Species distribution models (SDM)

AZTI

2023-01-18

Contents

About

This is a short tutorial for constructing species distribution models in R using shape-constrained generalized additive models.

The code is available in AZTI's github repository repository and the book is readily available here.

To cite this book, please use:

BLA BLA BLA

Chapter 1

Introduction

Species distribution models (SDMs) are numerical tools that combine observations of species occurrence or abundance at known locations with information on the environmental and/or spatial characteristics of those locations [Elith and Leathwick, 2009]. They are also known as ecological niche models (ENM) or habitat suitability models or…

A wide variety of methods have been used …

Reviews of SDM literature include …

One of the common problems is that, the fitted models do not agree with the ecological niche theory…

This book provides a tutorial on how to use shape-constrained generalized additive models to build SDMs. It is organised following the key steps in good modeling practice of SDMs [Elith and Leathwick, 2009]. First, presence data of a selected species are downloaded from GBIF/OBIS datasets and pseudo-absence data are created. Then, environmental data are downloaded from public repositories and extracted at each of the presence/pseudo-absence data points. Based on this dataset, an exploratory analysis is conducted to help deciding on the best modelling approach. The model is fitted to the dataset and the quality of the fit and the realism of the fitted response function are evaluated. After selecting a threshold to transform the continuous probability predictions into binary responses, the model is validated using a k-fold approach. Finally, the predicted maps are generated for visualization.

Chapter 2

Libraries

Load libraries that will be used

```
library("scam")
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-39. For overview type 'help("mgcv-package")'.
```

```
## This is scam 1.2-12.
```

Note that all the libraries must be installed. If some library is not installed, run:

```
install.packages("scam")
```

```
## Warning: package 'scam' is in use and will not be installed
```

Chapter 3

Presence-absence data

Bla bla bla

3.1 Download presence data

Download from GBIF OBIS. Mireia

3.2 Create pseudo-absence data

Prevalence 50%

See code from ANICHO (mantaining some space around presences). Leire C.

Ref [Barbet-Massin et al., 2012]

Copio aqui el codigo de anicho tal cual, luego lo limpiaré para este caso:

```
# Script information -------------------------------------------------------

# Title: Generate pseudo-absences for IM-18-ANICHO
# Last modified by Leire Ibaibarriaga (libaibarriaga@azti.es) and Leire Citores (lcitores@azti.es

# Load libraries -----------------------------------------------------------

library(tidyverse)
library(ggplot2)
library(scales)
library(here)
library(ggridges)

library(maps)        # some basic country maps
```

```
library(mapdata)     # higher resolution maps
library(mapproj)
library(marmap)      # access global topography data
library(mapplots)    # ices rectangles
library(sf)
library(gridExtra)
library(lubridate)

# general settings for ggplot (black-white background, larger base_size)

theme_set(theme_bw(base_size = 16))

# Set directories ------------------------------------------------------

# final data set created by Nerea Goikoetxea after combining logboook and VMS data are
# \\dok\nas\K\AZTIMAR\PROYECTOS\Funcionamiento de los ecosistemas marinos\IM-18-ANICHO
# note that these data do not have yet the environmental variables. We have to repeat
# because some dates were wrong in the previous file (anicho_landings_final_FECHASMAL.

data.dir <- here("data")

# Data frame with environmental variables ------------------------------

# read data-file

df <- read.csv(file.path(data.dir,"df_2010_2019_DEF.csv"), header=T, sep=";", dec=",")

dim(df) # 26336 rows and 11 columns
head(df)
tail(df)
summary(df)

# change the names

names(df) <- c("CODEUEBUQUE","FECHA_CAPT","DOY","WEEK","MONTH","YEAR","LAND","OK","PES

# format

df$FECHA_CAPT <- as.Date(df$FECHA_CAPT, format="%Y-%m-%d")

# check duplicates in all the columns

dupli <- duplicated(df)
table(dupli) # there are 13 duplicated rows!!
df[dupli, ] # this shows only the rows that are duplicated (ie the second/third/... ti
```

```
dupli <- duplicated(df) | duplicated(df, fromLast=T) # to see the duplicates and the first time t
table(dupli) # 26, ie each duplicated row appears twice
df[dupli, ]

# LIC: tras hablar con Lucia parece que aunque parezcan replicados, son datos oficiales de SGPM y
# tendrian que ser correctos. Asi que los agrupo y sumo la variable PESO

df <- df %>%
  group_by_at(vars(-PESO_ANE)) %>%
  summarise(PESO_ANE=sum(PESO_ANE))
dim(df) # 26085 observations

# select only points from March to July

df <- subset(df, MONTH %in% c(3:7)) # 25036 observations
dim(df) # 25036 observations

# Depth, ICES statistical rectangles etc ----------------------------------


# read shapefile with ices divisions

ices.areas.shp <- st_read("C:/use/proyectos/IM-18-ANICHO/datos/ICES_shapefiles/ICES_areas")
st_crs(ices.areas.shp)
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
ices.areas.shp <- st_transform(ices.areas.shp, wgs)

ices.rect.shp <- st_read("C:/use/proyectos/IM-18-ANICHO/datos/ICES_shapefiles/ICES_rectangles")
st_crs(ices.rect.shp)
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
ices.rect.shp <- st_transform(ices.rect.shp, wgs)

ices.rectareas.shp <- st_read("C:/use/proyectos/IM-18-ANICHO/datos/ICES_shapefiles/ICES_StatRec_m
st_crs(ices.rectareas.shp)
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
ices.rectareas.shp <- st_transform(ices.rectareas.shp, wgs)

# get bathymetry data

bathy <- getNOAA.bathy(lon1=-18,lon2=0,lat1=41,lat2=51, resolution = 1,
                       keep=FALSE, antimeridian=FALSE)
class(bathy)
autoplot(bathy)
bathy.df <- fortify(bathy)
class(bathy.df)
```

```
str(bathy.df)

# add Depth from marmap according to Lon and Lat

idx <- which(!is.na(df$LON) & !is.na(df$LAT) & df$LON > -18 & df$LON < 0 & df$LAT >41 &
df$DEPTH <- NA
df$DEPTH[idx] <- get.depth(bathy, df[idx,c("LON","LAT")], locator=F)$depth

# Maps ----------------------------------------------------------------------

# basic map data

global <- map_data("worldHires")

# basic ggplot

p0 <- ggplot() +
  geom_contour(data=bathy.df, aes(x,y,z=z), breaks=c(-100, -200), col="grey")+
  annotation_map(map=global, fill="grey")+
  geom_sf(data=fortify(ices.areas.shp[1]), fill=NA)+
  scale_x_continuous(minor_breaks = seq(-10, 0, 1), breaks = seq(-10, 0, 1))+   # ices
  scale_y_continuous(minor_breaks = seq(42, 50, 0.5), breaks=seq(42, 50, 1))+   # ices
  coord_sf(xlim=c(-10,0), ylim=c(42,50))+
  xlab("")+
  ylab("")
print(p0)

# EGSP: Transformation to UTM -----------------------------------------------

# function to find your UTM. Taken from Nerea Goikoetxea

lonlat2UTM = function(lonlat) {
  utm = (floor((lonlat[1] + 180) / 6) %% 60) + 1
  if(lonlat[2] > 0) {
    utm + 32600
  } else{
    utm + 32700
  }
}

EPSG_2_UTM <- lonlat2UTM(c(mean(df$LON), mean(df$LAT)))
EPSG_2_UTM # 32630 --> UTM zone 30N;
#           WGS84 Bounds: -6.0000, 0.0000, 0.0000, 84.0000
#           Projected Bounds: 166021.4431, 0.0000, 833978.5569, 9329005.1825
```

```
# Visualize areas for generating pseudo-absences ---------------------------

# check class
class(ices.rectareas.shp)

# select multipolygon object from the shapefile
aux1 <- ices.areas.shp[ices.areas.shp$Area_27 %in% c("8.b", "8.c"), ]

# create a polygon for intersection with ices areas, so that we can select from 6Â°W to the east

aux2 <- st_sfc(st_polygon( list(rbind(c(-6, 40), c(-6, 50), c(1, 50), c(1,40), c(-6, 40)))))
aux2 <- st_set_crs(aux2, "+proj=longlat +datum=WGS84 +ellps=WGS84")
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
aux2 <- st_transform(aux2, wgs)

# create a polygon for union with ices areas, so that we can add two rectangles from 4Â°W to the

aux3 <- st_sfc(st_polygon( list(rbind(c(-4, 44), c(-4, 46), c(-2, 46), c(-2,44), c(-4, 44)))))
# aux3 <- st_sfc(st_polygon( list(rbind(c(-4, 44.5), c(-4, 45.5), c(-2, 45.5), c(-2,44.5), c(-4,
aux3 <- st_set_crs(aux3, "+proj=longlat +datum=WGS84 +ellps=WGS84")
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
aux3 <- st_transform(aux3, wgs)

# transform the catch data points into sf and add the CRS

df.sf <- st_as_sf(df, coords=c("LON","LAT"))
df.sf <- st_set_crs(df.sf, "+proj=longlat +datum=WGS84 +ellps=WGS84")
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
df.sf <- st_transform(df.sf, wgs)

# transform to UTMs (in m)

aux1.utm <- st_transform(aux1, EPSG_2_UTM)
aux2.utm <- st_transform(aux2, EPSG_2_UTM)
aux3.utm <- st_transform(aux3, EPSG_2_UTM)
df.sf.utm <- st_transform(df.sf, EPSG_2_UTM)

# convex hull of presence data
# df_buff <- st_convex_hull(st_union(df.sf.utm))
# plot(df_buff)

# create buffers of 10km (10000) around the points and join the resulting polygons

buffer <- st_buffer(df.sf.utm, dist=10000)
buffer <- st_union(buffer)
```

```
plot(buffer)

# intersect the ices divisions and the squares

aux <- st_intersection(x=st_union(st_union(aux1.utm), aux3.utm), y=aux2.utm)
plot(aux, col=2)
plot(buffer, add=T)

# intersect the result with the buffers around the catch data points
aux0 <- st_difference(aux, buffer)
plot(aux0, col=2)

# ggplot for all data

p <- p0 +
  geom_sf(data=aux0, fill="red", alpha=0.3)+
  coord_sf(xlim=c(-10,0), ylim=c(42,50))
ggsave(file.path("plots","pseudo",paste0("area_pseudo_all.png")), p, device="png")

# # randomly sample inside the new polygon
#
# kk <-  st_sample(aux0, size=100, type="random")
#
# # plot
#
# p <- p0 +
#   geom_sf(data=aux0, col="red", alpha=0.3)+
#   geom_sf(data=fortify(kk))+
#   coord_sf(xlim=c(-10,0), ylim=c(42,50))
# print(p)
#
# # extract coordinates as data.frame
#
# st_coordinates(kk)

# loop to calculate area to generate pseudo-absences by year

for (yy in sort(unique(df$YEAR))){
  df.sub <- subset(df.sf.utm, YEAR==yy)
  if (nrow(df.sub)>0){
    buffer.sub <- st_buffer(df.sub, dist=10000)
    buffer.sub <- st_union(buffer.sub)
    aux0.sub <- st_difference(aux, buffer.sub)
    p <- p0 +
      geom_sf(data=aux0.sub, fill="red", alpha=0.3)+
```

```
      #geom_sf(data=df.sub)+
      coord_sf(xlim=c(-6,0), ylim=c(43,46))
    ggsave(file.path("plots","pseudo",paste0("area_pseudo_",yy,".png")), p, device="png")
  }
}

# loop to calculate area to generate pseudo-absences by month and year

for (yy in sort(unique(df$YEAR))){
  for (mm in seq(3,7,by=1)){
    df.sub <- subset(df.sf.utm, YEAR==yy & MONTH==mm)
    if (nrow(df.sub)>0){
      buffer.sub <- st_buffer(df.sub, dist=10000)
      buffer.sub <- st_union(buffer.sub)
      aux0.sub <- st_difference(aux, buffer.sub)
      p <- p0 +
        geom_sf(data=aux0.sub, fill="red", alpha=0.3)+
        #geom_sf(data=df.sub)+
        coord_sf(xlim=c(-6,0), ylim=c(43,46))
      ggsave(file.path("plots","pseudo",paste0("area_pseudo_",yy,"_",mm,".png")), p, device="png"
    }
  }
}

# Remove points outside ices 8b, 8c or in land ----------------------------

# number of points within the study area

df.in <- st_intersects(df.sf.utm, aux, sparse=FALSE)
df$INSIDE <- df.in[,1]
mean(df$INSIDE) # 0.9743969 points inside the area of study

table(df$INSIDE)
table(df$DEPTH>0)
table(df$INSIDE, df$DEPTH>0)

p <- p0 +
  geom_point(data=subset(df, INSIDE==1 & DEPTH <=0), aes(x=LON, y=LAT), col="red", alpha=0.3)+
  coord_sf(xlim=c(-10,0), ylim=c(43,46))+
  ggtitle("Points inside area with Depth<=0")
  ggsave(file.path("plots","pseudo",paste0("points_inside_depthneg.png")), p, device="png")

p <- p0 +
  geom_point(data=subset(df, INSIDE==1 & DEPTH >0), aes(x=LON, y=LAT), col="red", alpha=0.3)+
  coord_sf(xlim=c(-10,0), ylim=c(43,46))+
```

```
  ggtitle("Points inside area with Depth>0")
  ggsave(file.path("plots","pseudo",paste0("points_inside_depthpos.png")), p, device="
p <- p0 +
  geom_point(data=subset(df, INSIDE==0 & DEPTH <= 0), aes(x=LON, y=LAT), col="red", al
  coord_sf(xlim=c(-10,0), ylim=c(43,46))+
  ggtitle("Points outside area with Depth<=0")
  ggsave(file.path("plots","pseudo",paste0("points_outside_depthneg.png")), p, device=
p <- p0 +
  geom_point(data=subset(df, INSIDE==0 & DEPTH > 0), aes(x=LON, y=LAT), col="red", alp
  coord_sf(xlim=c(-10,0), ylim=c(43,46))+
#  coord_sf(xlim=c(-10,0), ylim=c(42,50))+
  ggtitle("Points outside area with Depth>0")
  ggsave(file.path("plots","pseudo",paste0("points_outside_depthpos.png")), p, device=
# So, we keep only the points INSIDE the area with DEPTH<0

df <- subset(df, INSIDE==1 & DEPTH <= 0)
dim(df) # 24147 observations

# remove the columns that are not going to be used

df$LAND <- df$OK <- df$INSIDE <- NULL

# Generate pseudo-absences ------------------------------------------------

# we will use the positive database to generate the pseudo-absences

dfpos <- subset(df, PESO_ANE>0)
nbpoints <- nrow(dfpos) # 23678 out of 24147 are positive observations (98%)

# we compute again the buffer but only for the positive data points

# transform the catch data points into sf and add the CRS

dfpos.sf <- st_as_sf(dfpos, coords=c("LON","LAT"))
dfpos.sf <- st_set_crs(dfpos.sf, "+proj=longlat +datum=WGS84 +ellps=WGS84")
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
dfpos.sf <- st_transform(dfpos.sf, wgs)
dfpos.sf.utm <- st_transform(dfpos.sf, EPSG_2_UTM) # transform to UTMs (in m)

# Generate the pseudo-absence data frame

pseudo <- matrix(data=NA, nrow=nbpoints, ncol=10)
```

```
pseudo <- data.frame(pseudo)
names(pseudo) <- c("CODEUEBUQUE","FECHA_CAPT","DOY","WEEK","MONTH","YEAR","LAT","LON","PESO_ANE",

# set the seed

set.seed(1)

# sample the date

pseudo$FECHA_CAPT <- sample(x=dfpos$FECHA_CAPT, size=nbpoints, replace = TRUE)
pseudo$FECHA_CAPT <- as.Date(pseudo$FECHA_CAPT, format="%Y-%m-%d")
pseudo$DOY <- yday(pseudo$FECHA_CAPT)
pseudo$WEEK <- week(pseudo$FECHA_CAPT)
pseudo$MONTH <- month(pseudo$FECHA_CAPT)
pseudo$YEAR <- year(pseudo$FECHA_CAPT)

# loop by month and year

for (yy in sort(unique(dfpos$YEAR))){
  for (mm in sort(unique(dfpos$MONTH))){
    idx <- which(pseudo$YEAR==yy & pseudo$MONTH==mm)
    if (length(idx)>0){
      df.sub <- subset(dfpos.sf.utm, YEAR==yy & MONTH==mm)
      buffer.sub <- st_buffer(df.sub, dist=10000)
      buffer.sub <- st_union(buffer.sub)
      aux0.sub <- st_difference(aux, buffer.sub)
      rp.sf <- st_sample(aux0.sub, size=length(idx), type="random") # randomly sample points
      rp.sf <- st_transform(rp.sf, 4326)
      rp <- as.data.frame(st_coordinates(rp.sf)) # transform to lat&lon and extract coordinates o
      pseudo$LON[idx] <- rp$X
      pseudo$LAT[idx] <- rp$Y
      p <- p0 +
        geom_sf(data=aux0.sub, fill=2, alpha=0.3)+
        geom_sf(data=df.sub, col=4, alpha=0.3)+
        geom_sf(data=rp.sf, col=1, shape=4)+
        coord_sf(xlim=c(-6,0), ylim=c(43,46))+
        ggtitle(paste("ANE",yy,mm))
      ggsave(file.path("plots","pseudo",paste0("pseudo_",yy,"_",mm,".png")), p, device="png")
    }
  }
}


# complete the rest of columns
```

```
pseudo$CODEUEBUQUE <- "ESPxxxxxxxxx" # generic name to distinguish the pseudo-absence
pseudo$PESO_ANE <- 0
pseudo$DEPTH <- get.depth(bathy, pseudo[ ,c("LON","LAT")], locator=F)$depth

# there might be still some locations with Depth>0
summary(pseudo$DEPTH)
sum(pseudo$DEPTH>0)  # 197 obs
mean(pseudo$DEPTH>0) # 0.008316447 very small proportion

# transform to sf object (lat&lon) to plot

pseudo.sf <- st_as_sf(pseudo, coords=c("LON","LAT"))
pseudo.sf <- st_set_crs(pseudo.sf, "+proj=longlat +datum=WGS84 +ellps=WGS84")
wgs<-"+proj=longlat +datum=WGS84 +ellps=WGS84"
pseudo.sf <- st_transform(pseudo.sf, wgs)

p <- p0 +
  geom_sf(data=pseudo.sf, aes(col=(DEPTH>0)), col="red", alpha=0.3)+
  coord_sf(xlim=c(-6,0), ylim=c(42,46))
  #  coord_sf(xlim=c(-10,0), ylim=c(42,50))+
print(p)

# plot all pseudo-absences

p <- p0 +
  geom_sf(data=aux0, fill=2, alpha=0.3)+
  geom_sf(data=dfpos.sf, col=4, alpha=0.3)+
  geom_sf(data=pseudo.sf, col=1, shape=4, alpha=0.3)+
  coord_sf(xlim=c(-6,0), ylim=c(43,46))
print(p)
ggsave(file.path("plots","pseudo","pseudo_all.png"), p, device="png")

# plot pseudo-absences by year

p <- p0 +
  geom_sf(data=aux0, fill=2, alpha=0.3)+
  geom_sf(data=dfpos.sf, col=4, alpha=0.3)+
  geom_sf(data=pseudo.sf, col=1, shape=4, alpha=0.3)+
  coord_sf(xlim=c(-6,0), ylim=c(43,46))+
  facet_wrap(~YEAR)
print(p)
ggsave(file.path("plots","pseudo","pseudo_all_by_year.png"), p, device="png")


# Save the final dataset including the pseudo-absences ---------------------
```

```
head(df)
head(pseudo)

# Join the two data sets and save the final dataset

dat <- rbind(df, pseudo)
write.table(dat, file=file.path("data","dfpseudo_ane_2010_2019.csv"), row.names=F, sep=";", dec="

# End of script ------------------------------------------------------------
```

Chapter 4

Environmental data

Bla bla bla

4.1 Download from public repositories

Download from Bio-oracle.

```
### Download Environmental data from Bio-oracle

# for mapping ...
library(maptools)
library(rgrass7)
library(raster)
library(sp)
library(ggplot2)
library(maps)

# libraries for bio-oracle
library(rgdal)
library(sdmpredictors)
library(leaflet)

# http://bio-oracle.org/code.php
install.packages("sdmpredictors")
#install.packages("leaflet")

# Load package
library(sdmpredictors)

# Species Data example from GBIF
```

```
mydata <- occ_data(scientificName = "Anisakis", hasCoordinate = TRUE)$data  # 02/10/20
mydata.gbif <- subset(mydata, select=c(acceptedScientificName,genus,specificEpithet,de
mydata.gbif.ll <- cbind(mydata.gbif$decimalLongitude, mydata.gbif$decimalLatitude)
mydata.gbif.ll <- as.data.frame(mydata.gbif.ll)
names(mydata.gbif.ll) <- c("Lon", "Lat")

# Explore datasets in the package
list_datasets()
list_layers("Bio-ORACLE")
mytab <- list_layers("Bio-ORACLE")

## Download specific layers to the current directory

myBioracle.layers <- load_layers(c("BO_chlomean", "BO_damean", "BO_salinity", "BO_sstme
#save(myBioracle.layers, file="myBioracle.layers.Rdata")
#load(file="myBioracle.layers.Rdata") # creo que no funciona
myBioracle.layers

# Check layer statistics
layer_stats("myBioracle.layers")

# Crop raster to fit the North East Atlantic window for estimating SST range
# lat: 36.49 to 67.00
# lon: -16.000 to 9.000
my.NEatlantic.ext <- extent(-100, 45, -90, 90) # xmin, xmax, ymin, ymax
myBioracle.layers.cropNEatlantic <- crop(myBioracle.layers, my.NEatlantic.ext)
my.colors = colorRampPalette(c("#5E85B8","#EDF0C0","#C13127"))
plot(myBioracle.layers.cropNEatlantic,col=my.colors(1000),axes=F, box=F)
summary(myBioracle.layers.cropNEatlantic)

# Generate a nice color ramp and plot the map
my.colors = colorRampPalette(c("#5E85B8","#EDF0C0","#C13127"))

plot(myBioracle.layers.cropNEatlantic,col=my.colors(1000),axes=F, box=F)

image(log(myBioracle.layers.cropNEatlantic$BO_chlomean),col=my.colors(1000),axes=T, yla
map("world",add=T, fill=T, col="white",lwd=0.1); box()

image(log(myBioracle.layers.cropNEatlantic$BO_damean),col=my.colors(1000),axes=T, ylab=
map("world",add=T, fill=T, col="white",lwd=0.1); box()

image(myBioracle.layers.cropNEatlantic$BO_salinity,col=my.colors(1000),axes=T, ylab=NA
map("world",add=T, fill=T, col="white",lwd=0.1); box()

image(myBioracle.layers.cropNEatlantic$BO_sstmean,col=my.colors(1000),axes=T, ylab=NA,
```

```
map("world",add=T, fill=T, col="white",lwd=0.1); box()

image(myBioracle.layers.cropNEatlantic$BO_sstrange,col=my.colors(1000),axes=T, ylab=NA,xlab=NA,ma
map("world",add=T, fill=T, col="white",lwd=0.1); box()

image(myBioracle.layers.cropNEatlantic$BO_bathymean,col=my.colors(1000),axes=T, ylab=NA,xlab=NA,m
map("world",add=T, fill=T, col="white",lwd=0.1); box()


### Extract environmental values from layers

mydata1.env <- raster::extract(x=myBioracle.layers,y=mydata.gbif.ll, df=T) # sin buffer (se extra

# con Bilinear extraigo los sites de los 4 nearest cells
mydata1.env.bil <- raster::extract(x=myBioracle.layers,y=mydata.gbif.ll, method="bilinear", na.rm

mydata.all <- cbind(mydata.gbif.ll, mydata1.env.bil)

summary(mydata.all) # 448 sites : 84 are NAs
```

We can also download the bathymetry from the `marmap` library

```
library(marmap)

bathy <- getNOAA.bathy(lon1=-11,lon2=0,lat1=41,lat2=51, resolution = 1,
                       keep=FALSE, antimeridian=FALSE)
bathy.df <- fortify(bathy) # LI: maybe this is not needed here

df$Depth <- get.depth(bathy, df[,c("Lon","Lat")], locator=F)$depth
```

4.2 Operations with rasters (maybe not needed)

We can complete this a bit more later on, though not necessary right now

for example, given a raster, we can calculate gradients in the vertical or depth at which the max is found

```
# Auxiliary functions ------------------------------------------------

# Taken from:
# https://gis.stackexchange.com/questions/114723/subset-netcdf-based-on-last-valid-variable-by-le

# deepestValid: function to that, for each cell,
# - returns NA in case all depth level values are NA
# - returns the value of the last available depth level in case no NA occur
# - else returns the value of the last non-NA depth level
```

```
deepestValid <- function(x) {
  na <- is.na(x)
  if (all(na)) {
    return(NA)
  } else if (all(!na)) {
    return(x[length(x)])
  } else {
    first_na <- which(na)[1]
    last_valid <- first_na - 1
    return(x[last_valid])
  }
}

# read the data files using the library raster

b1 <- brick(file.path(hdata.dir, paste0("thetao_",dd, " 12:00:00.nc")))
b2 <- brick(file.path(hdata.dir, paste0("so_",dd, " 12:00:00.nc")))
b3 <- brick(file.path(hdata.dir, paste0("mlotst_",dd, " 12:00:00.nc")))
b4 <- brick(file.path(hdata.dir, paste0("chl_",dd, " 12:00:00.nc")))
b5 <- brick(file.path(hdata.dir, paste0("o2_",dd, " 12:00:00.nc")))

# temperature at surface (exactly at 0.49m)
# see names(b1)

bb1a <- b1[[1]]
names(bb1a) <- "TEMP0m"

# temperature at 100m (layer 22, which is at 92.33m) or at the deepest available
# see names(b1)

bb1b <- calc(subset(b1, 1:22), fun=deepestValid)
names(bb1b) <- "TEMP100m"

# salinity at surface (exactly at 0.49m)
# see names(b2)

bb2 <- b2[[1]]
names(bb2) <- "so1"

# logarithm of ocean mixed layer thickness (m)

bb3 <- b3[[1]]
bb3 <- log(bb3)
# or equivalently:
# bb3 <- calc(bb3, fun=function(x) {log(x)})
```

```
names(bb3) <- "logmlotst1"

# logarithm of the chlorophyll integrated until 100m (layer 22, which is at 92.33m) or at the dee
# see names(b4)

b4 <- subset(b4, 1:22) # take only values in the first 22 layers corresponding up to depth 100m
depth.lev <- as.numeric(as.character(sapply(names(b4), substring, first=2))) # extract depth valu
depth.lev <- c(0, depth.lev)
w <- diff(depth.lev)
bb4 <- calc(b4, function(x){sum(x*w, na.rm=T)}) # compute integrated value
bb4 <- log(bb4)
names(bb4) <- "logCHL100m"

# oxygen integrated until 100m (layer 22, which is at 92.33m) or at the deepest available
# see names(b5)

b5 <- subset(b5, 1:22) # take only values in the first 22 layers corresponding up to depth 100m
depth.lev <- as.numeric(as.character(sapply(names(b5), substring, first=2))) # extract depth valu
depth.lev <- c(0, depth.lev)
w <- diff(depth.lev)
bb5 <- calc(b5, function(x){sum(x*w, na.rm=T)}) # compute integrated value
names(bb5) <- "O100m"

# unique stack (because they all have the same extent and resolution)

b <- stack(bb1a, bb1b, bb2, bb3, bb4, bb5)

# Step 2: Add bathymetry ------------------------------------------------

# get the extent of the raster to extract the corresponding bathymetry

e <- extent(b)

# load bathymetry of the area (get wider limits)
# resolution = 5 (resolucion en minutos, corresponde a 1/12 º)

# mybathy <- getNOAA.bathy(lon1=floor(e@xmin), lon2=ceiling(e@xmax), lat1=floor(e@ymin), lat2=ce
#                          resolution=1, keep=F)
# save(mybathy, file="mybathy.RData")

load(file.path("mybathy.RData"))

# transform to data frame

pred.dat <- raster::as.data.frame(b, xy=T)
```

```
# include bathymetry into the prediction data frame

pred.dat$DEPTH <- get.depth(mybathy, pred.dat[ ,c("x","y")], locator=F)$depth
pred.dat$DEPTH[pred.dat$DEPTH>=0] <- NA
pred.dat$logDEPTH<-log(-pred.dat$DEPTH)

# transform the bathymetry to raster format

b7 <- rasterFromXYZ(pred.dat[,c("x","y","logDEPTH")])
names(b7) <- "logDEPTH"

# include the bathymetry raster into the prediction stack

b <- stack(b, b7)


# Step 3: Create latitude, longitude and doy rasters ----------------------
# these are needed to predict according to the spatio-temporal model

b8 <- b9 <- b10 <- b[[1]] # create rasters with same structure

b8[] <- coordinates(b8)[,1] #longitude
names(b8) <- "LON"

b9[] <- coordinates(b8)[,2] #latitude
names(b9) <- "LAT"

b10[] <- doy # day of the year
names(b10) <- "DOY"

# include longitude, latitude and day of the year into the prediction stack,
# they are needed for the spatio-temporal model prediction

b <- stack(b, b8, b9, b10)
```

Chapter 5

Prepare final dataset

Bla bla bla

5.1 Extract environmental data associated to presence-absence data

5.2 Exploratory plots

Chapter 6

Shape Constrained-Generalized Additive Models

One citation is [Citores et al., 2020]

Mention there is an alternative using `mboost` that won't be further developed here.

6.1 Model fit

6.2 Model selection

Chapter 7

Model validation

Bla bla

7.1 Optimum threshold

7.2 k-fold validation

Chapter 8

Prediction and maps

predict from fitted models and produce maps

Bibliography

Morgane Barbet-Massin, Frédéric Jiguet, Cécile Hélène Albert, and Wilfried Thuiller. Selecting pseudo-absences for species distribution models: how, where and how many? *Methods in Ecology and Evolution*, 3(2):327–338, 2012. ISSN 2041210X. doi: 10.1111/j.2041-210X.2011.00172.x.

L. Citores, L. Ibaibarriaga, D. J. Lee, M. J. Brewer, M. Santos, and G. Chust. Modelling species presence–absence in the ecological niche theory framework using shape-constrained generalized additive models. *Ecological Modelling*, 418:108926, 2020.

Jane Elith and John R. Leathwick. Species distribution models: Ecological explanation and prediction across space and time. *Annual Review of Ecology, Evolution, and Systematics*, 40(1):677–697, 2009. ISSN 1543-592X 1545-2069. doi: 10.1146/annurev.ecolsys.110308.120159.