Ecological valuation assessment

Marie-Catherine Bouquieaux, Willem Boone, Hanneloor Heyndericks

01/10/2024

Contents

Project: MARBEFES

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

```r
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

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

Load packages

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

2. Input parameters (Settings)

REMINDER
Please ensure that you clean your R-environment before running the next ecological component.

2.1 Define the working environment

InputFile: Input file containing the data for an Ecological component; make sure to provide the entire filepath to it.

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

OutputFileName: Base path to the result files (maps and shapefile)

BBT_coordinates: File containing your BBT coordinates; make sure to provide the entire filepath to it.

BBT_layer: If your coordinates file contains several layers, give the name of the layer to be used. If only one layer is present, please leave empty: BBT_layer = " "

SubzoneShapeName: For working with an existing grid enter name of input shapefile with grid polygons; make sure to provide the entire filepath to it.

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.

continuous: Do you want your results to be displayed on a continuous scale (from 0 to 5) or on a categorical scale (5 categories: 1, 2, 3, 4 and 5)

expert_judgement, **modelling_data** and **monitoring_data**: define what type of data has been used. You must choose one of the following option: expert_judgement, modelling data or monitoring data. Put TRUE for the one you used and FALSE for the other.

component_habitat: Define if the ecological component analysed is the benthic habitats. Put TRUE if it is, or FALSE for any other ecological component.

```r
### Path settings for input and output data
InputFile <- "~/Documents/EVA_process/Data/EC_bird/EC_bird.csv"
Output <- "~/Documents/EVA_process/output/"
OutputFileName <- "EVA_EC_bird"

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

### 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. If there are no different
# layer, please leave empty: BBT_layer = ''
BBT_layer <- ""
# In case of grid already existing add here
# the paht and name of the file containing
# it
SubzoneShapeName <- c("")

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

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

# What type of data are you using?
expert_judgement <- FALSE
modelling_data <- FALSE
monitoring_data <- TRUE

# Benthic habitat?
component_habitat <- FALSE
```

- **CreateGrid:** create a grid for subzones. TRUE or FALSE parameter. (alternative to using shapefile polygons).
- **GridSize:** Desired gridcell size in meters (Note: calculations become rapidly slower when using smaller gridcell sizes).

```r
CreateGrid <- TRUE
GridSize <- 3000  #3km=3000 or 250m=250
```

2.2 Define the ecological significant species to be used. Put the first letter of each 'genus' in capitals.

If you do not have any ecological significant species, please leave *ESS <- c(" ")* as it is and run the code with the vector empty.

```r
### Example of species used for the North
### Sea BBT Bird ESS ESS <- c('Arenaria
### interpres', 'Larus argentatus', 'Larus
### fuscus', 'Sternula albifrons', 'Sterna
### hirundo', 'Sterna sandvicensis') Benthos
### ESS ESS <- c('Abra alba', 'Asterias
### rubens', 'Bathyporeia pilosa',
### 'Bathyporeia sarsi', 'Eumida sanguinea',
### 'Eurydice affinis', 'Eurydice pulchra',
### 'Kurtiella bidentata', 'Nephtys
### hombergii', 'Pariambus typicus',
### 'Pygospio elegans', 'Scolelepis
### squamata', 'Fabulina fabula') Fish ESS
### ESS <- c('Pomatoschistus lozanoi',
### 'Pomatoschistus microps',
### 'Pomatoschistus minutus')

### Enter your own list here:
ESS <- c()
```

2.3 Define the habitat forming keystone species. Put the first letter of each 'genus' in capitals.

If you do not have any habitat forming/keystone species, please leave *HFKS <- c(" ")* as it is and run the code with the vector empty.

```
### Example of species used for the North
### Sea BBT Benthos HFKS HFKS <-
### c('Arenicola marina', 'Jassa falcata',
### 'Lanice conchilega', 'Owenia
### fusiformis', 'Lagis koreni', 'Spiophanes
### bombyx','Spisula subtruncata')

### Enter your own list here:
HFKS <- c()
```

2.4 Define the mutualist or symbiotic species. Put the first letter of each 'genus' in capitals.

If you do not have any mutualist or symbiotic species, please leave *MSS <- c(" ")* as it is and run the code with the vector empty.

```
### Enter your own list here:
MSS <- c()
```

3. Import Dataset

Import of the dataset from the file defined above.

```
EcoComp1.0 <- fread(InputFile, data.table = FALSE)
```

4. Prepare the dataset

4.1. Format quality control

Match the column names of your dataframe to the column names listed in the table below. Note: the names need to be identical to those listed here! This is needed for further downstream calculations. The fields in **bold** are mandatory, while the fields in *italic* are only required if the 'Date' is not available or if the 'Abundance' is not available.

Information	Column name
Longitude	Longitude
Latitude	Latitude
Date	EventDate
FieldNumber	FieldNumber
Year	YearCollected
Month	MonthCollected
Day	DayCollected
Scientific name	ScientificName
Abundance	Density
Individual count	ObservedIndividualCount
Sample size/effort	SampleSize

The **FieldNumber** can be describe as containing an unique identifier for a location at a specific time, if this information is not present in your data, it will be calculated below but the column in itself **must** be present in your data.

Density If the data used comes from modelling data, it is assumed that density represents the percentage of coverage of a specific feature over a subzone. This percentage must then be divided by 100 so that the density ranges from 0 to 1.

Declare if the 'Date' or 'Abundance' still needs to be computed (TRUE or FALSE).

```r
# Compute date?
date_to_create <- FALSE
# Calculate abundance?
abundance_to_create <- FALSE
```

If your 'Date' field is missing, use the following code to create it.

```r
if (date_to_create == TRUE) {
    if ("YearCollected" %in% colnames(EcoComp1.0) &
        "MonthCollected" %in% colnames(EcoComp1.0) &
        "DayCollected" %in% colnames(EcoComp1.0)) {
        EcoComp1.0$eventDate <- paste(ifelse(is.na(EcoComp1.0$YearCollected),
            "", EcoComp1.0$YearCollected), ifelse(is.na(EcoComp1.0$MonthCollected),
            "", EcoComp1.0$MonthCollected), ifelse(is.na(EcoComp1.0$DayCollected),
            "", EcoComp1.0$DayCollected), collapse = NULL)
    }
    EcoComp1.0$eventDate <- replace(EcoComp1.0$eventDate,
        EcoComp1.0$eventDate == "", NA)
}
```

If your 'Abundance' field is missing use the following code to create it.

```r
if (abundance_to_create == TRUE) {
    EcoComp1.0$Density <- EcoComp1.0$ObservedIndividualCount/EcoComp1.0$SampleSize
}
```

This step is **NOT** optional. Create a field number if your dataframe doesn't have one yet.

```r
EcoComp1.0$FieldNumber <- ifelse(is.na(EcoComp1.0$FieldNumber),
    paste0((round(EcoComp1.0$Latitude, digits = 4))/100 +
        10000 * (round(EcoComp1.0$Longitude, digits = 4)),
        EcoComp1.0$EventDate), as.character(EcoComp1.0$FieldNumber))
```

Select the columns that are needed for the downstream analysis.

```r
EcoComp1.1 <- data.frame(Longitude = EcoComp1.0$Longitude,
    Latitude = EcoComp1.0$Latitude, EventDate = EcoComp1.0$EventDate,
    FieldNumber = EcoComp1.0$FieldNumber, ScientificName = EcoComp1.0$ScientificName,
    Density = EcoComp1.0$Density, stringsAsFactors = FALSE)
```

4.2. Geographic data quality control

Check if the longitude and latitude information is valid.

```r
EcoComp1.1 <- EcoComp1.1 %>%
    filter((Longitude >= -90 & Longitude <= 90) |
        (Latitude >= -180 & Latitude <= 180) |
        (Longitude >= (mean(Longitude) - 4 * sd(Longitude)) &
            Longitude <= (mean(Longitude) + 4 *
                sd(Longitude))) | (Latitude >=
        (mean(Latitude) - 4 * sd(Latitude)) &
        Latitude <= (mean(Latitude) + 4 * sd(Latitude))))
```

5. Prepare map and create spatial grid

5.1. Get BBT coordinates

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

```r
# 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

```r
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))
```

Map of your BBT

```r
Polygon_coordinated_4326 <- Polygon_coordinated_32631 %>%
    sf::st_transform(crs = "EPSG:4326")
print(ggplot() + geom_sf(data = Polygon_coordinated_4326$geom,
    fill = NA, color = "midnightblue") + geom_sf(data = world_data$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)))
```

5.2. Create grid

Create a grid if no polygons were provided (see settings defined in 2.1).

```r
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 = GridSize, 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)
}
```

Map of your grid

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

print(ggplot() + geom_sf(data = GRID_degree_4326$geometry,
    fill = NA, color = "midnightblue") + geom_sf(data = world_data$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)))
```

5.3 Map function

Function to create the different maps produced in the script

```r
### Map for distribution of dataset and rare species
produce_map <- function(dataset,data_used, title){
  print(ggplot()+
     geom_point(data = dataset,aes(Longitude,Latitude), shape = ".") +
     geom_sf(data = grid_polygons_sf$geometry, fill = NA) +
     geom_sf(data = world_data$geometry) +
     ggtitle(title) +
     theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
           panel.background = element_blank(),
           axis.text.x=element_text(angle=300,hjust=0),plot.title=element_text(hjust=0.5)))
  ggsave(filename=file.path(Output, paste0(OutputFileName,"_",data_used,"_distribution",".png")),
      width = NA,
      height = NA,
      units = "cm",
      dpi=1600)
}

### Map for AQ and total score
produce_map_sf <- function(dataset,class_column,name_AQ, continous){
  names(dataset)[names(dataset) %in% class_column] <- name_AQ
  # Combine the final score with the grid geometry.
  add_grid_geometry <- GRID_degree_4326 %>%
    left_join(dataset)
  add_grid_geometry[[name_AQ]] <- as.numeric(add_grid_geometry[[name_AQ]])

  if (continous == TRUE){
    print(ggplot()+
     geom_sf(data = add_grid_geometry$geometry,lwd = 0) +
     geom_sf(data = add_grid_geometry, aes(fill=.data[[name_AQ]]), colour = "grey90", lwd = 0) +
     scale_fill_viridis_c(name = name_AQ, na.value = "grey94", limits = c(0,5)) +
     geom_sf(data = world_data$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, paste0(OutputFileName,"_",name_AQ,".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
    add_grid_geometry$category_graph <- cut(
        add_grid_geometry[[name_AQ]],
        breaks = c(0, 1, 2, 3, 4, 5),
        include.lowest = TRUE,
        labels = categories,
        # Force creation of all levels
        right = FALSE,
        # Ensure that all levels are present
        levels = categories
    )
    print(ggplot()+
     geom_sf(data = add_grid_geometry$geometry,lwd = 0) +
     geom_blank() +
     geom_sf(data = add_grid_geometry, 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 = name_AQ, 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 = el
           axis.text.x=element_text(angle=300,hjust=0))
     )
  ggsave(filename=file.path(Output, paste0(OutputFileName,"_",name_AQ,".png")),
      width = NA,
      height = NA,
      units = "cm",
      dpi=1600)
  }

}
```

6. Mapping of data with grid

6.1 Mapping the sampling points with gridcells and add gridcell ID's to the dataset

In order to correctly link the dataset with the grid, we have to map the coordinates in the dataset to the grid_polygons.

```
EcoComp1.2 <- EcoComp1.1
EcoComp1.2$ID <- as.numeric(rownames(EcoComp1.2))

# Preparing SpatialPoints object
MyFields <- c("Longitude", "Latitude", "ID")
EcoComp1.2coord <- EcoComp1.2[MyFields]

# Create an sf object, this is required to
# merge grid and lat/long data together
```

```r
EcoComp1.2coord <- st_as_sf(EcoComp1.2coord, coords = c("Longitude",
    "Latitude"), crs = "EPSG:4326", remove = FALSE)

# Reproject the grid using degrees as unit
GRID_degree_4326 <- st_transform(grid_polygons_sf,
    crs = "EPSG:4326")

# Join the spatial points and the
# grid_polygons
combine_data_grid <- st_join(EcoComp1.2coord,
    GRID_degree_4326)

# Adding the grid_id to the original
# dataframe
EcoComp1.3a <- combine_data_grid %>%
    dplyr::select(ID, grid_id) %>%
    inner_join(EcoComp1.2, by = "ID")
# Select the data that is associated with a
# grid_id
EcoComp1.3 <- EcoComp1.3a[!is.na(EcoComp1.3a$grid_id),
    ]
# Create unique location numbers
EcoComp1.3$LocationNumber <- (round(EcoComp1.3$Latitude,
    digits = 4))/100 + 10000 * (round(EcoComp1.3$Longitude,
    digits = 4))
assessment_data <- EcoComp1.3
```

6.2 Distribution map of your data

```r
produce_map(assessment_data, "all_dataset", "Point of all the data available")
```

6.3 Dataset qualitative AQs/quantitative AQs

```r
# Qualitative AQs can be answered using all
# available data THis dataset is the
# assessment_data created above

# Quantitative AQs can only be answered
# using abundance data
Quantitative_data <- EcoComp1.3[which(EcoComp1.3$Density >
    0), ]
```

Distribution map of your quantitative data

```r
produce_map(Quantitative_data, "quantitative_dataset",
    "Abundance data available")
```

7. Calculating assessment questions (AQs)

It is **mandatory** to run all chunk of codes of this section (7.1 to 7.16), even if no ecologically significant species are present for example (AQ 10-11).

```
# Create a dataframe with the unique PolIDs
# that are linked to the dataset
PolIDs <- data.frame(grid_id = sort(unique(EcoComp1.3$grid_id),
    decreasing = FALSE))

# Number of GridPolygons (PolIDs) with data
NpolID <- dim(PolIDs)[1]

# Rescaling of the results x is the
# dataframe and y is the column; z is the
# name you want the added column to have
QClassify <- function(x, y, z) {
    max_value <- max(y, na.rm = TRUE)
    x <- x %>%
        mutate(class_1_5 = (5 * y)/max_value)
    names(x)[names(x) == "class_1_5"] <- z
    return(x)
}
```

7.1 Get a list of rare and common species

```


# Creating a list of unique grid_id -
# combinations
RareSpecList1 <- aggregate(ID ~ grid_id + ScientificName,
    data = assessment_data, length)

# Counting the number of grid_ids where each
# species occurs
RareSpecList2 <- aggregate(grid_id ~ ScientificName,
    data = RareSpecList1, length)

# Listing those species that occur in less
# then 5% of the gridcells
RareSpecList <- RareSpecList2[which(RareSpecList2$grid_id <=
    (NpolID * 5/100)), ]

# Listing those species that occur in more
# then 5% of the gridcells
CommonSpecList <- RareSpecList2[which(RareSpecList2$grid_id >
    (NpolID * 5/100)), ]

# Dataframe required for AQ9
EcoCompCommon_quant <- Quantitative_data[Quantitative_data$ScientificName %in%
    CommonSpecList$ScientificName, ]
```

7.1.1 Rare and common species list per subzone. Distribution map of your locally rare species

```r
Rarespec_loc <- assessment_data[assessment_data$ScientificName %in%
    RareSpecList$ScientificName, ]
st_write(Rarespec_loc, file.path(Output, "locally_rare_species_distribution.shp"),
    append = FALSE)
produce_map(Rarespec_loc, "locally_rare_sp", "Locally rare species data available")

Rarespec_loc_quantitative <- Quantitative_data[Quantitative_data$ScientificName %in%
    RareSpecList$ScientificName, ]
st_write(Rarespec_loc_quantitative, file.path(Output,
    "locally_rare_species_abundance_distribution.shp"),
    append = FALSE)
produce_map(Rarespec_loc_quantitative, "locally_rare_sp_abundance",
    "Locally rare species abundance data available")
```

Write species list and count number of occurrence/abundance data available

```r
Rarespec_loc_count <- Rarespec_loc %>%
    group_by(ScientificName) %>%
    summarise(occurence_data_present = n(), .groups = "drop") %>%
    st_drop_geometry()

Rarespec_loc_quantitative_count <- Rarespec_loc_quantitative %>%
    group_by(ScientificName) %>%
    summarise(abundance_data_present = n(), .groups = "drop") %>%
    st_drop_geometry()

print(kbl(list(Rarespec_loc_count, Rarespec_loc_quantitative_count)) %>%
    kable_classic(full_width = F, html_font = "Cambria"))

write.csv(Rarespec_loc_count, file = file.path(Output,
    paste0(OutputFileName, "_local_rare_sp_count.csv")),
    row.names = FALSE)
write.csv(Rarespec_loc_quantitative_count, file = file.path(Output,
    paste0(OutputFileName, "_local_rare_sp_abundance_count.csv")),
    row.names = FALSE)
```

7.1.2 Rare species list on Regional Scale

This part needs extra steps:

- Create the required regional grid
- Map the data using the new grid

```r
# Create the regional grid
GRD_region <- st_make_grid(Polygon_coordinated_32631$geom,
    cellsize = 50000, square = TRUE)

# Create a dataframe with the grid_polygon
# and grid_id for each subzone
grid_polygons_sf_region <- st_sf(grid_id = seq_along(GRD_region[Polygon_coordinated_32631$geom]),
    geometry = GRD_region[Polygon_coordinated_32631$geom],
    crs = 32631)
# Get Spatialpoint from dataset Done in
# point 6: EcoComp1.2coord

# Reproject the grid using degrees as unit
GRID_region_degree_4326 <- st_transform(grid_polygons_sf_region,
    crs = "EPSG:4326")

# Joining the spatial points and the grid
# polygons
combine_data_grid_region <- st_join(EcoComp1.2coord,
    GRID_region_degree_4326)

# Add the PolIDs to the original dataframe
EcoComp_regiona <- combine_data_grid_region %>%
    dplyr::select(ID, grid_id) %>%
    inner_join(EcoComp1.2, by = "ID")
# Select the data that is associated with a
# grid_id
EcoComp_region <- EcoComp_regiona[!is.na(EcoComp_regiona$grid_id),
    ]
# Create unique location numbers
EcoComp_region$LocationNumber <- (round(EcoComp_region$Latitude,
    digits = 4))/100 + 10000 * (round(EcoComp_region$Longitude,
    digits = 4))

# Creating a list of unique grid_id -
# ScientificName combinations
RareSpecList1_region <- aggregate(ID ~ grid_id +
    ScientificName, data = EcoComp_region, length)

# Counting the number of grid_ids where each
# species occurs
RareSpecList2_region <- aggregate(grid_id ~ ScientificName,
    data = RareSpecList1_region, length)

# Listing those species that occur in less
# then 2% of the gridcells
RareSpecList_region <- RareSpecList2_region[which(RareSpecList2_region$grid_id <=
    (NpolID * 2/100)), ]
```

Distribution map of your regionally rare species

```r
Rarespec_reg <- assessment_data[assessment_data$ScientificName %in%
    RareSpecList_region$ScientificName, ]
st_write(Rarespec_reg, file.path(Output, "regionally_rare_species_distribution.shp"),
    append = FALSE)
produce_map(Rarespec_reg, "regionally_rare_sp",
    "Regionally rare species data available")

Rarespec_reg_quantitative <- Quantitative_data[Quantitative_data$ScientificName %in%
    RareSpecList_region$ScientificName, ]
st_write(Rarespec_reg_quantitative, file.path(Output,
    "regionally_rare_species_abundance_distribution.shp"),
    append = FALSE)
```

```r
produce_map(Rarespec_reg_quantitative, "regionally_rare_sp_abundance",
    "Regionally rare species abundance data available")
```

Write species list and count number of occurrence/abundance data available

```r
Rarespec_reg_count <- Rarespec_reg %>%
    group_by(ScientificName) %>%
    summarise(occurence_data_present = n(), .groups = "drop") %>%
    st_drop_geometry()

Rarespec_reg_quantitative_count <- Rarespec_reg_quantitative %>%
    group_by(ScientificName) %>%
    summarise(abundance_data_present = n(), .groups = "drop") %>%
    st_drop_geometry()

print(kbl(list(Rarespec_reg_count, Rarespec_reg_quantitative_count)) %>%
    kable_classic(full_width = F, html_font = "Cambria"))


write.csv(Rarespec_reg_count, file = file.path(Output,
    paste0(OutputFileName, "_region_rare_sp_count.csv")),
    row.names = FALSE)
write.csv(Rarespec_reg_quantitative_count, file = file.path(Output,
    paste0(OutputFileName, "_region_rare_sp_abundance_count.csv")),
    row.names = FALSE)
```

7.1.3 Rare species list on National Scale This part need extra steps:

- Create the required national grid
- Map the data using the new grid

```r
# Create a regional grid
GRD_nation <- st_make_grid(Polygon_coordinated_32631$geom,
    cellsize = 10000, square = TRUE)

# Create a dataframe with grid_polygons and
# grid_ids for each subzone
grid_polygons_sf_nation <- st_sf(grid_id = seq_along(GRD_nation[Polygon_coordinated_32631$geom]),
    geometry = GRD_nation[Polygon_coordinated_32631$geom],
    crs = 32631)
# Get Spatialpoint from dataset Done in
# point 6: EcoComp1.2coord

# Reproject the grid using degrees as unit
GRID_nation_degree_4326 <- st_transform(grid_polygons_sf_nation,
    crs = "EPSG:4326")

# Join the spatial points and the grid
# polygons
combine_data_grid_nation <- st_join(EcoComp1.2coord,
    GRID_nation_degree_4326)
```

```r
# Adding PolIDs to the original dataframe
EcoComp_nationa <- combine_data_grid_nation %>%
    dplyr::select(ID, grid_id) %>%
    inner_join(EcoComp1.2, by = "ID")
# Select the data that is associated with a
# grid_id
EcoComp_nation <- EcoComp_nationa[!is.na(EcoComp_nationa$grid_id),
    ]
# Create unique location numbers
EcoComp_nation$LocationNumber <- (round(EcoComp_nation$Latitude,
    digits = 4))/100 + 10000 * (round(EcoComp_nation$Longitude,
    digits = 4))

# Creating a list of unique grid_id -
# ScientificName combinations
RareSpecList1_nation <- aggregate(ID ~ grid_id +
    ScientificName, data = EcoComp_nation, length)

# Counting the number of grid_ids where each
# species occurs
RareSpecList2_nation <- aggregate(grid_id ~ ScientificName,
    data = RareSpecList1_nation, length)

# Listing those species that occur in less
# then 0.5% of the gridcells
RareSpecList_nation <- RareSpecList2_nation[which(RareSpecList2_nation$grid_id <=
    (NpolID * 0.5/100)), ]
```

Distribution map of your nationally rare species

```r
Rarespec_nat <- assessment_data[assessment_data$ScientificName %in%
    RareSpecList_nation$ScientificName, ]
st_write(Rarespec_nat, file.path(Output, "nationally_rare_species_distribution.shp"),
    append = FALSE)
produce_map(Rarespec_nat, "nationally_rare_sp",
    "Nationally rare species data available")

Rarespec_nat_quantitative <- Quantitative_data[Quantitative_data$ScientificName %in%
    RareSpecList_nation$ScientificName, ]
st_write(Rarespec_nat_quantitative, file.path(Output,
    "nationally_rare_species_abundance_distribution.shp"),
    append = FALSE)
produce_map(Rarespec_nat_quantitative, "nationally_rare_sp_abundance",
    "Nationally rare species abundance data available")
```

Write species list and count number of occurrence/abundance data available

```r
Rarespec_nat_count <- Rarespec_nat %>%
    group_by(ScientificName) %>%
    summarise(occurence_data_present = n(), .groups = "drop") %>%
    st_drop_geometry()
```

```r
Rarespec_nat_quantitative_count <- Rarespec_nat_quantitative %>%
    group_by(ScientificName) %>%
    summarise(abundance_data_present = n(), .groups = "drop") %>%
    st_drop_geometry()

print(kbl(list(Rarespec_nat_count, Rarespec_nat_quantitative_count)) %>%
    kable_classic(full_width = F, html_font = "Cambria"))


write.csv(Rarespec_nat_count, file = file.path(Output,
    paste0(OutputFileName, "_nation_rare_sp_count.csv")),
    row.names = FALSE)
write.csv(Rarespec_nat_quantitative_count, file = file.path(Output,
    paste0(OutputFileName, "_nation_rare_sp_abundance_count.csv")),
    row.names = FALSE)
```

7.2 AQ1: Is the subzone characterised by the presence of many locally rare features?

```r
if (!length(RareSpecList$ScientificName) == 0) {
    QPRS1 <- data.frame()
    # Loop through the whole data grid to
    # check for the presence of rare
    # species.  A value 5 is given if a rare
    # species is detected in the grid, if
    # not, a value 0 is given.
    for (i in 1:NpolID) {
        grid_evaluated <- assessment_data[assessment_data$grid_id %in%
            PolIDs$grid_id[i], ]
        grid_vector <- PolIDs$grid_id[i]
        for (j in 1:length(RareSpecList$ScientificName)) {
            if (RareSpecList$ScientificName[j] %in%
                grid_evaluated$ScientificName) {
                grid_vector <- append(grid_vector,
                  5)
            } else {
                grid_vector <- append(grid_vector,
                  0)
            }
        }
        QPRS1 <- rbind(QPRS1, grid_vector)
    }

    # Calculation of the average value for
    # each subzone
    QPRS2 <- QPRS1 %>%
        mutate(average_value = rowMeans(.[-1])) %>%
        rename(grid_id = 1) %>%
        dplyr::select(grid_id, average_value)

    # Classification into five categories
    QPRS3 <- QClassify(QPRS2, QPRS2$average_value,
        "QPRSCLASS")
    QPRS <- QPRS3[c("grid_id", "QPRSCLASS")]

    # Produce map of the results
    produce_map_sf(QPRS, "QPRSCLASS", "AQ1", continous)
} else {
    if (!exists("QPRS")) {
        QPRS <- PolIDs
        QPRS$QPRSCLASS <- NA
    }
}
```

7.3 AQ2: Is the subzone characterised by a high abundance of many locally rare features?

```
if (!length(RareSpecList$ScientificName) == 0) {

    QHARSa <- data.frame(grid_id = PolIDs)
    RareSpecList_quant <- RareSpecList$ScientificName

    # Confirm if there is abundance data
    # present for the rare species, if not,
    # remove it from list.
    for (i in RareSpecList$ScientificName) {
        info_presence <- Quantitative_data[which(Quantitative_data$ScientificName %in%
            i), ]
        if (dim(info_presence)[1] == 0) {
            RareSpecList_quant <- RareSpecList_quant[!RareSpecList_quant ==
                i]
        }
    }
    for (j in RareSpecList_quant) {
        # Select data for the rare species
        lrf_abundance <- Quantitative_data[Quantitative_data$ScientificName %in%
            j, ] %>%
            st_drop_geometry()
        # Get the average abundance values
        # of the rare species for each
        # subzone
        df_lrf_abundance_average <- lrf_abundance %>%
            select(grid_id, ScientificName, Density) %>%
            group_by(grid_id) %>%
            summarise_at(vars(Density), list(average = mean),
                .groups = "drop")
        # Get the lrf value for each subzone
        df_lrf_value <- df_lrf_abundance_average %>%
            inner_join(unique(lrf_abundance[,
                names(lrf_abundance) %in% c("grid_id",
                  "ScientificName")]), by = "grid_id") %>%
            mutate(lrf_value = 5 * (average/max(average)))
        # Add the lrf_value to the final
        # dataframe
        QHARSa <- left_join(QHARSa, df_lrf_value[,
```

```r
            c(1, 4)], by = "grid_id")
    }

    # Average the lrf_value results of all
    # the species for each subzone into
    # final metrics
    QHARSb <- QHARSa %>%
        mutate(average_lrf_value = rowMeans(.[-1],
            na.rm = TRUE)) %>%
        mutate_all(~ifelse(is.na(.), 0, .)) %>%
        select(grid_id, average_lrf_value)

    # Classification into five categories
    QHARSc <- QClassify(QHARSb, QHARSb$average_lrf_value,
        "QHARSCLASS")

    QHARS <- QHARSc[c("grid_id", "QHARSCLASS")]

    # Produce map of the results
    produce_map_sf(QHARS, "QHARSCLASS", "AQ2",
        continous)
} else {
    if (!exists("QHARS")) {
        QHARS <- PolIDs
        QHARS$QHARSCLASS <- NA
    }
}
```

7.4 AQ3: Is the subzone characterised by the presence of many regionally rare features?

```r
if (!length(RareSpecList_region$ScientificName) ==
    0) {

    QRRS1 <- data.frame()
    # Loop through the whole data grid to
    # check for the presence of rare species
    # A value 5 is given if the rare species
    # is detected in the grid, if not, a
    # value 0 is given
    for (i in 1:NpolID) {
        grid_evaluated <- assessment_data[assessment_data$grid_id %in%
            PolIDs$grid_id[i], ]
        grid_vector <- PolIDs$grid_id[i]
        for (j in 1:length(RareSpecList_region$ScientificName)) {
            if (RareSpecList_region$ScientificName[j] %in%
                grid_evaluated$ScientificName) {
                grid_vector <- append(grid_vector,
                  5)
            } else {
                grid_vector <- append(grid_vector,
                  0)
            }
        }
        QRRS1 <- rbind(QRRS1, grid_vector)
    }

    # Calculation of the average value for
    # each subzone
    QRRS2 <- QRRS1 %>%
        mutate(average_value = rowMeans(.[-1])) %>%
        rename(grid_id = 1) %>%
        select(grid_id, average_value)

    # Classification into five categories
    QRRS3 <- QClassify(QRRS2, QRRS2$average_value,
        "QRRSCLASS")

    QRRS <- QRRS3[c("grid_id", "QRRSCLASS")]

    # Produce map of the results
    produce_map_sf(QRRS, "QRRSCLASS", "AQ3", continous)
} else {
    if (!exists("QRRS")) {
        QRRS <- PolIDs
        QRRS$QRRSCLASS <- NA
    }
}
```

7.5 AQ4: Is the subzone characterised by high abundance of many regionally rare features?

```
if (!length(RareSpecList_region$ScientificName) ==
    0) {
    QARRSa <- data.frame(grid_id = PolIDs)
    RareSpecList_region_quant <- RareSpecList_region$ScientificName

    # Confirm if there is abundance data
    # present for the rare species, if not,
    # remove it from list.
    for (i in RareSpecList_region$ScientificName) {
        info_presence <- Quantitative_data[which(Quantitative_data$ScientificName %in%
            i), ]
        if (dim(info_presence)[1] == 0) {
            RareSpecList_region_quant <- RareSpecList_region_quant[!RareSpecList_region_quant ==
                i]
        }
    }

    for (j in RareSpecList_region_quant) {
        # Select data for the rare species
        lrf_abundance <- Quantitative_data[Quantitative_data$ScientificName %in%
            j, ] %>%
            st_drop_geometry()
        # Get the average abundance values
        # of the rare species for each
        # subzone
        df_lrf_abundance_average <- lrf_abundance %>%
            select(grid_id, ScientificName, Density) %>%
            group_by(grid_id) %>%
            summarise_at(vars(Density), list(average = mean),
                .groups = "drop")
        # Get the lrf value for each subzone
        df_lrf_value <- df_lrf_abundance_average %>%
            inner_join(unique(lrf_abundance[,
                names(lrf_abundance) %in% c("grid_id",
                  "ScientificName")]), by = "grid_id") %>%
            mutate(lrf_value = 5 * (average/max(average)))
        # Add the lrf value to the final
        # dataframe
        QARRSa <- left_join(QARRSa, df_lrf_value[,
            c(1, 4)], by = "grid_id")
    }

    # Average lrf value results of all the
    # species for each subzone into final
    # metrics
    QARRSb <- QARRSa %>%
        mutate(average_lrf_value = rowMeans(.[-1],
            na.rm = TRUE)) %>%
        mutate_all(~ifelse(is.na(.), 0, .)) %>%
        select(grid_id, average_lrf_value)

    # Classification into five categories
    QARRSc <- QClassify(QARRSb, QARRSb$average_lrf_value,
        "QARRSCLASS")

    QARRS <- QARRSc[c("grid_id", "QARRSCLASS")]

    # Produce map of the results
    produce_map_sf(QARRS, "QARRSCLASS", "AQ4",
        continous)
} else {
    if (!exists("QARRS")) {
        QARRS <- PolIDs
        QARRS$QARRSCLASS <- NA
    }
}
```

7.6 AQ5: Is the subzone characterised by the presence of many nationally rare features?

```
if (!length(RareSpecList_nation$ScientificName) ==
    0) {
    QNRS1 <- data.frame()
    # Loop through the whole data grid to
    # check for the presence of rare species
    # A value 5 is given if the rare species
    # is detected in the grid, if not, a
    # value 0 is given
    for (i in 1:NpolID) {
        grid_evaluated <- assessment_data[assessment_data$grid_id %in%
            PolIDs$grid_id[i], ]
        grid_vector <- PolIDs$grid_id[i]
        for (j in 1:length(RareSpecList_nation$ScientificName)) {
            if (RareSpecList_nation$ScientificName[j] %in%
                grid_evaluated$ScientificName) {
                grid_vector <- append(grid_vector,
                  5)
            } else {
                grid_vector <- append(grid_vector,
                  0)
            }
        }
        QNRS1 <- rbind(QNRS1, grid_vector)
    }

    # Calculation of the average value for
    # each subzone
    QNRS2 <- QNRS1 %>%
        mutate(average_value = rowMeans(.[-1])) %>%
        rename(grid_id = 1) %>%
        select(grid_id, average_value)

    # Classification into five categories
    QNRS3 <- QClassify(QNRS2, QNRS2$average_value,
        "QNRSCLASS")

    QNRS <- QNRS3[c("grid_id", "QNRSCLASS")]

    # Produce map of the results
    produce_map_sf(QNRS, "QNRSCLASS", "AQ5", continous)
} else {
    if (!exists("QNRS")) {
        QNRS <- PolIDs
        QNRS$QNRSCLASS <- NA
    }
}
```

7.7 AQ6: Is the subzone characterised by high abundance of many nationally rare features?

```
if (!length(RareSpecList_nation$ScientificName) ==
    0) {
    QANRSa <- data.frame(grid_id = PolIDs)
    RareSpecList_nation_quant <- RareSpecList_nation$ScientificName

    # Test if the rare species has abundance
    # data, if not, remove from list
    for (i in RareSpecList_nation$ScientificName) {
        info_presence <- Quantitative_data[which(Quantitative_data$ScientificName %in%
            i), ]
        if (dim(info_presence)[1] == 0) {
            RareSpecList_nation_quant <- RareSpecList_nation_quant[!RareSpecList_nation_quant ==
                i]
        }
    }

    for (j in RareSpecList_nation_quant) {
        # get data for that species
        lrf_abundance <- Quantitative_data[Quantitative_data$ScientificName %in%
            j, ] %>%
            st_drop_geometry()
        # get average abundance for each
        # subzones for that species
        df_lrf_abundance_average <- lrf_abundance %>%
            select(grid_id, ScientificName, Density) %>%
            group_by(grid_id) %>%
            summarise_at(vars(Density), list(average = mean),
                .groups = "drop")
        # get the lrf value for each
        # subzones
        df_lrf_value <- df_lrf_abundance_average %>%
            inner_join(unique(lrf_abundance[,
                names(lrf_abundance) %in% c("grid_id",
                  "ScientificName")]), by = "grid_id") %>%
            mutate(lrf_value = 5 * (average/max(average)))
        # add lrf value to final dataframe
        QANRSa <- left_join(QANRSa, df_lrf_value[,
            c(1, 4)], by = "grid_id")
    }

    # Average lrf_value results of all the
    # species for each subzone into final
    # metrics
    QANRSb <- QANRSa %>%
        mutate(average_lrf_value = rowMeans(.[-1],
            na.rm = TRUE)) %>%
        mutate_all(~ifelse(is.na(.), 0, .)) %>%
        select(grid_id, average_lrf_value)

    # Classification into five categories
    QANRSc <- QClassify(QANRSb, QANRSb$average_lrf_value,
        "QANRSCLASS")

    QANRS <- QANRSc[c("grid_id", "QANRSCLASS")]

    # Produce map of the results
    produce_map_sf(QANRS, "QANRSCLASS", "AQ6",
        continous)
} else {
    if (!exists("QANRS")) {
        QANRS <- PolIDs
        QANRS$QANRSCLASS <- NA
    }
```

```
}
```

7.8 AQ7: Is the number of features in the subzone high?

```
QNFH1 <- data.frame()
features_present <- unique(assessment_data$ScientificName)
# Loop through all the grid with data, to
# check for the presence of rare species THe
# value 5 is given if the rare species is
# detected in the grid, if not, the value 0
# is given
for (i in 1:NpolID) {
    grid_evaluated <- assessment_data[assessment_data$grid_id %in%
        PolIDs$grid_id[i], ]
    grid_vector <- PolIDs$grid_id[i]
    for (j in features_present) {
        if (j %in% grid_evaluated$ScientificName) {
            grid_vector <- append(grid_vector,
                5)
        } else {
            grid_vector <- append(grid_vector,
                0)
        }
    }
    QNFH1 <- rbind(QNFH1, grid_vector)
}

# Calculation of the average value for each
# subzone
QNFH2 <- QNFH1 %>%
    mutate(average_value = rowMeans(.[-1])) %>%
    rename(grid_id = 1) %>%
    select(grid_id, average_value)


# Classification into five categories
QNFH3 <- QClassify(QNFH2, QNFH2$average_value,
    "QNFHCLASS")

QNFH <- QNFH3[c("grid_id", "QNFHCLASS")]

# Produce map of the results
produce_map_sf(QNFH, "QNFHCLASS", "AQ7", continous)

if (!exists("QNFH")) {
    QNFH <- PolIDs
    QNFH$QNFHCLASS <- NA
}
```

7.9 AQ8: Is the subzone characterised by high counts of many species?

```
### QHCMS Step1 ### Determine the species
### which are regularly occurring in your
### study area by selecting all species
### which occur in more than 5 % of your
### records (this is done to exclude rare
### species from the species list).

# See above

QHCMSa <- data.frame(grid_id = PolIDs)
CommonSpecList_quant <- CommonSpecList$ScientificName

for (i in CommonSpecList$ScientificName) {
    info_presence <- Quantitative_data[which(Quantitative_data$ScientificName %in%
        i), ]
    if (dim(info_presence)[1] == 0) {
        CommonSpecList_quant <- CommonSpecList_quant[!CommonSpecList_quant ==
            i]
    }
}

if (length(CommonSpecList_quant) > 0) {
    for (j in CommonSpecList_quant) {
        # get data for that species
        lrf_abundance <- Quantitative_data[Quantitative_data$ScientificName %in%
            j, ] %>%
            st_drop_geometry()
        if (dim(lrf_abundance)[1] > 2) {
            # get average abundance for each
            # subzones for that species
            df_lrf_abundance_average <- lrf_abundance %>%
                dplyr::select(grid_id, ScientificName,
                  Density) %>%
                group_by(grid_id) %>%
                summarise_at(vars(Density), list(average = mean),
                  .groups = "drop")
            # get the lrf value for each
            # subzones
            df_lrf_value <- df_lrf_abundance_average %>%
                inner_join(unique(lrf_abundance[,
                  names(lrf_abundance) %in% c("grid_id",
                    "ScientificName", "ScientificName")]),
                  by = "grid_id") %>%
                mutate(lrf_value = ifelse(is.na(average),
                  NA, 5 * (average/max(average,
                    na.rm = TRUE))))
        } else next

        # add lrf value to final dataframe
        QHCMSa <- left_join(QHCMSa, df_lrf_value[,
            c(1, 4)], by = "grid_id")
    }

    # average lrf_value results of all the
    # species for each subzone into final
    # metrics
    QHCMSb <- QHCMSa %>%
        mutate(average_lrf_value = rowMeans(.[-1],
            na.rm = TRUE)) %>%
        mutate_all(~ifelse(is.na(.), 0, .)) %>%
        dplyr::select(grid_id, average_lrf_value)

    # CLassify into 5 categories
    QHCMSc <- QClassify(QHCMSb, QHCMSb$average_lrf_value,
        "QHCMSCLASS")

    QHCMS <- QHCMSc[c("grid_id", "QHCMSCLASS")]
    # Produce map of the results
    produce_map_sf(QHCMS, "QHCMSCLASS", "AQ8",
        continous)
} else {
    if (!exists("QHCMS")) {
        QHCMS <- PolIDs
        QHCMS$QHCMSCLASS <- NA
    }
}
```

7.10 AQ 9: Is the abundance of certain species very high in the subzone?

```
# At list 10 records with abundance data are
# needed, if less, this questions will not
# be calculated

if (dim(EcoCompCommon_quant)[1] > 10) {
    ### QHACS Step1 ### Determine the
    ### species which are regularly
    ### occurring in your study area by
    ### selecting all species which occur in
    ### more than 5 % of the subzones (this
    ### is done to exclude rare species from
    ### the species list).

    # see EcoCompCommon

    ### QHACS Step2 ### Determine the mean
    ### density of every species for the
    ### whole study area (= X).
    top5Perc <- (round(NpolID * 5/100, digits = 0))

    QHACS1 <- aggregate(Density ~ grid_id + LocationNumber +
        FieldNumber + ScientificName, data = EcoCompCommon_quant,
        sum)
    QHACS1a <- aggregate(Density ~ grid_id + LocationNumber +
```

```r
    FieldNumber + ScientificName, data = EcoCompCommon_quant,
    length)
QHACS1$AvgDens <- QHACS1$Density/QHACS1a$Density

QHACS2 <- aggregate(AvgDens ~ grid_id + LocationNumber +
    ScientificName, data = QHACS1, sum)
QHACS2a <- aggregate(AvgDens ~ grid_id + LocationNumber +
    ScientificName, data = QHACS1, length)
QHACS2$AvgDensLoc <- QHACS2$AvgDens/QHACS2a$AvgDens

QHACS3 <- aggregate(AvgDensLoc ~ grid_id +
    ScientificName, data = QHACS2, sum)
QHACS3a <- aggregate(AvgDensLoc ~ grid_id +
    ScientificName, data = QHACS2, length)
QHACS3$AvgDensPol <- QHACS3$AvgDensLoc/QHACS3a$AvgDensLoc  #Xi

# per species sum of densities of all
# subzones
QHACS4 <- aggregate(AvgDensPol ~ ScientificName,
    data = QHACS3, sum)

# Z (number of subzones for each
# species)
QHACS4a <- aggregate(AvgDensPol ~ ScientificName,
    data = QHACS3, length)

# X (density of entire area)
QHACS4$AvgDensTot <- QHACS4$AvgDensPol/QHACS4a$AvgDensPol

### QHACS Step3 ### Calculate the mean
### density of every species for every
### subzone (= Xi).

# see QHACS3

### QHACS Step4 ### Calculate the ratio
### Xi/X for every species in each
### subzone.

QHACS5 <- merge(QHACS3, QHACS4, by = "ScientificName")
QHACS5$XioverX <- QHACS5$AvgDensPol.x/QHACS5$AvgDensTot  #Xi/X

### QHACS Step5 ### Determine the 5 %
### subzones with the highest
### ratio.Calculate the percentage of
### the density of every species that
### occurs in the 5 % most important
### subzones.

QHACS6 <- xtabs(XioverX ~ grid_id + ScientificName,
    data = QHACS5)
SumTop5 <- function(t) sum(sort(t, decreasing = TRUE)[1:top5Perc])
QHACS6a <- apply(QHACS6, 2, SumTop5)
```

```r
    QHACS6b <- apply(QHACS6, 2, sum)
    QHACS7 <- as.data.frame(cbind(QHACS4a, QHACS6a/QHACS6b))
    names(QHACS7)[2:3] <- c("Z", "Y")

    ### QHACS Step6 ### Determine in how
    ### many subzones every species occurs
    ### (= Z).

    # see QHACS4a

    ### QHACS Step7 ### Calculate the ratio
    ### Y/Z which is the aggregation
    ### coefficient for each species.

    QHACS7$YoverZ <- QHACS7$Y/QHACS7$Z  #(Y/Z)
    QHACS8 <- merge(QHACS5, QHACS7, by = "ScientificName")

    ### QHACS Step8 ### Multiply the ratio
    ### Y/Z with the ratio Xi/X and divide
    ### these values in 5 classes with
    ### values between 1 and 5 (with an
    ### equal amount of subzones in each
    ### class).

    QHACS8$XioverXYoverZ <- QHACS8$XioverX * QHACS8$YoverZ  #Xi*Y/Z
    QHACS9 <- aggregate(XioverXYoverZ ~ grid_id,
        data = QHACS8, sum)

    # Classification into five categories
    QHACS <- QClassify(QHACS9, QHACS9$XioverXYoverZ,
        "QHACSCLASS")

    QHACS <- merge(PolIDs, QHACS, by = "grid_id",
        all.x = TRUE)
    QHACS <- QHACS[c("grid_id", "QHACSCLASS")]

    # Produce map of the results
    produce_map_sf(QHACS, "QHACSCLASS", "AQ9",
        continous)
} else {
    if (!exists("QHACS")) {
        QHACS <- PolIDs
        QHACS$QHACSCLASS <- NA
    }
}
```

7.11 AQ 10: Is the presence of ecologically significant features high in the subzone?

```r
if (length(ESS) > 0) {
    ESS_list <- ESS

    # Test if the ES species is found in the
    # data
    for (j in ESS) {
        info_presence <- assessment_data[which(assessment_data$ScientificName %in%
            j), ]
        if (dim(info_presence)[1] == 0) {
            ESS_list <- ESS_list[!ESS_list ==
                j]
        }
    }

    qual_QESS1 <- data.frame()
    # Loop through the whole data grid to
    # check for the presence of the ES
    # species A value 5 is given if the ES
    # species is detected in the grid, if
    # not, the value 0 is given
    for (i in 1:NpolID) {
        grid_evaluated <- assessment_data[assessment_data$grid_id %in%
            PolIDs$grid_id[i], ]
        grid_vector <- PolIDs$grid_id[i]
        for (k in ESS_list) {
            if (k %in% grid_evaluated$ScientificName) {
                grid_vector <- append(grid_vector,
                  5)
            } else {
                grid_vector <- append(grid_vector,
                  0)
            }
        }
        qual_QESS1 <- rbind(qual_QESS1, grid_vector)
    }

    # Calculation of the average value for
    # each subzone
    qual_QESS2 <- qual_QESS1 %>%
        mutate(average_value = rowMeans(.[-1])) %>%
        rename(grid_id = 1) %>%
        select(grid_id, average_value)

    # Classification into five categories
    qual_QESS3 <- QClassify(qual_QESS2, qual_QESS2$average_value,
        "qual_QESSCLASS")

    qual_QESS <- qual_QESS3[c("grid_id", "qual_QESSCLASS")]

    # Produce map of the results
    produce_map_sf(qual_QESS, "qual_QESSCLASS",
        "AQ10", continous)
} else {
    qual_QESS <- PolIDs
    qual_QESS$qual_QESSCLASS <- NA
}
```

7.12 AQ 11: Is the abundance of ecologically significant features high in the subzone?

```
if (length(ESS) > 0) {
    ESS_list <- ESS

    for (j in ESS) {
        info_presence <- Quantitative_data[which(Quantitative_data$ScientificName %in%
            j), ]
        if (dim(info_presence)[1] == 0) {
            ESS_list <- ESS_list[!ESS_list ==
                j]
        }
    }

    QESSa <- data.frame(grid_id = PolIDs)

    for (k in ESS_list) {
        # get data for that species
        lrf_abundance <- Quantitative_data[Quantitative_data$ScientificName %in%
            k, ] %>%
            st_drop_geometry()
        # get average abundance for each
        # subzones for that species
        df_lrf_abundance_average <- lrf_abundance %>%
            select(grid_id, ScientificName, Density) %>%
            group_by(grid_id) %>%
            summarise_at(vars(Density), list(average = mean),
                .groups = "drop")
        # get the lrf value for each
        # subzones
        df_lrf_value <- df_lrf_abundance_average %>%
            inner_join(unique(lrf_abundance[,
                names(lrf_abundance) %in% c("grid_id",
                  "ScientificName", "ScientificName")]),
                by = "grid_id") %>%
            mutate(lrf_value = ifelse(is.na(average),
                NA, 5 * (average/max(average,
                  na.rm = TRUE))))
        # add lrf value to final dataframe
        QESSa <- left_join(QESSa, df_lrf_value[,
            c(1, 4)], by = "grid_id")
    }

    # average lrf_value results of all the
    # species for each subzone into final
    # metrics
    QESSb <- QESSa %>%
        mutate(average_lrf_value = rowMeans(.[-1],
            na.rm = TRUE)) %>%
        mutate_all(~ifelse(is.na(.), 0, .)) %>%
        select(grid_id, average_lrf_value)

    # Classification into five categories
    QESSc <- QClassify(QESSb, QESSb$average_lrf_value,
        "QESSCLASS")

    QESS <- QESSc[c("grid_id", "QESSCLASS")]

    # Produce map of the results
    produce_map_sf(QESS, "QESSCLASS", "AQ11",
        continous)
} else {
    QESS <- PolIDs
    QESS$QESSCLASS <- NA
}
```

7.13 AQ 12: Is the presence of habitat-forming species (or species formed habitats i.e. biogenic habitats) high in the subzone?

```
if (length(HFKS) > 0) {
    HFKS_list <- HFKS

    # Test if the HFK species is present in
    # the data
    for (j in HFKS) {
        info_presence <- assessment_data[which(assessment_data$ScientificName %in%
            j), ]
        if (dim(info_presence)[1] == 0) {
            HFKS_list <- HFKS_list[!HFKS_list ==
                j]
        }
    }

    qual_QHFKS1 <- data.frame()
    # Loop through the whole date grid to
    # check for the presence of HFK species
    # A value 5 is given if the HFK species
    # is detected in the grid, if not, the
    # value 0 is given
    for (i in 1:NpolID) {
        grid_evaluated <- assessment_data[assessment_data$grid_id %in%
            PolIDs$grid_id[i], ]
        grid_vector <- PolIDs$grid_id[i]
        for (k in HFKS_list) {
            if (k %in% grid_evaluated$ScientificName) {
                grid_vector <- append(grid_vector,
                  5)
            } else {
                grid_vector <- append(grid_vector,
                  0)
            }
        }
        qual_QHFKS1 <- rbind(qual_QHFKS1, grid_vector)
    }
    # Calculation of the average value for
    # each subzones
    qual_QHFKS2 <- qual_QHFKS1 %>%
        mutate(average_value = rowMeans(.[-1])) %>%
        rename(grid_id = 1) %>%
        select(grid_id, average_value)

    # Classification into five categories
    qual_QHFKS3 <- QClassify(qual_QHFKS2, qual_QHFKS2$average_value,
        "qual_QHFKSCLASS")

    qual_QHFKS <- qual_QHFKS3[c("grid_id", "qual_QHFKSCLASS")]

    # Produce map of the results
    produce_map_sf(qual_QHFKS, "qual_QHFKSCLASS",
        "AQ12", continous)
} else {
    qual_QHFKS <- PolIDs
    qual_QHFKS$qual_QHFKSCLASS <- NA
}
```

7.14 AQ 13: Is the abundance of habitat-forming species (or species formed habitats i.e. biogenic habitats) high in the subzone?

```
if (length(HFKS) > 0) {

    QHFKS <- list(1)  #just creating a list object
    HFKS_list <- HFKS

    for (j in HFKS) {
        info_presence <- Quantitative_data[which(Quantitative_data$ScientificName %in%
            j), ]
        if (dim(info_presence)[1] == 0) {
            HFKS_list <- HFKS_list[!HFKS_list ==
                j]
        }
    }

    QHFKSa <- data.frame(grid_id = PolIDs)

    for (k in HFKS_list) {
        # get data for that species
        lrf_abundance <- Quantitative_data[Quantitative_data$ScientificName %in%
            k, ] %>%
            st_drop_geometry()
        # get average abundance for each
        # subzones for that species
        df_lrf_abundance_average <- lrf_abundance %>%
            select(grid_id, ScientificName, Density) %>%
            group_by(grid_id) %>%
            summarise_at(vars(Density), list(average = mean),
                .groups = "drop")
        # get the lrf value for each
        # subzones
        df_lrf_value <- df_lrf_abundance_average %>%
            inner_join(unique(lrf_abundance[,
                names(lrf_abundance) %in% c("grid_id",
                  "ScientificName", "ScientificName")]),
                by = "grid_id") %>%
            mutate(lrf_value = ifelse(is.na(average),
                NA, 5 * (average/max(average,
                  na.rm = TRUE))))
        # add lrf value to final dataframe
        QHFKSa <- left_join(QHFKSa, df_lrf_value[,
            c(1, 4)], by = "grid_id")
    }

    # average lrf_value results of all the
    # species for each subzone into final
    # metrics
    QHFKSb <- QHFKSa %>%
        mutate(average_lrf_value = rowMeans(.[-1],
            na.rm = TRUE)) %>%
        mutate_all(~ifelse(is.na(.), 0, .)) %>%
        select(grid_id, average_lrf_value)

    # Classification into five categories
    QHFKSc <- QClassify(QHFKSb, QHFKSb$average_lrf_value,
        "QHFKSCLASS")

    QHFKS <- QHFKSc[c("grid_id", "QHFKSCLASS")]

    # Produce map of the results
    produce_map_sf(QHFKS, "QHFKSCLASS", "AQ13",
        continous)
} else {
    QHFKS <- PolIDs
    QHFKS$QHFKSCLASS <- NA
}
```

7.15 AQ 14: Is the presence of mutualist or symbiotic species high in the subzone?

```
if (length(MSS) > 0) {
    MSS_list <- MSS

    # Test if the HFK species is found in
    # the data
    for (j in MSS) {
        info_presence <- assessment_data[which(assessment_data$ScientificName %in%
            j), ]
        if (dim(info_presence)[1] == 0) {
            MSS_list <- MSS_list[!MSS_list ==
                j]
        }
    }

    qual_QMSS1 <- data.frame()
    # Loop through the whole data grid to
    # check for the presence of HFK species
    # A value 5 is given if the HFK species
    # is detected in the grid, if not, the
    # value 0 is given
    for (i in 1:NpolID) {
        grid_evaluated <- assessment_data[assessment_data$grid_id %in%
            PolIDs$grid_id[i], ]
        grid_vector <- PolIDs$grid_id[i]
        for (k in MSS_list) {
            if (k %in% grid_evaluated$ScientificName) {
                grid_vector <- append(grid_vector,
                  5)
            } else {
                grid_vector <- append(grid_vector,
                  0)
            }
        }
        qual_QMSS1 <- rbind(qual_QMSS1, grid_vector)
    }

    # Calculation of the average value for
    # each subzones
    qual_QMSS2 <- qual_QMSS1 %>%
        mutate(average_value = rowMeans(.[-1])) %>%
        rename(grid_id = 1) %>%
        select(grid_id, average_value)

    # Classification into five categories
    qual_QMSS3 <- QClassify(qual_QMSS2, qual_QMSS2$average_value,
        "qual_QMSSCLASS")

    qual_QMSS <- qual_QMSS3[c("grid_id", "qual_QMSSCLASS")]

    # Produce map of the results
    produce_map_sf(qual_QMSS, "qual_QMSSCLASS",
        "AQ14", continous)
} else {
    qual_QMSS <- PolIDs
    qual_QMSS$qual_QMSSCLASS <- NA
}
```

7.16 AQ 15: Is the abundance of symbiotic species high in the subzone?

```
if (length(MSS) > 0) {
    QMSS <- list(1)  #just creating a list object
    MSS_list <- MSS

    for (j in MSS) {
        info_presence <- Quantitative_data[which(Quantitative_data$ScientificName %in%
            j), ]
        if (dim(info_presence)[1] == 0) {
            MSS_list <- MSS_list[!MSS_list ==
                j]
        }
    }

    QMSSa <- data.frame(grid_id = PolIDs)

    for (k in MSS_list) {
        # get data for that species
        lrf_abundance <- Quantitative_data[Quantitative_data$ScientificName %in%
            k, ] %>%
            st_drop_geometry()
        # get average abundance for each
        # subzones for that species
        df_lrf_abundance_average <- lrf_abundance %>%
            select(grid_id, ScientificName, Density) %>%
            group_by(grid_id) %>%
            summarise_at(vars(Density), list(average = mean),
                .groups = "drop")
        # get the lrf value for each
        # subzones
        df_lrf_value <- df_lrf_abundance_average %>%
            inner_join(unique(lrf_abundance[,
                names(lrf_abundance) %in% c("grid_id",
                  "ScientificName", "ScientificName")]),
                by = "grid_id") %>%
            mutate(lrf_value = ifelse(is.na(average),
                NA, 5 * (average/max(average,
                  na.rm = TRUE))))
        # add lrf value to final dataframe
        QMSSa <- left_join(QMSSa, df_lrf_value[,
            c(1, 4)], by = "grid_id")
    }

    # Average lrf value results of all the
    # species for each subzone into final
    # metrics
    QMSSb <- QMSSa %>%
        mutate(average_lrf_value = rowMeans(.[-1],
            na.rm = TRUE)) %>%
        mutate_all(~ifelse(is.na(.), 0, .)) %>%
        select(grid_id, average_lrf_value)

    # Classification into five categories
    QMSSc <- QClassify(QMSSb, QMSSb$average_lrf_value,
        "QMSSCLASS")

    QMSS <- QMSSc[c("grid_id", "QMSSCLASS")]

    # Produce map of the results
```

```r
    produce_map_sf(QMSS, "QMSSCLASS", "AQ15",
        continous)
} else {
    QMSS <- PolIDs
    QMSS$QMSSCLASS <- NA
}
```

8. Final data frame with all AQ values

The final dataframe consists out of the following two columns:

Assessment Question	Variable name and results
AQ1	QPRS$QPRSCLASS
AQ2	QHARS$QHARSCLASS
AQ3	QRRS$QRRSCLASS
AQ4	QARRS$QARRSCLASS
AQ5	QNRS$QNRSCLASS
AQ6	QANRS$QANRSCLASS
AQ7	QNFH$QNFHCLASS
AQ8	QHCMS$QHCMSCLASS
AQ9	QHACS$QHACSCLASS
AQ10	qual_QESS$qual_QESSCLASS
AQ11	QESS$QESSCLASS
AQ12	qual_QHFKS$qual_QHFKSCLASS
AQ13	QHFKS$QHFKSCLASS
AQ14	qual_QMSS$qual_QMSSCLASS
AQ15	QMSS$QMSSCLASS

```r
# Final dataframe with all the results
EcoCompQ_base <- data.frame(QPRS$QPRSCLASS, QHARS$QHARSCLASS,
    QRRS$QRRSCLASS, QARRS$QARRSCLASS, QNRS$QNRSCLASS,
    QANRS$QANRSCLASS, QNFH$QNFHCLASS, QHCMS$QHCMSCLASS,
    QHACS$QHACSCLASS, qual_QESS$qual_QESSCLASS,
    QESS$QESSCLASS, qual_QHFKS$qual_QHFKSCLASS,
    QHFKS$QHFKSCLASS, qual_QMSS$qual_QMSSCLASS,
    QMSS$QMSSCLASS)

names(EcoCompQ_base) <- c("AQ1", "AQ2", "AQ3",
    "AQ4", "AQ5", "AQ6", "AQ7", "AQ8", "AQ9",
    "AQ10", "AQ11", "AQ12", "AQ13", "AQ14", "AQ15")
```

9. Confidence assessment

This score is calculated based on the equation present in the guideline. It needs 3 parameters: AQi, N and wi.

AQi: identifies the ith Assessment Question, which in the equation is assigned a value of 1 if it has been answered or 0 if not answered.

N: total number of questions defined for each EC (e.g., N=15 for macrobenthos, N=13 for marine benthic habitats)

wi: weight assigned to AQ, the value is between 1 and 5 and depends on the type of data (modelling/monitoring data or expert judgment) and the data availability.

```r
# Get N
if (component_habitat == TRUE) {
    N <- 13
} else {
    N <- 15
}

# Get AQi for each subzones
AQi_results <- EcoCompQ_base %>%
    mutate(across(.cols = everything(), ~ifelse(is.na(.x),
        0, 1)))

# Get wi for each subzones Count number of
# samples for monitoring data
DAV <- assessment_data %>%
    group_by(grid_id) %>%
    summarise(nb_sample = length(unique(FieldNumber))) %>%
    st_drop_geometry()
if (modelling_data == TRUE) {
    DAV <- assessment_data %>%
        group_by(grid_id) %>%
        summarise(area_covered = sum(Density,
            na.rm = TRUE)) %>%
        st_drop_geometry()
}


# If the number of subzones covered by the
# data is less than the number of subzones
# present in the bbt, than the minimum of
# the range will always be 0
if (monitoring_data == TRUE) {
    if (NpolID < length(GRID_degree_4326$grid_id)) {
        range_data <- max(DAV$nb_sample) - 0
        min_data <- 0
    } else if (NpolID == length(GRID_degree_4326$grid_id)) {
        range_data <- max(DAV$nb_sample) - min(DAV$nb_sample)
        min_data <- min(DAV$nb_sample)
    }

    X_data <- range_data/5
    max_data <- max(DAV$nb_sample)
}


# change nb_sample into categories (low,
# Medium-Low, Medium, Medium-High, High),
# depending on the data used
if (expert_judgement == TRUE) {
    DAV2 <- DAV
} else if (modelling_data == TRUE) {
    DAV2 <- DAV %>%
        mutate(categories_dav = ifelse(area_covered >=
            0 & area_covered <= 0.2, "Low", ifelse(area_covered >
            0.2 & area_covered <= 0.4, "Medium-Low",
            ifelse(area_covered > 0.4 & area_covered <=
                0.6, "Medium", ifelse(area_covered >
                0.6 & area_covered <= 0.8, "Medium-High",
                ifelse(area_covered > 0.8 & area_covered <=
                  1, "High", NA)))))) %>%
        select(grid_id, categories_dav)
} else if (monitoring_data == TRUE) {
    DAV2 <- DAV %>%
        mutate(categories_dav = ifelse(nb_sample >=
            min_data & nb_sample <= (min_data +
            X_data), "Low", ifelse(nb_sample >
            (min_data + X_data) & nb_sample <=
            (min_data + (2 * X_data)), "Medium-Low",
            ifelse(nb_sample > (min_data + (2 *
                X_data)) & nb_sample <= (min_data +
                (3 * X_data)), "Medium", ifelse(nb_sample >
                (min_data + (3 * X_data)) & nb_sample <=
                (min_data + (4 * X_data)), "Medium-High",
                ifelse(nb_sample > (min_data +
                  (4 * X_data)) & nb_sample <=
                  max_data, "High", NA)))))) %>%
        select(grid_id, categories_dav)
}

# Mutate those categories into wi, based on
# the type of data used

if (expert_judgement == TRUE) {
    wi_results <- DAV2 %>%
        mutate(wi = 1) %>%
        select(grid_id, wi)
} else if (modelling_data == TRUE) {
    wi_results <- DAV2 %>%
        mutate(wi = ifelse(categories_dav == "Low",
            1, ifelse(categories_dav == "Medium-Low",
                1, ifelse(categories_dav == "Medium",
                  2, ifelse(categories_dav ==
                    "Medium-High", 3, ifelse(categories_dav ==
                    "High", 3, NA)))))) %>%
        select(grid_id, wi)
} else if (monitoring_data == TRUE) {
    wi_results <- DAV2 %>%
        mutate(wi = ifelse(categories_dav == "Low",
            1, ifelse(categories_dav == "Medium-Low",
                2, ifelse(categories_dav == "Medium",
                  3, ifelse(categories_dav ==
                    "Medium-High", 4, ifelse(categories_dav ==
                    "High", 5, NA)))))) %>%
        select(grid_id, wi)
```

```r
}

EcoComp_DAV1 <- data.frame(cbind(wi_results, AQi_results))

# Calculate confidence scores
EcoComp_DAV2 <- EcoComp_DAV1 %>%
    mutate(across(starts_with("AQ"), ~.x * wi))

EcoComp_DAV2$conf_score <- rowSums(EcoComp_DAV2[,
    3:length(colnames(EcoComp_DAV2))], na.rm = TRUE)/N
```

Plot results obtained for the confidence score

```r
EcoComp_DAV_map <- GRID_degree_4326 %>%
  left_join(EcoComp_DAV2)

if (continous == TRUE){
  print(ggplot()+
   geom_sf(data = EcoComp_DAV_map$geometry,lwd = 0) +
   geom_sf(data = EcoComp_DAV_map, aes(fill=conf_score), colour = "grey90", lwd = 0) +
   scale_fill_viridis_c(name = "Confidence Score", na.value = "grey94", limits = c(0,5)) +
   geom_sf(data = world_data$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, paste0(OutputFileName,"_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
  EcoComp_DAV_map$category_graph <- cut(
      EcoComp_DAV_map$conf_score,
      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 = EcoComp_DAV_map$geometry,lwd = 0) +
   geom_blank() +
   geom_sf(data = EcoComp_DAV_map, aes(fill=category_graph), colour = "grey90", lwd = 0,
           show.legend = TRUE) +
     # show legend allow to have colors in legend even for categories without data points
   scale_fill_manual(values = mycolors, name = "Confidence Score", limits = categories,
                     na.value = "grey94", labels = categories, drop = FALSE) +
   geom_sf(data = world_data$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, paste0(OutputFileName,"_confidence_score.png")),
    width = NA,
    height = NA,
    units = "cm",
    dpi=1600)
}
```

10. Calculate the final score and add the calculated data to the grid

```
# Remove the column of questions where only
# NA are present
EcoCompQ <- EcoCompQ_base %>%
    select(where(function(x) any(!is.na(x))))


LenQ <- apply(EcoCompQ, 1, function(t) length(t) -
    length(t[t[] == "NA"]))
SumQ <- apply(EcoCompQ, 1, sum, na.rm = TRUE)
EcoCompQ$SumQ <- SumQ
EcoCompQ$LenQ <- LenQ
if (modelling_data == TRUE) {
    DAV$nb_sample <- 0
}

EcoCompQ$nb_sample <- DAV$nb_sample


EcoCompQ$Total <- SumQ/LenQ
# Add grid ID to the dataframe
EcoCompQ <- data.frame(cbind(PolIDs, EcoCompQ,
    EcoComp_DAV2$conf_score))

# Combine the final score with the grid
# geometry.
result.new <- GRID_degree_4326 %>%
    left_join(EcoCompQ)

names(result.new)[names(result.new) == "EcoComp_DAV2.conf_score"] <- "conf_score"


Result_final <- result.new

# Export the results to a shapefile
st_write(Result_final, file.path(Output, paste0(OutputFileName,
    "_layers.shp")), append = FALSE)
```

Map of the total score

```r
if (continous == TRUE){
  print(ggplot()+
   geom_sf(data = Result_final$geometry,lwd = 0) +
   geom_sf(data = Result_final, aes(fill=Result_final$Total), colour = "grey90", lwd = 0) +
   scale_fill_viridis_c(name = "Total", na.value = "grey94", limits = c(0,5)) +
   geom_sf(data = world_data$geometry) +
   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
         panel.background = element_blank(),
         axis.text.x=element_text(angle=300,hjust=0)))
ggsave(filename=file.path(Output, paste0(OutputFileName,"_totalscore.png")),
    width = NA,
    height = NA,
    units = "cm",
    dpi=1600)
} else{
  # Define the categories and colors
  categories <- c("0 - 1", "1 - 2", "2 - 3", "3 - 4", "4 - 5")
  mycolors <- viridisLite::viridis(5)
  # Create categories with explicit levels
  Result_final$category_graph <- cut(
      Result_final$Total,
      breaks = c(0, 1, 2, 3, 4, 5),
      include.lowest = TRUE,
      labels = categories,
      # Force creation of all levels
      right = FALSE,
      # Ensure that all levels are present
      levels = categories
  )
  print(ggplot()+
   geom_sf(data = Result_final$geometry,lwd = 0) +
   geom_blank() +
   geom_sf(data = Result_final, 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 = "Total", 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, paste0(OutputFileName,"_totalscore.png")),
    width = NA,
    height = NA,
    units = "cm",
    dpi=1600)
}
```