SEACAR Coral Analysis: Species Richness

Last compiled on 11 May, 2023

Table of Contents

# Important Notes

These scripts were created by [J.E. Panzik](mailto:jepanzik@usf.edu) ([jepanzik@usf.edu](mailto:jepanzik@usf.edu)) for SEACAR.

All scripts and outputs can be found on the SEACAR GitHub repository:

<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses>

This markdown file is designed to be compiled by [SEACAR\_CoastalWetlands\_SpeciesRichness\_ReportRender.R](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Coral/SEACAR_Coral_SpeciesRichness_ReportRender.R) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Coral/SEACAR_Coral_SpeciesRichness_ReportRender.R>).

# Libraries and Settings

Loads libraries used in the script. The inclusion of scipen option limits how frequently R defaults to scientific notation. Sets default settings for displaying warning and messages in created document, and sets figure dpi.

library(knitr)  
library(data.table)  
library(dplyr)  
library(lubridate)  
library(ggplot2)  
library(scales)  
library(tidyr)  
library(gridExtra)  
#library(tidyverse)  
library(ggpubr)  
library(scales)  
options(scipen=999)  
opts\_chunk$set(warning=FALSE, message=FALSE, dpi=200)

# File Import

Imports file that is determined in the SEACAR\_Coral\_SpeciesRichness\_ReportRender.R script.

The command fread is used because of its improved speed while handling large data files. Only columns that are used by the script are imported from the file, and are designated in the select input.

The script then gets the name of the parameter as it appears in the data file and units of the parameter.

The latest version of Coral data is available at: <https://usf.box.com/s/8hyj2ur5arothlifg1isnq2gxisjzbdg>

The file being used for the analysis is: **SpeciesRichness - DRY TORT-2021-Jul-26.csv, SpeciesRichness - FLA KEYS-2021-Jul-26.csv, SpeciesRichness - SE FL-2021-Jul-26.csv**

for(i in 1:length(file\_list)){  
 if(i==1){  
 data <- fread(file\_list[i], sep=",", header=TRUE, stringsAsFactors=FALSE,  
 na.strings="")  
 data$`[PercentCover-SpeciesComposition\_%]`<- as.numeric(data$`[PercentCover-SpeciesComposition\_%]`)  
   
 } else{  
 temp <- fread(file\_list[i], sep=",", header=TRUE,  
 stringsAsFactors=FALSE, na.strings="")  
 temp$`[PercentCover-SpeciesComposition\_%]`<- as.numeric(temp$`[PercentCover-SpeciesComposition\_%]`)  
 data <- bind\_rows(data, temp)  
 rm(temp)  
 }  
}  
  
cat(paste("The data file(s) used:", file\_short, sep="\n"))

## The data file(s) used:  
## SpeciesRichness - DRY TORT-2021-Jul-26.csv The data file(s) used:  
## SpeciesRichness - FLA KEYS-2021-Jul-26.csv The data file(s) used:  
## SpeciesRichness - SE FL-2021-Jul-26.csv

# Data Filtering

The processing and filtering that is done to the data is as follows:

1. Set parameter names to Species Richness
2. [PercentCover-SpeciesComposition\_%] column is renamed perccov
3. Changes ManagedArea column name to ManagedAreaName for consistency
4. Sets units
5. Replaces any NA values that were imported as blank character strings or "NA" character strings with the proper NA value
6. Removes rows that contains NA values in ManagedAreaName, GenusName, SpeciesName, Month, Year, SpeciesGroup1, and removes invasive species data
7. Sets perccov to be numeric values and removes rows where percent cover is 0
8. Removes duplicates (MADup==1)
9. Combines genus and species names
10. Corrects some managed area names to match what is being used with other habitats
11. Creates species richness dataset

* Grouped based on common ManagedAreaName, ProgramID, ProgramName, ProgramLocationID, and SampleDate
* SpeciesRichness determined based on the number of unique species (gensp) in each group

1. Merges data with managed area data to determine correct AreaID
2. Writes to file with “\_UsedData” file name to indicate what data was used for species richness.

# Create ParameterName Column  
data$ParameterName <- "Species Richness"  
parameter <- "Species Richness"  
  
# Changes column name to perccov for ease moving forward  
colnames(data)[colnames(data) == "[PercentCover-SpeciesComposition\_%]"] <-  
 "perccov"  
  
# Changes "ManagedArea" column name to "ManagedAreaName" for consistency  
colnames(data)[colnames(data) == "ManagedArea"] <- "ManagedAreaName"  
  
# Sets units for species richness  
unit <- "# of species"  
data$ParameterUnits <- unit  
  
# Replace instances where NA values imported as blank character string or as "NA"  
data <- replace(data, data=="", NA)  
data <- replace(data, data=="NA", NA)  
  
# Remove rows with missing ManagedAreaName  
data <- data[!is.na(data$ManagedAreaName),]  
data <- data[data$ManagedAreaName!="NA",]  
# Remove rows with missing GenusName  
data <- data[!is.na(data$GenusName),]  
# Remove rows with missing SpeciesName  
data <- data[!is.na(data$SpeciesName),]  
# Remove rows with missing Months  
data <- data[!is.na(data$Month),]  
# Remove rows with missing Years  
data <- data[!is.na(data$Year),]  
# Remove rows with missing SpeciesGroup1  
data <- data[!is.na(data$SpeciesGroup1),]  
# Remove rows with invasive species  
data <- data[data$SpeciesGroup1!="Invasive",]  
# Set perccov to be a number value  
data$perccov <- as.numeric(data$perccov)  
# Remove rows where perccov is 0 and missing  
data <- data[data$perccov!=0,]  
data <- data[!is.na(data$perccov),]  
# Remove duplicate rows  
data <- data[data$MADup==1,]  
# Create variable that combines the genus and species name  
data$gensp <- paste(data$GenusName, data$SpeciesName, sep=" ")  
# Corrects Managed Area names to be consistent with official names  
data$ManagedAreaName[data$ManagedAreaName=="Florida Keys NMS"] <-  
 "Florida Keys National Marine Sanctuary"  
data$ManagedAreaName[data$ManagedAreaName==  
 "Biscayne Bay-Cape Florida to Monroe County Line"] <-  
 "Biscayne Bay-Cape Florida to Monroe County Line Aquatic Preserve"  
data$ManagedAreaName[data$ManagedAreaName=="Coupon Bight"] <-  
 "Coupon Bight Aquatic Preserve"  
data$ManagedAreaName[data$ManagedAreaName=="Coral ECA"] <-  
 "Southeast Florida Coral Reef Ecosystem Conservation Area"  
  
# Remove timestamp and makes sure SampleDate is being stored as a Date object  
data$SampleDate <- sub(" 12:00AM", "", data$SampleDate)  
data$SampleDate <- as.Date(data$SampleDate, format="%b %d %Y")  
  
# Create Species Richness values for groups of unique combinations of  
# ManagedAreaName, ProgramID, ProgramName, ProgramLocationID, and SampleDate.  
data <- data %>%  
 group\_by(ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,  
 SampleDate) %>%  
 summarise(ParameterName=parameter,  
 Year=unique(Year), Month=unique(Month),  
 SpeciesRichness=length(unique(gensp)))  
  
# Adds AreaID for each managed area by combining the MA\_All datatable to the  
# data based on ManagedAreaName  
data <- merge.data.frame(MA\_All[,c("AreaID", "ManagedAreaName")],  
 data, by="ManagedAreaName", all=TRUE)  
  
# Writes this data that is used by the rest of the script to a text file  
fwrite(data, paste0(out\_dir,"/CoastalWetlands\_", param\_file, "\_UsedData.txt"),  
 sep="|")  
  
# Makes sure SampleDate is being stored as a Date object  
data$SampleDate <- as.Date(data$SampleDate)  
  
# Creates a variable with the names of all the managed areas that contain  
# species observations  
MA\_Include <- unique(data$ManagedAreaName[!is.na(data$SpeciesRichness)])  
  
# Puts the managed areas in alphabetical order  
MA\_Include <- MA\_Include[order(MA\_Include)]  
  
# Determines the number of managed areas used  
n <- length(MA\_Include)

# Managed Area Statistics

Gets summary statistics for each managed area. Uses piping from dplyr package to feed into subsequent steps. The following steps are performed:

1. Group data that have the same ManagedAreaName, Year, and Month.
   * Second summary statistics do not use the Month grouping and are only for ManagedAreaName and Year.
   * Third summary statistics do not use Year grouping and are only for ManagedAreaName and Month
   * Fourth summary statistics are only grouped based on ManagedAreaName
     + Determines the years that the minimum and maximum species richness occurred
2. For each group, provide the following information: Parameter Name (ParameterName), Number of Entries (N\_Data), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, and a list of all Programs included in these measurements.
3. Sort the data in ascending (A to Z and 0 to 9) order based on ManagedAreaName then Year then Month
4. Write summary stats to a pipe-delimited .txt file in the output directory
   * [Coral Species Richness Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/SpeciesRichness) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/SpeciesRichness>)

# Create summary statistics for each managed area based on Year and Month  
# intervals.  
MA\_YM\_Stats <- data %>%  
 group\_by(AreaID, ManagedAreaName, Year, Month) %>%  
 summarize(ParameterName=parameter,  
 N\_Data=length(na.omit(SpeciesRichness)),  
 Min=min(SpeciesRichness),  
 Max=max(SpeciesRichness),  
 Median=median(SpeciesRichness),  
 Mean=mean(SpeciesRichness),  
 StandardDeviation=sd(SpeciesRichness),  
 Programs=paste(sort(unique(ProgramName), decreasing=FALSE),  
 collapse=', '),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
# Puts the data in order based on ManagedAreaName, Year, then Month  
MA\_YM\_Stats <- as.data.table(MA\_YM\_Stats[order(MA\_YM\_Stats$ManagedAreaName,  
 MA\_YM\_Stats$Year,  
 MA\_YM\_Stats$Month), ])  
# Writes summary statistics to file  
fwrite(MA\_YM\_Stats, paste0(out\_dir,"/Coral\_", param\_file,  
 "\_MA\_MMYY\_Stats.txt"), sep="|")  
# Removes variable storing data to improve computer memory  
rm(MA\_YM\_Stats)  
  
# Create summary statistics for each managed area based on Year intervals  
MA\_Y\_Stats <- data %>%  
 group\_by(AreaID, ManagedAreaName, Year) %>%  
 summarize(ParameterName=parameter,  
 N\_Data=length(na.omit(SpeciesRichness)),  
 Min=min(SpeciesRichness),  
 Max=max(SpeciesRichness),  
 Median=median(SpeciesRichness),  
 Mean=mean(SpeciesRichness),  
 StandardDeviation=sd(SpeciesRichness),  
 Programs=paste(sort(unique(ProgramName), decreasing=FALSE),  
 collapse=', '),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
# Puts the data in order based on ManagedAreaName then Year  
MA\_Y\_Stats <- as.data.table(MA\_Y\_Stats[order(MA\_Y\_Stats$ManagedAreaName,  
 MA\_Y\_Stats$Year), ])  
# Writes summary statistics to file  
fwrite(MA\_Y\_Stats, paste0(out\_dir,"/Coral\_", param\_file,  
 "\_MA\_Yr\_Stats.txt"), sep="|")  
  
# Create summary statistics for each managed area based on Month intervals.  
MA\_M\_Stats <- data %>%  
 group\_by(AreaID, ManagedAreaName, Month) %>%  
 summarize(ParameterName=parameter,  
 N\_Data=length(na.omit(SpeciesRichness)),  
 Min=min(SpeciesRichness),  
 Max=max(SpeciesRichness),  
 Median=median(SpeciesRichness),  
 Mean=mean(SpeciesRichness),  
 StandardDeviation=sd(SpeciesRichness),  
 Programs=paste(sort(unique(ProgramName), decreasing=FALSE),  
 collapse=', '),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
# Puts the data in order based on ManagedAreaName then Month  
MA\_M\_Stats <- as.data.table(MA\_M\_Stats[order(MA\_M\_Stats$ManagedAreaName,  
 MA\_M\_Stats$Month), ])  
# Writes summary statistics to file  
fwrite(MA\_M\_Stats, paste0(out\_dir,"/Coral\_", param\_file,  
 "\_MA\_Mo\_Stats.txt"), sep="|")  
# Removes variable storing data to improve computer memory  
rm(MA\_M\_Stats)  
  
# Create summary overall statistics for each managed area.  
MA\_Ov\_Stats <- data %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 summarize(ParameterName=parameter,  
 N\_Years=length(unique(na.omit(Year))),  
 EarliestYear=min(Year),  
 LatestYear=max(Year),  
 N\_Data=length(na.omit(SpeciesRichness)),  
 Min=min(SpeciesRichness),  
 Max=max(SpeciesRichness),  
 Median=median(SpeciesRichness),  
 Mean=mean(SpeciesRichness),  
 StandardDeviation=sd(SpeciesRichness),  
 Programs=paste(sort(unique(ProgramName), decreasing=FALSE),  
 collapse=', '),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
# Puts the data in order based on ManagedAreaName  
MA\_Ov\_Stats <- as.data.table(MA\_Ov\_Stats[order(MA\_Ov\_Stats$ManagedAreaName), ])  
# Creates Year\_MinRichness and Year\_MaxRichness columns  
MA\_Ov\_Stats$Year\_MinRichness <- NA  
MA\_Ov\_Stats$Year\_MaxRichness <- NA  
  
# Loops through each ManagedAreaName.  
# Determines what year the minimum and maximum species richness occurred  
for(m in 1:nrow(MA\_Ov\_Stats)){  
 # Stores ManagedAreaName for this row  
 ma <- MA\_Ov\_Stats$ManagedAreaName[m]  
   
 # Skips to next row if there are no data for this combination  
 if(MA\_Ov\_Stats$N\_Data[m]==0){  
 next  
 }  
 # Gets subset of data from MA\_Y\_Stats (yearly summary stats) with this  
 # ManagedAreaName  
 ds <- MA\_Y\_Stats[MA\_Y\_Stats$ManagedAreaName==ma,]  
 # Gets the minimum and maximum Mean (yearly averages)  
 min <- min(ds$Mean)  
 max <- max(ds$Mean)  
 #Determines what years those minimum and maximum values occured  
 year\_min <- ds$Year[ds$Mean==min]  
 year\_max <- ds$Year[ds$Mean==max]  
 # Stores the occurrence years of the minimum and maximum into the overall  
 # stats for this row  
 MA\_Ov\_Stats$Year\_MinRichness[m] <- year\_min  
 MA\_Ov\_Stats$Year\_MaxRichness[m] <- year\_max  
}  
# Replaces blank ProgramIDs with NA (missing values)  
MA\_Ov\_Stats$ProgramIDs <- replace(MA\_Ov\_Stats$ProgramIDs,  
 MA\_Ov\_Stats$ProgramIDs=="", NA)  
MA\_Ov\_Stats$Programs <- replace(MA\_Ov\_Stats$Programs,  
 MA\_Ov\_Stats$Programs=="", NA)  
# Write overall statistics to file  
fwrite(MA\_Ov\_Stats, paste0(out\_dir,"/Coral\_", param\_file,  
 "\_MA\_Overall\_Stats.txt"), sep="|")  
# Removes entries from the overall statistics that do not have data.  
# Based on presence or absence of EarliestYear  
MA\_Ov\_Stats <- MA\_Ov\_Stats[!is.na(MA\_Ov\_Stats$EarliestYear), ]

# Appendix I: Managed Area Species Richness

The plots shown here are the species richness for each managed area with a yearly average.

1. Set common plot theme.
2. Determine the earliest and latest year of the data to create x-axis scale and intervals
3. Determine the upper and lower limit of the plot for better y-axis labels
4. Add the plot line
5. Set the plot type as a point plot with the size of the points
6. Create the title, x-axis, y-axis
7. Set the y and x limits
8. Apply common plot theme
9. Add table with summary statistics below each figure
   * Numerical non-integer values are rounded to 2 decimal places
   * StandardDeviation is renamed StDev for space reasons
10. Create file name to save figure
11. Save figure as png file

* [Coral Species Richness Figures in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/SpeciesRichness/Figures) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/SpeciesRichness/Figures>)

# Defines standard plot theme: black and white, no major or minor grid lines,  
# Arial font. Title is centered, size 12, and blue (hex coded). Subtitle is  
# centered, size 10, and blue (hex coded). Legend title is size 10 and the  
# legend is left-justified. X-axis title is size 10 and the margins are padded  
# at the top and bottom to give more space for angled axis labels. Y-axis title  
# is size 10 and margins are padded on the right side to give more space for  
# axis labels. Axis labels are size 10 and the x-axis labels are rotated -45  
# degrees with a horizontal justification that aligns them with the tick mark  
plot\_theme <- theme\_bw() +  
 theme(panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 text=element\_text(family="Arial"),  
 plot.title=element\_text(hjust=0.5, size=12, color="#314963"),  
 plot.subtitle=element\_text(hjust=0.5, size=10, color="#314963"),  
 legend.title=element\_text(size=10),  
 legend.text.align = 0,  
 axis.title.x = element\_text(size=10, margin = margin(t = 5, r = 0,  
 b = 10, l = 0)),  
 axis.title.y = element\_text(size=10, margin = margin(t = 0, r = 10,  
 b = 0, l = 0)),  
 axis.text=element\_text(size=10),  
 axis.text.x=element\_text(angle = -45, hjust = 0))  
  
# Color palette for SEACAR  
color\_palette <- c("#005396", "#0088B1", "#00ADAE", "#65CCB3", "#AEE4C1",  
 "#FDEBA8", "#F8CD6D", "#F5A800", "#F17B00")  
  
  
  
# Loop that cycles through each managed area with data  
if(n==0){  
 # Prints a statement if there are no managed areas with appropriate data  
 print("There are no monitoring locations that qualify.")  
} else {  
 for (i in 1:n) {  
 # Gets data for target managed area  
 plot\_data <- MA\_Y\_Stats[MA\_Y\_Stats$ManagedAreaName==MA\_Include[i]]  
 # Determines most recent year with available data for managed area  
 t\_max <- max(MA\_Ov\_Stats$LatestYear[MA\_Ov\_Stats$ManagedAreaName==  
 MA\_Include[i]])  
 # Determines earliest recent year with available data for managed area  
 t\_min <- min(MA\_Ov\_Stats$EarliestYear[MA\_Ov\_Stats$ManagedAreaName==  
 MA\_Include[i]])  
 # Determines how many years of data are present  
 t <- t\_max-t\_min  
   
 # Creates break intervals for plots based on number of years of data  
 if(t>=30){  
 # Set breaks to every 10 years if more than 30 years of data  
 brk <- -10  
 }else if(t<30 & t>=10){  
 # Set breaks to every 5 years if between 30 and 10 years of data  
 brk <- -5  
 }else if(t<10 & t>=4){  
 # Set breaks to every 2 years if between 10 and 4 years of data  
 brk <- -2  
 }else if(t<4){  
 # Set breaks to every year if less than 4 years of data  
 brk <- -1  
 }  
 # Determine range of data values for the managed area  
 y\_range <- max(plot\_data$Mean) - min(plot\_data$Mean)  
   
 # Determines lower bound of y-axis based on data range. Set based on  
 # relation of data range to minimum value. Designed to set lower boundary  
 # to be 10% of the data range below the minimum value  
 y\_min <- if(min(plot\_data$Mean)-(0.1\*y\_range)<0){  
 # If 10% of the data range below the minimum value is less than 0,  
 # set as 0  
 y\_min <- 0  
 } else {  
 # Otherwise set minimum bound as 10% data range below minimum value  
 y\_min <- min(plot\_data$Mean)-(0.1\*y\_range)  
 }  
   
 # Sets upper bound of y-axis to be 10% of the data range above the  
 # maximum value.  
 y\_max <- max(plot\_data$Mean)+(0.1\*y\_range)  
   
   
 # Creates plot object using plot\_data.  
 # Data is plotted as symbols with connected lines.  
 p1 <- ggplot(data=plot\_data) +  
 geom\_line(aes(x=Year, y=Mean), color=color\_palette[1],  
 size=0.75, alpha=1) +  
 geom\_point(aes(x=Year, y=Mean), fill=color\_palette[1],  
 shape=21, size=2, color="#333333", alpha=1) +  
 labs(title="Coral Species Richness",  
 subtitle=MA\_Include[i],  
 x="Year", y="Richness (# of species)") +  
 scale\_x\_continuous(limits=c(t\_min-0.25, t\_max+0.25),  
 breaks=seq(t\_max, t\_min, brk)) +  
 scale\_y\_continuous(limits=c(y\_min, y\_max),  
 breaks=pretty\_breaks(n=5)) +  
 plot\_theme  
 # Sets file name of plot created  
 outname <- paste0("Coral\_", gsub(" ", "", MA\_Include[i]), "\_",  
 param\_file, ".png")  
 # Saves plot as a png image  
 png(paste0(out\_dir, "/Figures/", outname),  
 width = 8,  
 height = 4,  
 units = "in",  
 res = 200)  
 print(p1)  
 dev.off()  
   
 # Creates a data table object to be shown underneath plots in report  
 ResultTable <-  
 MA\_Ov\_Stats[MA\_Ov\_Stats$ManagedAreaName==MA\_Include[i],]  
 # Removes location, and parameter information because it is in plot  
 # labels  
 ResultTable <- ResultTable[,-c("AreaID", "ManagedAreaName",  
 "ProgramIDs", "Programs", "ParameterName")]  
 # Renames StandardDeviation to StDev to save horizontal space  
 ResultTable <- ResultTable %>%  
 rename("StDev"="StandardDeviation")  
 # Converts all non-integer values to 2 decimal places for space  
 ResultTable$Min <- round(ResultTable$Min, digits=2)  
 ResultTable$Max <- round(ResultTable$Max, digits=2)  
 ResultTable$Median <- round(ResultTable$Median, digits=2)  
 ResultTable$Mean <- round(ResultTable$Mean, digits=2)  
 ResultTable$StDev <- round(ResultTable$StDev, digits=2)  
 # Stores as plot table object  
 t1 <- ggtexttable(ResultTable, rows = NULL,  
 theme=ttheme(base\_size=7))  
 # Combines plot and table into one figure  
 print(ggarrange(p1, t1, ncol=1, heights=c(0.85, 0.15)))  
   
 # Add extra space at the end to prevent the next figure from being too  
 # close  
 cat("\n \n \n \n")  
 }  
}



 