Ecological valuation assessment

Marie-Catherine Bouquieaux, Willem Boone, Hanneloor Heynderickx

01/10/2024

Contents

Project: MARBEFES

Vlaams Instituut voor de Zee/Flanders Marine Institute
Jacobsenstraat 1
8400 Oostende, Belgium

1. Install & load required libraries

If you like to knit the Rmarkdown script into a PDF an additional installation of a LaTeX program is required. Option to use tinytex::install_tinytex(). In addition, this Rmarkdown was created and ran using the following version:

- R: 4.3.2
- Rstudio: 2023.12.1
- Computer: Windows 11
- Package: last version available on the 08 of March 2024, except for ggplot2, for which the version is specified below

List of packages used

```
my_packages <- c("data.table", "sf", "raster",
    "tidyr", "dplyr", "rnaturalearth", "rnaturalearthdata",
    "devtools", "knitr", "kableExtra", "ggplot2")
```

To run only if some packages mentioned above are not already installed

```
install.packages(my_packages)
```

Load packages

```
lapply(c(my_packages), require, character.only = TRUE)
```

2. Input parameters (Settings)

###2.1 Define the working environment

Input: Filepath containing the shapefile **and** shx, .dbf and .prj for each Ecological Component.

Output directory: Filepath for output files; make sure directory exists.

list_EC_result: List of the files with the EC results; please add the extension of the file as well (for example "EVA_EC_bird_layers.shp")

xmin/xmax: Minimum and maximum longitude value to be used as limit on the map; must be provided in degrees.

ymin/ymax: Minimum and maximum latitude value to be used as limit on the map; must be provided in degrees.

quantile_classification: Please indicate TRUE if you prefer to utilize a distribution-based method for classifying your results, where the categories are determined by quantiles (20,40,60,80,100). If FALSE is enter, specific categories will be used and can be changed/defined in point 7.

CreateGrid: create a grid for subzones. TRUE or FALSE parameter. (alternative to using shapefile polygons).

```
## Path settings for input and output data
Input <- "~/Documents/EVA_process/output/"
Output <- "~/Documents/EVA_process/final_EV/"

## Test if output path exists, if not create
## it
if (!dir.exists(Output)) {
    dir.create(Output, recursive = TRUE)
}

# List of files:
list_EC_result_3km <- c("EVA_EC_bird_layers.shp",
    "EVA_EC_fish_layers.shp", "EVA_EC_mammal_layers.shp",
    "EVA_EC_phytoplankton_layers.shp", "EVA_EC_zooplankton_layers.shp")
list_EC_result_250m <- c("EVA_EC_benthos_layers.shp")

# Max/min latitude and longitude
xmin <- 1.5
xmax <- 3.5
ymin <- 51
ymax <- 55.5


## Path settings for the coordinates of the
```

```
## BBT Complete path and file name
## containing the BBT information
BBT_coordinates <- "~/Documents/EVA_process/BBT_coordinates/centerlinev2_20kmbuffer.gpkg"
# If a specific layers need to be used,
# enter name here:
BBT_layer <- ""

# Create grid?
CreateGrid <- TRUE

# Show result with continous scale or with
# discrete variables
continous <- TRUE
```

2.2 Get surrounding country map information and BBT/grid coordinates

Loading the coordinates of the BBT and compute its projection in the appropriate format.

```
# You should have specify the path/name of
# your file above, as well as the layer name
# is any is present
if (nchar(BBT_layer) > 0) {
    polygon_bbt <- st_read(BBT_coordinates, layer = BBT_layer)
} else {
    polygon_bbt <- st_read(BBT_coordinates)
}
# Change projection of the coordinates to
# ease the grid calculation
Polygon_coordinated_32631 <- st_transform(polygon_bbt,
    crs = st_crs(32631))
```

Get surrounding country map information

```
world_data <- ne_countries(scale = "medium", returnclass = "sf") |>
    sf::st_transform(crs = "EPSG:4326") |>
    sf::st_crop(c(xmin = xmin, xmax = xmax, ymin = ymin,
        ymax = ymax))
```

Get GRID coordinates

```
Polygon_coordinated_4326 <- Polygon_coordinated_32631 %>%
    sf::st_transform(crs = "EPSG:4326")

if (CreateGrid == FALSE) {
    # to check If your file contains
    # different layers, specify which one
    # should be used by entering the option
    # : layer = 'name'' If there are no
    # different layers present, remove the
    # option layer = 'name'
    GRD <- st_read(SubzoneShapeName)
    # Change projection of the coordinates
```

```
    # to ease the grid calculation.
    Polygon_coordinated_32631 <- st_transform(polygon_bbt,
        crs = st_crs(32631))
    # Create a dataframe with both the
    # grid_polygon and grid_id for each
    # subzone.
    grid_polygons_sf <- st_sf(grid_id = seq_along(GRD[Polygon_coordinated_32631$geom]),
        geometry = GRD[Polygon_coordinated_32631$geom],
        crs = 32631)

} else if (CreateGrid == TRUE) {
    # Create grid
    GRD <- st_make_grid(Polygon_coordinated_32631$geom,
        cellsize = 3000, square = FALSE)

    # Create a dataframe with both the
    # grid_polygon and grid_id for each
    # subzone.
    grid_polygons_sf <- st_sf(grid_id = seq_along(GRD[Polygon_coordinated_32631$geom]),
        geometry = GRD[Polygon_coordinated_32631$geom],
        crs = 32631)
}

GRID_degree_4326 <- st_transform(grid_polygons_sf,
    crs = "EPSG:4326")
```

3. Rescaling of the 250m grid

If you have EC that have been calculated using a 250m grid, please run the following code to resize it to a 3km grid.

3.1 Resize the smaller grid

```
if (any(nchar(list_EC_result_250m) > 0)) {
    # You should have specify the path/name
    # of your file above, as well as the
    # layer name is any is present
    if (nchar(BBT_layer) > 0) {
        polygon_bbt <- st_read(BBT_coordinates,
            layer = BBT_layer)
    } else {
        polygon_bbt <- st_read(BBT_coordinates)
    }
    # Change projection of the coordinates
    # to ease the grid calculation
    Polygon_coordinated_32631 <- st_transform(polygon_bbt,
        crs = st_crs(32631))

    # 250m grid
    GRD_250 <- st_make_grid(Polygon_coordinated_32631$geom,
        cellsize = 250, square = FALSE)
```

```
    # Create dataframe with both the grid
    # polygon and grid id for each subzones
    grid_polygons_sf_250 <- st_sf(grid_id = seq_along(GRD_250[Polygon_coordinated_32631$geom]),
        geometry = GRD_250[Polygon_coordinated_32631$geom],
        crs = 32631)
    # 3km grid
    GRD_3km <- st_make_grid(Polygon_coordinated_32631$geom,
        cellsize = 3000, square = FALSE)
    # Create dataframe with both the grid
    # polygon and grid id for each subzones
    grid_polygons_sf_3km <- st_sf(grid_id_3km = seq_along(GRD_3km[Polygon_coordinated_32631$geom]),
        geometry = GRD_3km[Polygon_coordinated_32631$geom],
        crs = 32631)

    hexagons_250 <- st_transform(grid_polygons_sf_250,
        crs = "EPSG:4326")
    hexagons_3000 <- st_transform(grid_polygons_sf_3km,
        crs = "EPSG:4326")

    # 250m grid associated to one 3km
    # hexagon
    inside_grid <- st_join(st_centroid(hexagons_250),
        hexagons_3000, join = st_intersects)
    inside_grid <- st_drop_geometry(inside_grid)

    data_EC_rescaled <- data.frame()

    for (j in length(list_EC_result_250m)) {
        data_to_rescale <- st_read(paste0(Input,
            list_EC_result_250m[j]))
        data_rescaled <- data_to_rescale %>%
            dplyr::select(grid_id, Total) %>%
            inner_join(inside_grid, by = "grid_id") %>%
            st_drop_geometry() %>%
            group_by(grid_id_3km) %>%
            summarise(Total_3k = mean(Total, na.rm = TRUE)) %>%
            mutate_all(~ifelse(is.nan(.), NA,
                .)) %>%
            drop_na(grid_id_3km) %>%
            full_join(grid_polygons_sf_3km, by = "grid_id_3km") %>%
            rename(grid_id = grid_id_3km)

        data_rescaled_classified <- QClassify(data_rescaled,
            data_rescaled$Total_3k, "Total")
        data_rescaled_classified$EC <- list_EC_result_250m[j]
        data_rescaled_classified <- data_rescaled_classified[,
            c("grid_id", "Total", "EC", "geometry")]
        data_EC_rescaled <- rbind(data_EC_rescaled,
            data_rescaled_classified)
    }
}
```

4. Computation of the final EV score

```
### Load shapefile with the results from
### each EC ###

data_EC <- data.frame()
for (i in 1:length(list_EC_result_3km)) {
    data_current <- st_read(paste0(Input, list_EC_result_3km[i]))
    data_current <- data_current[, c("grid_id",
        "Total", "geometry", "conf_score", "nb_sample")]
    data_current$EC <- list_EC_result_3km[i]
    data_EC <- rbind(data_EC, data_current)
}

if (exists("data_EC_rescaled") && is.data.frame(get("data_EC_rescaled"))) {

    data_geometry <- st_as_sf(data_EC_rescaled)
    st_crs(data_geometry) <- st_crs(data_EC)
    data_EC <- rbind(data_EC, data_geometry)
}

BBT_EC_EV <- data_EC %>%
    group_by(grid_id, geometry) %>%
    summarise(Total_EV = mean(Total, na.rm = TRUE),
        conf_score_EV = mean(conf_score, na.rm = TRUE),
        total_sample = sum(nb_sample, na.rm = TRUE),
        EC_used = sum(!is.na(conf_score)))

# Export the results to a shapefile
st_write(BBT_EC_EV, file.path(Output, "BBT_final_EV_layers.shp"),
    append = FALSE)
```

5. Map of the EV score

Plot EV score

```
if (continous == TRUE){
  print(ggplot()+
   geom_sf(data = BBT_EC_EV$geometry,lwd = 0) +
   geom_sf(data = BBT_EC_EV, aes(fill=BBT_EC_EV$Total_EV), colour = "grey90", lwd = 0) +
   scale_fill_viridis_c(name = "Ecological Valuation score", na.value = "grey94", limits = c(0,5)) +
   geom_sf(data = world_data$geometry) +
   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
         panel.background = element_blank(),
         axis.text.x=element_text(angle=300,hjust=0)))
ggsave(filename=file.path(Output, "Total_Valutation_Score.png"),
    width = NA,
    height = NA,
    units = "cm",
    dpi=1600)
} else{
  # Define the categories and colors
```

```
  categories <- c("0 - 1", "1 - 2", "2 - 3", "3 - 4", "4 - 5")
  mycolors <- viridisLite::viridis(5)
  # Create categories with explicit levels
  BBT_EC_EV$category_graph <- cut(
      BBT_EC_EV$Total_EV,
      breaks = c(0, 1, 2, 3, 4, 5),
      include.lowest = TRUE,
      labels = categories,
      # Force creation of all levels
      right = FALSE,
      # Ensure that all levels are present
      levels = categories
  )
  print(ggplot()+
   geom_sf(data = BBT_EC_EV$geometry,lwd = 0) +
   geom_blank() +
   geom_sf(data = BBT_EC_EV, aes(fill=category_graph), colour = "grey90", lwd = 0, show.legend = TRUE) +
     # show.legend allow to have colors in legend even for categories without data points
   scale_fill_manual(values = mycolors, name = "Ecological Valuation score",
                     limits = categories, na.value = "grey94",
                     labels = categories, drop = FALSE) +
   geom_sf(data = world_data$geometry) +
   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
         panel.background = element_blank(),
         axis.text.x=element_text(angle=300,hjust=0))
   )
ggsave(filename=file.path(Output, "Total_Valutation_Score.png"),
    width = NA,
    height = NA,
    units = "cm",
    dpi=1600)
}
```

Plot EV confidence score

```
if (continous == TRUE){
  print(ggplot()+
   geom_sf(data = BBT_EC_EV$geometry,lwd = 0) +
   geom_sf(data = BBT_EC_EV, aes(fill=conf_score_EV), colour = "grey90", lwd = 0) +
   scale_fill_viridis_c(name = "Confidence Score", na.value = "grey94", limits = c(0,5)) +
   geom_sf(data = world_data$geometry) +
   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
         panel.background = element_blank(),
         axis.text.x=element_text(angle=300,hjust=0)))
ggsave(filename=file.path(Output, "Total_confidence_score.png"),
    width = NA,
    height = NA,
    units = "cm",
    dpi=1600)
} else{
  # Define the categories and colors
  categories <- c("0 - 1", "1 - 2", "2 - 3", "3 - 4", "4 - 5")
  mycolors <- viridisLite::viridis(5)
```

```
  # Create categories with explicit levels
  BBT_EC_EV$category_graph <- cut(
      BBT_EC_EV$conf_score_EV,
      breaks = c(0, 1, 2, 3, 4, 5),
      include.lowest = TRUE,
      labels = categories,
      # Force creation of all levels
      right = FALSE,
      # Ensure that all levels are present
      levels = categories
  )
  print(ggplot()+
   geom_sf(data = BBT_EC_EV$geometry,lwd = 0) +
   geom_blank() +
   geom_sf(data = BBT_EC_EV, aes(fill=category_graph), colour = "grey90", lwd = 0, show.legend = TRUE) +
     # show legend allow to have colors in legend even for categories without data points
   scale_fill_manual(values = mycolors, name = "Confidence Score",
                     limits = categories, na.value = "grey94",
                     labels = categories, drop = FALSE) +
   geom_sf(data = world_data$geometry) +
   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
         panel.background = element_blank(),
         axis.text.x=element_text(angle=300,hjust=0))
   )
ggsave(filename=file.path(Output, "Total_confidence_score.png"),
    width = NA,
    height = NA,
    units = "cm",
    dpi=1600)
}
```