid search
4. save the new dataframe as an ESRI shapefile that can be plotted as a map

In [ ]:
```
setwd("path of your root folder")
getwd()
```

In [ ]:
```
library(USE) #devtools::install_github("danddr/USE")
library(terra)
library(raster)
library(sf)
#library(tidyverse) #for data wrangling and plotting
```

In [ ]:
```
envdata <- do.call(brick, lapply(list.files(
  path = "./Data/Input/Processed/Resampled/guan",
  pattern = "*.tif", full.names = T),raster))
#reads all the tif files in a folder called /guan/

e <- extent(envdata)
#get the lat-lon range or extent of all your rasters
```

In [ ]:
```
zyg_pres <- read.csv("./Data/Input/Raw/GBIF/gbif_zyg_brev01.csv")
process_gbif <- function(presence_data, e, name_longitude, name_latitude, date){
  gbif <- data.frame(lon = rep(NA, nrow(presence_data),
                               lat = rep(NA, nrow(presence_data))))
  gbif$lon <- presence_data[,name_longitude]
  gbif$lat <- presence_data[,name_latitude]
  gbif <- unique(gbif)
  gbif <- gbif[which(gbif$lon>=e[1] & gbif$lon<=e[2]),]
  gbif <- gbif[which(gbif$lat>=e[3] & gbif$lat<=e[4]),]
  return(gbif)
}
zyg_pres <- zyg_pres %>% dplyr::select(lon, lat) %>% dplyr::mutate(CLASS = rep(1, nrow(.)))

zyg_pres <- process_gbif(zyg_pres, e = e, "lon", "lat", NA)

myPres <- st_as_sf(zyg_pres, coords=c("lon", "lat"), crs=4326)
```

In [ ]:
```
zyg.psAbs <- USE::paSampling(env.rast=envdata,
                             pres=myPres,
                             thres=0.75,
                             H=NULL,
                             grid.res=1,
                             n.tr = as.numeric(nrow(myPres)),
                             prev=NULL,
                             sub.ts=F,
                             n.ts=NULL,
                             plot_proc=F,
                             verbose=T)

# from R-vignette
# env.rast
# A RasterStack, RasterBrick or a SpatRaster object comprising the variables
# describing the environmental space.

# pres
# A SpatialPointsDataframe, a SpatVector or an sf object including the presence-only
# observations of the species of interest.

# thres
# (double) This value identifies the quantile value used to specify the boundary of
# the kernel density estimate (default thres=0.75 ). Thus, probability values higher
# than the threshold should indicate portions of the multivariate space likely associated
# with presence points.

# H
# The kernel bandwidth (i.e., the width of the kernel density function that defines
#                       its shape) excluding the portion of the environmental space
# associated with environmental conditions likely suitable for the species. It can be
# either defined by the user or automatically estimated by paSampling via ks::Hpi.

# grid.res
# (integer) resolution of the sampling grid. The resolution can be arbitrarily selected
# or defined using the optimRes function.

# n.tr
# (integer) number of pseudo-absences for the training dataset to sample in each cell
# of the sampling grid

# sub.ts
# (logical) sample the validation pseudo-absences

# n.ts
# (integer; optional) number of pseudo-absences for the testing dataset to sample in
# each cell of the sampling grid. sub.ts argument must be TRUE.

# prev
# (double) prevalence value to be specified instead of n.tr and n.ts

# plot_proc
# (logical) plot progress of the sampling, default FALSE

# verbose
# (logical) Print verbose
```

In [ ]:
```
## functions to process presence-absence
process_absences <- function(abs_df, pres_df, e_df, coord_ref,
                             path_write, layer_name, driver_write) {
  abs_df <- abs_df %>% select(lon = x,lat = y)
  abs_df <- as.data.frame(abs_df[,1:2])
  abs_df$geometry <- NULL
  pres_df <- pres_df %>% select("lon", "lat")
  pres_df <- process_gbif(pres_df, e = e_df, "lon", "lat", NA)
  train <- rbind(pres_df, abs_df)
  pa_train <- c(rep(1, nrow(pres_df)), rep(0, nrow(abs_df)))
  train <- data.frame(cbind(CLASS=pa_train, train))
  crs <- crs(coord_ref)
  train <- train[sample(nrow(train)),]
  class.pa <- data.frame(train[,1])
  colnames(class.pa) <- 'CLASS'
  dataMap.gbif  <- SpatialPointsDataFrame(train[,c(2,3)], class.pa,
                                          proj4string =crs)
  st_write(as(dataMap.gbif, "sf"),
           path_write,layer_name,
           driver = driver_write, append = F)
  return(dataMap.gbif)
}
```

In [ ]:
```
zyg_gbif <- process_absences(abs_df = zyg.psABs,
                             pres_df = zyg_pres,
                             e_df = e,
                             coord_ref = envdata,
                             path_write = "./Data/Input/Processed/GBIF/paSampling",
                             layer_name = "zyg_pa",
                             driver_write = "ESRI Shapefile")
```

In [ ]:
```
ggplot(st_as_sf(zyg_gbif), aes(color = factor(CLASS))) + geom_sf() + theme_linedraw()
```