

GAM-NICHE: Shape-Constrained GAMs to
build Species Distribution Models under the
ecological niche theory

AZTI

2025-02-17

Contents

About	5
1 Introduction	7
2 Presence-absence data	9
2.1 Download presence data	10
2.2 Create pseudo-absence data	19
3 Environmental data	29
3.1 Download from public repositories	30
4 Prepare the final dataset	35
4.1 Extract environmental data associated to species distribution data	36
4.2 Exploratory plots of environmental variables	39
4.3 Exploratory plots of environmental variables depending on species distribution data	42
4.4 Correlation analysis	44
4.5 Variance Inflation Factor (VIF)	46
5 Shape Constrained-Generalized Additive Models	49
5.1 Model fit	50
5.2 Model selection	57
6 Model validation	63
6.1 Optimum threshold	64
6.2 k-fold validation	66

7 Prediction and maps	69
7.1 Prepare environmental data	70
7.2 Projection	71
7.3 Mapping	71
8 Acknowledgements	73

About

This is a short tutorial for constructing Species Distribution Models in R using Shape-Constrained Generalized Additive Models [Pya and Wood, 2015], based on the development and application to marine fish by Citores et al. [2020].

The code is available in AZTI's github repository and the book is readily available here. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0)

To cite this book, please use:

Valle, M., Citores, L., Ibaibarriaga, L., Chust, C. (2023) GAM-NICHE: Shape-Constrained GAMs to build Species Distribution Models under the ecological niche theory. AZTI. <https://doi.org/10.57762/fzpy-6w51>

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]. SDMs are widely used as a tool for understanding species spatial ecology and are also known as ecological niche models (ENM) or habitat suitability models.

According to ecological niche theory, species response curves are unimodal with respect to environmental gradients [Hutchinson, 1957]. While a variety of statistical methods have been developed for species distribution modelling, a general problem with most of these habitat modelling approaches is that the estimated response curves can display biologically implausible shapes which do not respect ecological niche theory. This is because species response curves are fit statistically with any assumption or restriction, which sometimes do not respect the ecological niche theory. To better understand species response to environmental changes, SDMs should consider theoretical background such as the ecological niche theory and pursue the unimodality of the response curves with respect to environmental gradients.

This book provides a tutorial on how to use Shape-Constrained Generalized Additive Models (SC-GAMs) [Pya and Wood, 2015] to build SDMs under the ecological niche theory framework [Citores et al., 2020]. SC-GAMs impose monotonicity and concavity constraints in the linear predictor of the GAMs and avoid overfitting. SC-GAM is an effective alternative to fitting nonsymmetric parametric response curves, while retaining the unimodality constraint, required by ecological niche theory, for direct variables and limiting factors.

The book is organised following the key steps in good modelling practice of SDMs [Elith and Leathwick, 2009]. First, presence data of a selected species are downloaded from GBIF/OBIS global public 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

Presence-absence data

In this chapter we first, download occurrence data from global open-access datasets such as Global Biodiversity Information Facility (GBIF, <https://www.gbif.org/>) and Ocean Biodiversity Information System (OBIS, <https://obis.org/>); second, clean downloaded data reformatting, renaming fields and removing outliers data; and lastly, we generate a set of pseudoabsence points along the defined study area.

First we load a list of required libraries.

```
requiredPackages <- c("here", "rstudioapi",
  "ggplot2", "robis", "rgbif", "CoordinateCleaner",
  "sf", "data.table", "dplyr", "tidyverse",
  "marmap", "tidyverse", "scales", "ggridges",
  "maps", "mapdata", "mapproj", "mapplots",
  "gridExtra", "lubridate", "raster")
```

We run a function to install the required packages that are not in our system and load all the required packages.

```
install_load_function <- function(pkg) {
  new.pkg <- pkg[!(pkg %in% installed.packages() [
    "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}

install_load_function(requiredPackages)
```

```
##          here      rstudioapi     ggplot2       robis
```

```

##          TRUE      TRUE      TRUE      TRUE
##   rgbif CoordinateCleaner      sf  data.table
##          TRUE      TRUE      TRUE      TRUE
##   dplyr    tidyverse      marmap  tidyverse
##          TRUE      TRUE      TRUE      TRUE
##   scales    ggridges      maps  mapdata
##          TRUE      TRUE      TRUE      TRUE
##   mapproj    mapplots  gridExtra  lubridate
##          TRUE      TRUE      TRUE      TRUE
##   raster
##          TRUE

```

We define some overall settings.

```

# General settings for ggplot
# (black-white background, larger
# base_size)
theme_set(theme_bw(base_size = 16))

```

2.1 Download presence data

In this section we download presence data from global public datasets.

To do so, we first define a study area, in this case we select the Atlantic Ocean based on the The Food and Agriculture Organization (FAO) Major Fishing Areas for Statistical Purposes and we remove Black Sea subarea.

```

# we could download the shapefile with
# the FAO fishing areas from the url
# where FAO shapefile is stored
# uncommenting the next two lines:

# url<-'https://www.fao.org/fishery/geoserver/area/ows?service=WFS&version=1.0.0&request=GetFeature&featureType=FAO_AREAS'
# download.file(url, 'data/spatial/FAO_AREAS.zip', mode='wb')

# Unzip downloaded file
unzip(here::here("data", "spatial", "FAO_AREAS.zip"),
       exdir = "data/spatial")

# Load FAO (spatial multipolygon)
FAO <- st_read(file.path("data", "spatial",
                        "FAO_AREAS.shp"))

## Reading layer `FAO_AREAS` from data source

```

```

## `C:\USE\GitHub\gam-niche\data\spatial\FAO_AREAS.shp` using driver `ESRI Shapefile'
## Simple feature collection with 50 features and 15 fields
## Geometry type: MULTIPOLYGON
## Dimension: XY
## Bounding box: xmin: -180 ymin: -85.58276 xmax: 180 ymax: 89.99
## Geodetic CRS: WGS 84

# Select Atlantic Ocean FAO Area
FAO_Atl <- FAO[FAO$OCEAN == "Atlantic", ]

# Select Black Sea subarea
Black_Sea <- FAO_Atl[FAO_Atl$ID == "20",
  ]

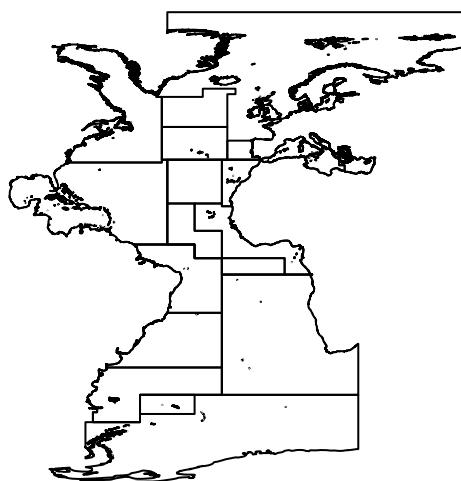
# Transform to sf objects
FAO_Atl.sf <- st_as_sf(FAO_Atl)
Black_Sea.sf <- st_as_sf(Black_Sea)

# Remove Black sea using st_difference
# (reverse of st_intersection)
FAO_Atl_no_black_sea <- st_difference(FAO_Atl.sf,
  Black_Sea.sf) %>%
  dplyr::select(F_AREA)

# Transform to spatial polygons
# dataframe
study_area <- sf:::as_Spatial(FAO_Atl_no_black_sea)

plot(study_area)

```



```
# Remove unused files
rm(FAO, FAO_Atl, FAO_Atl.sf, FAO_Atl_no_black_sea,
   Black_Sea, Black_Sea.sf)
```

Download occurrence data from OBIS and GBIF using scientific name.

In this case we select Albacore tuna species (*Thunnus alalunga*). This can take a long time, so we upload data downloaded previously.

```
# To get data from OBIS
# mydata.obis<-robis::occurrence(scientificname='Thunnus
# alalunga')

# To get data from GBIF
# mydata.gbif<-occ_data(scientificName='Thunnus
# alalunga', hasCoordinate = TRUE,
# limit=100000)$data

# Given that this can take a long time,
# we upload data downloaded previously
load(here::here("data", "occurrences", "mydata.obis.RData"))
load(here::here("data", "occurrences", "mydata.gbif.RData"))
```

We now check the downloaded data and select the fields of interest.

```
# Check names for GBIF data
names(mydata.gbif)
```

```
## [1] "key"                                "scientificName"
## [3] "decimalLatitude"                      "decimalLongitude"
## [5] "issues"                               "datasetKey"
## [7] "publishingOrgKey"                     "installationKey"
## [9] "publishingCountry"                    "protocol"
## [11] "lastCrawled"                          "lastParsed"
## [13] "crawlId"                             "hostingOrganizationKey"
## [15] "basisOfRecord"                        "occurrenceStatus"
## [17] "taxonKey"                            "kingdomKey"
## [19] "phylumKey"                           "orderKey"
## [21] "familyKey"                           "genusKey"
## [23] "speciesKey"                          "acceptedTaxonKey"
## [25] "acceptedScientificName"              "kingdom"
## [27] "phylum"                             "order"
## [29] "family"                            "genus"
## [31] "species"                           "genericName"
## [33] "specificEpithet"                   "taxonRank"
```

```

## [35] "taxonomicStatus"
## [37] "dateIdentified"
## [39] "year"
## [41] "day"
## [43] "modified"
## [45] "references"
## [47] "isInCluster"
## [49] "recordedBy"
## [51] "geodeticDatum"
## [53] "country"
## [55] "identifier"
## [57] "informationWithheld"
## [59] "collectionCode"
## [61] "occurrenceID"
## [63] "catalogNumber"
## [65] "eventTime"
## [67] "identificationID"
## [69] "stateProvince"
## [71] "occurrenceRemarks"
## [73] "datasetID"
## [75] "footprintWKT"
## [77] "county"
## [79] "nameAccordingTo"
## [81] "individualCount"
## [83] "waterBody"
## [85] "otherCatalogNumbers"
## [87] "recordNumber"
## [89] "vernacularName"
## [91] "language"
## [93] "identificationRemarks"
## [95] "municipality"
## [97] "higherGeography"
## [99] "island"
## [101] "locality"
## [103] "startDayOfYear"
## [105] "higherClassification"
## [107] "disposition"
## [109] "organismQuantity"
## [111] "samplingProtocol"
## [113] "coordinatePrecision"
## [115] "nomenclaturalCode"
## [117] "taxonRemarks"
## [119] "bibliographicCitation"
## [121] "locationRemarks"
## [123] "http://unknown.org/license"
## [125] "http://unknown.org/rightsHolder"
"iucnRedListCategory"
"coordinateUncertaintyInMeters"
"month"
"eventDate"
"lastInterpreted"
"license"
"datasetName"
"identifiedBy"
"countryCode"
"rightsHolder"
"http://unknown.org/nick"
"verbatimEventDate"
"gbifID"
"taxonID"
"institutionCode"
"http://unknown.org/captive"
"continent"
"verbatimLocality"
"lifeStage"
"eventID"
"originalNameUsage"
"identificationVerificationStatus"
"networkKeys"
"elevation"
"institutionKey"
"preparations"
"acceptedNameUsage"
"institutionID"
"type"
"projectId"
"collectionKey"
"georeferenceProtocol"
"endDayOfYear"
"fieldNumber"
"collectionID"
"materialSampleID"
"programmeAcronym"
"organismQuantityType"
"locationAccordingTo"
"georeferencedDate"
"associatedReferences"
"ownerInstitutionCode"
"habitat"
"depth"
"taxonConceptID"
"depthAccuracy"

```

```

## [127] "dynamicProperties"           "elevationAccuracy"
## [129] "rights"                     "georeferenceSources"
## [131] "georeferenceRemarks"         "name"
## [133] "associatedSequences"        "establishmentMeans"
## [135] "georeferenceVerificationStatus" "accessRights"
## [137] "georeferencedBy"             "verbatimSRS"
## [139] "previousIdentifications"     "locationID"
## [141] "acceptedNameUsageID"          "http://unknown.org/language"
## [143] "http://unknown.org/modified"   "samplingEffort"
## [145] "verbatimDepth"                "behavior"
## [147] "eventRemarks"                 "footprintSRS"
## [149] "namePublishedInYear"          "verbatimCoordinateSystem"
## [151] "parentNameUsage"              "http://unknown.org/taxonRankID"
## [153] "http://unknown.org/species"    "higherGeographyID"
## [155] "islandGroup"                  "organismID"
## [157] "distanceFromCentroidInMeters" "http://unknown.org/orders"
## [159] "typeStatus"

# Select columns of interest
mydata.gbif <- mydata.gbif %>%
  dplyr::select("acceptedScientificName",
    "decimalLongitude", "decimalLatitude",
    "year", "month", "day", "eventDate",
    "depth")

# Check names in for OBIS data
names(mydata.obis)

```

```

## [1] "infraphylum"                  "country"
## [3] "date_year"                    "scientificNameID"
## [5] "year"                         "scientificName"
## [7] "dropped"                      "gigaclassid"
## [9] "aphiaID"                       "decimalLatitude"
## [11] "subclassid"                   "gigaclass"
## [13] "infraphylumid"                "phylumid"
## [15] "familyid"                     "catalogNumber"
## [17] "basisOfRecord"                 "terrestrial"
## [19] "id"                            "day"
## [21] "parvphylum"                   "order"
## [23] "dataset_id"                   "locality"
## [25] "decimalLongitude"              "collectionCode"
## [27] "date_end"                     "speciesid"
## [29] "date_start"                   "month"
## [31] "genus"                        "eventDate"
## [33] "brackish"                     "absence"

```

```

## [35] "subfamily"
## [37] "originalScientificName"
## [39] "subphylumid"
## [41] "institutionCode"
## [43] "class"
## [45] "waterBody"
## [47] "classid"
## [49] "species"
## [51] "subclass"
## [53] "category"
## [55] "parvphylumid"
## [57] "flags"
## [59] "shoredistance"
## [61] "bathymetry"
## [63] "minimumDepthInMeters"
## [65] "depth"
## [67] "verbatimCoordinates"
## [69] "individualCount"
## [71] "modified"
## [73] "occurrenceID"
## [75] "taxonRank"
## [77] "associatedReferences"
## [79] "coordinateUncertaintyInMeters"
## [81] "footprintWKT"
## [83] "recordNumber"
## [85] "stateProvince"
## [87] "recordedBy"
## [89] "language"
## [91] "license"
## [93] "datasetName"
## [95] "accessRights"
## [97] "county"
## [99] "lifeStage"
## [101] "samplingProtocol"
## [103] "eventID"
## [105] "identifiedBy"
## [107] "minimumElevationInMeters"
## [109] "dateIdentified"
## [111] "georeferenceProtocol"
## [113] "organismQuantity"
## [115] "startDayOfYear"
## [117] "typeStatus"
## [119] "acceptedNameUsage"
## [121] "countryCode"
## [123] "references"
## [125] "endDayOfYear"
"genusid"
"marine"
"subfamilyid"
"date_mid"
"orderid"
"kingdom"
"phylum"
"subphylum"
"family"
"kingdomid"
"node_id"
"sss"
"sst"
"maximumDepthInMeters"
"sex"
"coordinatePrecision"
"occurrenceRemarks"
"occurrenceStatus"
"materialSampleID"
"scientificNameAuthorship"
"datasetID"
"bibliographicCitation"
"vernacularName"
"specificEpithet"
"georeferenceRemarks"
"continent"
"rightsHolder"
"type"
"ownerInstitutionCode"
"geodeticDatum"
"dynamicProperties"
"samplingEffort"
"footprintsRS"
"parentEventID"
"eventRemarks"
"georeferencedBy"
"maximumElevationInMeters"
"behavior"
"verbatimDepth"
"organismQuantityType"
"collectionID"
"institutionID"
"preparations"
"otherCatalogNumbers"
"fieldNumber"
"eventTime"

```

```

## [127] "locationID"                      "taxonRemarks"
## [129] "georeferencedDate"                 "verbatimEventDate"
## [131] "identificationRemarks"             "nomenclaturalCode"
## [133] "taxonomicStatus"                  "higherClassification"
## [135] "higherGeography"                  "verbatimLatitude"
## [137] "verbatimLongitude"                "establishmentMeans"
## [139] "verbatimLocality"                 "georeferenceVerificationStatus"
## [141] "island"                           "identificationQualifier"
## [143] "identificationID"                 "informationWithheld"
## [145] "islandGroup"                     "acceptedNameUsageID"
## [147] "locationRemarks"                 "verbatimSRS"
## [149] "previousIdentifications"

# Select columns of interest
mydata.obis <- mydata.obis %>%
  dplyr::select("scientificName", "decimalLongitude",
    "decimalLatitude", "date_year", "month",
    "day", "eventDate", "depth", "bathymetry",
    "occurrenceStatus", "sst")

```

Reformat the data adding a new field and renaming some columns from mydata.gbif dataframe in order to have the same columns and be able to join both downloaded datasets.

```

mydata.gbif <- mydata.gbif %>%
  dplyr::rename(scientificName = "acceptedScientificName") %>%
  dplyr::rename(date_year = "year") %>%
  dplyr::mutate(bathymetry = NA) %>%
  dplyr::mutate(occurrenceStatus = 1) %>%
  dplyr::mutate(sst = NA)

# Join data from OBIS and GBIF
mydata.fus <- rbind(mydata.obis, mydata.gbif)

# Assign unique scientific name
mydata.fus <- mydata.fus %>%
  dplyr::mutate(scientificName = paste(mydata.obis$scientificName[1]))

# Remove unused files
rm(mydata.gbif, mydata.obis)

```

We now clean downloaded raw data.

```

# Give date format to eventDate and
# fill out month and date_year columns
mydata.fus$eventDate <- as.Date(mydata.fus$eventDate)
mydata.fus$date_year <- as.numeric(mydata.fus$date_year)
mydata.fus$month <- as.numeric(mydata.fus$month)

# Assign 1 value to occurrenceStatus
mydata.fus <- mydata.fus %>%
  dplyr::mutate(occurrenceStatus = 1)

```

We remove outliers based on distance method (total distance= 1000 km) available in the CoordinateCleaner package.

```

out.dist <- cc_outl(x = mydata.fus, lon = "decimalLongitude",
  lat = "decimalLatitude", species = "scientificName",
  method = "distance", tdi = 1000, thinning = T,
  thinning_res = 0.5, value = "flagged")

# Remove outliers from the data
mydata.fus <- mydata.fus[out.dist, ]

```

Remove duplicates.

```

# First create a vector containing
# longitude, latitude and event date
# information
date <- cbind(mydata.fus$decimalLongitude,
  mydata.fus$decimalLatitude, mydata.fus$eventDate)

# Remove the duplicated records
mydata.fus <- mydata.fus[!duplicated(date),
  ]

# Remove unused files
rm(date)

```

Mask retrieved occurrence data to our study area and add bathymetry value to each occurrence point.

```

# Assign coordinate format and
# projection to be able to use FAO
# Atlantic as a mask
dat <- data.frame(cbind(mydata.fus$decimalLongitude,
  mydata.fus$decimalLatitude))

```

```

ptos <- as.data.table(dat, keep.columnnames = TRUE)

coordinates(ptos) <- ~X1 + X2

# Assign projection
proj4string(ptos) <- proj4string(study_area)

# Select only occurrences from FAO
# Atlantic
match2 <- data.frame(subset(mydata.fus, !is.na(over(ptos,
study_area)[, 1])))

# Extract the FAO area of each point
match3 <- data.frame(subset(over(ptos, study_area),
!is.na(over(ptos, study_area)[, 1])))

# Create data frame with area, name,
# long, lat and year
df0 <- cbind(F_AREA = match3$F_AREA, match2[,,
c("F_AREA", "scientificName", "decimalLongitude",
"decimalLatitude", "date_year", "occurrenceStatus")])

# Rename some columns
names(df0)[3:5] <- c("LON", "LAT", "YEAR")

options(timeout = 600)

# Add bathymetry from NOAA
bathy <- marmap::getNOAA.bathy(lon1 = -100,
lon2 = 30, lat1 = -41, lat2 = 55, resolution = 1,
keep = TRUE, antimeridian = FALSE, path = here::here("data",
"spatial"))

# instead, we could upload a file
# already downloaded
# load(here::here('data',
# 'spatial','bathy.RData'))

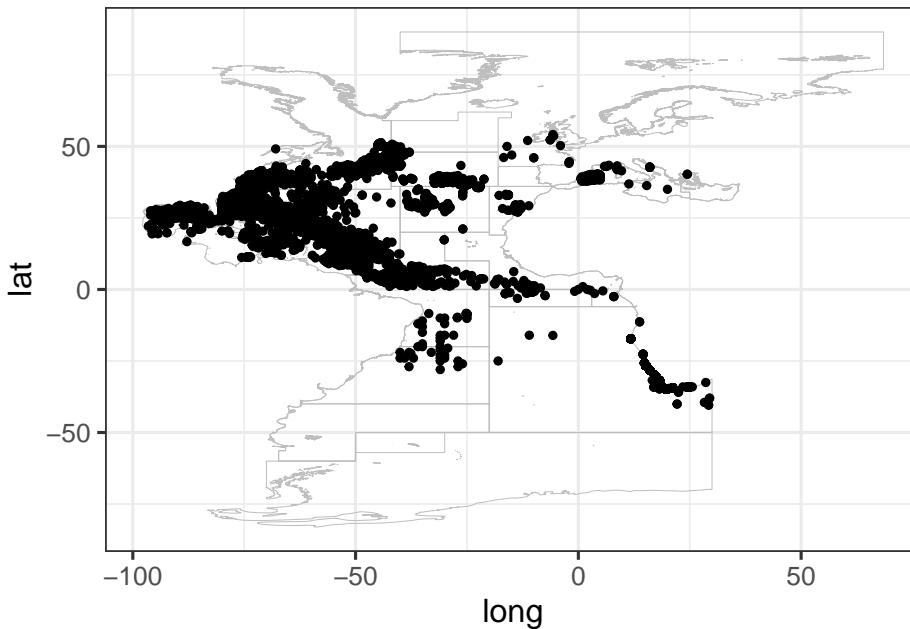
df0$bathymetry <- get.depth(bathy, df0[,,
c("LON", "LAT")], locator = F)$depth

# Remove unused files
rm(mydata.fus, match2, match3, dat, ptos)

```

We plot the occurrence data.

```
ggplot() + geom_path(data = study_area, aes(x = long,
y = lat, group = group), color = "gray",
linewidth = 0.2) + geom_point(data = df0,
aes(x = LON, y = LAT))
```



And, we save the data.

```
save(df0, file = "data/occurrences/occ.RData")
save(study_area, file = "data/spatial/study_area.RData")
```

2.2 Create pseudo-absence data

After saving presence data for the species of interest we need to generate absence information in order to work with logistic regression SDM afterwards. In this case we create pseudo-absence data with a constant prevalence of 50% [McPherson et al., 2004]; [Barbet-Massin et al., 2012].

For that, we generate a buffer around each presence data point, with a radius of 100km, where no points can be generated.

Before starting the pseudo-absence generation process, we delete observations in land (positive bathymetry) and select data between 2000 and 2014 (same temporal range as the environmental data that we will download later).

```
# Remove points in land
df0 <- subset(df0, bathymetry < 0)

# Select only years from 2000 to 2014
df0 <- subset(df0, YEAR <= 2014 & YEAR >=
  2000)
```

Then we transform the presence data frame to “SpatialPointsDataFrame” class object and to “sf” class object so we can operate and plot easily with spatial data tools. We can see the characteristics of the object through its summary.

```
# Convert to spatial point data frame
df <- df0
coordinates(df) <- ~LON + LAT
crs(df) <- crs(study_area)

# Convert to sf
df.sf <- st_as_sf(df)
study_area.sf <- st_union(st_as_sf(study_area))

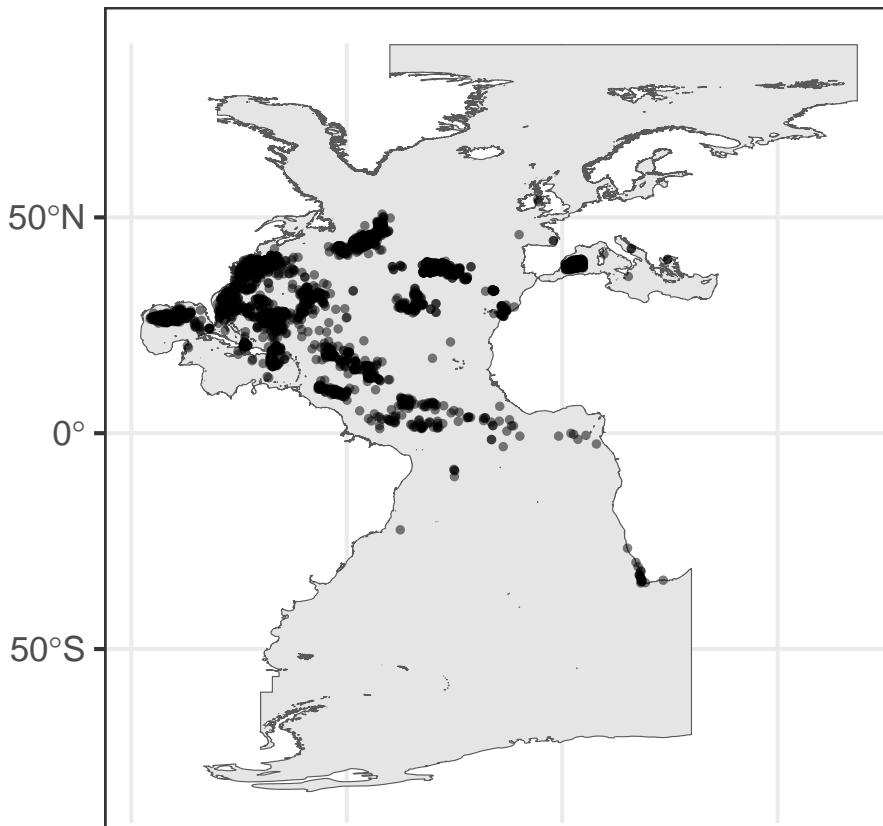
summary(df)

## Object of class SpatialPointsDataFrame
## Coordinates:
##      min     max
## LON -95.65 24.45000
## LAT -34.71 54.15014
## Is projected: FALSE
## proj4string : [+proj=longlat +datum=WGS84 +no_defs]
## Number of points: 14903
## Data attributes:
##      F_AREA      scientificName      YEAR occurrenceStatus
## Length:14903 Length:14903    Min.   :2000  Min.   :1
## Class :character Class :character  1st Qu.:2001  1st Qu.:1
## Mode  :character Mode  :character  Median :2002  Median :1
##                                         Mean   :2003  Mean   :1
##                                         3rd Qu.:2004  3rd Qu.:1
##                                         Max.   :2013  Max.   :1
##      bathymetry
##      Min.   :-8404.09
##      1st Qu.:-1916.92
##      Median :-1030.77
##      Mean   :-1433.61
##      3rd Qu.:-351.32
```

```
## Max. : -1.13
```

We can easily plot sf objects using ggplot, here we plot the area of study and the presence data points.

```
ggplot(study_area_sf) + geom_sf() + geom_sf(data = st_union(df.sf),
      size = 1, alpha = 0.5)
```

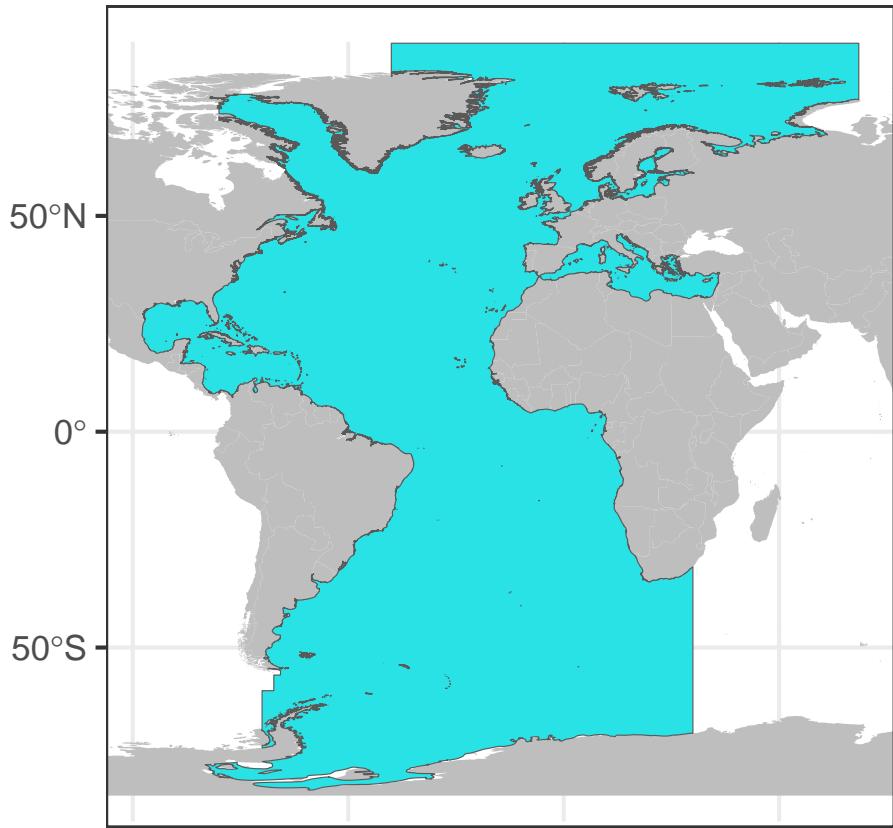


We save a base plot (p0) with the world map and our area of study in blue.

```
# Basic ggplot
global <- map_data("worldHires")

p0 <- ggplot() + annotation_map(map = global,
      fill = "grey") + geom_sf(data = study_area_sf,
      fill = 5)

print(p0)
```



In order to generate a buffer around each occurrence data point, we need to work with euclidean distances, so first, we need to transform the decimal latitude and longitude values to UTM.

```
# Function to find your UTM.
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)))
```

```
## [1] 32623
```

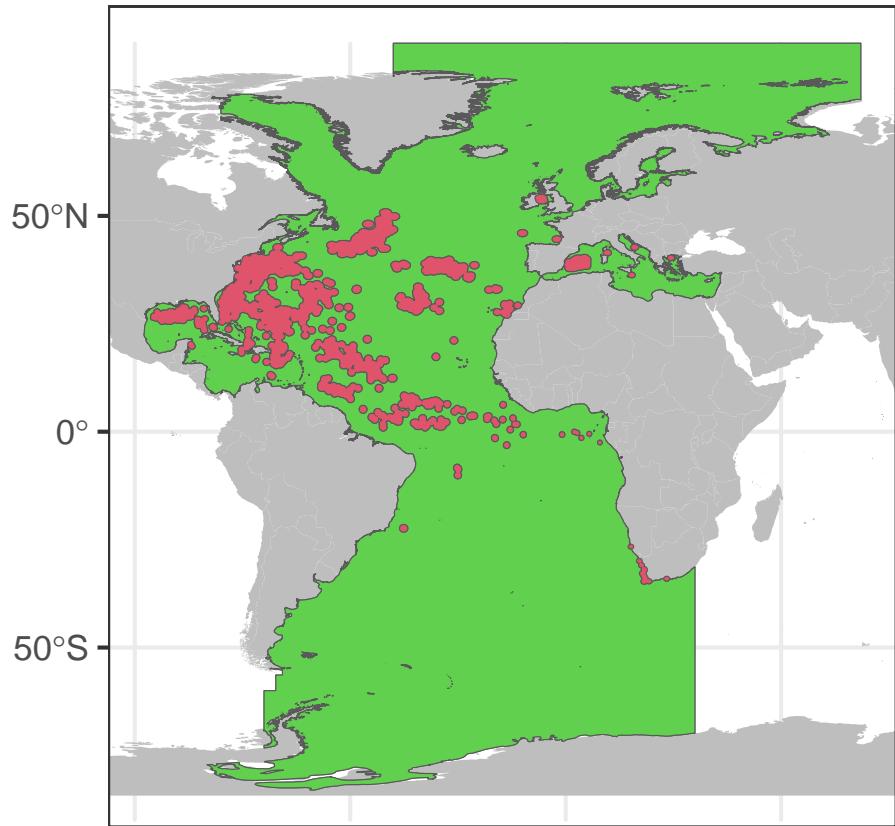
```
# Transform study_area and data points
# to UTM (in m)
aux <- st_transform(study_area.sf, EPSG_2_UTM)
df.sf.utm <- st_transform(df.sf, EPSG_2_UTM)
```

Now, we can create buffers of 100 km around the points and join the resulting polygons. Then this buffer is intersected with the area of study defining the area where the pseudo-absences can be generated. To visualize the defined areas, we plot the buffers in red and the area that we will use to generate pseudo-absences in green.

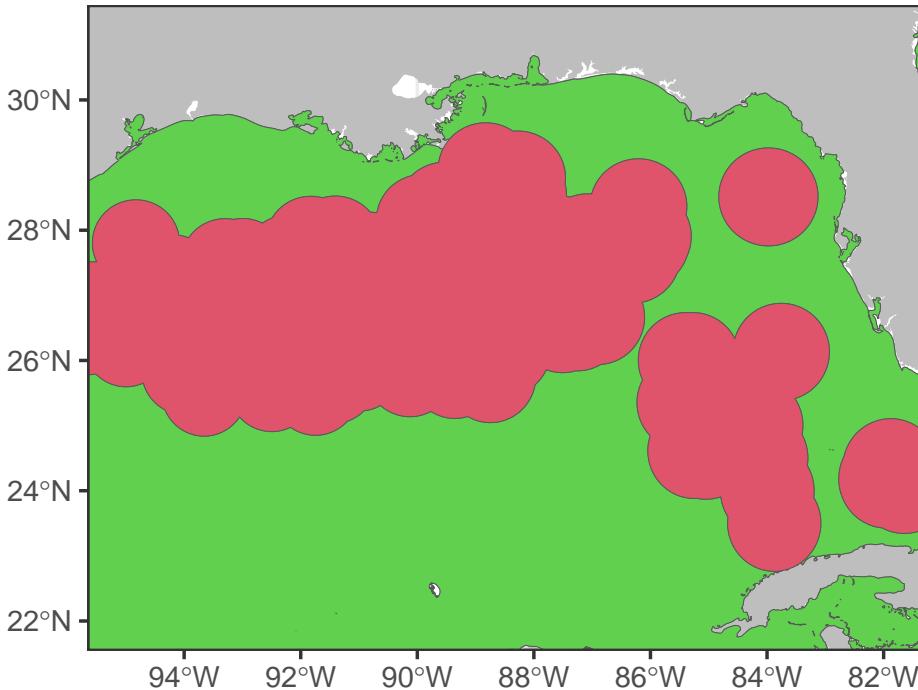
```
# Create buffers of 100000m
buffer <- st_buffer(df.sf.utm, dist = 1e+05)
buffer <- st_union(buffer)

# Intersect the area with the buffer
aux0 <- st_difference(aux, buffer)

# ggplot for all data
p0 + geom_sf(data = aux0, fill = 3) + geom_sf(data = buffer,
fill = 2)
```



```
# zoom
p0 + geom_sf(data = aux0, fill = 3) + geom_sf(data = buffer,
      fill = 2) + coord_sf(xlim = c(-95, -82),
      ylim = c(22, 31))
```



We create a data frame for pseudo-absences with the same dimensions as the presences data frame.

```
# Generate the pseudo-absence data
# frame
pseudo <- matrix(data = NA, nrow = dim(df0)[1],
                   ncol = dim(df0)[2])
pseudo <- data.frame(pseudo)
names(pseudo) <- names(df0)
```

To generate the pseudo-absence data points, we sample randomly from the defined area and we extract their latitude and longitude to incorporate them in the created data frame. We set the occurrenceStatus equal to 0 as they are absences.

```
# Set the seed
set.seed(1)

# Sample from the defined area
rp.sf <- st_sample(aux0, size = dim(df.sf.utm)[1],
                     type = "random") # randomly sample points

# Transform to lat and lon and extract
```

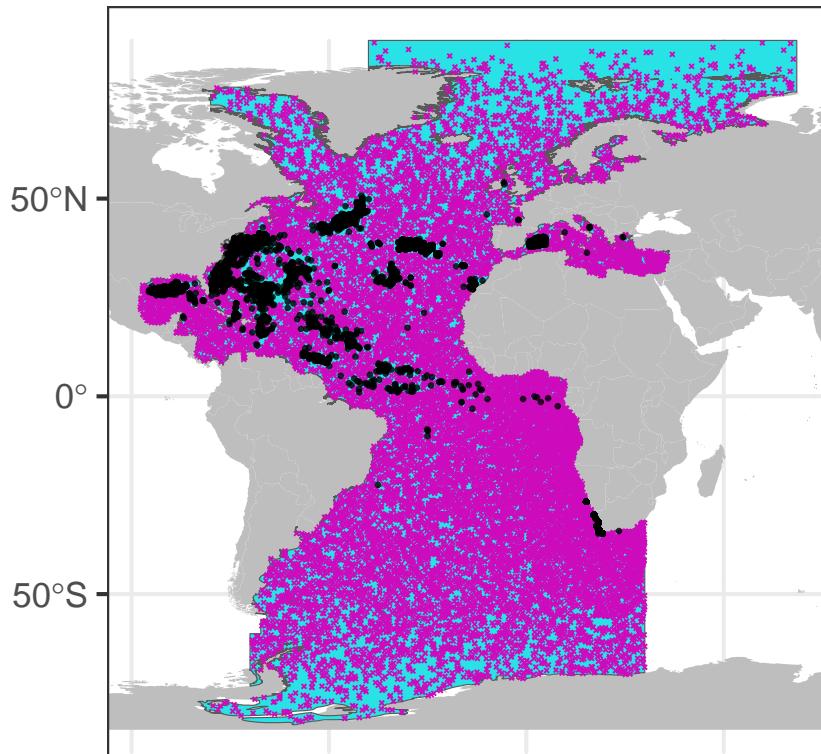
```
# coordinates as data.frame
rp.sf <- st_transform(rp.sf, 4326)
rp <- as.data.frame(st_coordinates(rp.sf))
pseudo$LON <- rp$X
pseudo$LAT <- rp$Y

# Complete the rest of columns
pseudo$scientificName <- df0$scientificName
pseudo$occurrenceStatus <- 0
```

We can plot the generated pseudo-absence data (in pink) in the map, together with the presence data points (in black).

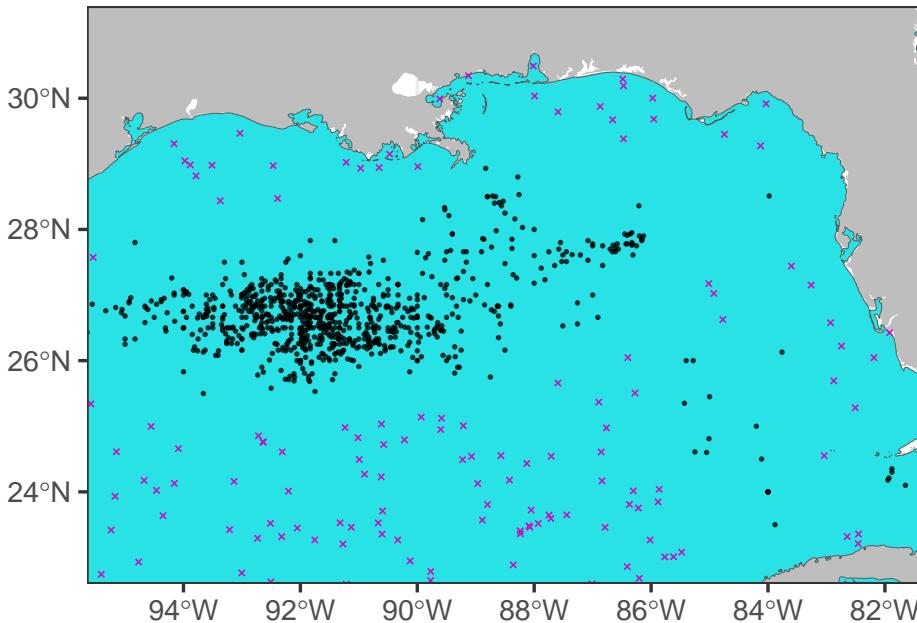
```
p0 + geom_sf(data = rp.sf, col = 6, shape = 4,
size = 0.5) + geom_sf(data = df.sf.utm,
col = 1, alpha = 0.8, size = 0.5) + ggtitle(unique(df$scientificName))
```

Thunnus alalunga



```
# Zoom
p0 + geom_sf(data = rp.sf, col = 6, shape = 4,
  size = 1) + geom_sf(data = df.sf.utm,
  col = 1, alpha = 0.8, size = 0.5) + coord_sf(xlim = c(-95,
-82), ylim = c(23, 31)) + ggtitle(unique(df$scientificName))
```

Thunnus alalunga



Finally we join the presence and pseudo-absence data frames selecting the columns of interest and save the new data frame.

```
# Join the two data sets
PAdat <- rbind(df0, pseudo)[, c("scientificName",
  "LON", "LAT", "YEAR", "occurrenceStatus")]

# Save the final dataset of occurrence
# and pseudo-absence points
save(list = c("PAdat"), file = file.path("data",
  "outputs_for_modelling", file = "PAdat.RData"))
```


Chapter 3

Environmental data

In this chapter we first, download environmental data from a public repository; second, we crop the data to our area of interest, and we save it as raster stack.

As in Chapter 2, first, we load a list of required libraries.

```
requiredPackages <- c("here", "rstudioapi",
  "ggplot2", "knitr", "kableExtra", "raster",
  "dplyr", "sdmpredictors")
```

We run a function to install the required packages that are not in our system and load all the required packages.

```
install_load_function <- function(pkg) {
  new.pkg <- pkg[!(pkg %in% installed.packages() [
    "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}

install_load_function(requiredPackages)
```

```
##      here    rstudioapi     ggplot2      knitr    kableExtra
##      TRUE        TRUE        TRUE        TRUE        TRUE
##      raster      dplyr  sdmpredictors      TRUE        TRUE
##      TRUE        TRUE        TRUE        TRUE
```

We define some overall settings.

dataset_code	terrestrial	marine	url	description
WorldClim	TRUE	FALSE	http://www.worldclim.org/	WorldClim is a set of global climate datasets.
Bio-ORACLE	FALSE	TRUE	https://bio-oracle.org/	Bio-ORACLE is a set of ocean rasters.
MARSPEC	FALSE	TRUE	http://marspec.org/	MARSPEC is a set of high-resolution ocean rasters.
ENVIREM	TRUE	FALSE	https://envirem.github.io/	The ENVIREM dataset is a collection of environmental datasets.
Freshwater	TRUE	FALSE	https://www.earthenv.org/streams	NA

```
# General settings for ggplot
# (black-white background, larger
# base_size)
theme_set(theme_bw(base_size = 16))
```

3.1 Download from public repositories

Environmental data can be available from different sources. In this case, we used the Bio-ORACLE (ocean rasters for analyses of climate and environment) database [Tyberghein et al., 2012, Assis et al., 2017]. These data are publicly available and are easily accessible from the `sdmpredictors` package.

We can check the list of available datasets with the function `list_datasets` from `sdmpredictors` package, as follows:

```
mydat <- list_datasets()

# to solve problems with encoding in
# column 5:
mydat[, 5] <- iconv(mydat[, 5], to = "UTF-8")

kable(mydat) %>%
  kable_styling("striped") %>%
  scroll_box(height = "600px", width = "100%")
```

By default this function returns all the supported datasets. To return only marine datasets we can set the `marine` argument equal to TRUE or equivalently we could set the `terrestrial` and `freshwater` arguments equal to FALSE:

```
mydat <- list_datasets(marine = T)
# or equivalently: mydat <-
# list_datasets(terrestrial=F,
# freshwater=F) mydat <-
# list_datasets(marine=T,
# terrestrial=F, freshwater=F)
```

There are two datasets (Bio-ORACLE and MARSPEC) that have marine data. The function `list_layers` returns information on the layers of one or more datasets. So, we can see the layers available in the Bio-ORACLE dataset as follows:

```
mytab <- list_layers("Bio-ORACLE")

kable(mytab) %>%
  kable_styling("striped") %>%
  scroll_box(height = "600px", width = "100%")
```

Once we identify the dataset and the layers we are interested on, we can extract their details from the list. In this case, we download data on chlorophyll, salinity, diffuse attenuation coefficient and sea surface temperature.

```
target <- c("B02_chlomean_ss", "B02_salinitymean_ss",
          "B0_damean", "B0_sstmean")

# Extract details from the list
myvars <- mytab %>%
  dplyr::filter(mytab$layer_code %in% target)

myvars$name

## [1] "Diffuse attenuation coefficient at 490 nm (mean)"
## [2] "Sea surface temperature (mean)"
## [3] "Chlorophyll concentration (mean)"
## [4] "Sea surface salinity (mean)"
```

And we can download them using the function `load_layers`

```
# Download layers
myBioracle.layers <- load_layers(c("B02_chlomean_ss",
                                     "B02_salinitymean_ss", "B0_damean", "B0_sstmean"))
```

Note that the resulting object is a `rasterStack`, where each variable is a layer

```
class(myBioracle.layers)
```

```
## [1] "RasterStack"
## attr(,"package")
## [1] "raster"
```

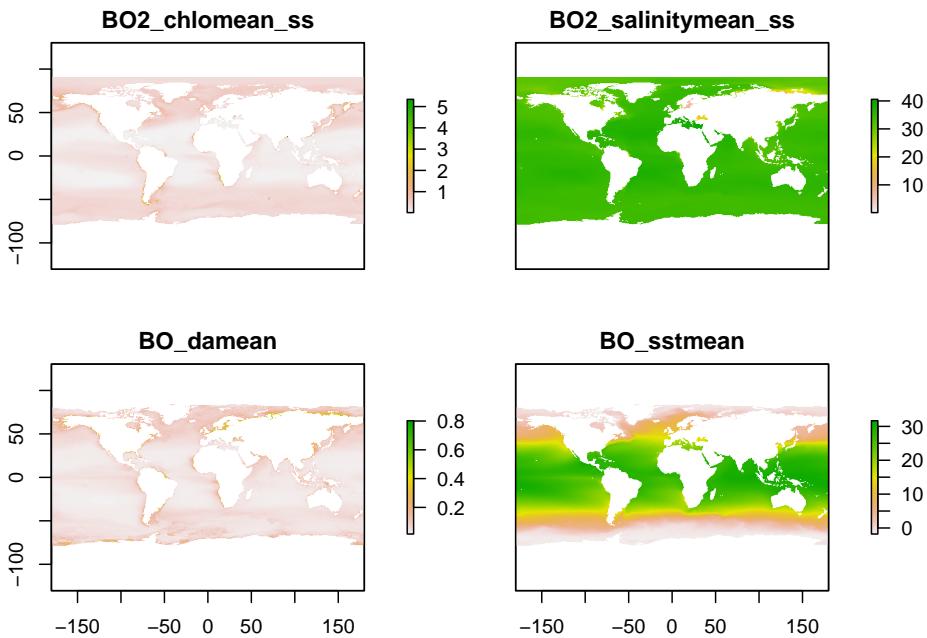
	dataset_code	layer_code	name
74	Bio-ORACLE	BO_cloudmax	Cloud fraction (maximum)
452	Bio-ORACLE	BO22_calcite	Calcite (mean)
453	Bio-ORACLE	BO22_chlomax	Cloud fraction (maximum)
454	Bio-ORACLE	BO22_chlomean	Cloud fraction (mean)
455	Bio-ORACLE	BO22_chlomin	Cloud fraction (minimum)
456	Bio-ORACLE	BO22_damax	BO_damax
457	Bio-ORACLE	BO22_damean	BO_damean
458	Bio-ORACLE	BO22_damin	BO_damin
459	Bio-ORACLE	BO22_parmax	BO_parmax
460	Bio-ORACLE	BO22_parmeans	BO_parmeans
461	Bio-ORACLE	BO22_ph	BO_ph
462	Bio-ORACLE	BO_calcite	Calcite (mean)
463	Bio-ORACLE	BO_chlomax	Chlorophyll A (maximum)
464	Bio-ORACLE	BO_chlomean	Chlorophyll A (mean)
465	Bio-ORACLE	BO_chlomin	Chlorophyll A (minimum)
466	Bio-ORACLE	BO_chlorange	Chlorophyll A (range)
467	Bio-ORACLE	BO_cloudmean	Cloud fraction (mean)
468	Bio-ORACLE	BO_cloudmin	Cloud fraction (minimum)
469	Bio-ORACLE	BO_damax	Diffuse attenuation coefficient at 490 nm
470	Bio-ORACLE	BO_damean	Diffuse attenuation coefficient at 490 nm
471	Bio-ORACLE	BO_damin	Diffuse attenuation coefficient at 490 nm
472	Bio-ORACLE	BO_dissox	Dissolved oxygen
473	Bio-ORACLE	BO_nitrate	Nitrate
474	Bio-ORACLE	BO_parmax	Photosynthetically available radiation (maximum)
475	Bio-ORACLE	BO_parmeans	Photosynthetically available radiation (mean)
476	Bio-ORACLE	BO_ph	pH
477	Bio-ORACLE	BO_phosphate	Phosphate
478	Bio-ORACLE	BO_salinity	Salinity
479	Bio-ORACLE	BO_silicate	Silicate
480	Bio-ORACLE	BO_sstmax	Sea surface temperature (maximum)
481	Bio-ORACLE	BO_sstmean	Sea surface temperature (mean)
482	Bio-ORACLE	BO_sstmin	Sea surface temperature (minimum)
483	Bio-ORACLE	BO_sstrange	Sea surface temperature (range)
484	Bio-ORACLE	BO_bathymin	Bathymetry (minimum)
485	Bio-ORACLE	BO_bathymax	Bathymetry (maximum)
486	Bio-ORACLE	BO_bathymean	Bathymetry (mean)
487	Bio-ORACLE	BO2_chlomax_bdmax	Chlorophyll concentration (maximum at mean)
488	Bio-ORACLE	BO2_chlomax_bdmean	Chlorophyll concentration (maximum at mean)
489	Bio-ORACLE	BO2_chlomax_bdmin	Chlorophyll concentration (maximum at mean)
490	Bio-ORACLE	BO2_chlomean_bdmax	Chlorophyll concentration (mean at maximum)
491	Bio-ORACLE	BO2_chlomean_bdmean	Chlorophyll concentration (mean at mean)
492	Bio-ORACLE	BO2_chlomean_bdmin	Chlorophyll concentration (mean at minimum)
493	Bio-ORACLE	BO2_chlomin_bdmax	Chlorophyll concentration (minimum at maximum)
494	Bio-ORACLE	BO2_chlomin_bdmean	Chlorophyll concentration (minimum at mean)
495	Bio-ORACLE	BO2_chlomin_bdmin	Chlorophyll concentration (minimum at minimum)
496	Bio-ORACLE	BO2_chlorange_bdmax	Chlorophyll concentration (range at maximum)
497	Bio-ORACLE	BO2_chlorange_bdmean	Chlorophyll concentration (range at mean)
498	Bio-ORACLE	BO2_chlorange_bdmin	Chlorophyll concentration (range at minimum)
499	Bio-ORACLE	BO2_chloltmax_bdmax	Chlorophyll concentration (longterm maximum)
500	Bio-ORACLE	BO2_chloltmax_bdmean	Chlorophyll concentration (longterm mean)
501	Bio-ORACLE	BO2_chloltmax_bdmin	Chlorophyll concentration (longterm minimum)
502	Bio-ORACLE	BO2_chloltmin_bdmax	Chlorophyll concentration (longterm maximum)
503	Bio-ORACLE	BO2_chloltmin_bdmean	Chlorophyll concentration (longterm mean)

```
myBioracle.layers
```

```
## class      : RasterStack
## dimensions : 2160, 4320, 9331200, 4  (nrow, ncol, ncell, nlayers)
## resolution : 0.08333333, 0.08333333  (x, y)
## extent     : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## names      : BO2_chlomean_ss, BO2_salinitymean_ss, BO_damean, BO_sstmean
## min values :          0.015806,          0.059304,  0.012000, -1.801000
## max values :          5.394896,         40.651300,  0.972000, 32.918000
```

And we can plot it:

```
raster::plot(myBioracle.layers)
```



Note that the functions `dataset_citations` and `layer_citations` provide the bibliographic entries of the datasets and layers for proper citation:

```
print(dataset_citations("Bio-ORACLE"))
```

```
## [1] "Assis J, Tyberghein L, Bosch S, Heroen V, Serrão E, De Clerck O, Tittensor D (2018). \"Bio-ORACLE: A Global Benthic Oracle for Marine Benthos. PLoS ONE 13(1): e0192002. https://doi.org/10.1371/journal.pone.0192002"
## [2] "Tyberghein L, Heroen V, Pauly K, Troupin C, Mineur F, De Clerck O (2012). \"Bio-ORACLE: a global benthic oracle for marine benthos. PLoS ONE 7(10): e46522. https://doi.org/10.1371/journal.pone.0046522"
```

```
print(layer_citations("BO2_chlomean_ss"))
```

```
## [1] "Assis J, Tyberghein L, Bosch S, Heroen V, Serrão E, De Clerck O, Tittensor D (2012). Biogeographic patterns of phytoplankton chlorophyll-a concentration in the world's oceans. Biogeosciences, 9(1), 1-16. doi:10.5194/bg-9-1-2012. URL: https://doi.org/10.5194/bg-9-1-2012"
## [2] "Tyberghein L, Heroen V, Pauly K, Troupin C, Mineur F, De Clerck O (2012). Biogeographic patterns of phytoplankton chlorophyll-a concentration in the world's oceans. Biogeosciences, 9(1), 1-16. doi:10.5194/bg-9-1-2012. URL: https://doi.org/10.5194/bg-9-1-2012"
```

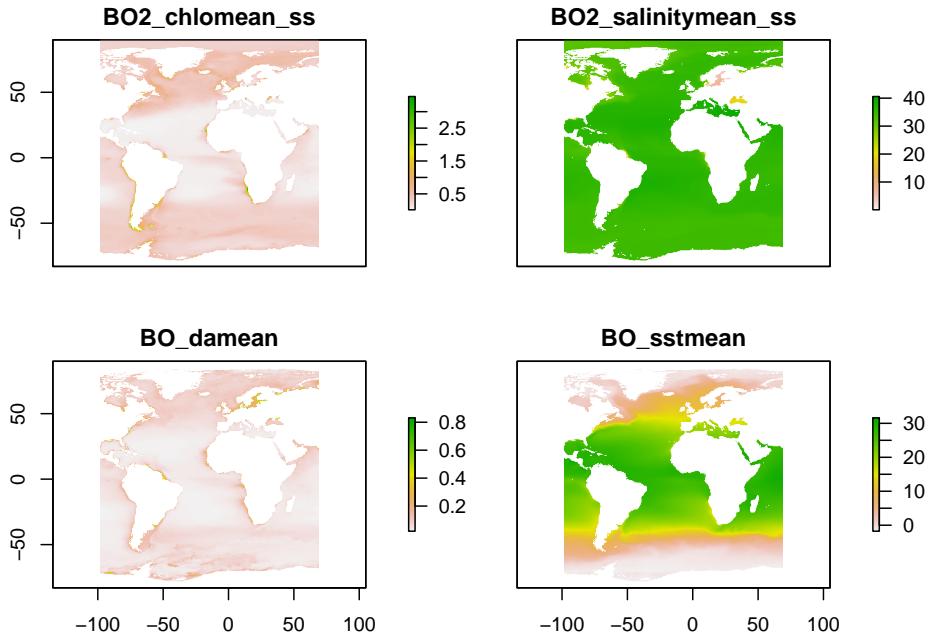
In case we are not interested on the whole area, we can crop the raster objects to the area of interest.

For example, we can load the `study_area` object that is a `SpatialPolygonsDataFrame` that has been created previously and defines the extent of our spatial data and we can crop the `rasterStack` to the same extent:

```
load(here::here("data", "spatial", "study_area.RData"))

mylayers <- crop(myBioracle.layers, extent(study_area))

plot(mylayers)
```



To facilitate subsequent access, the `rasterStack` with the downloaded data is saved in a local folder:

```
writeRaster(mylayers, filename = "data/env/mylayers.tif",
            options = "INTERLEAVE=BAND", overwrite = TRUE)
```

Chapter 4

Prepare the final dataset

In this chapter we first, extract environmental data associated to the presence/pseudo-absence data, we explore the data we got, we check correlation between variables and we calculate the Variance Inflation Factor (VIF) to make a selection of the variables we are going to use in the model.

First, we load a list of required libraries.

```
requiredPackages <- c("here", "rstudioapi",
  "sp", "raster", "dplyr", "tidyverse", "ggplot2",
  "ggcorrplot", "GGally", "HH")
```

We run a function to install the required packages that are not in our system and load all the required packages.

```
install_load_function <- function(pkg) {
  new.pkg <- pkg[!(pkg %in% installed.packages() [
    "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}

install_load_function(requiredPackages)

##      here rstudioapi      sp      raster      dplyr      tidyverse      ggplot2
##      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE
## ggcorrplot      GGally      HH
##      TRUE      TRUE      TRUE
```

We define some overall settings.

```
# General settings for ggplot
# (black-white background, larger
# base_size)
theme_set(theme_bw(base_size = 16))
```

4.1 Extract environmental data associated to species distribution data

Once we have prepared our species distribution data (occurrences and pseudo-absences) and the environmental rasters, we need to merge both sources of data. First, we load the objects created in previous sections:

```
# Load presence-absence data
load(here::here("data", "outputs_for_modelling",
    "PAdatas.RData"))

# Load environmental rasters
mylayers <- stack(here::here("data", "env",
    "mylayers.tif"))
```

Now we can extract the environmental data associated to each of the species data points using the function `extract` from the `raster` package. The method employed is `bilinear` that returns the interpolated value from the four nearest raster cells.

```
raster_ex <- raster::extract(x = mylayers,
    y = PAdatas[, c("LON", "LAT")], method = "bilinear",
    na.rm = TRUE, df = T)

colnames(raster_ex)[-1] <- c("B02_chlomean_ss",
    "B02_salinitymean_ss", "B0_damean", "B0_sstmean")

head(raster_ex)

##   ID B02_chlomean_ss B02_salinitymean_ss B0_damean B0_sstmean
## 1  1      1.20410309          35.42666 0.1973854 16.99439
## 2  2      1.20410309          35.42666 0.1973854 16.99439
## 3  3      0.08455528          36.43070 0.0290000 25.52963
## 4  4      1.20410309          35.42666 0.1973854 16.99439
## 5  5      0.72283228          34.20426 0.0817736 15.21486
## 6  6      1.20410309          35.42666 0.1973854 16.99439
```

4.1. EXTRACT ENVIRONMENTAL DATA ASSOCIATED TO SPECIES DISTRIBUTION DATA37

We merge the presence/pseudo-absence data and the environmental data:

```
data <- cbind(PAdata, raster_ex)
```

We can conduct some quick checks on the new dataset:

```
dim(data)
```

```
## [1] 29806     10
```

```
str(data)
```

```
## 'data.frame': 29806 obs. of 10 variables:
## $ scientificName : chr "Thunnus alalunga" "Thunnus alalunga" "Thunnus alalunga" "Thunnus
## $ LON          : num 18.5 18.5 -76.6 18.5 -69.3 ...
## $ LAT          : num -34.4 -34.4 28.4 -34.4 39.9 ...
## $ YEAR         : num 2004 2004 2000 2000 2000 ...
## $ occurrenceStatus : num 1 1 1 1 1 1 1 1 1 1 ...
## $ ID           : num 1 2 3 4 5 6 7 8 9 10 ...
## $ B02_chlomean_ss : num 1.2041 1.2041 0.0846 1.2041 0.7228 ...
## $ B02_salinitymean_ss: num 35.4 35.4 36.4 35.4 34.2 ...
## $ B0_damean      : num 0.1974 0.1974 0.029 0.1974 0.0818 ...
## $ B0_sstmean     : num 17 17 25.5 17 15.2 ...
```

```
head(data)
```

	scientificName	LON	LAT	YEAR	occurrenceStatus	ID	B02_chlomean_ss	
## 1	Thunnus alalunga	18.4972	-34.3569	2004		1	1	1.20410309
## 5	Thunnus alalunga	18.4972	-34.3569	2004		1	2	1.20410309
## 8	Thunnus alalunga	-76.6000	28.4000	2000		1	3	0.08455528
## 10	Thunnus alalunga	18.4972	-34.3569	2000		1	4	1.20410309
## 11	Thunnus alalunga	-69.3100	39.8800	2000		1	5	0.72283228
## 12	Thunnus alalunga	18.4972	-34.3569	2001		1	6	1.20410309
		B02_salinitymean_ss	B0_damean	B0_sstmean				
## 1		35.42666	0.1973854	16.99439				
## 5		35.42666	0.1973854	16.99439				
## 8		36.43070	0.0290000	25.52963				
## 10		35.42666	0.1973854	16.99439				
## 11		34.20426	0.0817736	15.21486				
## 12		35.42666	0.1973854	16.99439				

```
summary(data)
```

	LON	LAT	YEAR
## scientificName			
## Length:29806	Min. :-97.805	Min. :-82.187	Min. :2000
## Class :character	1st Qu.:-69.221	1st Qu.:-29.933	1st Qu.:2001
## Mode :character	Median :-28.967	Median :24.006	Median :2002
##	Mean :-31.118	Mean : 7.678	Mean :2003
##	3rd Qu.: 2.933	3rd Qu.: 38.250	3rd Qu.:2004
##	Max. : 67.648	Max. : 89.418	Max. :2013
##			NA's :14903
## occurrenceStatus	ID	B02_chlomean_ss	B02_salinitymean_ss
## Min. :0.0	Min. : 1	Min. :0.01585	Min. : 0.1632
## 1st Qu.:0.0	1st Qu.: 7452	1st Qu.:0.08960	1st Qu.:34.5032
## Median :0.5	Median :14904	Median :0.24648	Median :35.4267
## Mean :0.5	Mean :14904	Mean :0.42505	Mean :35.2388
## 3rd Qu.:1.0	3rd Qu.:22355	3rd Qu.:0.68378	3rd Qu.:36.2156
## Max. :1.0	Max. :29806	Max. :3.44767	Max. :39.2467
##		NA's :98	NA's :98
## B0_damean	B0_sstmean		
## Min. :0.02180	Min. :-1.705		
## 1st Qu.:0.03505	1st Qu.:14.852		
## Median :0.05039	Median :18.898		
## Mean :0.07833	Mean :18.117		
## 3rd Qu.:0.08800	3rd Qu.:25.373		
## Max. :0.69655	Max. :30.663		
## NA's :139	NA's :139		

The new dataset has 29806 rows and 10 columns, and there are 139 NA's in the environmental dataset. We remove the points with NA's:

```
# we remove year column because
# pseudoabsences miss this info
data <- data %>%
  dplyr::select(-YEAR) %>%
  na.omit()
```

We check again the dataset:

```
dim(data)
```

```
## [1] 29661     9
```

```
summary(data)

##  scientificName           LON             LAT          occurrenceStatus
##  Length:29661      Min.   :-97.805   Min.   :-77.508   Min.   :0.0000
##  Class  :character    1st Qu.:-69.330   1st Qu.:-29.933   1st Qu.:0.0000
##  Mode   :character    Median :-28.946   Median : 24.088   Median :1.0000
##              Mean   :-31.138   Mean   : 7.792   Mean   :0.5024
##              3rd Qu.: 2.938   3rd Qu.: 38.242   3rd Qu.:1.0000
##              Max.   : 67.241   Max.   : 83.660   Max.   :1.0000
## 
##  ID            B02_chlomean_ss  B02_salinitymean_ss  B0_damean
##  Min.   : 1   Min.   :0.01585   Min.   : 0.1632   Min.   :0.02180
##  1st Qu.: 7416 1st Qu.:0.08954   1st Qu.:34.5100   1st Qu.:0.03504
##  Median :14831 Median :0.24526   Median :35.4267   Median :0.05037
##  Mean   :14865 Mean   :0.42527   Mean   :35.2420   Mean   :0.07833
##  3rd Qu.:22313 3rd Qu.:0.68457   3rd Qu.:36.2163   3rd Qu.:0.08800
##  Max.   :29806 Max.   :3.44767   Max.   :39.2467   Max.   :0.69655
## 
##  B0_sstmean
##  Min.   :-1.705
##  1st Qu.:14.853
##  Median :18.899
##  Mean   :18.120
##  3rd Qu.:25.374
##  Max.   :30.663
```

The resulting dataset has 29661. We save this dataset in a local file to work on it in subsequent steps.

```
save(list = "data", file = "data/outputs_for_modelling/PAdata_with_env.RData")
```

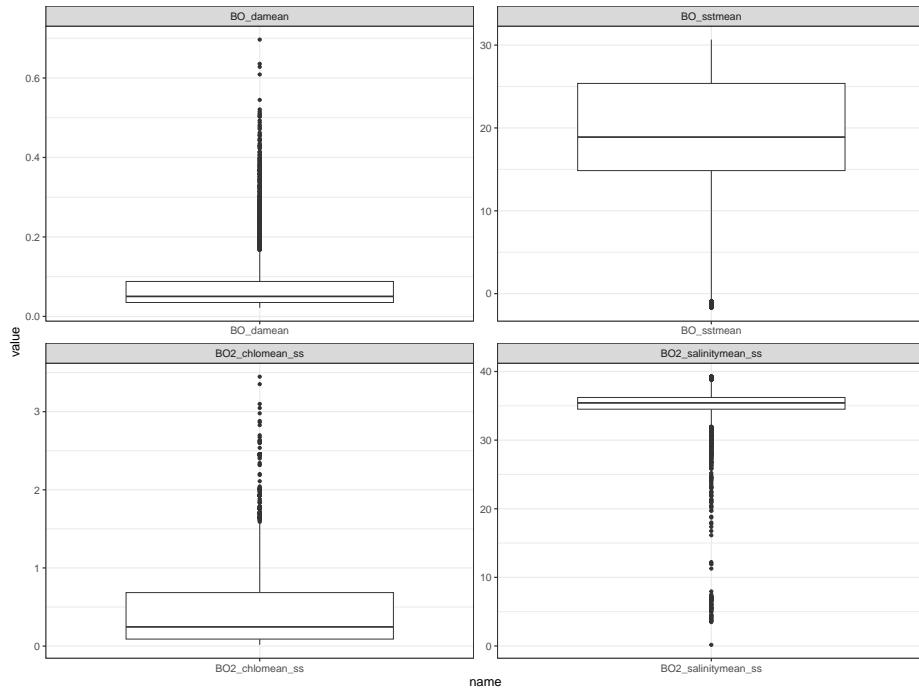
4.2 Exploratory plots of environmental variables

Before starting the modelling process, we are going to explore the individual variables in the dataset.

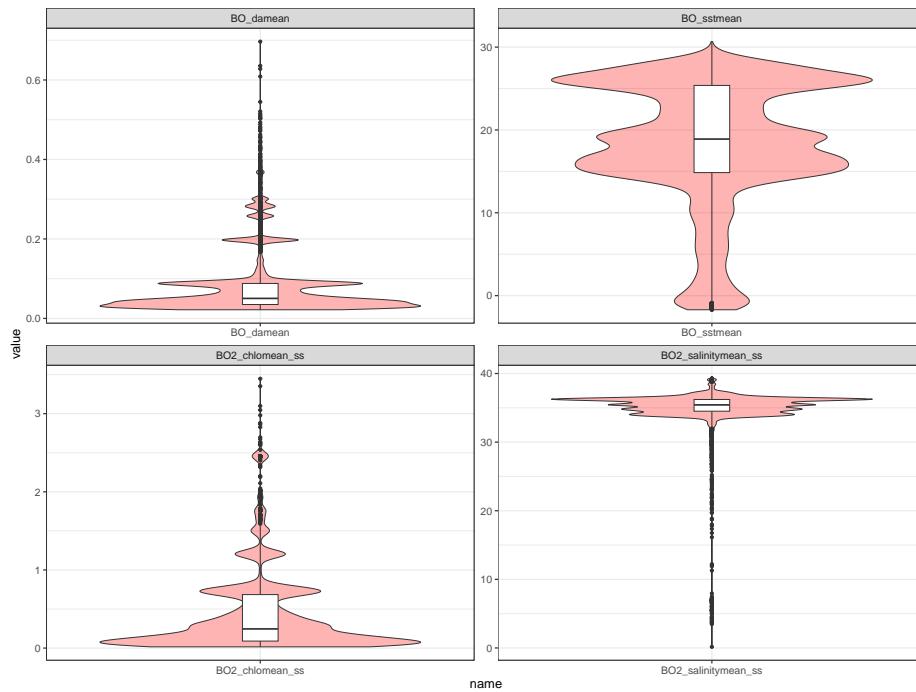
We can explore the distributions of each of the environmental variables by looking at the violin and boxplots and at the histograms and density plots as follows:

```
tmp <- data[, c("B02_chlomean_ss", "B02_salinitymean_ss",
               "B0_damean", "B0_sstmean")]
tmp <- pivot_longer(data = tmp, cols = everything())

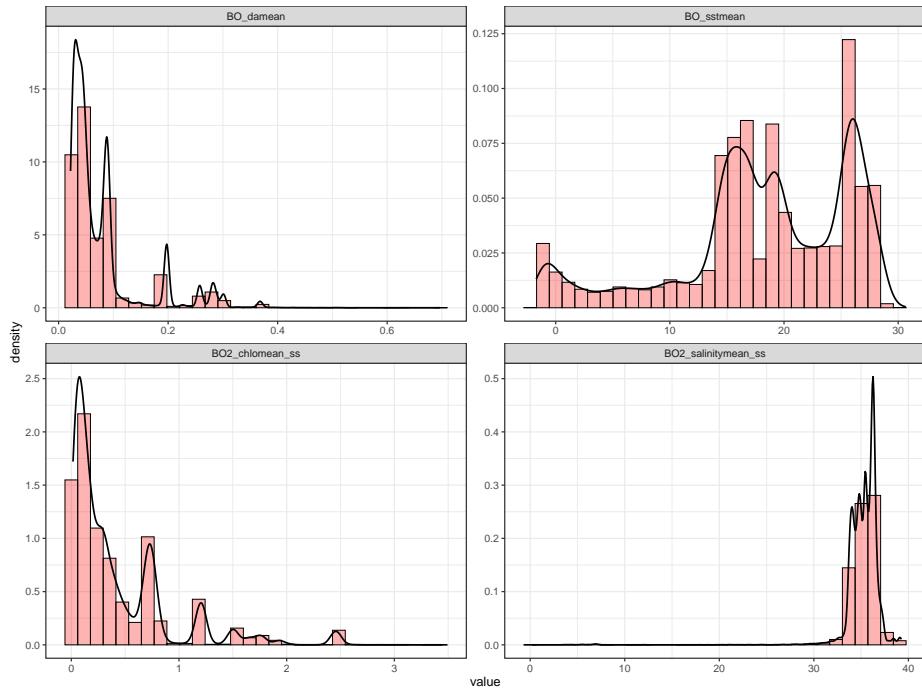
ggplot(data = tmp, aes(x = name, y = value)) +
  geom_boxplot() + facet_wrap(~name, scales = "free")
```



```
ggplot(data = tmp, aes(x = name, y = value)) +
  geom_violin(fill = "red", alpha = 0.3) +
  geom_boxplot(width = 0.1) + facet_wrap(~name,
  scales = "free")
```



```
ggplot(data = tmp, aes(x = value)) + geom_histogram(aes(y = after_stat(density)),
  colour = 1, fill = "red", alpha = 0.3) +
  geom_density(lwd = 1) + facet_wrap(~name,
  scales = "free")
```



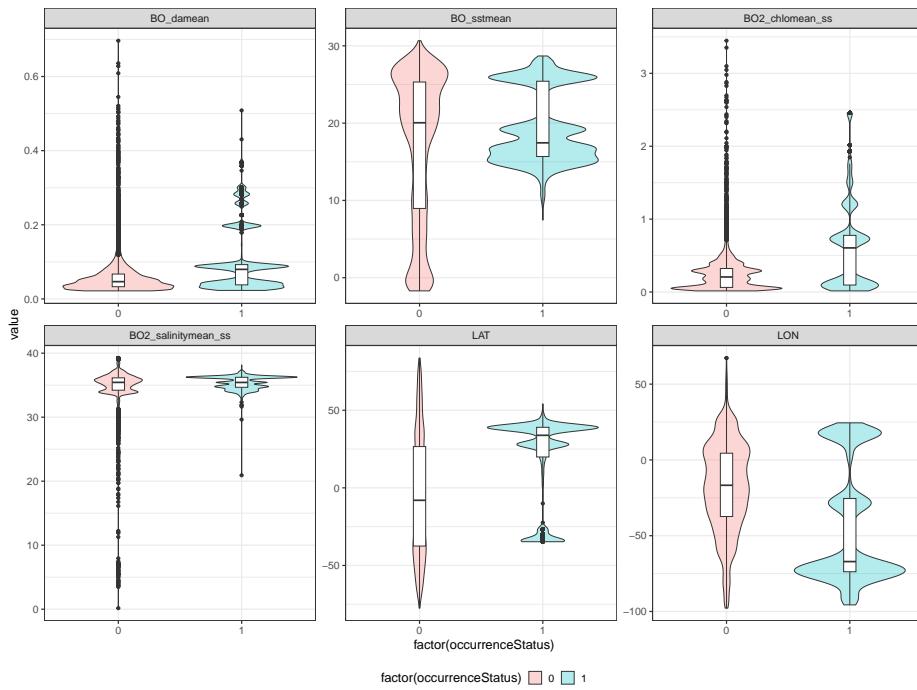
4.3 Exploratory plots of environmental variables depending on species distribution data

To analyse if there are preferences for certain ranges of the environmental variables, we compare the distribution of the environmental variables for presence and pseudo-absence data:

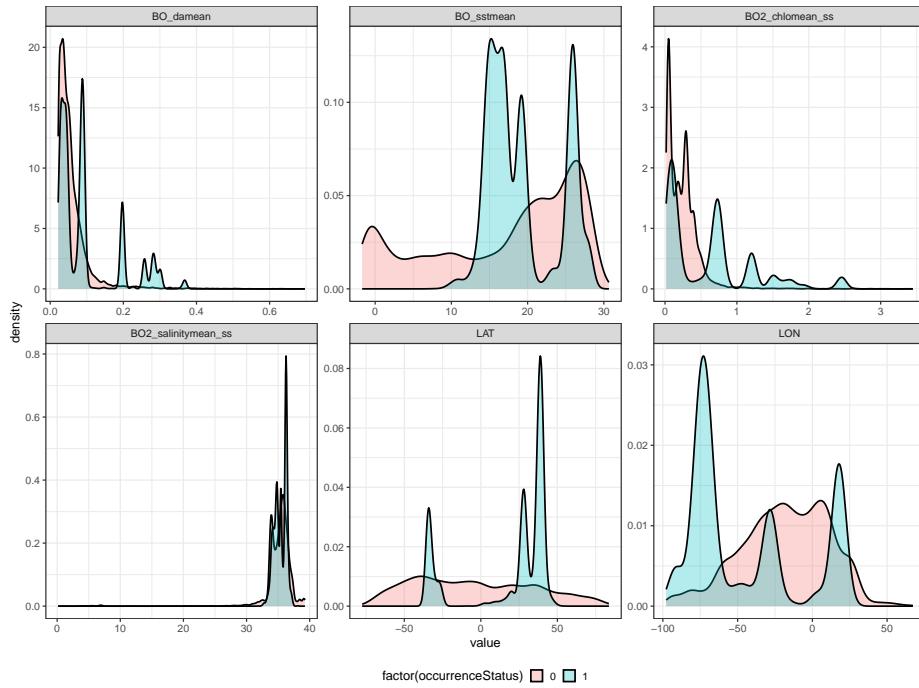
```
tmp <- data[, c("LON", "LAT", "B02_chlomean_ss",
  "B02_salinitymean_ss", "B0_damean", "B0_sstmean",
  "occurrenceStatus")]
tmp <- pivot_longer(data = tmp, cols = !occurrenceStatus)

ggplot(data = tmp, aes(x = factor(occurrenceStatus),
  y = value, fill = factor(occurrenceStatus),
  group = factor(occurrenceStatus))) +
  geom_violin(alpha = 0.3) + geom_boxplot(fill = "white",
  width = 0.1) + facet_wrap(~name, scales = "free") +
  theme(legend.position = "bottom", legend.background = element_rect(fill = "white",
  colour = NA))
```

4.3. EXPLORATORY PLOTS OF ENVIRONMENTAL VARIABLES DEPENDING ON SPECIES DISTRIBUTION



```
ggplot(data = tmp, aes(x = value, fill = factor(occurrenceStatus),
group = factor(occurrenceStatus))) +
  geom_density(lwd = 1, alpha = 0.3) +
  facet_wrap(~name, scales = "free") +
  theme(legend.position = "bottom", legend.background = element_rect(fill = "white",
colour = NA))
```

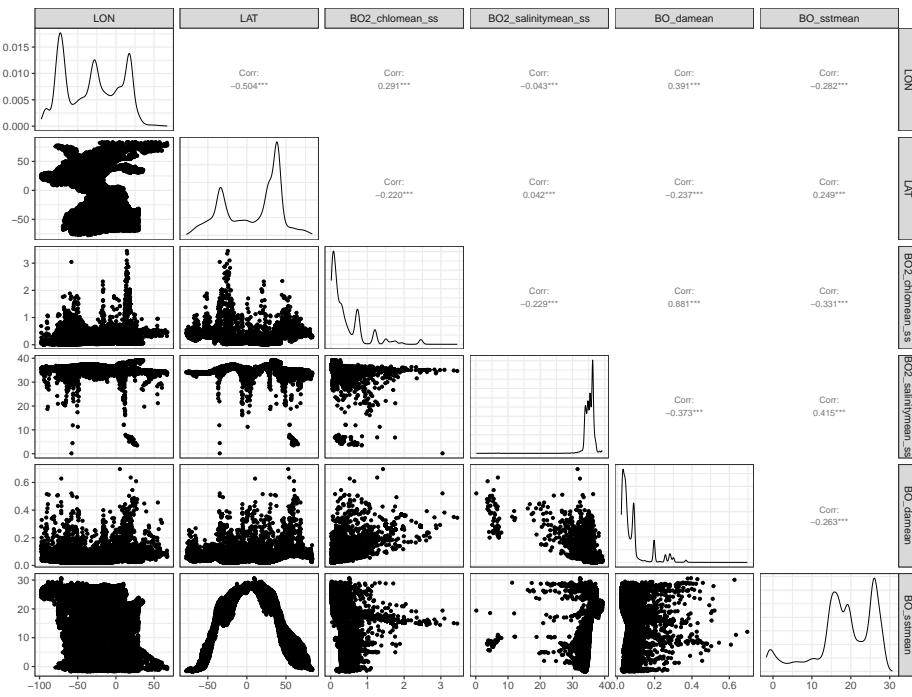


4.4 Correlation analysis

Some of the environmental variables can be correlated. The `GGally` package allows to easily produce pairplots of the variables and their correlation.

```
tmp <- data[, c("LON", "LAT", "BO2_chlomean_ss",
               "BO2_salinitymean_ss", "BO_damean", "BO_sstmean")]

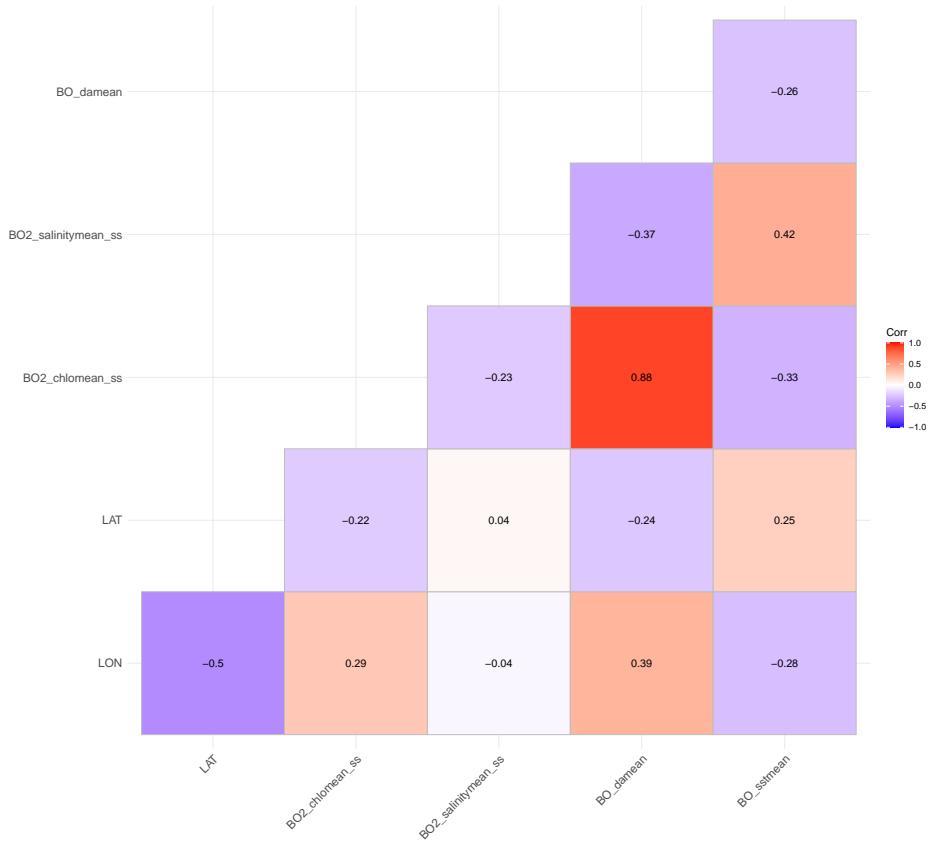
ggpairs(tmp) #this takes some minutes
```



A more detailed analysis of the potential correlations can be conducted using the package `ggcorrplot`:

```
mat <- cor(tmp, use = "complete.obs")
p.mat <- cor_pmat(tmp)

ggcorrplot(mat, type = "lower", lab = T,
           p.mat = p.mat)
```



4.5 Variance Inflation Factor (VIF)

Furthermore, multicollinearity in regression analysis can be explored using the VIF (Variance Inflation Factor). The value of the VIF statistics indicate the level of multicollinearity with the rest of the variables:

- VIF equal to 1 = variables are not correlated
- VIF between 1 and 5 = variables are moderately correlated
- VIF greater than 5 = variables are highly correlated

There are several packages in R that allows to calculate the VIF statistics. In this case we use the package `HH`:

```
# Select variables for VIF calculation
v.table <- data %>%
  dplyr::select(BO2_salinitymean_ss, BO_sstmean,
```

```
B02_chlomean_ss, B0_damean)

# Get VIF results
out.vif <- vif(v.table)
sort(out.vif)

##           B0_sstmean B02_salinitymean_ss      B02_chlomean_ss      B0_damean
##           1.375837          1.493366          5.270639          5.478410
```

We remove the variable that has the highest VIF value and we test again the multicollinearity:

```
v.table <- v.table %>%
  dplyr::select(-B0_damean)

# Get new VIF results
out.vif <- vif(v.table)
sort(out.vif)

##           B02_chlomean_ss B02_salinitymean_ss      B0_sstmean
##           1.136295          1.222233          1.300901
```

Now all the variables have VIF values that are acceptable. So, we proceed to remove BO_damean (Diffuse attenuation coefficient at 490 nm). And save the selected variables for the next modelling stages:

```
data <- data %>%
  dplyr::select(-B0_damean)
```

We save the dataset as our output for modelling.

```
save(list = "data", file = "data/outputs_for_modelling/PAdata_with_env.RData")
```


Chapter 5

Shape Constrained-Generalized Additive Models

In order to fit SDM in agreement with the ecological niche theory, the proposed Shape-Constrained Generalized Additive Models (SC-GAMs) in [Cidores et al., 2020] are fitted in this section. SC-GAMs are based on Generalized Additive Models, allowing us to impose shape-constraints to the linear predictor function. The R package SCAM implements the general framework developed by [Pya and Wood, 2015] using shape-constrained P-splines. Monotonicity and concavity/convexity constraints can be imposed on the sign of the first and/or the second derivatives of the smooth terms. For fitting Species Distribution Models in agreement with the ecological niche theory, we imposed concavity constraints ($f''(x) \leq 0$), so that the response can presents at most a single mode.

Alternatively, the R package `mboost` fits SC-GAMs using boosting methods. We are not going to develop this alternative here.

First, we load the list of required libraries.

```
requiredPackages <- c("here", "rstudioapi",
  "stringr", "RColorBrewer", "ggplot2",
  "dplyr", "fields", "maps", "raster",
  "scam", "plotmo", "pkgbuild", "dismo")
```

We run a function to install the required packages that are not in our system and load all the required packages.

```

install_load_function <- function(pkg) {
  new.pkg <- pkg[!(pkg %in% installed.packages() [,
    "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}

install_load_function(requiredPackages)

##      here    rstudioapi     stringr   RColorBrewer    ggplot2    dplyr
##      TRUE      TRUE        TRUE        TRUE        TRUE      TRUE
##      fields     maps       raster       scam      plotmo    pkgbuild
##      TRUE      TRUE        TRUE        TRUE      TRUE      TRUE
##      dismo
##      TRUE

```

We define some overall settings.

```

# General settings for ggplot
# (black-white background, larger
# base_size)
theme_set(theme_bw(base_size = 16))

```

5.1 Model fit

We set the working directory to the folder where the current script is located and we load the dataset (PAdat_a_with_env.Rdata) containing the presence-absence data together with the environmental data.

```

load(here::here("data", "outputs_for_modelling",
  "PAdat_a_with_env.Rdata"))

```

To fit a logistic regression model in the SC-GAMs framework, we use the `scam` function, where we set the binomial family with the logit link function. Our response variable is the presence-absence data and the selected three explanatory variables are the SST, chlorophyll and salinity. Each variable is included in the model through an spline function where the concavity constraint is set using `bs="cv"`. The details about this option can be found in the section “Constructor for concave P-splines in SCAMs” of the SCAM manual (<https://cran.r-project.org/web/packages/scam/scam.pdf>). The number of knots (k) is fixed at 8 in this example for a good balance between flexibility and computation time.

UNIVARIATE MODELS

Before fitting the model with the selected three environmental variables, we can fit univariate model as follows.

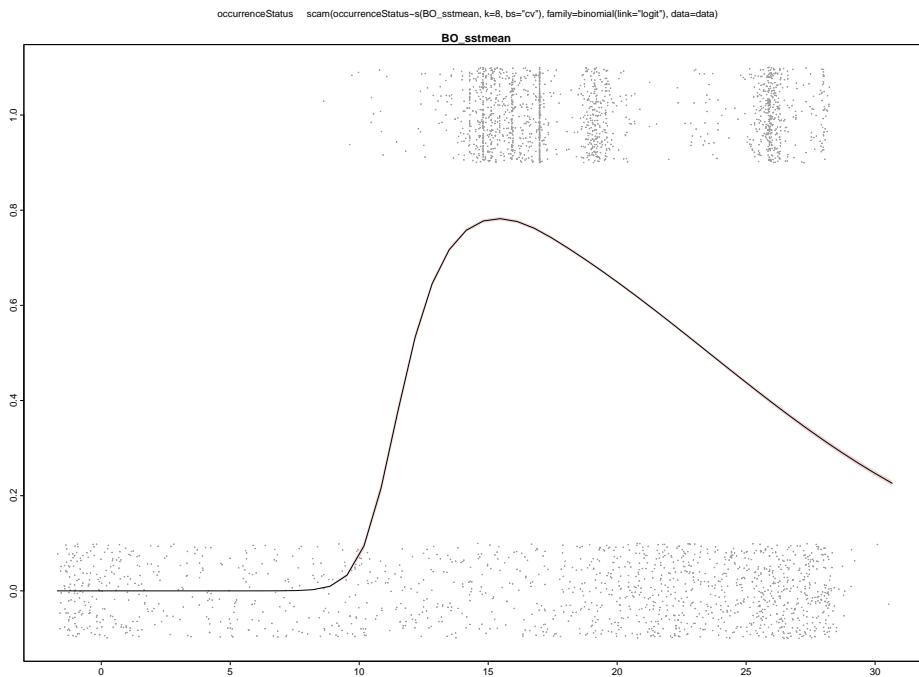
We fit the univariate model for SST, we print the summary of the model fit, and look at the fitted curve in the response scale.

```
model_sst <- scam(occurrenceStatus ~ s(B0_sstmean,
  k = 8, bs = "cv"), family = binomial(link = "logit"),
  data = data)
summary(model_sst)

##
## Family: binomial
## Link function: logit
##
## Formula:
## occurrenceStatus ~ s(B0_sstmean, k = 8, bs = "cv")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.24702   0.08182 -27.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq p-value
## s(B0_sstmean)    2      2 1078  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.2607  Deviance explained = 23.1%
## UBRE score = 0.066363  Scale est. = 1          n = 29661

plotmo(model_sst, level = 0.95, pt.col = 8)
```

52CHAPTER 5. SHAPE CONSTRAINED-GENERALIZED ADDITIVE MODELS



We repeat the same steps for the rest of the variables.

```
model_chl <- scam(occurrenceStatus ~ s(B02_chlomean_ss,
  k = 8, bs = "cv"), family = binomial(link = "logit"),
  data = data)
summary(model_chl)

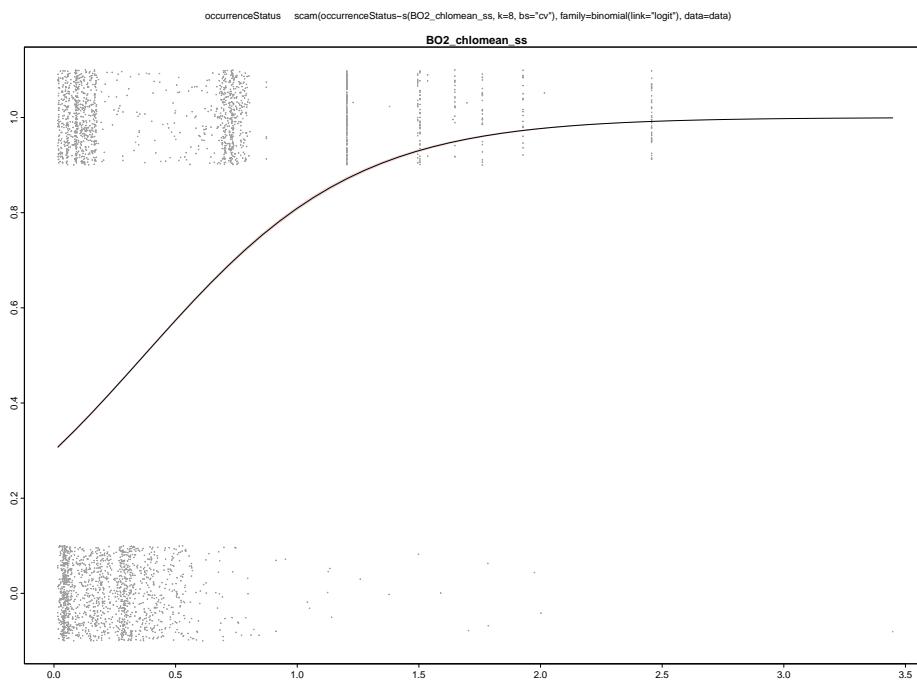
##
## Family: binomial
## Link function: logit
##
## Formula:
## occurrenceStatus ~ s(B02_chlomean_ss, k = 8, bs = "cv")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.12663   0.01333  9.499 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq p-value
## s(B02_chlomean_ss) 1      1 3263 <2e-16 ***
## ---
```

```

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.1546  Deviance explained = 12.4%
## UBRE score = 0.21423  Scale est. = 1          n = 29661

plotmo(model_chl, level = 0.95, pt.col = 8)

```



Due to convergence issues, sometimes it is necessary to fix the smoothing parameter (sp) at small value, i.e. 10^{-5} , as here when introducing salinity as an explanatory variable. If no value is provided, the smoothing parameter is estimated within the model.

```

model_sal <- scam(occurrenceStatus ~ s(B02_salinitymean_ss,
  k = 8, bs = "cv"), family = binomial(link = "logit"),
  data = data, sp = 1e-05)
summary(model_sal)

```

```

##
## Family: binomial
## Link function: logit
##
## Formula:
## occurrenceStatus ~ s(B02_salinitymean_ss, k = 8, bs = "cv")

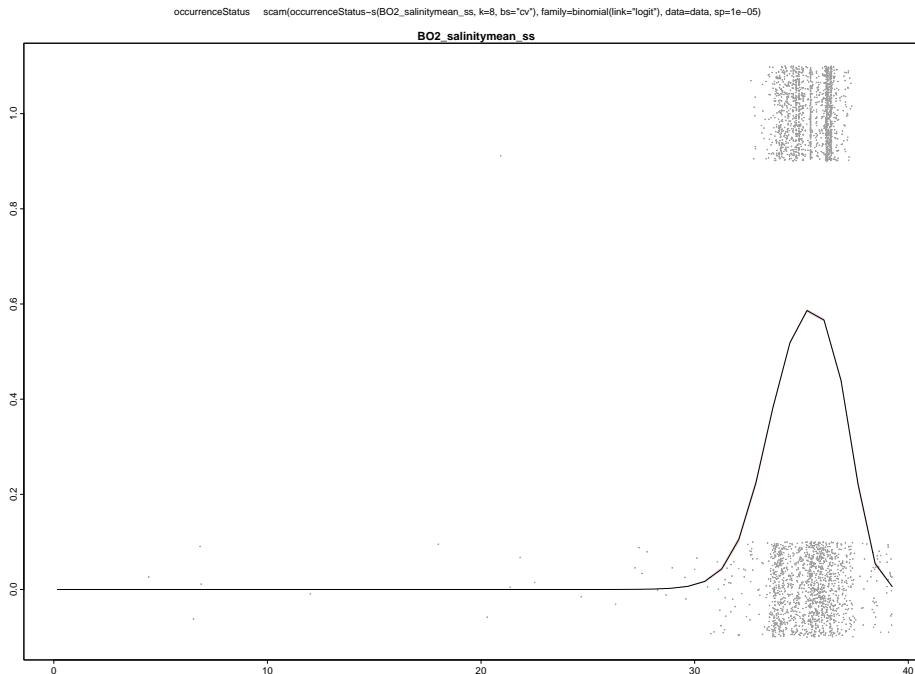
```

```

## 
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.1445     0.0138 -10.47  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Approximate significance of smooth terms:
##                      edf Ref.df Chi.sq p-value
## s(BO2_salinitymean_ss)    2      2 893.8 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## R-sq.(adj) =  0.0514   Deviance explained = 4.38%
## Scale est. = 1           n = 29661

plotmo(model_sal, level = 0.95, pt.col = 8)

```



MODEL WITH ALL VARIABLES

Now we fit the model including the three selected variables.

```

model <- scam(occurrenceStatus ~ s(BO_sstmean,
k = 8, bs = "cv") + s(BO2_salinitymean_ss,

```

```

k = 8, bs = "cv") + s(B02_chlomean_ss,
k = 8, bs = "cv"), family = binomial(link = "logit"),
data = data, sp = rep(1e-05, 3))
summary(model)

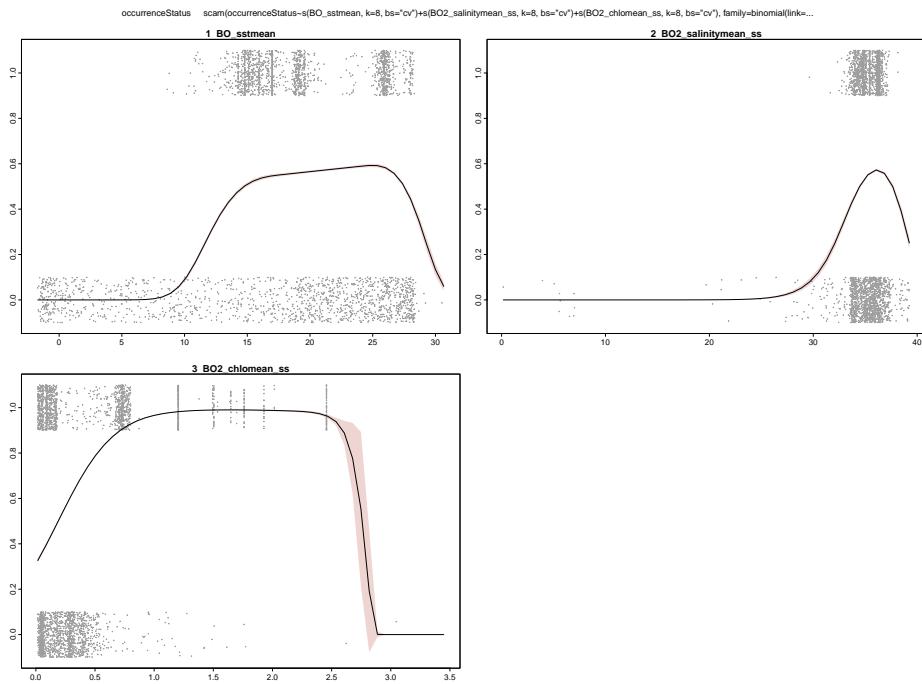
##
## Family: binomial
## Link function: logit
##
## Formula:
## occurrenceStatus ~ s(B0_sstmean, k = 8, bs = "cv") + s(B02_salinitymean_ss,
##      k = 8, bs = "cv") + s(B02_chlomean_ss, k = 8, bs = "cv")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.813     3.163   -0.573    0.567
##
## Approximate significance of smooth terms:
##                      edf Ref.df Chi.sq p-value
## s(B0_sstmean)        4.001 4.003  765.8 <2e-16 ***
## s(B02_salinitymean_ss) 2.004 2.008  178.7 <2e-16 ***
## s(B02_chlomean_ss)    3.987 4.003 1778.0 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.3491  Deviance explained =   32%
## Scale est. = 1           n = 29661

plotmo(model, level = 0.95, pt.col = 8)

##
## plotmo grid:      B0_sstmean B02_salinitymean_ss B02_chlomean_ss
##                   18.89921          35.42666          0.2452632

```

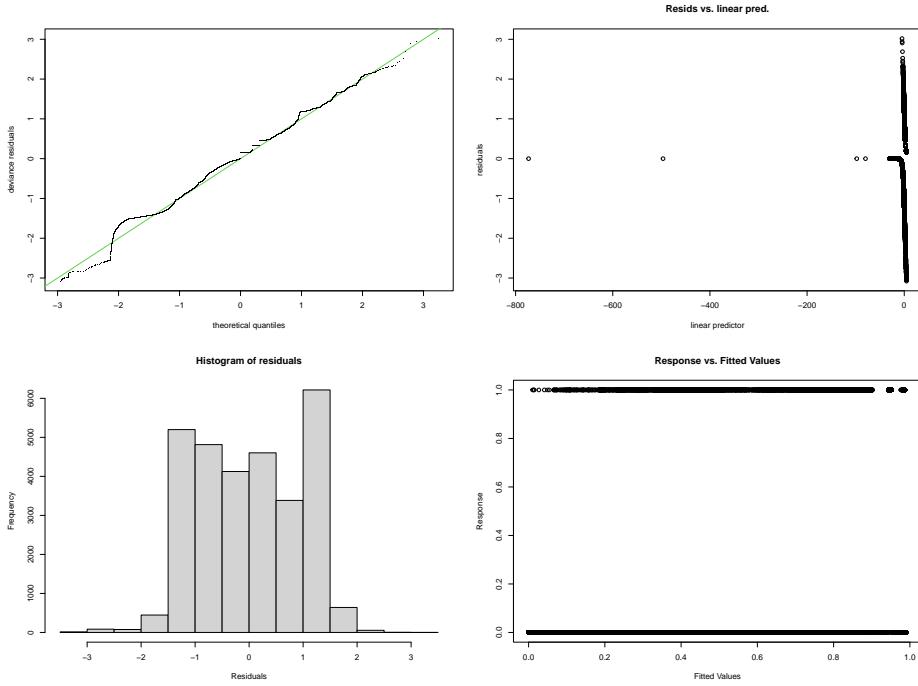
56CHAPTER 5. SHAPE CONSTRAINED-GENERALIZED ADDITIVE MODELS



We can see in the summary of the fit, that all included variables are statistically significant (with $p < 0.05$) and present a unimodal response curve.

For a more detailed check of the fitting, we can use the `scam.check` function:

```
old.par <- par(mfrow = c(2, 2))
scam.check(model)
```



```
##  
## Method: UBRE    Optimizer: NA newton  
## The optimal smoothing parameter(s): 1e-05 1e-05 1e-05 .
```

5.2 Model selection

When several explanatory variables are available, a variable selection process can be carried out. Here we provide as an example, a function that performs forward variable selection (modsel.scam) based on the significance of variables and AIC values of the fits.

```
source("function/function_scam_selection_optimized.R")
```

We save the names of the variables we want to introduce for the variable selection process as a vector:

```
vars <- c("B0_sstmean", "B02_salinitymean_ss",  
       "B02_chlomean_ss")
```

The default AIC tolerance is 2 and there is not a limit on selected terms in this example. These options can be modified through aic.tol and vmax arguments in the function. The number of knots and the sp can be also modified.

58 CHAPTER 5. SHAPE CONSTRAINED-GENERALIZED ADDITIVE MODELS

```
model_SCGAM <- try(modsel.scam(basef = "occurrenceStatus ~ 1",
                                vars = vars, dat = data, sp = 1e-05),
                                silent = T)

## [1] "Fitting models with 1 terms"
## occurrenceStatus ~ 1+s(B0_sstmean,m=2,bs='cv',k=8)
## occurrenceStatus ~ 1+s(B02_salinitymean_ss,m=2,bs='cv',k=8)
## occurrenceStatus ~ 1+s(B02_chlomean_ss,m=2,bs='cv',k=8)
## [1] "Fitting models with 2 terms"
## occurrenceStatus ~ 1+s(B0_sstmean,m=2,bs='cv',k=8)+s(B02_salinitymean_ss,m=2,bs='cv',k=8)
## occurrenceStatus ~ 1+s(B0_sstmean,m=2,bs='cv',k=8)+s(B02_chlomean_ss,m=2,bs='cv',k=8)
## [1] "Fitting models with 3 terms"
## occurrenceStatus ~ 1+s(B0_sstmean,m=2,bs='cv',k=8)+s(B02_chlomean_ss,m=2,bs='cv',k=8)+s(B02_salinitymean_ss,m=2,bs='cv',k=8)
```

We check results of the selected model, such as, selected variable names:

```
model_SCGAM$svars

## [1] "B0_sstmean"           "B02_chlomean_ss"      "B02_salinitymean_ss"
```

AICs of the fitted models:

```
sapply(model_SCGAM$smod, AIC)

##               null          B0_sstmean        B02_chlomean_ss  B02_salinitymean_ss
## 41120.17       31629.40          28318.30          27990.07
```

Explained deviance of fitted models:

```
sapply(model_SCGAM$smod, function(x) summary(x)$dev.expl)

##               null          B0_sstmean        B02_chlomean_ss  B02_salinitymean_ss
## -1.769524e-16 2.309141e-01          3.117433e-01          3.198121e-01
```

Formulas of the fitted models:

```
lapply(model_SCGAM$smod, formula)

## $null
## occurrenceStatus ~ 1
##
```

```

## $B0_sstmean
## occurrenceStatus ~ 1 + s(B0_sstmean, m = 2, bs = "cv", k = 8)
## <environment: 0x0000028eec32ac0>
##
## $B02_chlomean_ss
## occurrenceStatus ~ 1 + s(B0_sstmean, m = 2, bs = "cv", k = 8) +
##   s(B02_chlomean_ss, m = 2, bs = "cv", k = 8)
## <environment: 0x0000028eec32ac0>
##
## $B02_salinitymean_ss
## occurrenceStatus ~ 1 + s(B0_sstmean, m = 2, bs = "cv", k = 8) +
##   s(B02_chlomean_ss, m = 2, bs = "cv", k = 8) + s(B02_salinitymean_ss,
##   m = 2, bs = "cv", k = 8)
## <environment: 0x0000028eec32ac0>

```

Summaries of the fitted models:

```
lapply(model_SCGAM$smod, summary)
```

```

## $null
##
## Family: binomial
## Link function: logit
##
## Formula:
## occurrenceStatus ~ 1
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.009777  0.011613  0.842     0.4
##
## 
## R-sq.(adj) =      0   Deviance explained = -1.77e-14%
## UBRE = 0.38634  Scale est. = 1           n = 29661
##
## $B0_sstmean
##
## Family: binomial
## Link function: logit
##
## Formula:
## occurrenceStatus ~ 1 + s(B0_sstmean, m = 2, bs = "cv", k = 8)
## <environment: 0x0000028eec32ac0>
##
## Parametric coefficients:

```

60CHAPTER 5. SHAPE CONSTRAINED-GENERALIZED ADDITIVE MODELS

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.24678    0.08094 -27.76  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq p-value
## s(B0_sstmean)    2      2 1082  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.2607   Deviance explained = 23.1%
## Scale est. = 1           n = 29661
##
##
## $B02_chlomean_ss
##
## Family: binomial
## Link function: logit
##
## Formula:
## occurrenceStatus ~ 1 + s(B0_sstmean, m = 2, bs = "cv", k = 8) +
##     s(B02_chlomean_ss, m = 2, bs = "cv", k = 8)
## <environment: 0x0000028eeec32ac0>
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.057     9.021  -0.339    0.735
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq p-value
## s(B0_sstmean)    4.000  4.000  939.7  <2e-16 ***
## s(B02_chlomean_ss) 4.221  3.952 2225.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.341   Deviance explained = 31.2%
## Scale est. = 1           n = 29661
##
##
## $B02_salinitymean_ss
##
## Family: binomial
## Link function: logit
##
## Formula:
```

```

## occurrenceStatus ~ 1 + s(B0_sstmean, m = 2, bs = "cv", k = 8) +
##   s(B02_chlomean_ss, m = 2, bs = "cv", k = 8) + s(B02_salinitymean_ss,
##   m = 2, bs = "cv", k = 8)
## <environment: 0x0000028eeec32ac0>
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.813     3.163   -0.573    0.567
##
## Approximate significance of smooth terms:
##                      edf Ref.df Chi.sq p-value
## s(B0_sstmean)      4.001 4.003  765.8 <2e-16 ***
## s(B02_chlomean_ss) 3.987 4.003 1778.1 <2e-16 ***
## s(B02_salinitymean_ss) 2.004 2.008  178.7 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.3491  Deviance explained =   32%
##   Scale est. = 1           n = 29661

```

The last model of the list, is the selected one. Which in this case contains the considered three variables.

```
selected_model <- model_SCGAM$smod[[length(model_SCGAM$smod)]]
```

We save the selected model

```
save(list = "selected_model", file = "models/selected_model.RData")
```

Note that there are multiple options and criteria for model selection that are not reviewed here. Any model selection technique used for GAMs can be used also for SC-GAMs.

Chapter 6

Model validation

In this section, model validation is performed in order to assess the predictive performance of the selected model. This validation is conducted via k-fold cross-validation. The data set is divided into k equally sized groups [Hijmans, 2012], using a percentage of randomly selected observations to run the model and the remaining for validation, iteratively for each fold.

First, we load the list of required libraries.

```
requiredPackages <- c("here", "rstudioapi",
  "stringr", "RColorBrewer", "ggplot2",
  "dplyr", "tidyverse", "R.utils", "ggpubr",
  "hrbrthemes", "fields", "maps", "raster",
  "scam", "plotmo", "pkgbuild", "dismo",
  "SDMTools")
```

We run a function to install the required packages that are not in our system and load all the required packages.

```
install_load_function <- function(pkg) {
  new.pkg <- pkg[!(pkg %in% installed.packages() [
    "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
    sapply(pkg, require, character.only = TRUE)
}

install_load_function(requiredPackages)

##          here      rstudioapi      stringr      RColorBrewer      ggplot2      dplyr
```

```

##      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE
## tidyverse R.utils ggpubr hrbrthemes fields maps
##      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE
## raster   scam   plotmo pkgbuild dismo SDMTools
##      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE

```

Note: in case `SDMTools` failed to load you will have to first manually install `RTools` and then run this code below:

```

# find_rtools()

# install.packages('remotes')
# remotes::install_version('SDMTools',
#   version = '1.1-221.2')

library(SDMTools)

```

We define some overall settings.

```

# General settings for ggplot
# (black-white background, larger
# base_size)
theme_set(theme_bw(base_size = 16))

```

We load output from the selected model saved in the previous step.

```
load(here::here("models", "selected_model.Rdata"))
```

6.1 Optimum threshold

We generate a data frame with the data used in the selected model and we add the predicted values.

```

# PAdat.environ used in the
# selected model
data <- selected_model$model

# Predict
scgam.pred <- predict(selected_model, newdata = data,
  type = "response")

# Add the prediction to the data object
data$scgam.pred <- as.vector(scgam.pred)
head(data)

```

```

## occurrenceStatus B0_sstmean B02_chlomean_ss B02_salinitymean_ss scgam.pred
## 1                 1  16.99439    1.20410309      35.42666  0.9814526
## 5                 1  16.99439    1.20410309      35.42666  0.9814526
## 8                 1  25.52963    0.08455528      36.43070  0.4357622
## 10                1  16.99439    1.20410309      35.42666  0.9814526
## 11                1  15.21486    0.72283228      34.20426  0.8506453
## 12                1  16.99439    1.20410309      35.42666  0.9814526

```

The threshold for presence-absence classification for each species is obtained as the values maximizing sensitivity plus specificity [Jiménez-Valverde and Lobo, 2007]. If the result was a range (instead of a single value), we would select the mean value of the range.

```

# Optimizing the threshold probability
obs <- data$occurrenceStatus
predSCGAM_P <- data$scgam.pred

# Threshold optimizing
myoptim <- optim.thresh(obs, predSCGAM_P)
myoptim

## $min.occurrence.prediction
## [1] 0.01055724
##
## $mean.occurrence.prediction
## [1] 0.671467
##
## $`10.percent.omission`
## [1] 0.37
##
## $`sensitivity=specificity`
## [1] 0.45
##
## $`max.sensitivity+specificity`
## [1] 0.43
##
## $maxKappa
## [1] 0.43
##
## $max.prop.correct
## [1] 0.43
##
## $min.ROC.plot.distance
## [1] 0.43

```

```
# Select the threshold that maximizes
# the sum of sensitivity and
# specificity
myThreshold <- as.numeric((myoptim[["max.sensitivity+specificity"]]))
```

Accuracy indicators, such as AUC (Area Under the Receiver Operating Characteristic—ROC—curve), sensitivity (true predicted presences) and specificity (true predicted absences) are first computed for the all observations.

```
# Accuracy values with all observations
accuracy(obs, predSCGAM_P, threshold = myThreshold)
```

```
##   threshold      AUC omission.rate sensitivity specificity prop.correct
## 1      0.43 0.7409775      0.1978796    0.8021204    0.6798347    0.7412764
##       Kappa
## 1 0.4822374
```

```
# Create confusion matrix with all
# observations
confusion.matrix(obs, predSCGAM_P, threshold = myThreshold)
```

```
##      obs
## pred      0      1
##   0 10033 2949
##   1 4725 11954
## attr(,"class")
## [1] "confusion.matrix"
```

6.2 k-fold validation

In this case we use a 5-fold cross-validation.

```
# Number of groups
k <- 5

# Generate groups
groups <- kfold(data, k, by = data$occurrencestatus)
```

The model is run for each of the 5 random subset (with a 20% of the observations) and indicators are then computed using the remaining 80% of the observations. Indicators are the averaged across folds.

```

# Initialise the confusion matrix and
# the accuracy table:
myCM <- NULL
myACC <- NULL

# Get the formula of the selected model
formula <- summary(selected_model)[["formula"]]

# Get the smoothing parameters of the
# selected model
sp <- selected_model$sp

# Loop for each group k
for (j in 1:k) {
  # Preparation of Training Sites
  p_Training <- data[groups != j, ]

  # Model fit
  selected_model.sp.j <- scam(formula,
    family = binomial(link = "logit"),
    data = p_Training, sp = c(sp))

  # Predict Model
  p_validacion <- data[groups == j, ]

  selected_model.sp.j.pred <- predict(selected_model.sp.j,
    newdata = p_validacion, type = "response")
  p_validacion$Pred <- selected_model.sp.j.pred

  # Confussion matrix and accuracy
  # table for fold j
  obs <- p_validacion$occurrenceStatus
  predSCGAM <- p_validacion$Pred
  myCM <- rbind(myCM, as.numeric(confusion.matrix(obs,
    predSCGAM, threshold = myThreshold)))
  myACC <- rbind(myACC, accuracy(obs, predSCGAM,
    threshold = myThreshold))
}

# Mean values across k-folds
validation_summary <- cbind(Threshold = myThreshold,
  mean_AUC = mean(myACC$AUC), mean_Omission = mean(myACC$omission.rate),
  mean_sensitivity = mean(myACC$sensitivity),
  mean_specificity = mean(myACC$specificity),
  mean_Prop.Corr = mean(myACC$prop.correct))

```

```
validation_summary
```

```
##      Threshold  mean_AUC mean_Omission mean_sensitivity mean_specificity
## [1,]      0.43  0.7396357   0.1912325       0.8087675      0.6705039
##      mean_Prop.Corr
## [1,]      0.7398605
```

We save the validation summary object.

```
save(validation_summary, file = here::here("models/validation_summary.RData"))
```

Chapter 7

Prediction and maps

In this chapter we predict from the fitted model and produce final SDMs maps.

First, we load a list of required libraries.

```
requiredPackages <- c(  
  "here",  
  "rstudioapi",  
  "ggplot2",  
  "tidyverse",  
  # "rgdal",  
  "raster",  
  "maps",  
  "RColorBrewer",  
  "scam",  
  "ggpubr"  
)
```

We run a function to install the required packages that are not in our system and load all the required packages.

```
install_load_function <- function(pkg) {  
  new.pkg <- pkg[!(pkg %in% installed.packages() [,  
    "Package"])]  
  if (length(new.pkg))  
    install.packages(new.pkg, dependencies = TRUE)  
  sapply(pkg, require, character.only = TRUE)  
}  
  
install_load_function(requiredPackages)
```

```
##      here    rstudioapi     ggplot2     tidyverse     raster     maps
##      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE
## RColorBrewer      scam     ggpubr      TRUE
##      TRUE      TRUE      TRUE
```

We define some overall settings.

```
# General settings for ggplot
# (black-white background, larger
# base_size)
theme_set(theme_bw(base_size = 16))
```

7.1 Prepare environmental data

In previous steps (see Chapter 2), we have defined the study area that defines the extent of our spatial data. We load the `study_area` object that is a `SpatialPolygonsDataFrame` class:

```
load(here::here("data", "spatial", "study_area.RData"))
```

And we load the `rasterStack` with the downloaded environmental data.

```
mylayers <- stack(here::here("data", "env",
"mylayers.tif"))
```

We transform the environmental data set first into a data frame, and then into a `SpatialDataFrame`.

```
env_dataframe <- raster::as.data.frame(mylayers,
```

```
xy = TRUE)
```

```
summary(env_dataframe)
```

	x	y	mylayers_1	mylayers_2
## Min.	-97.79	-82.96	Min. :0.0	Min. : 0.1
## 1st Qu.	-56.23	-39.73	1st Qu.:0.1	1st Qu.:33.8
## Median	-14.67	3.50	Median :0.3	Median :34.6
## Mean	-14.67	3.50	Mean :0.3	Mean :34.4
## 3rd Qu.	26.90	46.73	3rd Qu.:0.4	3rd Qu.:35.6
## Max.	68.46	89.96	Max. :3.6	Max. :40.7
##			NA's :1501044	NA's :1501044
## mylayers_3		mylayers_4		

```

##   Min.    :0.0      Min.    :-1.8
##  1st Qu.:0.0      1st Qu.: 1.9
##  Median :0.0      Median :15.1
##  Mean   :0.1      Mean   :13.7
##  3rd Qu.:0.1      3rd Qu.:24.1
##  Max.   :1.0      Max.   :32.3
##  NA's    :1652879  NA's   :1652879

names(env_dataframe) <- c("x", "y", "B02_chlomean_ss",
                          "B02_salinitymean_ss", "B0_damean", "B0_sstmean")

```

7.2 Projection

We load the selected model and predict into the whole environmental data.

```

# Load SC-GAM model
load(here::here("models", "selected_model.Rdata"))

# Predicting
predict <- predict(selected_model, newdata = env_dataframe,
                     type = "response", se.fit = T)

env_dataframe$fit <- predict$fit
env_dataframe$se.fit <- predict$se.fit

save(env_dataframe, file = "results/projection.Rdata")

```

7.3 Mapping

```

# Load PA data
load(here::here ("data", "outputs_for_modelling", "PAdatas_with_env.Rdata"))

proj_map <- ggplot() +
  geom_raster(data=subset(env_dataframe),
              aes(x,y,fill=fit)) +
  scale_fill_gradient2(low="blue",
                       mid="orange",
                       high="red",
                       midpoint = 0.5,
                       limits = c(0,1)) +

```

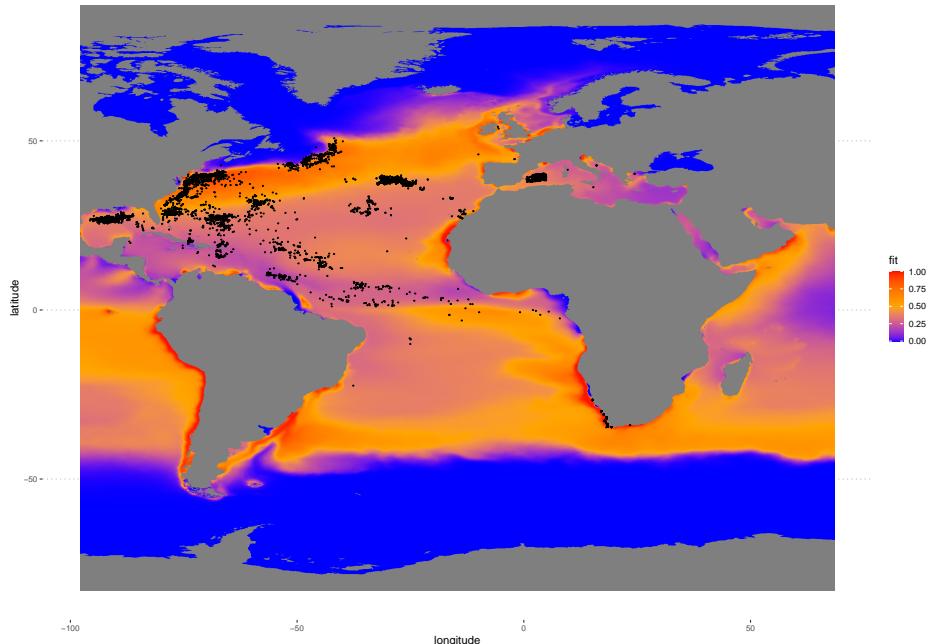
```

ggtitle("Occurrence probabilt Thunnus alalunga")+
  geom_point(data=subset(data,occurrenceStatus==1),
             aes(LON,LAT),
             col=1,
             size=0.3) +
  theme_pubclean(base_size = 14) +
  theme(panel.background = element_blank(),
        plot.title = element_text(face = "italic"),
        #text = element_text(size = 14),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        legend.position="right") +
  labs(y="latitude", x = "longitude")

print(proj_map)

```

Occurrence probability Thunnus alalunga



We finally save the projection map.

```

ggsave(filename = "Thunnus_alalunga_proj_map.tif",
       plot = proj_map, device = "tiff", path = here::here("plots",
       "projections"), height = 22, width = 30,
       units = "cm", dpi = 300)

```

Chapter 8

Acknowledgements

This tutorial has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreements No 862428 MISSION ATLANTIC project.

Bibliography

- J Assis, L Tyberghein, S Bosch, H Verbruggen, EA Serrão, O De Clerck, and D Tittensor. Bio-oracle v2.0: Extending marine data layers for bioclimatic modelling. *Global Ecology and Biogeography*, 27(3):277–284, 2017. ISSN 1466-822X 1466-8238. doi: 10.1111/geb.12693.
- M Barbet-Massin, F Jiguet, CH Albert, and W 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 Ibañarriaga, DJ Lee, MJ 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. doi: 10.1016/j.ecolmodel.2019.108926.
- J Elith and JR 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.
- RJ Hijmans. Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null model. *Ecology*, 93(3):679–688, 2012. doi: <https://doi.org/10.1890/11-0826.1>.
- GE Hutchinson. Concluding remarks. *Cold Spring Harbor Symposia on Quantitative Biology*, 22:415–427, 1957. doi: 10.1101/SQB.1957.022.01.039.
- A Jiménez-Valverde and JM Lobo. Threshold criteria for conversion of probability of species presence to either-or presence-absence. *Acta oecologica*, 31 (3):361–369, 2007. doi: <https://doi.org/10.1016/j.actao.2007.02.001>.
- JM McPherson, W Jetz, and DJ Rogers. The effects of species’ range sizes on the accuracy of distribution models: ecological phenomenon or statistical artefact? *Journal of applied ecology*, 41(5):811–823, 2004. doi: <https://doi.org/10.1111/j.0021-8901.2004.00943.x>.
- N Pya and SN Wood. Shape constrained additive models. *Statistics and computing*, 25:543–559, 2015.

L Tyberghein, H Verbruggen, K Pauly, C Troupin, F Mineur, and O De Clerck. Bio-oracle: a global environmental dataset for marine species distribution modelling. *Global Ecology and Biogeography*, 21(2):272–281, 2012. ISSN 1466822X. doi: 10.1111/j.1466-8238.2011.00656.x.