SEACAR Coral Analysis: Percent Cover

Last compiled on 06 June, 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\_Coral\_PercentCover\_ReportRender.R](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Corals/SEACAR_Coral_PercentCover_ReportRender.R) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Coral/SEACAR_Coral_PercentCover_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(hrbrthemes)  
library(nlme)  
library(ggpubr)  
options(scipen=999)  
opts\_chunk$set(warning=FALSE, message=FALSE, dpi=200)

# File Import

Imports file that is determined in the SEACAR\_Coral\_PercentCover\_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(s) being used for the analysis: **All\_CORAL\_Parameters-2023-Jun-06.txt**

data <- fread(file\_in, sep="|", header=TRUE, stringsAsFactors=FALSE,  
 na.strings="")  
  
cat(paste("The data file(s) used:", file\_short, sep="\n"))

## The data file(s) used:  
## All\_CORAL\_Parameters-2023-Jun-06.txt

# Data Filtering

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

1. Only take data rows that are Percent Cover measurements
2. Shorten parameter names to Percent Cover
3. Sets units
4. Removes any data that is not coral

* Only looks for Octocoral, Milleporans, or Scleractinian

1. Removes rows that contains NA values in ManagedAreaName, GenusName, SpeciesName, Month, Year, SpeciesGroup1, ResultValue, and SampleDate
2. Removes duplicates (MADup==1)
3. Combines genus and species names
4. Corrects some managed area names to match what is being used with other habitats
5. Merges data with managed area data to determine correct AreaID

# Only keep data for Percent Cover  
data <- data[data$ParameterName=="Percent Cover - Species Composition"]  
  
# Simplify ParametetrNamee to Percent Cover  
data$ParameterName <- "Percent Cover"  
parameter <- "Percent Cover"  
  
# Sets units for percent cover  
unit <- "%"  
data$ParameterUnits <- unit  
  
# Remove any rows that are not corals  
data <- data[SpeciesGroup1=="Octocoral"|  
 SpeciesGroup1=="Milleporans"|  
 SpeciesGroup1=="Scleractinian", ]  
# 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$SpeciesGroup1),]  
# Remove rows with missing ResultValue  
data <- data[!is.na(data$ResultValue),]  
# Remove rows with missing SampleDate  
data <- data[!is.na(data$SampleDate),]  
# Remove duplicate rows  
data <- data[data$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"  
  
# 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=c("AreaID", "ManagedAreaName"), all=TRUE)

# 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
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 Percent Cover Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/PercentCover) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/PercentCover>)
5. Determines if there is sufficient data to be analyzed based on having more than 5 years of records.
6. Gets list of managed areas to be analyzed.

# 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(ResultValue)),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 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(ResultValue)),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 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(ResultValue)),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 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))),  
 SufficientData=ifelse(N\_Years>=5, TRUE, FALSE),  
 EarliestYear=min(Year),  
 LatestYear=max(Year),  
 N\_Data=length(na.omit(ResultValue)),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 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), ])  
  
# 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="|")  
# Creates a variable with the names of all the managed areas that contain  
# species observations  
MA\_Include <- unique(MA\_Ov\_Stats$ManagedAreaName[!is.na(MA\_Ov\_Stats$Mean)&  
 MA\_Ov\_Stats$SufficientData==  
 TRUE])  
  
# 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)

# Linear Mixed Effects Models

Performs a linear mixed effects (LME) model on each managed area between using a relationship between percent cover and year.

The following steps are performed:

1. Create a blank data frame to store results
2. Sets the column names for the data to be stored from LME model
3. Starts a loop for each managed area included in the analysis
4. Gets data for the current managed area
5. Performs LME on current managed area
6. Stores information and fits into lme\_stats data frame for current managed area
7. Merges lme\_stats with MA\_Ov\_Stats to create a data frame with gegneral statistics and the LME parameters
8. Puts lme\_stats in alphabetical order by managed area name
9. Write lme\_stats to a pipe-delimited .txt file in the output directory
   * [Coral Percent Cover Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/PercentCover) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/PercentCover>)
10. Gets the start and endpoints for LME fit to be used in plots.

# Creates blank data frame with number of rows defined by how many managed areas  
# are going to be analyzed  
lme\_stats <- data.frame(matrix(ncol = 4, nrow = n))  
# Sets column names for blank data frame  
colnames(lme\_stats) <- c("AreaID", "ManagedAreaName", "LME\_Intercept",  
 "LME\_Slope")  
  
# Begins to loop through each managed area for analysis  
for(i in 1:n){  
 # Gets data for current managegd area  
 lme\_data <- data[data$ManagedAreaName==MA\_Include[i],]  
 # Perform LME for relation between ResultValue and Year for current managed area  
 AnyCoral<-lme(ResultValue ~ Year,  
 random =~1|ProgramLocationID,  
 na.action = na.omit,  
 data = lme\_data)  
 # Store information and model fits in appropriate row of data frame  
 lme\_stats$AreaID[i] <- unique(lme\_data$AreaID)  
 lme\_stats$ManagedAreaName[i] <- MA\_Include[i]  
 lme\_stats$LME\_Intercept[i] <- AnyCoral$coefficients$fixed[1]  
 lme\_stats$LME\_Slope[i] <- AnyCoral$coefficients$fixed[2]  
   
 # Clears temporary variables for memory  
 rm(lme\_data)  
 rm(AnyCoral)  
}  
  
# Merges LME stats with overall stats to complete stats for each managed area  
lme\_stats <- merge.data.frame(MA\_Ov\_Stats[,-c("Programs", "ProgramIDs")],  
 lme\_stats, by=c("AreaID", "ManagedAreaName"), all=TRUE)  
  
# Puts the data in order based on ManagedAreaName  
lme\_stats <- as.data.frame(lme\_stats[order(lme\_stats$ManagedAreaName), ])  
  
# Write lme statistics to file  
fwrite(lme\_stats, paste0(out\_dir,"/Coral\_", param\_file,  
 "\_LME\_Stats.txt"), sep="|")  
  
# Gets lower x and y values based on LME fit to use in plot  
lme\_plot <- lme\_stats %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 summarize(x=EarliestYear,  
 y=LME\_Slope\*x+LME\_Intercept)  
# Gets upper x and y values based on LME fit to use in plot  
lme\_plot2 <- lme\_stats %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 summarize(x=LatestYear,  
 y=LME\_Slope\*x+LME\_Intercept)  
# Merges LME fit values for plot into one data frame  
lme\_plot <- bind\_rows(lme\_plot, lme\_plot2)  
rm(lme\_plot2)  
# Puts LME plot data fram in alphabetical order by managed area  
lme\_plot <- as.data.frame(lme\_plot[order(lme\_plot$ManagedAreaName), ])  
lme\_plot <- lme\_plot[!is.na(lme\_plot$y),]

# Appendix I: Plots

The plots shown here are the percent cover for each managed area by year with the LME trendline.

1. Set common plot theme.
2. Starts a loops that creates plots for each managed area analyzed
3. Determine the earliest and latest year of the data to create x-axis scale and intervals
4. Determine the upper and lower limit of the plot for better y-axis labels
5. Set the plot type as a jitter plot with the size of the points to show concentration of data
6. Add LME trendline
7. Create the title, x-axis, y-axis
8. Set the y and x limits
9. Apply common plot theme
10. 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
11. Create file name to save figure
12. Save figure as png file

* [Coral Percent Cover Figures in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/PercentCover/Figures) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Coral/output/PercentCover/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 locations that qualify.")  
} else {  
 for (i in 1:n) {  
 # Gets data for target managed area  
 plot\_data <- data[data$ManagedAreaName==MA\_Include[i],]  
   
 lme\_plot\_data <- lme\_plot[lme\_plot$ManagedAreaName==MA\_Include[i],]  
 # Determines most recent year with available data for managed area  
 t\_max <- max(MA\_Ov\_Stats$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$ResultValue) - min(plot\_data$ResultValue)  
   
 # Sets y\_min to be -1  
 y\_min <- -1  
   
 # Sets upper bound of y-axis to be 10% of the data range above the  
 # maximum value.  
 y\_max <- max(plot\_data$ResultValue)+(0.1\*y\_range)  
   
   
 # Creates plot object using plot\_data.  
 # Data is plotted as symbols with connected lines.  
 p1 <- ggplot(data=plot\_data) +  
 geom\_jitter(aes(x=Year, y=ResultValue), shape=21, size=2,  
 color="#333333", fill="#cccccc", alpha=1) +  
 geom\_line(data=lme\_plot\_data, aes(x=x, y=y),  
 color="#000099", size=2, alpha=0.8) +  
 labs(title="Coral Percent Cover",  
 subtitle=MA\_Include[i],  
 x="Year", y="Percent cover (%)") +  
 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\_", param\_file, "\_", gsub(" ", "", MA\_Include[i]),  
 ".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 <-  
 lme\_stats[lme\_stats$ManagedAreaName==MA\_Include[i],]  
 # Removes location, and parameter information because it is in plot  
 # labels  
 ResultTable <- select(ResultTable, -c("AreaID", "ManagedAreaName",  
 "ParameterName"))  
 # Renames StandardDeviation to StDev to save horizontal space  
 ResultTable <- ResultTable %>%  
 rename("StDev"="StandardDeviation")  
 # Converts all non-integer values to 2 decimal places for space  
 ResultTable$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$StDev, digits=2)  
 ResultTable$LME\_Intercept <- round(ResultTable$LME\_Intercept, digits=2)  
 ResultTable$LME\_Slope <- round(ResultTable$LME\_Slope, 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")  
 }  
}



 