```
#GBIF DATA ACCESSED ON 2023-05-22
#GBIF data corresponding to this work are available at DOI: https://doi.org/10.15468/dd.zc83q5

#-------------------------------------------------------------------------------
#Title: "Future increase of filamentous cyanobacteria in coastal Baltic Sea predicted by
multiple realm models-
#of marine, terrestrial, and climate change scenarios"
#Author: "Abdelgadir et al.
#The R code file contains value extraction from raster layers
#-------------------------------------------------------------------------------

########################################
## Upload required packages
########################################
#BEFORE STARTING:
#Note: replace the working directory folder "R-Directory" (~\\R\\R-Directory\\) with your
desired working path.

library(tidyverse)
library(sdmpredictors)
library(raster)
library(sp)
library(dismo)
library(writexl)

#######################
## DOWNLOADING PREDICTORS from Bio-Oracle 2.2 database: https://bio-oracle.org/ -
# and Bio-climatic variables from WorldClim database: https://www.worldclim.org/data/bioclim
######################

#Create a folder in a desired location where you can save all raster layers (to avoid re-
downloading data in next sessions).
# Download raster layers to sdmpredictors/Meta folder and import into R

options(sdmpredictors_datadir = "C:/R-4.0.2/library/sdmpredictors/Meta/")

#First: upload the marine raster layers. Name them mar1 and mar2

mar1 <- load_layers( layercodes = c('BO22_parmax','BO22_parmean','BO22_nitratemax_ss',
                                    'BO22_nitratemean_ss','BO22_nitratemin_ss',
                                    'BO22_nitraterange_ss', 'BO22_nitrateltmin_ss',
                                    'BO22_nitrateltmax_ss','BO22_phosphatemax_ss',
                                    'BO22_phosphatemean_ss','BO22_phosphatemin_ss',
                                    'BO22_phosphaterange_ss','BO22_phosphateltmax_ss',
                                    'BO22_phosphateltmin_ss','BO22_ppmax_ss','BO22_ppmean_ss',
                                    'BO22_ppltmin_ss','BO22_ppmin_ss','BO22_pprange_ss',
                                    'BO22_ppltmax_ss','BO22_salinityltmax_ss',
                                    'BO22_salinityltmin_ss','BO22_salinitymax_ss',
                                    'BO22_salinitymean_ss','BO22_salinitymin_ss',
                                    'BO22_salinityrange_ss',
                                    'BO22_dissoxmax_ss','BO22_dissoxmean_ss',
                                    'BO22_dissoxmin_ss','BO22_dissoxrange_ss',
                                    'BO22_dissoxltmax_ss','BO22_dissoxltmin_ss'),
                                     equalarea=FALSE, rasterstack=TRUE)


mar2 <- load_layers( layercodes = c("MS_biogeo13_sst_mean_5m","MS_biogeo14_sst_min_5m",
                                    "MS_biogeo15_sst_max_5m","MS_biogeo16_sst_range_5m",
                                    "MS_biogeo17_sst_variance_5m","MS_bathy_5m"),
                                     equalarea=FALSE, rasterstack=TRUE)

#Stack mar1 and mar2

mar= stack(mar1,mar2)

#Next: upload the terrestrial raster layers. Name it (terr)
```

```
terr <- load_layers( layercodes = c("WC_bio1","WC_bio5","WC_bio6","WC_bio7",
                                    "WC_bio8","WC_bio9","WC_bio10","WC_bio11",
                                    "WC_bio12","WC_bio13","WC_bio14","WC_bio16",
                                    "WC_bio17","WC_bio18","WC_bio19",
                                    "WC_alt"),equalarea=FALSE,rasterstack=TRUE)

#############################################################################################
# Define a boundary
boundary = extent(c(xmin = 10, xmax = 30, ymin = 53, ymax = 66))

# Crop raster layers to boundary extent
mar = crop(mar, boundary)
terr = crop(terr, boundary)

mar
terr

plot(mar)
plot(terr)

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

#Prepare and import data file point data containing longitude and latitude points

Data <- read.csv("~/R/R-Directory/combinedDF.csv")

#Check the data file
Data

set.seed(123)

#Check the new dataframe and make SpatialPointsDataFrame
Data$decimalLongitude <- as.numeric(Data$decimalLongitude)
Data$decimalLatitude <- as.numeric(Data$decimalLatitude)
coordinates(Data) = ~ decimalLongitude + decimalLatitude


#Convert tibble to a SpatialPoints object and set coordinate reference system (CRS).

Data = SpatialPoints(Data, proj4string = CRS("+proj=longlat +datum=WGS84"))

Data

projection(Data) == projection(mar)

#Check points CRS matches raster CRS.
projection(Data) == projection(mar)


#Create a tibble or data.frame to store Bio-ORACLE marine data for each point.
Final.Data = tibble(ID = 1:nrow(Data@coords),
                     Lon = Data$decimalLongitude,
                     Lat = Data$decimalLatitude
)

Final.Data

#Extract data for each point
#Combine all rasters into one raster stack.
rasters = raster::stack(mar,terr)
nlayers(rasters)

#Extract data from each raster layer for each point and store in a list.

store_data = list()
for (i in 1:nlayers(rasters)){
  store_data[[i]] = raster::extract(rasters[[i]], Data)
}
```

```
#Add the extracted data as new columns to Final.Data.
# Name variables in the list and then combine data
names(store_data) = names(rasters)
Final.Data = bind_cols(Final.Data, as_tibble(store_data))
Final.Data

#Check for NA values and drop rows if required.

# Check each column for NA values
na.check = map_int(Final.Data, ~sum(is.na(.)))
summary(na.check > 0)

# if needed remove NA records (REMOVE # and run the line)
#Final.Data = Final.Data %>% drop_na

#If nedded, round values to three decimal places. (REMOVE # and run the line)
#Final.Data[-(1:4)] = apply(Final.Data[-(1:4)], MARGIN = 2, FUN = round, digits = 3)


#Export data to a csv file and name it "Current"

write_csv(Final.Data, path = "CurrentDataCyanobacteria.csv")

#OR

#Convert to excel file and name it "CURRENT"

write_xlsx(Final.Data, "~\\R\\R-Directory\\CurrentDataCyanobacteria.xlsx")


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

###################################
#Now we do the same and extract the future values
###################################

#-------------------------------------------------------
#Download the Future MAR raster layers for years 2050 and 2100 at RCP85
#-------------------------------------------------------

# Download raster layers to sdmpredictors/Meta folder and import into R
options(sdmpredictors_datadir = "C:/R-4.0.2/library/sdmpredictors/Meta/")

future1 <- load_layers( layercodes = c("BO22_RCP85_2050_templtmax_ss",
"BO22_RCP85_2050_templtmin_ss",
                                       "BO22_RCP85_2050_tempmax_ss",
"BO22_RCP85_2050_tempmin_ss",
                                       "BO22_RCP85_2050_tempmean_ss",
"BO22_RCP85_2050_temprange_ss"),
                        equalarea=FALSE, rasterstack=TRUE)

future2 <- load_layers( layercodes = c("BO22_RCP85_2100_templtmax_ss",
"BO22_RCP85_2100_templtmin_ss",
                                       "BO22_RCP85_2100_tempmax_ss",
"BO22_RCP85_2100_tempmin_ss",
                                       "BO22_RCP85_2100_tempmean_ss",
"BO22_RCP85_2100_temprange_ss"),
                        equalarea=FALSE, rasterstack=TRUE)

#Stack mar layers
marf = stack(future1, future2)

#-------------------------------------------------------
#Download the Future TERR raster layers for years 2050 and 2070 at RCP85
#-------------------------------------------------------

# Download raster layers to sdmpredictors/Meta folder and import into R
```

```
options(sdmpredictors_datadir = "C:/R-4.0.2/library/sdmpredictors/Meta/")

future3<- raster::getData('CMIP5', var='bio', res=2.5, rcp=85, year=50,model='AC')

#Drop layers Bio02, Bio03, Bio04, Bio15
future3<-dropLayer(future3, c("ac85bi502", "ac85bi503", "ac85bi504","ac85bi5015"))

future4<- raster::getData('CMIP5', var='bio', res=2.5, rcp=85, year=70,model='AC')

#Drop layers Bio02, Bio03, Bio04, Bio15
future4<-dropLayer(future4, c("ac85bi702", "ac85bi703", "ac85bi704","ac85bi7015"))

#Stack terr layers
terrf = stack(future3, future4)

###############################################################################################
# Define a boundary
boundary = extent(c(xmin = 10, xmax = 30, ymin = 53, ymax = 66))

# Crop rasters to boundary extent
marf = crop(marf, boundary)
terrf = crop(terrf, boundary)

marf
terrf

plot(marf)
plot(terrf)

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

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

#Prepare and import data file point data containing longitude and latitude points

Data <- read.csv("~/R/R-Directory/combinedDF.csv")

#Check the data file
Data

set.seed(123)

#Check the new dataframe and make SpatialPointsDataFrame
Data$decimalLongitude <- as.numeric(Data$decimalLongitude)
Data$decimalLatitude <- as.numeric(Data$decimalLatitude)
coordinates(Data) = ~ decimalLongitude + decimalLatitude

#Convert tibble to a SpatialPoints object and set coordinate reference system (CRS).

Data = SpatialPoints(Data, proj4string = CRS("+proj=longlat +datum=WGS84"))

Data

projection(Data) == projection(marf)

#Check points CRS matches raster CRS.
projection(Data) == projection(marf)


#Create a tibble or data.frame to store Bio-ORACLE marine data for each point.
Final.Data = tibble(ID = 1:nrow(Data@coords),
                     Lon = Data$decimalLongitude,
                     Lat = Data$decimalLatitude
)

Final.Data

#Extract data for each point and combine all rasters into one raster stack.
```

```
rasters = raster::stack(marf,terrf)
nlayers(rasters)

#Extract data from each raster layer for each point and store in a list.

store_data = list()
for (i in 1:nlayers(rasters)){
  store_data[[i]] = raster::extract(rasters[[i]], Data)
}

#Add the extracted data as new columns to Final.Data.
# Name variables in the list and then combine data
names(store_data) = names(rasters)
Final.Data = bind_cols(Final.Data, as_tibble(store_data))
Final.Data

#Check for NA values and drop rows if required.

# Check each column for NA values
na.check = map_int(Final.Data, ~sum(is.na(.)))
summary(na.check > 0)

# If needed, remove NA records (REMOVE the # and run the line).
#Final.Data = Final.Data %>% drop_na

#If needed, round values to three decimal places (REMOVE the # and run the line).
#Final.Data[-(1:4)] = apply(Final.Data[-(1:4)], MARGIN = 2, FUN = round, digits = 3)

#Export data to a csv file and name it "Future"

write_csv(Final.Data, path = "FutureDataCyanobacteria.csv")

#OR

#Convert to excel file and name it "Future"

write_xlsx(Final.Data, "~\\R\\R-Directory\\FutureDataCyanobacteria.xlsx")
```