Supporting Information Appendix S1 - Preparing presence-background data from non-systematic research surveys

S. Derville, L. Torres, C. Iovan, C. Garrigue

22 mai, 2018

Supporting information S1 to "Finding the right fit: Comparative cetacean distribution models using multiple data sources and statistical approaches", 2018, Diversity and Distributions. Contact information: solene.derville@ird.fr

Runs on R version 3.3.2 (2016-10-31) Platform: x86_64-pc-linux-gnu (64-bit)

Tracklines were segmented into on- and off-effort sections. The sections over which the boat was conducting a focal follow were considered off-effort as observers were not vigilant to the detection of new groups. A set of points, denoted background points (a.k.a "pseudo-absences" or "control points"), was sampled within the on-effort survey track strip-width, spanning 4 km to each side of the tracklines to reflect minimal detection distance (pers. comm. Garrigue). Given the spatial resolution of most environmental variables (minimum 500 x 500 m smoothed with a 5 km moving window), daily samples of background points were generated with a minimum distance of 1 km from each other. The number of background points per survey day was proportional to the encounter rate on that day (number of sightings per hour on-effort).

R Packages

```
library(raster)
library(rgdal)
library(plyr)
library(rgeos)
library(maptools)
library(sp)
library(geoR)
library(oce)
library(gstat)
library(viridis)
```

Brief description of input data

Boat tracklines and whale group positions

```
# groups.df is a dataframe containing information about whale groups encountered at sea:
# the time and position of encounter (in lat-long: gro_Xdeb, gro_Ydeb; and in a mercator
# projected CRS: mercX and mercY)
head(groups.df)
```

```
##     gro_ide    sor_dat            gro_Hdeb gro_Xdeb  gro_Ydeb    mercX
## 181 G03-001 2003-07-21 2003-07-21 10:45:00 167.0120 -22.45187 223972.4
## 182 G03-002 2003-07-21 2003-07-21 11:54:00 166.9472 -22.40007 216765.2
## 183 G03-003 2003-07-23 2003-07-23 08:28:00 166.8732 -22.38635 208521.9
```

```
## 184 G03-004 2003-07-25 2003-07-25 11:27:00 167.0327 -22.58678 226275.7
## 185 G03-005 2003-07-26 2003-07-26 09:23:00 167.0784 -22.49272 231368.1
## 186 G03-006 2003-07-26 2003-07-26 11:14:00 167.0086 -22.45628 223600.4
##        mercY
## 181 -2549552
## 182 -2543350
## 183 -2541708
## 184 -2565718
## 185 -2554445
## 186 -2550082
```

```
# sorties.df.eez is a dataframe containing the dates (sor_dat), start time (sor_dob) and
# end time (sor_fob) of all surveys undertaken in the New Caledonian eez
head(sorties.df.eez)
```

```
##     sor_ide    sor_dat             sor_dob             sor_fob year
## 493 S030720 2003-07-20 2003-07-20 07:48:00 2003-07-20 13:00:00 2003
## 494 S030721 2003-07-21 2003-07-21 07:25:00 2003-07-21 14:55:00 2003
## 495 S030722 2003-07-22 2003-07-22 07:33:00 2003-07-22 14:15:00 2003
## 496 S030723 2003-07-23 2003-07-23 08:20:00 2003-07-23 10:03:00 2003
## 498 S030725 2003-07-25 2003-07-25 07:37:00 2003-07-25 14:00:00 2003
## 499 S030726 2003-07-26 2003-07-26 07:41:00 2003-07-26 13:26:00 2003
```

```
# effort_df is a dataframe continaining the successive GPS positions (in lat-long: x and y)
# recorded along the boat tracklines. Each position along the track is characterized by a
# state: effort = 0 or effort = 1 whether the survey team was on-effort or not. Effort = 0
# can be either "navigation" or "groups". The latter charcaterizes parts of the tracks
# where the research team is following a group of whales.
str(effort_df)
```

```
## 'data.frame':    341611 obs. of  7 variables:
##  $ day     : POSIXct, format: "2003-07-20" "2003-07-20" ...
##  $ time    : POSIXct, format: "2003-07-20 07:12:21" "2003-07-20 07:13:21" ...
##  $ id      : Factor w/ 14 levels "2003","2004",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ x       : num  167 167 167 167 167 ...
##  $ y       : num  -22.3 -22.3 -22.3 -22.3 -22.3 ...
##  $ activity: chr  "navigating" "navigating" "navigating" "navigating" ...
##  $ effort  : num  0 0 0 0 0 0 0 0 0 0 ...
```

land_shp_merc and **reef_shp_merc** are polygons shapefiles of the coastline and the emerged reef areas in New Caledonia, projected in a Mercator coordinate system

```
plot(land_shp_merc, col = "gray13")
```

```
plot(reef_shp_merc, col = "grey")
```

1- BUFFERING BOAT TRACKLINES

The aim of this section is to create a list of polygons, one per daily survey. Each polygon is created by buffering the boat trackline for that day then removing portions of the buffer which are overlaid with land or reefs.

Function **Fun_robustBufferDiff** created to fix the "topology exception error" occuring when there is a ring self-intersection.

```
Fun_robustBufferDiff = function(spgeom1, spgeom2) {
  tryCatch(expr = rgeos::gDifference(spgeom1, spgeom2, byid = F),
           error = function(e) {w <- gSimplify(spgeom1, tol = 1000, topologyPreserve = T);
```

```
                                w <- gBuffer(w, byid = TRUE, width = 100);
                                ww <- rgeos::gDifference(w, spgeom2, byid = F);
                                return(ww)
                        }
  )
}
```

Create buffer polygons around boat tracklines

```
# only keep the portions of trackline "on-effort"
oneffort_sp <- effort_df[effort_df$effort == 1, ]

# convert the dataframe to sp object and project it in a custom mercator CRS
oneffort_sp <- dlply(oneffort_sp, ~day, function(d){
  coordinates(d) =~ x + y
  proj4string(d) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
  d <- spTransform(d, CRS("+proj=merc +lon_0=165 +lat_1=-21.5 +k=1 +x_0=0 +y_0=0
                           +ellps=WGS84 +datum=WGS84 +units=m +no_defs"))
  return(d)
})
```

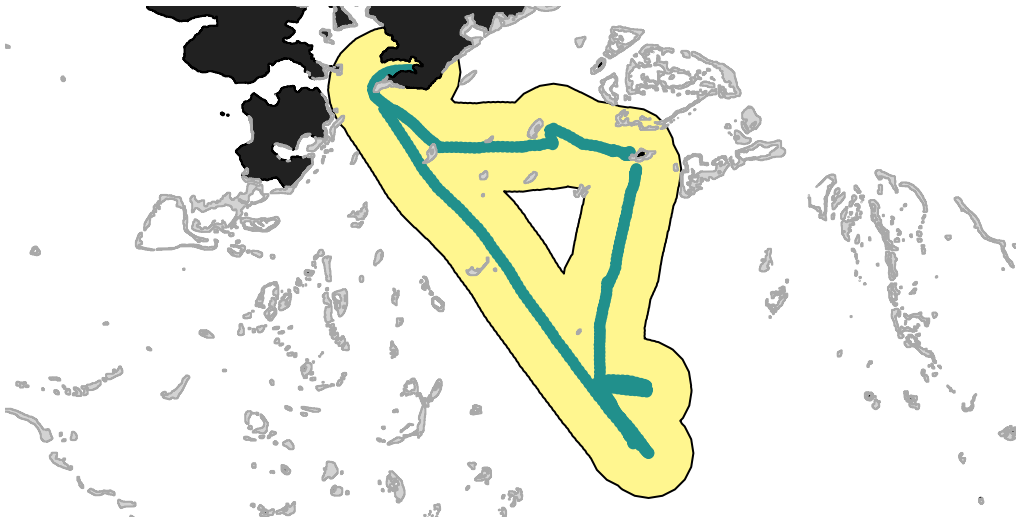
Create buffer of 4km around tracklines

```
bufferonland <- llply(oneffort_sp,function(x){rgeos::gBuffer(x, byid = F, width = 4000,
                                                capStyle = "ROUND", joinStyle = "ROUND")})
```

Remove portions of the polygon overlaid with land or reef

```
buffer_pol <- llply(bufferonland, function(x){Fun_robustBufferDiff(x, land_shp_merc)})
buffer_pol <- llply(buffer_pol, function(x){Fun_robustBufferDiff(x, reef_shp_merc)})
```

For instance, on day 2014-07-29, the surveyed polygon was represented as follows (with trackline in blue, buffer in yellow, reefs in grey and land in black):



2- SAMPLE BACKGROUND POINTS PER DAY OF SURVEY

The aim of this section is to sample background points in the area surveyed. The number of points generated per day is set to be proportional to the time spent on-effort that day. Background points are sampled

randomly within the surveyed polygons but are separated by a minimum distance of 1km.

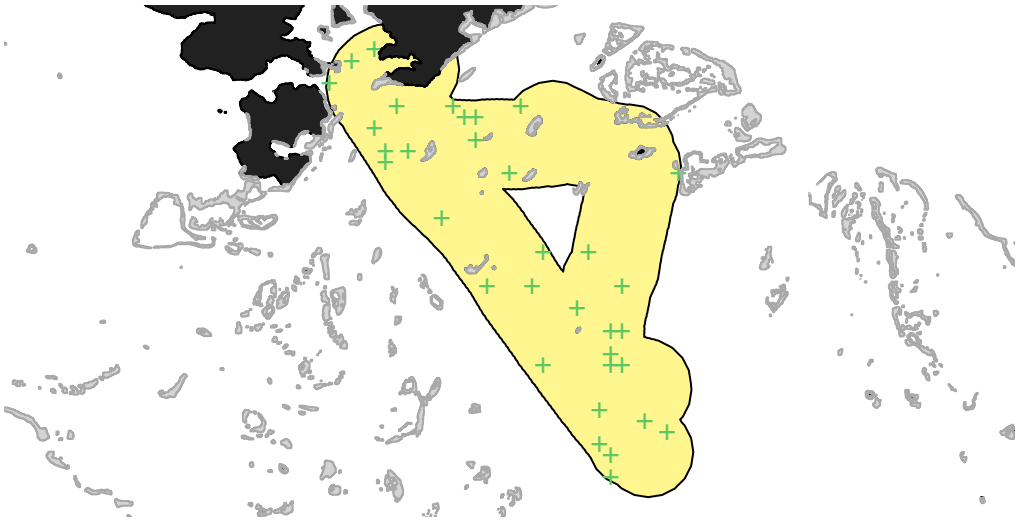
Assess time on effort per day of survey

```
# number of on-effort positions per day give an estimate of time on-effort
survey_time <- as.data.frame(table(effort_df[effort_df$effort == 1, ]$day))
names(survey_time) <- c("day", "time_on_effort")
survey_time$day <- as.POSIXct(survey_time$day, format = "%Y-%m-%d",
                              usetz = TRUE, tz = "SBT")
```

Sample background points in each buffer. The number of background points per day is proportional to the time on effort

```
# time on-effort histogram
hist(survey_time$time_on_effort, xlab="time ON-effort (min)", main = "")

i <- 0
background_sp <- lapply(buffer_pol,function(x){ i <<- i + 1
    # time on effort i that day
    t <- survey_time$time_on_effort[i]
    # generate a regular grid of point separated by 5km
    s <- sp::spsample(x, type = "regular", iter = 100000, cellsize = 5000)
    # randomly sample points from those points previously produced
    s2 <- sample(s, (round(t/15,0) + 5)) #sample four points per hour of effort, plus 5 points per day
    return(s2)
})
```

For instance, on day 2014-07-29:



Convert background points spatial objects to a single dataframe object reprojected in EPSG4326 (lat/long)

```
background_df <- ldply(background_sp, function(x){
    x <- spTransform(x, "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
    df <- data.frame(x@coords)
    df$presence <- 0
    return(df)
})
names(background_df) <- c("day","lon","lat","presence")
background_df$day <- as.POSIXct(background_df$day, format = "%Y-%m-%d",
                                tz = "SBT", usetz = TRUE)
```

3- CREATE FINAL BINOMIAL DATASET & EXTRACT ENVIRONMENTAL VARIABLES

Combine presence and background points in the same dataset

Prepare presence points dataframe

```
##            day      lon       lat presence    mercX    mercY
## 181 2003-07-21 167.0120 -22.45187        1 223972.4 -2549552
## 182 2003-07-21 166.9472 -22.40007        1 216765.2 -2543350
## 183 2003-07-23 166.8732 -22.38635        1 208521.9 -2541708
## 184 2003-07-25 167.0327 -22.58678        1 226275.7 -2565718
## 185 2003-07-26 167.0784 -22.49272        1 231368.1 -2554445
## 186 2003-07-26 167.0086 -22.45628        1 223600.4 -2550082
```

Prepare background points dataframe to match the same format

```
# adding a few columns to match the structure of groups.df = presences
background_df$day <- as.POSIXct(background_df$day,
                                format = "%Y-%m-%d", tz = "SBT", usetz = T)
background_df <- background_df[c("day", "lon", "lat", "presence")]

# projecting coordinates in a custom mercator CRS
background_df[c("mercX","mercY")] <- project(as.matrix(cbind(background_df$lon,
                                                             background_df$lat)),
                              "+proj=merc +lon_0=165 +lat_1=-21.5 +k=1 +x_0=0 +y_0=0
                              +ellps=WGS84 +datum=WGS84 +units=m +no_defs")

head(background_df)
```

```
##          day      lon       lat presence    mercX    mercY
## 1 2003-07-20 166.9811 -22.37266        0 220537.5 -2540069
## 2 2003-07-20 167.0260 -22.60639        0 225537.5 -2568069
## 3 2003-07-20 166.9632 -22.48123        0 218537.5 -2553069
## 4 2003-07-20 166.9991 -22.59805        0 222537.5 -2567069
## 5 2003-07-20 166.9991 -22.61473        0 222537.5 -2569069
## 6 2003-07-20 166.9811 -22.43113        0 220537.5 -2547069
```

Combine presences with background points

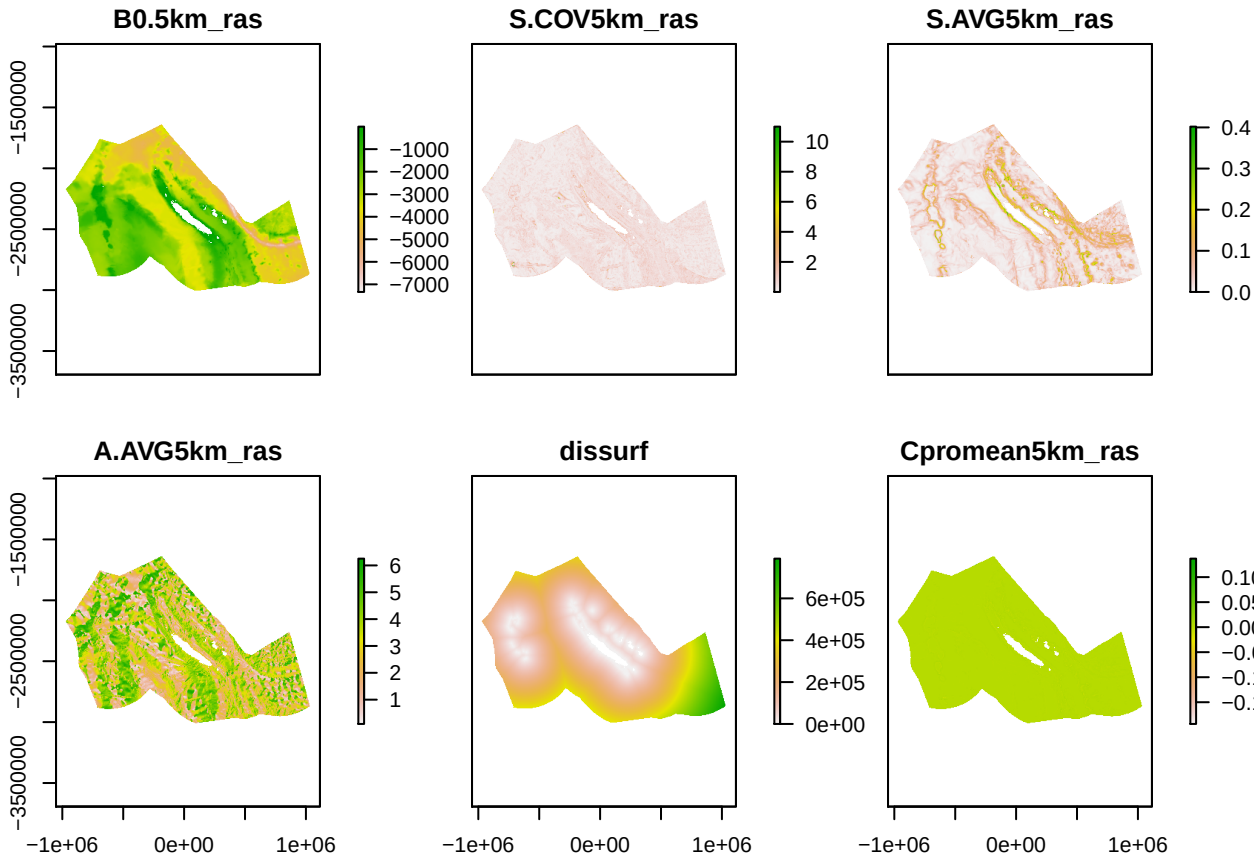
```
PAsurvey_df <- rbind(pres_df, background_df)
str(PAsurvey_df)
```

```
## 'data.frame':    18046 obs. of  6 variables:
##  $ day     : POSIXct, format: "2003-07-21" "2003-07-21" ...
##  $ lon     : num  167 167 167 167 167 ...
##  $ lat     : num  -22.5 -22.4 -22.4 -22.6 -22.5 ...
##  $ presence: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ mercX   : num  223972 216765 208522 226276 231368 ...
##  $ mercY   : num  -2549552 -2543350 -2541708 -2565718 -2554445 ...
```

Extract topographic environmental variables

Environmental conditions are extracted at the positions of presence and background points using a RasterStack object **env_stack** Environnemental shapefiles and rasters

```
# env_stack is a RasterStack object containing topographic environmental layers
# such as bathymetry, slope etc.
plot(env_stack[[c("B0.5km_ras","S.COV5km_ras", "S.AVG5km_ras", "A.AVG5km_ras",
                  "dissurf", "Cpromean5km_ras")]])
```



Function **Fun_Extract** designed to extract data from rasters and jitter points by a few pixels if necessary to extract the closest non-NA value. This function becomes useful when the whales have been observed very close to the reef or coast and the resolution of the rasters is not fine enough to place the position in a non-NA cell (the point has to be moved by a few hundred meters so that it does not appear to be on land).

```
# extracts data from a raster stack. Jitters points by a few pixels if necessary so that
# there are no NA left
# coordinates = vector of two column names = columns contain X and Y coordinates in the
# right projection
# layernames = vector of the names of the layers we want to extract from in the
# RasterStack
# stackR = RasterStack object
# d = dataframe from which coordinates are extracted and to which environmental columns
# will be appended
# search buffer is half the size of the extent from which we are going to pull a value if
# the pixel underlying a position is NA.
  # Default is set to 10km, that is to say we are looking in a box of 20km*20km.

Fun_Extract <- function(d, stackR, layernames, coordinates, search_buffer = 10000){
  for (v in layernames){
    # get raster from stack
    r <- stackR[[v]]
```

```
    # simple extraction
    d[,names(r)] <- raster::extract(r, d[coordinates], method = "simple")
    d_nona <- d[!is.na(d[,names(r)]), ]
    # some points' positions may be a little bit imprecise and fall in land polygons
    # OR small gaps in the raster
    d_na <- d[is.na(d[,names(r)]), ]
    # for these points we use the value of the closest non-NA pixel
    if (nrow(d_na)>0){
      d_na[,names(r)] <- apply(X = d_na[coordinates], MARGIN = 1, FUN = function(xy) {
        # cropping the raster to speed up the process (97% faster then if not cropped)
        r2 <- crop(r, extent(xy[1] - search_buffer, xy[1] + search_buffer,
                             xy[2] - search_buffer, xy[2] + search_buffer))
        v <- values(r2)[which.min(replace(distanceFromPoints(r2, xy), is.na(r2), NA))][1]
        return(v)
      })
    }
    # bind everything together --> no NA
    d <- rbind(d_nona, d_na)
  }
  return(d)
}
```

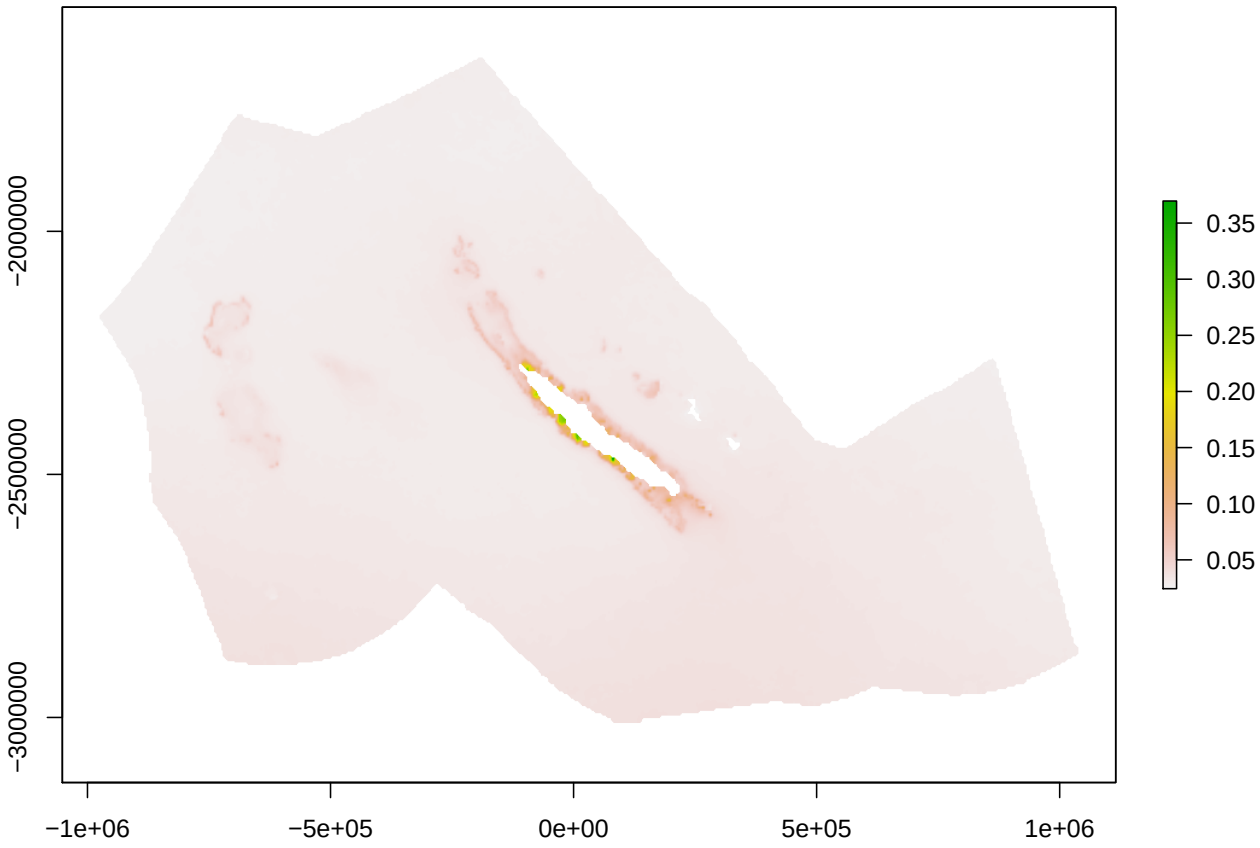
```
# variables to extract from env_stack, corresponding to bathymetry, slope
# coefficient of variation, slope average, aspect, distance to closest reef or
# land, and profile curvature
var <- c("BO.5km_ras","S.COV5km_ras", "S.AVG5km_ras",
         "A.AVG5km_ras","dissurf", "Cpromean5km_ras")

# apply the custom Fun_Extract function
PAsurvey_df <- Fun_Extract(d = PAsurvey_df, stack = env_stack, layernames = var,
                           coordinates = c("mercX", "mercY"))
```

Extract dynamic environmental variables: the case of diffuse attenuation - K490

Here, not only do we need to extract the environmental data at a given position, but also at a given time. The temporal resolution of this analysis is set to a month. Each position in the PAsurvey_df dataframe was recorded on a given day, on a given month (from June to October of each year). We retrieve the value of K490 from the corresponding monthly raster produced from MODIS remotely sensed satellite data (https://coastwatch.pfeg.noaa.gov/erddap/info/erdMH1kd4908day/index.html).

K490 data is stored as annual RasterBricks saved in a folder in the working directory (k490_path is the path to this folder)

K490 is a measure of turbidity, and in the New Caledonian region, it may be considered a proxy to the lagoon habitats. The average winter K490 from 2003 to 2016 is shown below:



Extracting values from RasterBricks

```
years <- seq(2003,2016,1)
count <- 0
PAsurvey_df <- adply(list.files(k490_path, full.names = T), .margins = 1, function(s){
  # read annual stacks of k490 data
  load(s) # load the yearly stack: k490_eez_ras
  count <<- count + 1
  # select points from that year
  d <- PAsurvey_df[as.character(as.POSIXlt(PAsurvey_df$day)$year+1900) == years[count], ]
  p <- ddply(d, ~ month, function(df){ # read the PAsurvey dataframe per month
    df$k490_m <- Fun_Extract(d = df, stack = k490_month, coordinates = c("lon","lat"),
                         layernames = paste("X", df$month[1], sep = ""))[, ncol(df) + 1]
    return(df)
  })
  return(p)
})
```

Scale all variables

Scale the variables and store them in the same dataframe (adding '.s' as a suffix to the column names)

```
PAsurvey_df[paste(names(PAsurvey_df[c(7:18)]), ".s", sep="")] <- base::scale(
  PAsurvey_df[c(7:18)], center = T, scale = T)

############################################################
save(PAsurvey_df, file = "./Outputs/PAsurvey_df.RData")
```

Create training and evaluation datasets

```
training.df.day <- list()
evaluation.df.day <- list()
for (i in 1:50){
  #remove 10% of days = equivalent to 64 days of survey
  days_out <- unique(PAsurvey_df$day)[
    sample(c(1:length(unique(PAsurvey_df$day))), 64, replace=F)]
  t <- PAsurvey_df[!(PAsurvey_df$day %in% days_out),]
  e <- PAsurvey_df[PAsurvey_df$day %in% days_out,]
  # add weights
  t$weights <- 1
  t[t$presence == 0, ]$weights <- nrow(t[t$presence == 1, ]) / nrow(t[t$presence == 0, ])
  e$weights <- 1
  e[e$presence == 0, ]$weights <- nrow(e[e$presence == 1, ]) / nrow(e[e$presence == 0, ])
  # save in list
  training.df.day <- c(training.df.day, list(t))
  evaluation.df.day <- c(evaluation.df.day, list(e))
}
```