SEACAR Oyster Analysis

Last compiled on 06 March, 2025

Contents

Important Notes

The purpose of this script is to group oyster data, create managed area statistics, perform lme analysis on density, percent live, and shell height, and create reports in pdf and Word document form for Oyster data.

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 Oyster_ReportRender.R (https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Oyster/Oyster_ReportRender.R).

This script is based off of code originally written by Stephen Durham.

Compiled and edited by J.E. Panzik and Tyler Hill for SEACAR.

Libraries and Settings

Loads libraries used in the script. Loads the Segoe UI font for use in the figures. 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(lubridate)
library(tidyverse)
library(data.table)
library(tictoc)
library(glue)
library(kableExtra)
options(scipen=999)
knitr::opts_chunk$set(
   warning=FALSE,
   message=FALSE,
   dpi=200,
   fig.pos = 'H'
)
options(kableExtra.auto_format = FALSE)
options(knitr.kable.NA = '-')
```

File Import

Imports file that is determined in the Oyster_ReportRender.R script.

The command `fread` is used because of its improved speed while handling large data files. Updates column names.

The latest version of Oyster data is available at: https://usf.box.com/s/6dtcrcdkrtc33fqw5kl5fjo6guo0anrw

The file being used for the analysis is: **All_OYSTER_Parameters-2024-Dec-08.txt**

```
oysterraw <- fread(file_in, sep="|", na.strings=c("NULL"))
oysterraw2 <- pivot_wider(oysterraw, names_from="ParameterName",
                          values_from="ResultValue")
```

```
setDT(oysterraw2)
setnames(oysterraw2, c("Density", "Percent Live", "Shell Height",
                       "Number of Oysters Counted - Live",
                       "Number of Oysters Counted - Dead",
                       "Number of Oysters Counted - Total", "Reef Height"),
         c("Density_m2", "PercentLive_pct", "ShellHeight_mm",
           "Number_of_Oysters_Counted_Live_Count",
           "Number_of_Oysters_Counted_Dead_Count",
           "Number_of_Oysters_Counted_Total_Count",
           "ReefHeight_mm"))
oysterraw2[, ObsIndex := seq(1:nrow(oysterraw2))]

oysterraw <- oysterraw2
rm(oysterraw2)
```

Generate Universal Reef IDs

Individual IDs are given to reefs based on location within GIS files. This process now occurs separately from the Oyster habitat scripts and the UniversalReefID will be maintained within the database.

```
# Output from oyster ID script (will make autonomous)
new_crosswalk2 <- readRDS("data/new_crosswalk2.rds")

oysterraw2 <- merge(oysterraw[, -c("UniversalReefID")],
                    new_crosswalk2 %>%
                      select(c("UniversalReefID", "LocationID")),
                    by = "LocationID")
oysterraw <- oysterraw2
rm(oysterraw2)
```

Data Setup & Filtering

Documentation on database filtering is provided here: SEACAR Documentation- Analysis Filters and Calculations.docx

Identifies and removes outliers in the data and various programs.

```
#Make sure column formats are correct-I am still getting an "NAs introduced
#by coercion" warning on the LiveDate calculation,
#but I'm not sure what is going on because when I spot-check the output,
#it does not look like it is introducing NAs...
oysterraw[, `:=` (RowID=as.integer(RowID),
                  ProgramID=as.integer(ProgramID),
                  LocationID=as.integer(LocationID),
                  ProgramName=as.character(ProgramName),
                  ProgramLocationID=as.character(ProgramLocationID),
                  QuadIdentifier=as.character(QuadIdentifier),
                  ReefIdentifier=as.character(ReefIdentifier),
                  UniversalReefID=as.factor(UniversalReefID),
                  LiveDate=as.integer(ifelse(!is.na(LiveDate_Qualifier) &
                                               str_detect(LiveDate,
                                                          "....-..-.."),
                                             paste0(str_sub(LiveDate, 1, 4)),
```

```
                                              round(as.numeric(LiveDate)))),
                  LiveDate_Qualifier=as.character(LiveDate_Qualifier),
                  LiveDate_MinEstDate=as.numeric(LiveDate_MinEstDate),
                  LiveDate_MaxEstDate=as.numeric(LiveDate_MaxEstDate),
                  SampleAge_Stdev=as.numeric(SampleAge_Stdev),
                  #GISUniqueID=as.logical(GISUniqueID),
                  Year=as.integer(Year),
                  Month=as.integer(Month),
                  ManagedAreaName=as.character(ManagedAreaName),
                  SurveyMethod=as.character(SurveyMethod),
                  PercentLiveMethod=as.character(PercentLiveMethod),
                  HabitatClassification=as.character(HabitatClassification),
                  MinimumSizeMeasured_mm=as.character(MinimumSizeMeasured_mm),
                  NumberMeasured_n=as.character(NumberMeasured_n),
                  QuadSize_m2=as.numeric(QuadSize_m2),
                  MADup=as.integer(MADup),
                  Density_m2=as.numeric(Density_m2),
                  PercentLive_pct=as.numeric(PercentLive_pct),
                  ShellHeight_mm=as.numeric(ShellHeight_mm),
                  Number_of_Oysters_Counted_Total_Count =
                    as.integer(Number_of_Oysters_Counted_Total_Count),
                  Number_of_Oysters_Counted_Live_Count =
                    as.integer(Number_of_Oysters_Counted_Live_Count),
                  Number_of_Oysters_Counted_Dead_Count =
                    as.integer(Number_of_Oysters_Counted_Dead_Count),
                  ObsIndex=as.integer(ObsIndex))]

#Calculate Density_m2 values for ProgramID==4016 & 4042
oysterraw[ProgramID==4016, Density_m2 :=
            Number_of_Oysters_Counted_Live_Count/as.numeric(QuadSize_m2)]
oysterraw[ProgramID==4042 & !is.na(Number_of_Oysters_Counted_Live_Count),
          Density_m2 :=
            Number_of_Oysters_Counted_Live_Count/as.numeric(QuadSize_m2)]

#Remove "25" values from total counts column, make all "PercentLiveMethod"
#values the same, and calculate estimated live Density for ProgramID==5074 and
oysterraw <- oysterraw[RowID %in%
                         setdiff(
                           oysterraw[, RowID],
                           oysterraw[ProgramID ==5074 &
                                       Number_of_Oysters_Counted_Total_Count==25, RowID]), ]
oysterraw[ProgramID==5074, PercentLiveMethod := "Estimated percent"]
oysterraw[ProgramID==5074, SampleDate :=
            unique(oysterraw[ProgramID==5074 &
                               !is.na(Number_of_Oysters_Counted_Total_Count),
                             SampleDate])[1]]

#Some PercentLiveMethod values for ID4042 are NA
oysterraw[ProgramID==4042 | ProgramID==4016,
          PercentLiveMethod := "Point-intercept"]

#make sure quadrat identifiers are unique
oysterraw[, QuadIdentifier_old := QuadIdentifier]
```

```
oysterraw[, QuadIdentifier := paste(UniversalReefID,
                                    LocationID, Year, Month,
                                    QuadIdentifier_old, sep="_")]

oysterraw[, MA_plotlab := paste0(ManagedAreaName, "_", HabitatClassification)]
subtidal <- c(4044, 5007, 5071, 5073)
oysterraw[, Subtidal := ifelse(ProgramID %in% subtidal, 1, 0)][, Subtidal := as.logical(Subtidal)]

#Create variables for relative year and size class category for data that
#should be included in analyses and counts of live oysters measured
for(i in unique(oysterraw$ManagedAreaName)){
  oysterraw[ManagedAreaName==i & !is.na(LiveDate), `:=`
            (RelYear=(LiveDate-min(LiveDate))+1,
              YearDiff=min(LiveDate)-1,
              #adding 1 to each RelYear to avoid min(RelYear)==0,
              #because it is used later as an index for plotting years so
              #it needs to start from 1
              SizeClass=fcase(ShellHeight_mm >= 25 &
                                ShellHeight_mm < 75, "25to75mm",
                              ShellHeight_mm >= 75, "o75mm",
                              default=NA))]

  oysterraw[ManagedAreaName==i & !is.na(LiveDate),
            counts := length(ShellHeight_mm), by=c("QuadIdentifier")]
}

#Remove unrealistically high shell heights from ID_5017
oysterraw <- setdiff(oysterraw, oysterraw[ProgramID==5017 & ShellHeight_mm >= 165, ])

#Create data table to save model results
oysterresults <- data.table(indicator=character(),
                            managed_area=character(),
                            habitat_class=character(),
                            size_class=character(),
                            live_date_qual=character(),
                            n_programs=integer(),
                            programs=list(),
                            filename=character(),
                            effect=character(),
                            component=character(),
                            group=character(),
                            term=character(),
                            estimate=numeric(),
                            std.error=numeric(),
                            conf.low=numeric(),
                            conf.high=numeric())

#How many years of data for each managed area/habitat class/indicator combination?
setDT(oysterraw)
oysterraw[!is.na(Density_m2), `:=` (nyrpar="Density_m2",
                                    nyears=length(unique(Year))),
          by=MA_plotlab]
oysterraw[!is.na(PercentLive_pct), `:=` (nyrpar="PercentLive_pct",
```

```
                                        nyears=length(unique(Year))),
          by=MA_plotlab]
oysterraw[!is.na(ShellHeight_mm), `:=` (nyrpar="ShellHeight_mm",
                                        nyears=length(unique(Year))),
          by=MA_plotlab]
MAinclude <- distinct(oysterraw[, .(MA_plotlab, nyrpar, nyears)])
# View(MAinclude[!is.na(nyrpar) & nyears >= 5, ])

oysterraw[str_detect(MA_plotlab, "Pine Island Sound"),
          `:=` (MA_plotlab=ifelse(str_detect(ProgramLocationID,
                                             "Reference") |
                                  str_detect(ProgramLocationID,
                                             "Control"),
                                "Pine Island Sound Aquatic Preserve_Natural",
                                "Pine Island Sound Aquatic Preserve_Restored"),
                HabitatClassification=ifelse(str_detect(ProgramLocationID,
                                                        "Reference") |
                                               str_detect(ProgramLocationID,
                                                          "Control"),
                                             "Natural", "Restored"))]
```

Managed Area Statistics

Gets summary statistics for each managed area. Uses piping from dplyr package to feed into subsequent steps. Sets of summary statistics are performed for Density, Shell Height, and Percent Living. The following steps are performed:

1. Group data that have the same `ManagedAreaName`, `Year`, `Month`, `nyrpar`, `LiveDate_Qualifier`, `SizeClass`, and `HabitatClassification`.
   - Second summary statistics do not use the `Month` grouping and are only for `ManagedAreaName`, `Year`, and the other oyster parameters.
   - Third summary statistics do not use `Year` grouping and are only for `ManagedAreaName`, `Month`, and the other oyster parameters.
   - Fourth summary statistics are only grouped based on `ManagedAreaName` and the other oyster parameters
     - 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
   - Oyster Output Files in SEACAR GitHub (https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Oyster/output)

##Density

```
oysterraw$SizeClass[oysterraw$SizeClass=="25to75mm"] <- "25-75mm"
oysterraw$SizeClass[oysterraw$SizeClass=="35to75mm"] <- "35-75mm"
oysterraw$SizeClass[oysterraw$SizeClass=="o75mm"] <- ">75mm"

# Create summary statistics for each managed area based on Year and Month
# intervals.
MA_YM_Stats <- oysterraw[oysterraw$nyrpar=="Density_m2",] %>%
   group_by(AreaID, ManagedAreaName, Year, Month, nyrpar,
```

```
            LiveDate_Qualifier, SizeClass, HabitatClassification) %>%
   dplyr::summarize(N_Data=length(Density_m2[!is.na(Density_m2)]),
             Min=min(Density_m2[!is.na(Density_m2)]),
             Max=max(Density_m2[!is.na(Density_m2)]),
             Median=median(Density_m2[!is.na(Density_m2)]),
             Mean=mean(Density_m2[!is.na(Density_m2)]),
             StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
             Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                            collapse=', '),
             ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                            collapse=', '))
setnames(MA_YM_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# 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$Month,
                                               MA_YM_Stats$ShellType,
                                               MA_YM_Stats$SizeClass,
                                               MA_YM_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_YM_Stats, paste0(out_dir,"/Density/Oyster_Dens_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 <- oysterraw[oysterraw$nyrpar=="Density_m2",] %>%
   group_by(AreaID, ManagedAreaName, Year, nyrpar, LiveDate_Qualifier,
            SizeClass, HabitatClassification) %>%
   dplyr::summarize(N_Data=length(Density_m2[!is.na(Density_m2)]),
             Min=min(Density_m2[!is.na(Density_m2)]),
             Max=max(Density_m2[!is.na(Density_m2)]),
             Median=median(Density_m2[!is.na(Density_m2)]),
             Mean=mean(Density_m2[!is.na(Density_m2)]),
             StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
             Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                            collapse=', '),
             ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                            collapse=', '))
setnames(MA_Y_Stats, c("nyrpar", "LiveDate_Qualifier",
                       "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Year
MA_Y_Stats <- as.data.table(MA_Y_Stats[order(MA_Y_Stats$ManagedAreaName,
                                             MA_Y_Stats$Year,
                                             MA_Y_Stats$ShellType,
                                             MA_Y_Stats$SizeClass,
```

```
                                              MA_Y_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_Y_Stats, paste0(out_dir,"/Density/Oyster_Dens_MA_Yr_Stats.txt"),
       sep="|")
# Removes variable storing data to improve computer memory
rm(MA_Y_Stats)

# Create summary statistics for each managed area based on Month intervals.
MA_M_Stats <- oysterraw[oysterraw$nyrpar=="Density_m2",] %>%
   group_by(AreaID, ManagedAreaName, Month, nyrpar,
            LiveDate_Qualifier, SizeClass,
            HabitatClassification) %>%
   dplyr::summarize(N_Data=length(Density_m2[!is.na(Density_m2)]),
             Min=min(Density_m2[!is.na(Density_m2)]),
             Max=max(Density_m2[!is.na(Density_m2)]),
             Median=median(Density_m2[!is.na(Density_m2)]),
             Mean=mean(Density_m2[!is.na(Density_m2)]),
             StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
             Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                            collapse=', '),
             ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                            collapse=', '))
setnames(MA_M_Stats, c("nyrpar", "LiveDate_Qualifier",
                       "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Month
MA_M_Stats <- as.data.table(MA_M_Stats[order(MA_M_Stats$ManagedAreaName,
                                             MA_M_Stats$Month,
                                             MA_M_Stats$ShellType,
                                             MA_M_Stats$SizeClass,
                                             MA_M_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_M_Stats, paste0(out_dir,"/Density/Oyster_Dens_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 <- oysterraw[oysterraw$nyrpar=="Density_m2",] %>%
   group_by(AreaID, ManagedAreaName, nyrpar,
            LiveDate_Qualifier, SizeClass,
            HabitatClassification) %>%
   dplyr::summarize(N_Years=length(unique(
     LiveDate[!is.na(LiveDate) & !is.na(Density_m2)])),
             SufficientData=ifelse(N_Years>=5, TRUE, FALSE),
             EarliestLiveDate=min(LiveDate[!is.na(Density_m2)]),
             LatestLiveDate=max(LiveDate[!is.na(Density_m2)]),
             LastSampleDate=max(SampleDate),
             N_Data=length(Density_m2[!is.na(Density_m2)]),
             Min=min(Density_m2[!is.na(Density_m2)]),
             Max=max(Density_m2[!is.na(Density_m2)]),
```

```
            Median=median(Density_m2[!is.na(Density_m2)]),
            Mean=mean(Density_m2[!is.na(Density_m2)]),
            StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
            Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                           collapse=', '),
            ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                           collapse=', '))
setnames(MA_Ov_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName
MA_Ov_Stats <- as.data.table(MA_Ov_Stats[order(MA_Ov_Stats$ManagedAreaName,
                                               MA_Ov_Stats$ShellType,
                                               MA_Ov_Stats$SizeClass,
                                               MA_Ov_Stats$HabitatType), ])

# 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,"/Density/Oyster_Dens_MA_Overall_Stats.txt"),
       sep="|")
# Removes variable storing data to improve computer memory
rm(MA_Ov_Stats)
```

##Shell Height

```
# Create summary statistics for each managed area based on Year and Month
# intervals.
MA_YM_Stats <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
   group_by(AreaID, ManagedAreaName, Year, Month, nyrpar,
            LiveDate_Qualifier, SizeClass, HabitatClassification) %>%
   dplyr::summarize(N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                           collapse=', '),
            ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                           collapse=', '))
setnames(MA_YM_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName, Year, then Month
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                               MA_YM_Stats$Year,
```

```
                                                MA_YM_Stats$Month,
                                                MA_YM_Stats$ShellType,
                                                MA_YM_Stats$SizeClass,
                                                MA_YM_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_YM_Stats, paste0(out_dir,"/Shell_Height/Oyster_SH_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 <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
   group_by(AreaID, ManagedAreaName, Year, nyrpar, LiveDate_Qualifier,
            SizeClass, HabitatClassification) %>%
   dplyr::summarize(N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                            collapse=', '),
             ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                            collapse=', '))
setnames(MA_Y_Stats, c("nyrpar", "LiveDate_Qualifier",
                       "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Year
MA_Y_Stats <- as.data.table(MA_Y_Stats[order(MA_Y_Stats$ManagedAreaName,
                                             MA_Y_Stats$Year,
                                             MA_Y_Stats$ShellType,
                                             MA_Y_Stats$SizeClass,
                                             MA_Y_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_Y_Stats, paste0(out_dir,"/Shell_Height/Oyster_SH_MA_Yr_Stats.txt"),
       sep="|")
# Removes variable storing data to improve computer memory
rm(MA_Y_Stats)

# Create summary statistics for each managed area based on Month intervals.
MA_M_Stats <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
   group_by(AreaID, ManagedAreaName, Month, nyrpar,
            LiveDate_Qualifier, SizeClass,
            HabitatClassification) %>%
   dplyr::summarize(N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
             Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
```

```
                              collapse=', '),
            ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                              collapse=', '))
setnames(MA_M_Stats, c("nyrpar", "LiveDate_Qualifier",
                       "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# 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,
                                             MA_M_Stats$ShellType,
                                             MA_M_Stats$SizeClass,
                                             MA_M_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_M_Stats, paste0(out_dir,"/Shell_Height/Oyster_SH_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 <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
   group_by(AreaID, ManagedAreaName, nyrpar,
            LiveDate_Qualifier, SizeClass,
            HabitatClassification) %>%
   dplyr::summarize(N_Years=length(unique(
     LiveDate[!is.na(LiveDate) & !is.na(ShellHeight_mm)])),
            SufficientData=ifelse(N_Years>=5, TRUE, FALSE),
            EarliestLiveDate=min(LiveDate[!is.na(ShellHeight_mm)]),
            LatestLiveDate=max(LiveDate[!is.na(ShellHeight_mm)]),
            LastSampleDate=max(SampleDate),
            N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
            Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                              collapse=', '),
            ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                              collapse=', '))
setnames(MA_Ov_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName
MA_Ov_Stats <- as.data.table(MA_Ov_Stats[order(MA_Ov_Stats$ManagedAreaName,
                                               MA_Ov_Stats$ShellType,
                                               MA_Ov_Stats$SizeClass,
                                               MA_Ov_Stats$HabitatType), ])

# 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,"/Shell_Height/Oyster_SH_MA_Overall_Stats.txt"),
       sep="|")
# Removes variable storing data to improve computer memory
rm(MA_Ov_Stats)
```

##Percent Live

```
# Create summary statistics for each managed area based on Year and Month
# intervals.
MA_YM_Stats <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
   group_by(AreaID, ManagedAreaName, Year, Month, nyrpar,
            LiveDate_Qualifier, SizeClass, HabitatClassification) %>%
   dplyr::summarize(N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
             Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
             Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
             Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
             Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
             StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)]),
             Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                            collapse=', '),
             ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                            collapse=', '))
setnames(MA_YM_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName, Year, then Month
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                               MA_YM_Stats$Year,
                                               MA_YM_Stats$Month,
                                               MA_YM_Stats$ShellType,
                                               MA_YM_Stats$SizeClass,
                                               MA_YM_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_YM_Stats, paste0(out_dir,"/Percent_Live/Oyster_PrcLive_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 <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
   group_by(AreaID, ManagedAreaName, Year, nyrpar, LiveDate_Qualifier,
            SizeClass, HabitatClassification) %>%
   dplyr::summarize(N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
             Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
             Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
             Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
             Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
```

```
            StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)]),
            Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                           collapse=', '),
            ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                           collapse=', '))
setnames(MA_Y_Stats, c("nyrpar", "LiveDate_Qualifier",
                       "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# 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,
                                             MA_Y_Stats$ShellType,
                                             MA_Y_Stats$SizeClass,
                                             MA_Y_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_Y_Stats, paste0(out_dir,"/Percent_Live/Oyster_PrcLive_MA_Yr_Stats.txt"),
       sep="|")
# Removes variable storing data to improve computer memory
rm(MA_Y_Stats)

# Create summary statistics for each managed area based on Month intervals.
MA_M_Stats <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
   group_by(AreaID, ManagedAreaName, Month, nyrpar,
            LiveDate_Qualifier, SizeClass,
            HabitatClassification) %>%
   dplyr::summarize(N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
            Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
            Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
            Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
            Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
            StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)]),
            Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                           collapse=', '),
            ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                           collapse=', '))
setnames(MA_M_Stats, c("nyrpar", "LiveDate_Qualifier",
                       "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Month
MA_M_Stats <- as.data.table(MA_M_Stats[order(MA_M_Stats$ManagedAreaName,
                                             MA_M_Stats$Month,
                                             MA_M_Stats$ShellType,
                                             MA_M_Stats$SizeClass,
                                             MA_M_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_M_Stats, paste0(out_dir,"/Percent_Live/Oyster_PrcLive_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 <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
   group_by(AreaID, ManagedAreaName, nyrpar,
            LiveDate_Qualifier, SizeClass,
            HabitatClassification) %>%
   dplyr::summarize(N_Years=length(unique(
     LiveDate[!is.na(LiveDate) & !is.na(PercentLive_pct)])),
             SufficientData=ifelse(N_Years>=5, TRUE, FALSE),
             EarliestLiveDate=min(LiveDate[!is.na(PercentLive_pct)]),
             LatestLiveDate=max(LiveDate[!is.na(PercentLive_pct)]),
             LastSampleDate=max(SampleDate),
             N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
             Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
             Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
             Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
             Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
             StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)]),
             Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                            collapse=', '),
             ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                            collapse=', '))
setnames(MA_Ov_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName
MA_Ov_Stats <- as.data.table(MA_Ov_Stats[order(MA_Ov_Stats$ManagedAreaName,
                                               MA_Ov_Stats$ShellType,
                                               MA_Ov_Stats$SizeClass,
                                               MA_Ov_Stats$HabitatType), ])

# 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,"/Percent_Live/Oyster_PrcLive_MA_Overall_Stats.txt"),
       sep="|")
# Removes variable storing data to improve computer memory
rm(MA_Ov_Stats)

# LiveDate Threshold ------------------------------------------------------
oysterraw <- oysterraw[oysterraw$LiveDate>=1960,]

for(i in unique(oysterraw$ManagedAreaName)){
      oysterraw[ManagedAreaName==i & !is.na(LiveDate), `:=`
                (RelYear=(LiveDate-min(LiveDate))+1)]
}
```

Functions

Sets universal them for plots and creates functions that are used to create figures and data files from data subsets and GLMM models. The functions are:

1. diagnosticplots

- Creates panel of summary plots for each analysis

2. meplots

- Creates and saves plots for density and live percent

3. modresults

- Takes input data and model results for density and live percent.
- Adds model results to compilation variable.
- Send data and model results to diagnosticplots and meplots

4. meplotssh

- Creates and saves plots for shell height

5. modresultssh

- Takes input data and model results for shell height.
- Adds model results to compilation variable.
- Send data and model results to diagnosticplots and meplotssh

```
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 = element_text(hjust = 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))

plot_jitter <- position_jitter(width = 0.1, height = 0.1, seed=42)
#Function to save diagnostic plots
diagnosticplots <- function(model, indicator, managedarea, sizeclass="",
                            historical=FALSE){
  ind <- case_when(str_detect(indicator, "ercent") ~ "Pct",
                   str_detect(indicator, "ensity") ~ "Den",
                   str_detect(indicator, "^S|^s") ~ "SH")

  ma <- managedarea

  if(sizeclass != ""){
    size <- case_when(str_detect(sizeclass, "25") &
                        str_detect(sizeclass, "75") ~ "25to75",
                      str_detect(sizeclass, "35") &
                        str_detect(sizeclass, "75") ~ "35to75",
```

```
                      str_detect(sizeclass, "25")==FALSE &
                        str_detect(sizeclass, "75") ~ "o75", TRUE ~ "raw")
  sizelab <- case_when(str_detect(sizeclass, "25") &
                         str_detect(sizeclass, "75") ~ "25-75mm",
                       str_detect(sizeclass, "35") &
                         str_detect(sizeclass, "75") ~ "35-75mm",
                       str_detect(sizeclass, "25")==FALSE &
                         str_detect(sizeclass, "75") ~ "\u2265 75mm",
                       TRUE ~ "raw")
}

#Save diagnostic plot(s) of chains
diag <- plot(model, plot=FALSE)

title <- textGrob(paste0(ma, " (", ind, " ", sizelab, ")"),
                  just="left",
                  gp=gpar(fontsize=10))

diag[[1]] <- gtable_add_rows(
  diag[[1]],
  heights=grobHeight(title)+unit(5, "mm"),
  pos=0
)

diag[[1]] <- gtable_add_grob(
  diag[[1]],
  title,
  clip="off",
  1, 1, 1, 1)

if(class(try(diag[[2]], silent=TRUE)) != "try-error"){
  diag[[2]] <- gtable_add_rows(
    diag[[2]],
    heights=grobHeight(title)+unit(5, "mm"),
    pos=0
  )
}

if(class(try(diag[[3]], silent=TRUE)) != "try-error"){
  diag[[3]] <- gtable_add_rows(
    diag[[3]],
    heights=grobHeight(title)+unit(5, "mm"),
    pos=0
  )
}

#save chains plots
jpeg(filename=paste0("output/Figures/", ind, "_AllDates_GLMM_", ma,
                     "_PDistandMChains_", ifelse(sizeclass != "",
                                                 paste0(size, "_"), ""),
                     ifelse(historical==TRUE, "hist_", "_"),
                     Sys.Date(), ".png"),
     width=6,
```

```
      height=ifelse(length(diag)==1, 6, ifelse(length(diag)==2, 12, 18)),
      units="in", quality=100, res=300)
  print(grid.arrange(grobs=diag, ncol=1))
  dev.off()

  #Save posterior predictive check plot
  postpc <- tryCatch(pp_check(model),
                     error=function(e) NA)
  k <- 1001

  while(is.na(postpc)==TRUE & k <= 1000){
    postpc <- tryCatch(pp_check(model),
                       error=function(e) NA)
    k <- k+1
  }

  if(!is.na(postpc)){
    postpc <- postpc +
      labs(title=paste0(ind, "_AllDates_GLMM_", ma, "_PPcheck_",
                        ifelse(sizeclass != "", paste0(size, "_"), ""),
                        ifelse(historical==TRUE, "_hist_", "_"),
                        Sys.Date(), ".png"))

    ma_short <- MA_All[ManagedAreaName==ma, Abbreviation]

    ggsave(paste0("output/Figures/", ind, "_AllDates_GLMM_", ma_short, "_PPcheck_",
                  ifelse(sizeclass != "", paste0(size, "_"), ""),
                  ifelse(historical==TRUE, "_hist_", "_"), Sys.Date(),
                  ".png"),
           postpc,
           width=6,
           height=6,
           units="in",
           dpi=300)
  }

  print(paste0("Plots saved."))

}

# Save marginal effects plots
meplots <- function(models, data, indicator, managedarea, sizeclass="",
                    zoom=FALSE){
  ind <- case_when(str_detect(indicator, "ercent") ~ "Pct",
                   str_detect(indicator, "ensity") ~ "Den",
                   str_detect(indicator, "^S|^s") ~ "SH")

  ma <- managedarea

  if(sizeclass != ""){
    size <- case_when(str_detect(sizeclass, "25") &
                        str_detect(sizeclass, "75") ~ "25to75",
                      str_detect(sizeclass, "35") &
```

```
                      str_detect(sizeclass, "75") ~ "35to75",
                    str_detect(sizeclass, "25")==FALSE &
                      str_detect(sizeclass, "75") ~ "o75", TRUE ~ "raw")
  sizelab <- case_when(str_detect(sizeclass, "25") &
                         str_detect(sizeclass, "75") ~ "25-75mm",
                       str_detect(sizeclass, "35") &
                         str_detect(sizeclass, "75") ~ "35-75mm",
                       str_detect(sizeclass, "25")==FALSE &
                         str_detect(sizeclass, "75") ~ "\u2265 75mm",
                       TRUE ~ "raw")
}



if(ind=="Den"){
  nyrs <- max(data$LiveDate)-min(data$LiveDate)+1
  maxyr <- max(data$LiveDate)
  minyr <- min(data$LiveDate)
  if(grepl("Natural", unique(data$MA_plotlab))==TRUE){
    type <- "Natural"
  } else{
    type <- "Restored"
  }
  yrdiff <- unique(data$YearDiff)
  # Creates break intervals for plots based on number of years of data
  if(nyrs>=40){
    # Set breaks to every 10 years if more than 40 years of data
    brk <- 10
  }else if(nyrs<40 & nyrs>=20){
    # Set breaks to every 5 years if between 40 and 20 years of data
    brk <- 5
  }else if(nyrs<20 & nyrs>=12){
    # Set breaks to every 3 years if between 20 and 12 years of data
    brk <- 3
  }else if(nyrs<12 & nyrs>=8){
    # Set breaks to every 2 years if between 12 and 8 years of data
    brk <- 2
  }else if(nyrs<8){
    # Set breaks to every year if less than 8 years of data
    brk <- 1
  }else if(nyrs<3){
    brk <- 1
    maxyr <- maxyr + 1
    minyr <- minyr - 1
  }
  yrlist <- seq(minyr,maxyr,brk)
  # nbreaks <- ifelse(nyrs < 11, nyrs+1, 12)
  # breaks <- if(minyr==0){
  #   c(minyr, round(minyr+c(1:(nbreaks-2))*((nyrs/nbreaks) +
  #                                            (nyrs/nbreaks)/nbreaks)),
  #     maxyr)+1
  # } else{
```

```
#   c(minyr, round(minyr+c(1:(nbreaks-2))*((nyrs/nbreaks) +
#                                          (nyrs/nbreaks)/nbreaks)),
#     maxyr)
# }

denplots <- plot(conditional_effects(models[[1]], re_formula=NULL),
                 plot=FALSE)

plot1 <- ggplot() +
  {if("meanDen_int" %in% colnames(data)){
    geom_point(data=data, aes(x=LiveDate,
                              y=meanDen_int), position=plot_jitter,
               shape=21, size=2, color="#333333", fill="#cccccc",
               alpha=1, inherit.aes=FALSE)
  } else{
    geom_point(data=data, aes(x=LiveDate,
                              y=Density_m2), position=plot_jitter,
               shape=21, size=2, color="#333333", fill="#cccccc",
               alpha=1, inherit.aes=FALSE)
  }
  } +
  list(geom_ribbon(data=denplots$RelYear$data,
                   aes(x=RelYear+yrdiff, y=Density_m2,
                       ymin=lower__, ymax=upper__),
                   fill="#000099", alpha=0.1, inherit.aes=FALSE),
       geom_line(data=denplots$RelYear$data,
                 aes(x=RelYear+yrdiff,
                     y=estimate__),
                 color="#000099", lwd=0.75, inherit.aes=FALSE)) +
  scale_x_continuous(limits=c(minyr-0.25, maxyr+0.25),
                     breaks=yrlist) +
  plot_theme +
  {if("meanDen_int" %in% colnames(data)){
    labs(title="Oyster Density",
         subtitle=managedarea,
         x="Year",
         y=bquote('Estimated density ('*~m^{-2}*')'))
  }else{
    labs(title="Oyster Density",
         subtitle=managedarea,
         x="Year",
         y=bquote('Density ('*~m^{-2}*')'))
  }}
# labs(title="Oyster Density",
#      subtitle=managedarea,
#      x="Year",
#      y=ifelse("meanDen_int" %in% colnames(data),
#                           "Estimated density (square meters)",
#                           bquote('Richness (species/100'*~m^{2}*')')))


ma_short <- MA_All[ManagedAreaName==ma, Abbreviation]
```

```
  ggsave(paste0("output/Density/Figures/Oyster_Dens_GLMM_", ma_short, "_", type,
                ifelse(sizeclass != "", paste0(size), "_raw"), ".png"),
         plot1,
         width=8,
         height=4,
         units="in",
         dpi=200)
}


#Marginal effects plot including random effects for percent live
if(ind=="Pct"){
  nyrs <- max(data$LiveDate)-min(data$LiveDate)+1
  maxyr <- max(data$LiveDate)
  minyr <- min(data$LiveDate)
  yrdiff <- unique(data$YearDiff)
  # Creates break intervals for plots based on number of years of data
  if(nyrs>=40){
    # Set breaks to every 10 years if more than 40 years of data
    brk <- 10
  }else if(nyrs<40 & nyrs>=20){
    # Set breaks to every 5 years if between 40 and 20 years of data
    brk <- 5
  }else if(nyrs<20 & nyrs>=12){
    # Set breaks to every 3 years if between 20 and 12 years of data
    brk <- 3
  }else if(nyrs<12 & nyrs>=8){
    # Set breaks to every 2 years if between 12 and 8 years of data
    brk <- 2
  }else if(nyrs<8){
    # Set breaks to every year if less than 8 years of data
    brk <- 1
  }else if(nyrs<3){
    brk <- 1
    maxyr <- maxyr + 1
    minyr <- minyr - 1
  }
  yrlist <- seq(minyr,maxyr,brk)
  # nbreaks <- ifelse(nyrs < 11, nyrs+1, 12)
  # breaks <- if(minyr==0){
  #   c(minyr, round(minyr+c(1:(nbreaks-2))*((nyrs/nbreaks) +
  #                                          (nyrs/nbreaks)/nbreaks)),
  #     maxyr)+1
  # } else{
  #   c(minyr, round(minyr+c(1:(nbreaks-2))*((nyrs/nbreaks) +
  #                                          (nyrs/nbreaks)/nbreaks)),
  #     maxyr)
  # }

  set.seed(987)
  pctplots <- plot(conditional_effects(models[[1]], re_formula=NULL),
                   plot=FALSE)
```

```
plot1 <- ggplot() +
  geom_point(data=data, aes(x=LiveDate,
                            y=100*PercentLive_dec), position=plot_jitter,
             shape=21, size=2, color="#333333", fill="#cccccc",
             alpha=1, inherit.aes=FALSE) +
  {if(names(pctplots$RelYear$data[2])=="PercentLive_dec"){
    list(geom_ribbon(data=pctplots$RelYear$data,
                     aes(x=RelYear+yrdiff,
                         y=100*PercentLive_dec, ymin=100*lower__,
                         ymax=100*upper__), fill="#000099",
                     alpha=0.1, inherit.aes=FALSE),
         geom_line(data=pctplots$RelYear$data,
                   aes(x=RelYear+yrdiff,
                       y=100*estimate__), color="#000099",
                   lwd=0.75, inherit.aes=FALSE))
  } else{
    list(geom_ribbon(data=pctplots$RelYear$data,
                     aes(x=RelYear+yrdiff,
                         y=100*LiveObs, ymin=100*lower__,
                         ymax=100*upper__), fill="#000099",
                     alpha=0.1, inherit.aes=FALSE),
         geom_line(data=pctplots$RelYear$data,
                   aes(x=RelYear+yrdiff,
                       y=100*estimate__), color="#000099",
                   lwd=0.75, inherit.aes=FALSE))
  }} +
  scale_x_continuous(limits=c(minyr-0.25, maxyr+0.25),
                     breaks=yrlist) +
  plot_theme +
  theme(legend.text=element_text(size=10),
        legend.title=element_text(size=10)) +
  {
    if(managedarea=="Lemon Bay Aquatic Preserve"){
      labs(title="Percent of Live vs. Dead Oysters",
           subtitle=managedarea,
           x="Year",
           y="Live vs. dead (%)")
    } else {
      labs(title="Oyster Percent Live Cover",
           subtitle=managedarea,
           x="Year",
           y="Live cover (%)")
    }
  }

ma_short <- MA_All[ManagedAreaName==ma, Abbreviation]

ggsave(paste0("output/Percent_Live/Figures/Oyster_PrcLive_GLMM_", ma_short,
              "_raw.png"),
       plot1,
       width=8,
       height=4,
       units="in",
```

```
        dpi=200)

  #Plot of modeled mean percent live
  if("Region" %in% names(pctplots)){
    meanPct <- pctplots$Region$data
    setnames(meanPct, "effect1__", "Region")

    meanpctplot <- ggplot(meanPct, aes(x=Region, y=estimate__,
                                       ymin=lower__, ymax=upper__)) +
      geom_pointinterval(fill="black", size=3,
                         fatten_point=4, shape=21,
                         color="black") +
      labs(title="Oyster Percent Live Cover",
           subtitle=managedarea,
           y="Live cover (%)",
           fill=NULL) +
      plot_theme +
      theme(legend.text=element_text(size=10),
            legend.title=element_text(size=10))

    ma_short <- MA_All[ManagedAreaName==ma, Abbreviation]

    ggsave(paste0("output/Percent_Live/Figures/Oyster_PrcLive_GLMM_", ma_short,
                  "_raw_MeanRes.png"),
           meanpctplot,
           width=8,
           height=4,
           units="in",
           dpi=200)
  }

  #Plot of RelYear * Region interaction
  if("RelYear:Region" %in% names(pctplots)){
    pctplots$RelYear$data$RelYear <-
      pctplots$RelYear$data$RelYear-
      (min(pctplots$RelYear$data$RelYear)-1)
    RelYrbyRegion <- pctplots$`RelYear:Region`


    intplot <- RelYrbyRegion +
      geom_point(data=data, aes(x=RelYear-(min(RelYear)-1),
                                y=PercentLive_dec,
                                fill=Region),
                 alpha=0.5, shape=21, size=3, color="black",
                 inherit.aes=FALSE) +
      scale_x_continuous(breaks=breaks,
                         labels=c(yrlist[breaks])) +
      labs(title=ma,
           x="Year",
           y="Proportion live",
           fill="Region") +
      plot_theme +
      theme(legend.text=element_text(size=12),
```

```
              legend.title=element_text(size=13),
              legend.position="none") +
        facet_wrap(~ Region, ncol=3, scales="free")

      ma_short <- MA_All[ManagedAreaName==ma, Abbreviation]

      ggsave(paste0("output/Percent_Live/Figures/Oyster_PrcLive_GLMM_", ma_short,
                    "_raw.png"),
             intplot,
             width=10,
             height=10,
             units="in",
             dpi=300)
    }
  }
}

# Create model results tables and save diagnostic plots
modresults <- function(datafile, models, indicator, meplotzoom=FALSE){
  for(m in seq_along(models)){
    modelobj <- models[[m]]
    sizeclass <- ifelse(str_detect(modelobj$file, "25to75|seed"),
                        "25-75mm",
                        ifelse(str_detect(modelobj$file, "35to75|seed"),
                               "35-75mm",
                               ifelse(str_detect(modelobj$file,
                                                 "o75|market"),
                                      ">75mm", "NA")))
    oyres_i <- setDT(broom.mixed::tidy(modelobj))
    #tidy() does not like that parameter values have underscores for
    #some reason, so the resulting table is incomplete

    if(nrow(oyres_i[effect=="fixed", ])-nrow(summary(modelobj)$fixed)==-1){
      missingrow <- data.table(effect="fixed",
                               component="cond",
                               #not sure what "cond" means in the tidy summary.
                               group=NA,
                               term=rownames(summary(modelobj)$fixed)[2],
                               estimate=summary(modelobj)$fixed$Estimate[2],
                               std.error=summary(modelobj)$fixed$Est.Error[2],
                               conf.low=summary(modelobj)$fixed$`l-95% CI`[2],
                               conf.high=summary(modelobj)$fixed$`u-95% CI`[2])
      oyres_i <- rbind(oyres_i, missingrow) %>% arrange(effect, group)
    }

    oyres_i[, `:=` (indicator=indicator,
                    managed_area=unique(datafile$ManagedAreaName),
                    habitat_class=unique(datafile$HabitatClassification),
                    size_class=sizeclass,
                    live_date_qual=ifelse(
                      str_detect(modelobj$file, "_hist"),
                      "Estimate", "Exact"),
                    n_programs=if(
```

```
                        class(try(datafile$LiveDate_Qualifier)) !=
                        "try-error"){
                        length(
                          unique(
                            datafile[LiveDate_Qualifier==
                                       ifelse(
                                         str_detect(
                                           modelobj$file,
                                           "_hist"),
                                         "Estimate",
                                         "Exact"),
                                     ProgramID]))
                      } else{length(unique(datafile[, ProgramID]))},
                      programs=if(class(try(
                        datafile$LiveDate_Qualifier)) != "try-error"){
                        list(unique(datafile[LiveDate_Qualifier==
                                               ifelse(
                                                 str_detect(
                                                   modelobj$file,
                                                   "_hist"),
                                                 "Estimate",
                                                 "Exact"),
                                             ProgramID]))
                      } else{list(unique(datafile[, ProgramID]))},
                      filename=modelobj$file)]
    oysterresults <<- rbind(oysterresults, oyres_i)

    # Save diagnostic plots
    #diagnosticplots(modelobj, indicator,
    #unique(datafile$ManagedAreaName), sizeclass,
    #ifelse(str_detect(modelobj$file, "_hist"), TRUE, FALSE))
  }

  # Save marginal effects plots
  meplots(models, datafile, indicator, unique(datafile$ManagedAreaName),
          sizeclass, meplotzoom)
}
# Marginal effects plots for shell height (attempt to combine models into one plot)
meplotssh <- function(models1, data1, sizeclass1="", models2, data2,
                      sizeclass2="", managedarea, indicator, zoom=FALSE){
  ind <- case_when(str_detect(indicator, "ercent") ~ "Pct",
                   str_detect(indicator, "ensity") ~ "Den",
                   str_detect(indicator, "^S|^s") ~ "SH")
  ma <- managedarea

  sizeclass1 <- unique(data1$SizeClass)
  sizeclass2 <- unique(data2$SizeClass)
  # Set size labels
  if(sizeclass1 != ""){
    size1 <- case_when(
      str_detect(sizeclass1, "25") & str_detect(sizeclass1, "75") ~ "25to75",
      str_detect(sizeclass1, "35") & str_detect(sizeclass1, "75") ~ "35to75",
      str_detect(sizeclass1, "25")==FALSE & str_detect(sizeclass1, "75") ~ "o75",
```

```
    TRUE ~ "raw")
  sizelab1 <- case_when(
    str_detect(sizeclass1, "25") & str_detect(sizeclass1, "75") ~ "25-75mm",
    str_detect(sizeclass1, "35") & str_detect(sizeclass1, "75") ~ "35-75mm",
    str_detect(sizeclass1, "25")==FALSE & str_detect(sizeclass1, "75") ~ "\u2265 75mm",
    TRUE ~ "raw")
}
if(sizeclass2 != ""){
  size2 <- case_when(
    str_detect(sizeclass2, "25") & str_detect(sizeclass2, "75") ~ "25to75",
    str_detect(sizeclass2, "35") & str_detect(sizeclass2, "75") ~ "35to75",
    str_detect(sizeclass2, "25")==FALSE & str_detect(sizeclass2, "75") ~ "o75",
    TRUE ~ "raw")
  sizelab2 <- case_when(
    str_detect(sizeclass2, "25") & str_detect(sizeclass2, "75") ~ "25-75mm",
    str_detect(sizeclass2, "35") & str_detect(sizeclass2, "75") ~ "35-75mm",
    str_detect(sizeclass2, "25")==FALSE & str_detect(sizeclass2, "75") ~ "\u2265 75mm",
    TRUE ~ "raw")
}

#Marginal effects plot including random effects
## Hist plot settings
y_max <- round(max(data2[!is.na(ShellHeight_mm), ShellHeight_mm]), -0)+1
y_breaks <- seq(0, 300, 50)
y_labs <- seq(0, 300, 50)
y_minor <- seq(0, 300, 25)
ylim_upper <- ceiling(y_max/25)*25

yrdiff1 <- unique(data1$YearDiff)
yrdiff2 <- unique(data2$YearDiff)

# function to set year breaks, type == "hist" or "live"
set_breaks <- function(type, data1, data2){
  ldq <- ifelse(type=="hist", "Estimate", "Exact")

  maxyr <- max(data1[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate],
               data2[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate])
  minyr <- min(data1[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate],
               data2[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate])
  nyrs <- (maxyr)-(minyr)+1

  # Creates break intervals for plots based on number of years of data
  if(nyrs>=40){
    # Set breaks to every 10 years if more than 30 years of data
    brk <- 10
  }else if(nyrs<40 & nyrs>=20){
    # Set breaks to every 5 years if between 30 and 15 years of data
    brk <- 5
  }else if(nyrs<20 & nyrs>=12){
    # Set breaks to every 3 years if between 15 and 9 years of data
    brk <- 3
  }else if(nyrs<12 & nyrs>=8){
    # Set breaks to every 2 years if between 9 and 6 years of data
```

```
    brk <- 2
  }else if(nyrs<8 & nyrs>=3){
    # Set breaks to every year if less than 6 years of data
    brk <- 1
  }else if(nyrs<3){
    brk <- 1
    maxyr <- maxyr + 1
    minyr <- minyr - 1
  }
  return(list("seq" = seq(minyr,maxyr,brk),"maxyr" = maxyr, "minyr" = minyr))
}

yrlist_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["seq"]]
maxyr_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["maxyr"]]
minyr_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["minyr"]]
yrlist_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["seq"]]
maxyr_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["maxyr"]]
minyr_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["minyr"]]

## Check data for Exact and Estimate
n_hist1 <- nrow(data1[data1$LiveDate_Qualifier=="Estimate" &
                        !is.na(data1$ShellHeight_mm),])
n_live1 <- nrow(data1[data1$LiveDate_Qualifier=="Exact" &
                        !is.na(data1$ShellHeight_mm),])
n_hist2 <- nrow(data2[data2$LiveDate_Qualifier=="Estimate" &
                        !is.na(data2$ShellHeight_mm),])
n_live2 <- nrow(data2[data2$LiveDate_Qualifier=="Exact" &
                        !is.na(data2$ShellHeight_mm),])

set.seed(987)
if(!is.null(models1)){
  liveplot_1 <- plot(conditional_effects(models1[[1]], re_formula=NULL), plot=FALSE)
}

if(!is.null(models2)){
  liveplot_2 <- plot(conditional_effects(models2[[1]], re_formula=NULL), plot=FALSE)
}

# Set boolean values for whether liveplot1&2 are available
liveplot1_avail <- class(try(liveplot_1, silent=TRUE)) != "try-error"
liveplot2_avail <- class(try(liveplot_2, silent=TRUE)) != "try-error"

# Set ribbon transparency value
a_ribb <- 0.2
# Set size and shapes for plots
p_shape <- c("size2"=24, "size1"=21)
sizelab <- c("size2"=sizelab2, "size1"=sizelab1)

check <- NA
check1 <- NA
check2 <- NA

if(exists("present1")){
```

```
  check1 <- c("size1"="#00374f")
} else{
  check1 <- c("size1"="#FFFFFF")
}

if(exists("present2")){
  check2 <- c("size2"="#0094b0")
} else{
  check2 <- c("size2"="#FFFFFF")
}
p_color <- c(check2, check1)

# Initial plots to set legends
plot_leg <- ggplot() +
  {if(liveplot1_avail){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear+yrdiff1, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__,
                         fill="size1"), alpha=a_ribb),
         geom_line(data=liveplot_1$RelYear$data,
                   aes(x=RelYear+yrdiff1, y=estimate__, color="size1"),
                   lwd=0.75))
  }} +
  {if(liveplot2_avail){
    list(geom_ribbon(data=liveplot_2$RelYear$data,
                     aes(x=RelYear+yrdiff2, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__,
                         fill="size2"), alpha=a_ribb),
         geom_line(data=liveplot_2$RelYear$data,
                   aes(x=RelYear+yrdiff2, y=estimate__, color="size2"),
                   lwd=0.75))
  }} +
  geom_point(data=data1[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
             position=plot_jitter, size=2, color="#333333", fill="#cccccc",
             alpha=0.2, inherit.aes=FALSE) +
  geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
             position=plot_jitter, size=2, color="#333333", fill="#cccccc",
             alpha=0.2, inherit.aes=FALSE) +
  plot_theme +
  theme(legend.position="right") +
  scale_shape_manual(name="Shell heights",
                     values=p_shape,
                     labels=sizelab) +
  scale_color_manual(name="Shell heights",
                     values=p_color,
                     labels=sizelab) +
  scale_fill_manual(name="Shell heights",
                    values=p_color,
                    labels=sizelab)

leg <-get_legend(plot_leg)
```

```
rm(plot_leg)

# Dead oyster shell plot
plot1 <- ggplot() +
  geom_hline(yintercept=75, size=1, color="grey") +
  {if(n_hist1>0){
    geom_point(data=data1[!is.na(RelYear) &
                          !is.na(LiveDate) &
                          LiveDate_Qualifier=="Estimate", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  {if(n_hist2>0){
    geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
                          LiveDate_Qualifier=="Estimate", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  scale_x_continuous(limits=c(minyr_hist-0.25, maxyr_hist+0.25),
                     breaks=yrlist_hist) +
  scale_y_continuous(breaks=y_breaks,
                     labels=y_labs, minor_breaks=y_minor) +
  plot_theme +
  theme(plot.subtitle=element_text(hjust=0, size=10, color="#314963"),
        legend.position="none",
  ) +
  labs(subtitle="Dead Oyster Shells",
       x="Estimated year",
       y="Shell height (mm)") +
  scale_shape_manual(name="Shell heights",
                     values=c("size1"=21, "size2"=24),
                     labels=c(sizelab1, sizelab2)) +
  scale_color_manual(name="Shell heights",
                     values=c("size1"="#00374f", "size2"="#0094b0"),
                     labels=c(sizelab1, sizelab2)) +
  scale_fill_manual(name="Shell heights",
                    values=c("size1"="#00374f", "size2"="#0094b0"),
                    labels=c(sizelab1, sizelab2)) +
  coord_cartesian(ylim=c(0, ylim_upper))

# Live oyster shell plot
plot2 <- ggplot() +
  geom_hline(yintercept=75, size=1, color="grey") +
  {if(n_live1>0){
    geom_point(data=data1[!is.na(RelYear) & !is.na(LiveDate) &
                          LiveDate_Qualifier=="Exact", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  {if(n_live2>0){
```

```
    geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
                            LiveDate_Qualifier=="Exact", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  {if(liveplot1_avail){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear+yrdiff1, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size1"),
                     alpha=a_ribb),
         geom_line(data=liveplot_1$RelYear$data,
                   aes(x=RelYear+yrdiff1, y=estimate__, color="size1"),
                   lwd=0.75))
  }} +
  {if(liveplot2_avail){
    list(geom_ribbon(data=liveplot_2$RelYear$data,
                     aes(x=RelYear+yrdiff2, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size2"),
                     alpha=a_ribb),
         geom_line(data=liveplot_2$RelYear$data,
                   aes(x=RelYear+yrdiff2, y=estimate__, color="size2"),
                   lwd=0.75))
  }} +
  scale_x_continuous(limits=c(minyr_live-0.25, maxyr_live+0.25),
                     breaks=yrlist_live) +
  scale_y_continuous(breaks=y_breaks,
                     labels=y_labs, minor_breaks=y_minor) +
  plot_theme +
  theme(plot.subtitle=element_text(hjust=0, size=10, color="#314963"),
        legend.position="none",
        axis.text.y=element_blank(),  #remove y-axis labels
        axis.ticks.y=element_blank(),  #remove y-axis ticks
        axis.title.y=element_blank()   #removes y-axis title
  ) +
  labs(subtitle="Live Oyster Shells",
       x="Year",
       y="Shell height (mm)") +
  scale_shape_manual(name="Shell heights",
                     values=c("size1"=21, "size2"=24),
                     labels=c(sizelab1, sizelab2)) +
  scale_color_manual(name="Shell heights",
                     values=c("size1"="#00374f", "size2"="#0094b0"),
                     labels=c(sizelab1, sizelab2)) +
  scale_fill_manual(name="Shell heights",
                    values=c("size1"="#00374f", "size2"="#0094b0"),
                    labels=c(sizelab1, sizelab2)) +
  coord_cartesian(ylim=c(0, ylim_upper))

# Set plot title
plot_title <- ggplot() +
  labs(title="Oyster Size Class", subtitle=ma) +
  plot_theme +
```

```
    theme(plot.subtitle=element_text(hjust=0.5, size=10, color="#314963"),
          panel.border=element_blank(),
          panel.grid.major=element_blank(),
          panel.grid.minor=element_blank(), axis.line=element_blank())

  plot_comb <- ggarrange(plot1, plot2, leg, nrow=1,
                         widths=c(0.46, 0.39, 0.15))

  plot_comb <- ggarrange(plot_title, plot_comb, ncol=1,
                         heights=c(0.125, 0.875))

  ma_abrev <- MA_All[ManagedAreaName==ma, Abbreviation]

  ggsave(paste0("output/Shell_Height/Figures/Oyster_SH_GLMM_", ma_abrev, ".png"),
         plot_comb,
         width=8,
         height=4,
         units="in",
         dpi=200,
         bg="white")
  rm(liveplot_1, liveplot_2)
}

# Marginal effects plots for shell height (attempt to combine models into one plot)
meplotssh <- function(models1, data1, sizeclass1="", models2, data2,
                      sizeclass2="", managedarea, indicator, zoom=FALSE){
  ind <- case_when(str_detect(indicator, "ercent") ~ "Pct",
                   str_detect(indicator, "ensity") ~ "Den",
                   str_detect(indicator, "^S|^s") ~ "SH")
  ma <- managedarea

  sizeclass1 <- unique(data1$SizeClass)
  sizeclass2 <- unique(data2$SizeClass)
  # Set size labels
  if(sizeclass1 != ""){
    size1 <- case_when(
      str_detect(sizeclass1, "25") & str_detect(sizeclass1, "75") ~ "25to75",
      str_detect(sizeclass1, "35") & str_detect(sizeclass1, "75") ~ "35to75",
      str_detect(sizeclass1, "25")==FALSE & str_detect(sizeclass1, "75") ~ "o75",
      TRUE ~ "raw")
    sizelab1 <- case_when(
      str_detect(sizeclass1, "25") & str_detect(sizeclass1, "75") ~ "25-75mm",
      str_detect(sizeclass1, "35") & str_detect(sizeclass1, "75") ~ "35-75mm",
      str_detect(sizeclass1, "25")==FALSE & str_detect(sizeclass1, "75") ~ "\u2265 75mm",
      TRUE ~ "raw")
  }
  if(sizeclass2 != ""){
    size2 <- case_when(
      str_detect(sizeclass2, "25") & str_detect(sizeclass2, "75") ~ "25to75",
      str_detect(sizeclass2, "35") & str_detect(sizeclass2, "75") ~ "35to75",
      str_detect(sizeclass2, "25")==FALSE & str_detect(sizeclass2, "75") ~ "o75",
      TRUE ~ "raw")
    sizelab2 <- case_when(
```

```
    str_detect(sizeclass2, "25") & str_detect(sizeclass2, "75") ~ "25-75mm",
    str_detect(sizeclass2, "35") & str_detect(sizeclass2, "75") ~ "35-75mm",
    str_detect(sizeclass2, "25")==FALSE & str_detect(sizeclass2, "75") ~ "\u2265 75mm",
    TRUE ~ "raw")
}

#Marginal effects plot including random effects
## Hist plot settings
y_max <- round(max(data2[!is.na(ShellHeight_mm), ShellHeight_mm]), -0)+1
y_breaks <- seq(0, 300, 50)
y_labs <- seq(0, 300, 50)
y_minor <- seq(0, 300, 25)
ylim_upper <- ceiling(y_max/25)*25

yrdiff1 <- unique(data1$YearDiff)
yrdiff2 <- unique(data2$YearDiff)

# function to set year breaks, type == "hist" or "live"
set_breaks <- function(type, data1, data2){
  ldq <- ifelse(type=="hist", "Estimate", "Exact")

  maxyr <- max(data1[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate],
               data2[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate])
  minyr <- min(data1[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate],
               data2[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate])
  nyrs <- (maxyr)-(minyr)+1

  # Creates break intervals for plots based on number of years of data
  if(nyrs>=40){
    # Set breaks to every 10 years if more than 30 years of data
    brk <- 10
  }else if(nyrs<40 & nyrs>=20){
    # Set breaks to every 5 years if between 30 and 15 years of data
    brk <- 5
  }else if(nyrs<20 & nyrs>=12){
    # Set breaks to every 3 years if between 15 and 9 years of data
    brk <- 3
  }else if(nyrs<12 & nyrs>=8){
    # Set breaks to every 2 years if between 9 and 6 years of data
    brk <- 2
  }else if(nyrs<8 & nyrs>=3){
    # Set breaks to every year if less than 6 years of data
    brk <- 1
  }else if(nyrs<3){
    brk <- 1
    maxyr <- maxyr + 1
    minyr <- minyr - 1
  }
  return(list("seq" = seq(minyr,maxyr,brk),"maxyr" = maxyr, "minyr" = minyr))
}

yrlist_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["seq"]]
maxyr_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["maxyr"]]
```

```
minyr_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["minyr"]]
yrlist_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["seq"]]
maxyr_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["maxyr"]]
minyr_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["minyr"]]

## Check data for Exact and Estimate
n_hist1 <- nrow(data1[data1$LiveDate_Qualifier=="Estimate" &
                        !is.na(data1$ShellHeight_mm),])
n_live1 <- nrow(data1[data1$LiveDate_Qualifier=="Exact" &
                        !is.na(data1$ShellHeight_mm),])
n_hist2 <- nrow(data2[data2$LiveDate_Qualifier=="Estimate" &
                        !is.na(data2$ShellHeight_mm),])
n_live2 <- nrow(data2[data2$LiveDate_Qualifier=="Exact" &
                        !is.na(data2$ShellHeight_mm),])

set.seed(987)
if(!is.null(models1)){
  liveplot_1 <- plot(conditional_effects(models1[[1]], re_formula=NULL), plot=FALSE)
}

if(!is.null(models2)){
  liveplot_2 <- plot(conditional_effects(models2[[1]], re_formula=NULL), plot=FALSE)
}

# Set boolean values for whether liveplot1&2 are available
liveplot1_avail <- class(try(liveplot_1, silent=TRUE)) != "try-error"
liveplot2_avail <- class(try(liveplot_2, silent=TRUE)) != "try-error"

# Set ribbon transparency value
a_ribb <- 0.2
# Set size and shapes for plots
p_shape <- c("size2"=24, "size1"=21)
sizelab <- c("size2"=sizelab2, "size1"=sizelab1)

check <- NA
check1 <- NA
check2 <- NA

if(exists("present1")){
  check1 <- c("size1"="#00374f")
} else{
  check1 <- c("size1"="#FFFFFF")
}

if(exists("present2")){
  check2 <- c("size2"="#0094b0")
} else{
  check2 <- c("size2"="#FFFFFF")
}
p_color <- c(check2, check1)

# Initial plots to set legends
plot_leg <- ggplot() +
```

```
  {if(liveplot1_avail){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear+yrdiff1, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__,
                         fill="size1"), alpha=a_ribb),
         geom_line(data=liveplot_1$RelYear$data,
                   aes(x=RelYear+yrdiff1, y=estimate__, color="size1"),
                   lwd=0.75))
  }} +
  {if(liveplot2_avail){
    list(geom_ribbon(data=liveplot_2$RelYear$data,
                     aes(x=RelYear+yrdiff2, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__,
                         fill="size2"), alpha=a_ribb),
         geom_line(data=liveplot_2$RelYear$data,
                   aes(x=RelYear+yrdiff2, y=estimate__, color="size2"),
                   lwd=0.75))
  }} +
  geom_point(data=data1[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
             position=plot_jitter, size=2, color="#333333", fill="#cccccc",
             alpha=0.2, inherit.aes=FALSE) +
  geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
             position=plot_jitter, size=2, color="#333333", fill="#cccccc",
             alpha=0.2, inherit.aes=FALSE) +
  plot_theme +
  theme(legend.position="right") +
  scale_shape_manual(name="Shell heights",
                     values=p_shape,
                     labels=sizelab) +
  scale_color_manual(name="Shell heights",
                     values=p_color,
                     labels=sizelab) +
  scale_fill_manual(name="Shell heights",
                    values=p_color,
                    labels=sizelab)

leg <-get_legend(plot_leg)
rm(plot_leg)

# Dead oyster shell plot
plot1 <- ggplot() +
  geom_hline(yintercept=75, size=1, color="grey") +
  {if(n_hist1>0){
    geom_point(data=data1[!is.na(RelYear) &
                            !is.na(LiveDate) &
                            LiveDate_Qualifier=="Estimate", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  {if(n_hist2>0){
```

```
    geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
                          LiveDate_Qualifier=="Estimate", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  scale_x_continuous(limits=c(minyr_hist-0.25, maxyr_hist+0.25),
                     breaks=yrlist_hist) +
  scale_y_continuous(breaks=y_breaks,
                     labels=y_labs, minor_breaks=y_minor) +
  plot_theme +
  theme(plot.subtitle=element_text(hjust=0, size=10, color="#314963"),
        legend.position="none",
  ) +
  labs(subtitle="Dead Oyster Shells",
       x="Estimated year",
       y="Shell height (mm)") +
  scale_shape_manual(name="Shell heights",
                     values=c("size1"=21, "size2"=24),
                     labels=c(sizelab1, sizelab2)) +
  scale_color_manual(name="Shell heights",
                     values=c("size1"="#00374f", "size2"="#0094b0"),
                     labels=c(sizelab1, sizelab2)) +
  scale_fill_manual(name="Shell heights",
                    values=c("size1"="#00374f", "size2"="#0094b0"),
                    labels=c(sizelab1, sizelab2)) +
  coord_cartesian(ylim=c(0, ylim_upper))

# Live oyster shell plot
plot2 <- ggplot() +
  geom_hline(yintercept=75, size=1, color="grey") +
  {if(n_live1>0){
    geom_point(data=data1[!is.na(RelYear) & !is.na(LiveDate) &
                          LiveDate_Qualifier=="Exact", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  {if(n_live2>0){
    geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
                          LiveDate_Qualifier=="Exact", ],
               aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
               position=plot_jitter, size=2, color="#333333", fill="#cccccc",
               alpha=0.2, inherit.aes=FALSE)
  }} +
  {if(liveplot1_avail){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear+yrdiff1, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size1"),
                     alpha=a_ribb),
         geom_line(data=liveplot_1$RelYear$data,
                   aes(x=RelYear+yrdiff1, y=estimate__, color="size1"),
                   lwd=0.75))
```

```
  }} +
  {if(liveplot2_avail){
    list(geom_ribbon(data=liveplot_2$RelYear$data,
                     aes(x=RelYear+yrdiff2, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size2"),
                     alpha=a_ribb),
         geom_line(data=liveplot_2$RelYear$data,
                   aes(x=RelYear+yrdiff2, y=estimate__, color="size2"),
                   lwd=0.75))
  }} +
  scale_x_continuous(limits=c(minyr_live-0.25, maxyr_live+0.25),
                     breaks=yrlist_live) +
  scale_y_continuous(breaks=y_breaks,
                     labels=y_labs, minor_breaks=y_minor) +
  plot_theme +
  theme(plot.subtitle=element_text(hjust=0, size=10, color="#314963"),
        legend.position="none",
        axis.text.y=element_blank(),  #remove y-axis labels
        axis.ticks.y=element_blank(),  #remove y-axis ticks
        axis.title.y=element_blank()   #removes y-axis title
  ) +
  labs(subtitle="Live Oyster Shells",
       x="Year",
       y="Shell height (mm)") +
  scale_shape_manual(name="Shell heights",
                     values=c("size1"=21, "size2"=24),
                     labels=c(sizelab1, sizelab2)) +
  scale_color_manual(name="Shell heights",
                     values=c("size1"="#00374f", "size2"="#0094b0"),
                     labels=c(sizelab1, sizelab2)) +
  scale_fill_manual(name="Shell heights",
                    values=c("size1"="#00374f", "size2"="#0094b0"),
                    labels=c(sizelab1, sizelab2)) +
  coord_cartesian(ylim=c(0, ylim_upper))

# Set plot title
plot_title <- ggplot() +
  labs(title="Oyster Size Class", subtitle=ma) +
  plot_theme +
  theme(plot.subtitle=element_text(hjust=0.5, size=10, color="#314963"),
        panel.border=element_blank(),
        panel.grid.major=element_blank(),
        panel.grid.minor=element_blank(), axis.line=element_blank())

plot_comb <- ggarrange(plot1, plot2, leg, nrow=1,
                       widths=c(0.46, 0.39, 0.15))

plot_comb <- ggarrange(plot_title, plot_comb, ncol=1,
                       heights=c(0.125, 0.875))

ma_abrev <- MA_All[ManagedAreaName==ma, Abbreviation]

ggsave(paste0("output/Shell_Height/Figures/Oyster_SH_GLMM_", ma_abrev, ".png"),
```

```
        plot_comb,
        width=8,
        height=4,
        units="in",
        dpi=200,
        bg="white")
  rm(liveplot_1, liveplot_2)
}


# Create model results tables and save diagnostic plots
modresultssh <- function(datafile1, models1, datafile2, models2, indicator,
                         meplotzoom=FALSE){
  datafile1$SizeClass[datafile1$SizeClass=="25to75mm" &
                        datafile1$MA_plotlab==
                        "St. Martins Marsh Aquatic Preserve_Natural"] <-
    "35-75mm"
  sizeclass1 <- unique(datafile1$SizeClass)
  for(m in seq_along(models1)){
    modelobj <- models1[[m]]
    oyres_i <- setDT(broom.mixed::tidy(modelobj))
    #tidy() does not like that parameter values have underscores
    #for some reason, so the resulting table is incomplete

    if(nrow(oyres_i[effect=="fixed", ])-nrow(summary(modelobj)$fixed)==-1){
      missingrow <- data.table(effect="fixed",
                               component="cond",
                               #not sure what "cond" means in the tidy summary.
                               group=NA,
                               term=rownames(summary(modelobj)$fixed)[2],
                               estimate=summary(modelobj)$fixed$Estimate[2],
                               std.error=summary(modelobj)$fixed$Est.Error[2],
                               conf.low=summary(modelobj)$fixed$`l-95% CI`[2],
                               conf.high=summary(modelobj)$fixed$`u-95% CI`[2])
      oyres_i <- rbind(oyres_i, missingrow) %>% arrange(effect, group)
    }

    setDT(oyres_i)
    oyres_i[, `:=` (indicator=indicator,
                    managed_area=unique(datafile1$ManagedAreaName),
                    habitat_class=unique(datafile1$HabitatClassification),
                    size_class=sizeclass1,
                    live_date_qual=ifelse(
                      str_detect(
                        modelobj$file, "_hist"), "Estimate",
                      "Exact"),
                    n_programs=if(class(
                      try(datafile1$LiveDate_Qualifier))!="try-error"){
                      length(unique(
                        datafile1[LiveDate_Qualifier==
                                    ifelse(str_detect(
                                      modelobj$file, "_hist"),
                                      "Estimate", "Exact"),
```

```
                                ProgramID]))
                    } else{length(unique(datafile1[, ProgramID]))},
                    programs=if(class(try(
                      datafile1$LiveDate_Qualifier)) != "try-error"){
                      list(unique(
                        datafile1[LiveDate_Qualifier==
                                    ifelse(
                                      str_detect(
                                        modelobj$file,
                                        "_hist"),
                                      "Estimate",
                                      "Exact"),
                                  ProgramID]))
                    } else{list(unique(datafile1[, ProgramID]))},
                    filename=modelobj$file)]

  oysterresults <<- rbind(oysterresults, oyres_i)

  # Save diagnostic plots
  #diagnosticplots(modelobj, indicator,
  #unique(datafile$ManagedAreaName), sizeclass,
  #ifelse(str_detect(modelobj$file, "_hist"), TRUE, FALSE))
}

datafile2$SizeClass[datafile2$SizeClass=="25to75mm" &
                      datafile2$MA_plotlab==
                      "St. Martins Marsh Aquatic Preserve_Natural"] <- "35-75mm"
sizeclass2 <- unique(datafile2$SizeClass)

for(m in seq_along(models2)){
  modelobj <- models2[[m]]
  oyres_i <- setDT(broom.mixed::tidy(modelobj))
  #tidy() does not like that parameter values have underscores for
  #some reason, so the resulting table is incomplete

  if(nrow(oyres_i[effect=="fixed", ])-nrow(summary(modelobj)$fixed)==-1){
    missingrow <- data.table(effect="fixed",
                             component="cond",
                             #not sure what "cond" means in the tidy summary.
                             group=NA,
                             term=rownames(summary(modelobj)$fixed)[2],
                             estimate=summary(modelobj)$fixed$Estimate[2],
                             std.error=summary(modelobj)$fixed$Est.Error[2],
                             conf.low=summary(modelobj)$fixed$`l-95% CI`[2],
                             conf.high=summary(modelobj)$fixed$`u-95% CI`[2])
    oyres_i <- rbind(oyres_i, missingrow) %>% arrange(effect, group)
  }

  oyres_i <- oyres_i %>%
    mutate(
      indicator = indicator,
      managed_area = unique(datafile2$ManagedAreaName),
      habitat_class = unique(datafile2$HabitatClassification),
```

```
        size_class = sizeclass2,
        live_date_qual = if_else(
          str_detect(modelobj$file, "_hist"), "Estimate", "Exact"
        ),
        n_programs = if (class(try(datafile2$LiveDate_Qualifier)) != "try-error") {
          datafile2 %>%
            filter(LiveDate_Qualifier == if_else(str_detect(modelobj$file, "_hist"), "Estimate", "Exact
            pull(ProgramID) %>%
            unique() %>%
            length()
        } else {
          datafile2 %>%
            pull(ProgramID) %>%
            unique() %>%
            length()
        },
        programs = if (class(try(datafile2$LiveDate_Qualifier)) != "try-error") {
          list(datafile2 %>%
                 filter(LiveDate_Qualifier == if_else(str_detect(modelobj$file, "_hist"), "Estimate", "
                 pull(ProgramID) %>%
                 unique())
        } else {
          list(datafile2 %>% pull(ProgramID) %>% unique())
        },
        filename = modelobj$file
      )
    oysterresults <<- rbind(oysterresults, oyres_i)

    # Save diagnostic plots
    #diagnosticplots(modelobj, indicator,
    #unique(datafile$ManagedAreaName), sizeclass,
    #ifelse(str_detect(modelobj$file, "_hist"), TRUE, FALSE))
  }

  # Save marginal effects plots
  meplotssh(models1, datafile1, sizeclass1, models2, datafile2, sizeclass2,
            unique(datafile1$ManagedAreaName), indicator, meplotzoom)
}
```

Oyster Shell Height Analysis

Subsets data for that which is related to shell height. The data is further subset for each managed area and shell size class. Appropriate GLMMs are called from file for each shell size class, or are created if they don't exist. Data and models are then sent to the modresultssh function to create figures and save data.

```
# Parallel, num cores setup
ncores <- 4
nchains <- 4
threads <- 10
iter <- 3000
warmup <- 1000

# At least 5 years of data are required in order to run model analyses
```

```r
# Function checks N years of data, returns T or F
suff_years <- function(data){length(unique(data$Year))>=5}

#summarize shell height data
sh_all_sum <- summarySE(oysterraw[!is.na(ShellHeight_mm), ],
                        measurevar='ShellHeight_mm',
                        groupvars=c('ManagedAreaName', 'LiveDate_Qualifier',
                                    'LiveDate'))



## Apalachicola Bay Aquatic Preserve_Natural ----------------------------------------

#Exclude the five samples that don't have counts less than the "NumberMeasured"
#value for the corresponding program (see variable exploration graphs in the
#25to75mm section for the rationale and graphs for this step.)
numValves <- unique(oysterraw[, c("ProgramID", "RelYear", "counts",
                                  "QuadIdentifier", "Subtidal", "QuadSize_m2",
                                  "LiveDate_Qualifier", "NumberMeasured_n")])
exclude_samps <- subset(numValves, numValves$NumberMeasured_n=="20" &
                          numValves$counts > 19)$QuadIdentifier
ab_sho25 <- oysterraw[!is.na(ShellHeight_mm) &
                        ShellHeight_mm >= 25 &
                        MA_plotlab=="Apalachicola Bay Aquatic Preserve_Natural" &
                        QuadIdentifier %in% setdiff(
                          oysterraw[!is.na(ShellHeight_mm) &
                                      ManagedAreaName==
                                      "Apalachicola Bay Aquatic Preserve",
                                    QuadIdentifier], exclude_samps), ]

saveRDS(ab_sho25, paste0('output/model_results/data/ab_sho25_', Sys.Date(), '.rds'))


### abap-25 to 75mm ----------------------------------------------------------

ab_sh25to75 <- ab_sho25[ShellHeight_mm < 75, ]
saveRDS(ab_sh25to75, paste0('output/model_results/data/ab_sh25to75_', Sys.Date(), '.rds'))

if(suff_years(subset(ab_sh25to75, ab_sh25to75$LiveDate_Qualifier!="Estimate"))){
  ab_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                            RelYear+QuadSize_m2+(1 | UniversalReefID),
                          data=subset(ab_sh25to75, ab_sh25to75$LiveDate_Qualifier!="Estimate"),
                          family=gaussian, cores=ncores,
                          control=list(adapt_delta=0.995, max_treedepth=20),
                          iter=iter, warmup=warmup, chains=nchains, thin=3, seed=5699,
                          backend="rstan", threads=threading(threads),
                          file="output/model_results/GLMMs/ab_sh25to75_glmm4b.rds")

  models1 <- list(ab_sh25to75_glmm)
} else {models1 <- NULL}

# Create model results tables and save diagnostic plots
data1 <- ab_sh25to75
```

```
### ABAP->75mm ------------------------------------------------------------

ab_sho75 <- ab_sho25[ShellHeight_mm >= 75, ]

saveRDS(ab_sho75, paste0('output/model_results/data/ab_sho75_', Sys.Date(), '.rds'))

if(suff_years(subset(ab_sho75, ab_sho75$LiveDate_Qualifier!="Estimate"))){
  ab_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                         RelYear+(1 | UniversalReefID),
                       data=subset(ab_sho75,
                                   ab_sho75$LiveDate_Qualifier!="Estimate"),
                       family=gaussian, cores=ncores,
                       control= list(adapt_delta=0.995, max_treedepth=20),
                       iter=iter, warmup=warmup, chains=nchains, thin=3, seed=3639,
                       backend="rstan", threads=threading(threads),
                       file="output/model_results/GLMMs/ab_sho75_glmm4b.rds")

  models2 <- list(ab_sho75_glmm)
} else {models2 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- ab_sho75
modresultssh(data1, models1, data2, models2, "Size class", meplotzoom=FALSE)

## Apalachicola National Estuarine Research Reserve_Natural -------------------

an_sho25 <- oysterraw[!is.na(ShellHeight_mm) &
                        !is.na(LiveDate) &
                        ShellHeight_mm >= 25 &
                        MA_plotlab==
                        "Apalachicola National Estuarine Research Reserve_Natural" &
                        QuadIdentifier %in%
                        setdiff(oysterraw[!is.na(ShellHeight_mm) &
                                            ManagedAreaName==
                                            "Apalachicola National Estuarine Research Reserve",
                                          QuadIdentifier], exclude_samps), ]

saveRDS(an_sho25, paste0('output/model_results/data/an_sho25_', Sys.Date(), '.rds'))


### ANERR-25 to 75mm ------------------------------------------------------

an_sh25to75 <- subset(an_sho25, an_sho25$ShellHeight_mm < 75)

saveRDS(an_sh25to75, paste0('output/model_results/data/an_sh25to75_',
                            Sys.Date(), '.rds'))

if(suff_years(subset(an_sh25to75, an_sh25to75$LiveDate_Qualifier!="Estimate"))){
  an_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                            RelYear+QuadSize_m2+(1 | UniversalReefID),
                          data=subset(an_sh25to75, an_sh25to75$LiveDate_Qualifier!="Estimate"),
                          family=gaussian, cores=ncores,
```

```
                          control=list(adapt_delta=0.995, max_treedepth=20),
                          iter=iter, warmup=warmup, chains=nchains, thin=3, seed=5699,
                          backend="rstan", threads=threading(threads),
                          file="output/model_results/GLMMs/an_sh25to75_glmm4b.rds")

  models1 <- list(an_sh25to75_glmm)
} else {models1 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- an_sh25to75

### ANERR->75mm ------------------------------------------------------

an_sho75 <- an_sho25[ShellHeight_mm >= 75, ]

saveRDS(an_sho75, paste0('output/model_results/data/an_sho75_', Sys.Date(), '.rds'))

if(suff_years(subset(an_sho75, an_sho75$LiveDate_Qualifier!="Estimate"))){
  an_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                         RelYear+(1 | UniversalReefID),
                       data=subset(an_sho75, an_sho75$LiveDate_Qualifier!=
                                   "Estimate"), family=gaussian, cores=ncores,
                       control= list(adapt_delta=0.995, max_treedepth=20),
                       iter=iter, warmup=warmup, chains=nchains, thin=3, seed=3639,
                       backend="rstan", threads=threading(threads),
                       file="output/model_results/GLMMs/an_sho75_glmm4b.rds")
  models2 <- list(an_sho75_glmm)
} else {models2 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- an_sho75
#modresults(data, models, "Size class", meplotzoom=TRUE)
modresultssh(data1, models1, data2, models2, "Size class", meplotzoom=FALSE)

## Estero Bay Aquatic Preserve_Natural ---------------------------------------

eb_sho25 <- oysterraw[!is.na(ShellHeight_mm) &
                        ShellHeight_mm >= 25 &
                        MA_plotlab=="Estero Bay Aquatic Preserve_Natural", ]

saveRDS(eb_sho25, paste0('output/model_results/data/eb_sho25_', Sys.Date(), '.rds'))


### EBAP-25 to 75mm ------------------------------------------------------

eb_sh25to75 <- subset(eb_sho25, eb_sho25$ShellHeight_mm < 75)
eb_sh25to75 <- eb_sh25to75[!is.na(eb_sh25to75$QuadSize_m2),]

saveRDS(eb_sh25to75, paste0('output/model_results/data/eb_sh25to75_',
                            Sys.Date(), '.rds'))

# Does not update models because this subset of data have NA for QuadSize_m2 values
if(suff_years(subset(eb_sh25to75,eb_sh25to75$LiveDate_Qualifier=="Exact"))){
```

```
  eb_sh25to75_glmm <- brm(formula=ShellHeight_mm ~
                            RelYear+QuadSize_m2+(0+RelYear | UniversalReefID),
                          data=subset(eb_sh25to75,
                                      eb_sh25to75$LiveDate_Qualifier=="Exact"),
                          family=gaussian, cores=ncores,
                          control= list(adapt_delta=0.995, max_treedepth=20),
                          iter=iter, warmup=warmup, chains=nchains, thin=3, seed=6881,
                          backend="rstan", threads=threading(threads),
                          file="output/model_results/GLMMs/eb_sh25to75_glmm5.rds")

  models1 <- list(eb_sh25to75_glmm)
} else {models1 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- eb_sh25to75

### EBAP->75mm ------------------------------------------------------

eb_sho75 <- eb_sho25[ShellHeight_mm >= 75, ]

saveRDS(eb_sho75, paste0('output/model_results/data/eb_sho75_', Sys.Date(), '.rds'))

if(suff_years(subset(eb_sho75, eb_sho75$LiveDate_Qualifier=="Exact"))){
  eb_sho75_glmm <- brm(formula=ShellHeight_mm ~
                         RelYear+(1 | UniversalReefID),
                       data=subset(eb_sho75, eb_sho75$LiveDate_Qualifier=="Exact"),
                       family=gaussian, cores=ncores,
                       control=list(adapt_delta=0.995, max_treedepth=20), iter=iter,
                       warmup=warmup, chains=nchains, thin=3, seed=3138,
                       backend="rstan", threads=threading(threads),
                       file="output/model_results/GLMMs/eb_sho75_glmm4.rds")
  models2 <- list(eb_sho75_glmm)
} else {models2 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- eb_sho75

modresultssh(data1, models1, data2, models2, "Size class", meplotzoom=FALSE)

## Guana River Marsh Aquatic Preserve_Natural ------------------------------

grm_sho25 <- oysterraw[!is.na(ShellHeight_mm) &
                         ShellHeight_mm >= 25 &
                         MA_plotlab==
                         "Guana River Marsh Aquatic Preserve_Natural", ]

saveRDS(grm_sho25, paste0('output/model_results/data/grm_sho25_',
                          Sys.Date(), '.rds'))


### GRMAP-25 to 75mm ------------------------------------------------------

grm_sh25to75 <- subset(grm_sho25, grm_sho25$ShellHeight_mm < 75)
```

```
saveRDS(grm_sh25to75, paste0('output/model_results/data/grm_sh25to75_',
                             Sys.Date(), '.rds'))



if(suff_years(subset(grm_sh25to75, grm_sh25to75$LiveDate_Qualifier=="Exact"))){
  # NumberMeasred_n contains NA, removed from formula
  grm_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                             RelYear+(1 | UniversalReefID),
                           data=subset(grm_sh25to75,
                                       grm_sh25to75$LiveDate_Qualifier=="Exact"),
                           family=gaussian, cores=ncores,
                           control= list(adapt_delta=0.8, max_treedepth=10),
                           iter=iter, warmup=warmup, chains=nchains, thin=3,
                           seed=3457, backend="rstan", threads=threading(threads),
                           file="output/model_results/GLMMs/grm_sh25to75_glmm4.rds")

  models1 <- list(grm_sh25to75_glmm)
} else {models1 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- grm_sh25to75

### GRMAP->75mm ------------------------------------------------------

grm_sho75 <- grm_sho25[ShellHeight_mm >= 75, ]

saveRDS(grm_sho75, paste0('output/model_results/data/grm_sho75_',
                          Sys.Date(), '.rds'))

if(suff_years(subset(grm_sho75, grm_sho75$LiveDate_Qualifier=="Exact"))){
  grm_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                          RelYear+(1 | UniversalReefID),
                        data=subset(grm_sho75,
                                    grm_sho75$LiveDate_Qualifier=="Exact"),
                        family=gaussian, cores=ncores,
                        control= list(adapt_delta=0.8, max_treedepth=10),
                        iter=iter, warmup=warmup, chains=nchains, thin=3,
                        seed=4352, backend="rstan", threads=threading(threads),
                        file="output/model_results/GLMMs/grm_sho75_glmm4.rds")
  models2 <- list(grm_sho75_glmm)
} else {models2 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- grm_sho75
modresultssh(data1, models1, data2, models2, "Size class", meplotzoom=FALSE)

## Guana Tolomato Matanzas National Estuarine Research Reserve_Natural --------

gtmn_sho25 <- oysterraw[!is.na(ShellHeight_mm) &
                          ShellHeight_mm >= 25 &
                          MA_plotlab==
                          "Guana Tolomato Matanzas National Estuarine Research Reserve_Natural", ]
```

```
saveRDS(gtmn_sho25, paste0('output/model_results/data/gtmn_sho25_',
                           Sys.Date(), '.rds'))

### GTMNERR-25 to 75mm ----------------------------------------------------

gtmn_sh25to75 <- subset(gtmn_sho25, gtmn_sho25$ShellHeight_mm < 75)

saveRDS(gtmn_sh25to75, paste0('output/model_results/data/gtmn_sh25to75_',
                              Sys.Date(), '.rds'))

if(suff_years(subset(gtmn_sh25to75, gtmn_sh25to75$LiveDate_Qualifier != "Estimate"))){
  gtmn_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                              RelYear+(1 | UniversalReefID),
                            data=subset(gtmn_sh25to75,
                                        gtmn_sh25to75$LiveDate_Qualifier != "Estimate"),
                            family=gaussian, cores=ncores,
                            control=list(adapt_delta=0.8, max_treedepth=20),
                            iter=iter, warmup=warmup, chains=nchains, thin=3,
                            seed=7844, backend="rstan", threads=threading(threads),
                            file="output/model_results/GLMMs/gtmn_sh25to75_glmm5.rds")

  models1 <- list(gtmn_sh25to75_glmm)
} else {models1 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- gtmn_sh25to75

### GTMNERR->75mm ---------------------------------------------------------

gtmn_sho75 <- gtmn_sho25[ShellHeight_mm >= 75, ]

saveRDS(gtmn_sho75, paste0('output/model_results/data/gtmn_sho75_',
                           Sys.Date(), '.rds'))

if(suff_years(subset(gtmn_sho75, gtmn_sho75$LiveDate_Qualifier != "Estimate"))){
  gtmn_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                           RelYear + (1 | UniversalReefID),
                         data=subset(gtmn_sho75,
                                     gtmn_sho75$LiveDate_Qualifier != "Estimate"),
                         family=gaussian,
                         cores=ncores, control=list(adapt_delta=0.995, max_treedepth=20),
                         iter=iter, warmup=warmup, chains=nchains, thin=3,
                         seed=5332, backend="rstan", threads=threading(threads),
                         file="output/model_results/GLMMs/gtmn_sho75_glmm6.rds")
  models2 <- list(gtmn_sho75_glmm)
} else {models2 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- gtmn_sho75
modresultssh(data1, models1, data2, models2, "Size class", meplotzoom=FALSE)

## Indian River-Vero Beach to Ft. Pierce Aquatic Preserve_Natural -------------
irvbfp_sho25 <- oysterraw[!is.na(ShellHeight_mm) &
```

```
                            ShellHeight_mm >= 25 &
                            MA_plotlab==
                            "Indian River-Vero Beach to Ft. Pierce Aquatic Preserve_Natural", ]

saveRDS(irvbfp_sho25, paste0('output/model_results/data/irvbfp_sho25_',
                             Sys.Date(), '.rds'))


### IRVBFPAP-25 to 75mm ------------------------------------------------------

irvbfp_sh25to75 <- subset(irvbfp_sho25, irvbfp_sho25$ShellHeight_mm < 75)

saveRDS(irvbfp_sh25to75, paste0('output/model_results/data/irvbfp_sh25to75_',
                                Sys.Date(), '.rds'))

if(suff_years(subset(irvbfp_sh25to75, irvbfp_sh25to75$LiveDate_Qualifier != "Estimate"))){
  irvbfp_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                                RelYear+(1 | UniversalReefID),
                              data=subset(irvbfp_sh25to75,
                                          irvbfp_sh25to75$LiveDate_Qualifier != "Estimate"),
                              family=gaussian, cores=ncores,
                              control=list(adapt_delta=0.8, max_treedepth=20),
                              iter=iter, warmup=warmup, chains=nchains, thin=3,
                              seed=7844, backend="rstan", threads=threading(threads),
                              file="output/model_results/GLMMs/irvbfp_sh25to75_glmm.rds")

  models1 <- list(irvbfp_sh25to75_glmm)
} else {models1 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- irvbfp_sh25to75

### IRVBFPAP->75mm -----------------------------------------------------------

irvbfp_sho75 <- irvbfp_sho25[ShellHeight_mm >= 75, ]

saveRDS(irvbfp_sho75, paste0('output/model_results/data/irvbfp_sho75_',
                             Sys.Date(), '.rds'))

if(suff_years(subset(irvbfp_sho75, irvbfp_sho75$LiveDate_Qualifier != "Estimate"))){
  irvbfp_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                             RelYear + (1 | UniversalReefID),
                           data=subset(irvbfp_sho75,
                                       irvbfp_sho75$LiveDate_Qualifier != "Estimate"),
                           family=gaussian,
                           cores=ncores, control=list(adapt_delta=0.995, max_treedepth=20),
                           iter=iter, warmup=warmup, chains=nchains, thin=3,
                           seed=5332, backend="rstan", threads=threading(threads),
                           file="output/model_results/GLMMs/irvbfp_sho75_glmm.rds")
  models2 <- list(irvbfp_sho75_glmm)
} else {models2 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
```

```
data2 <- irvbfp_sho75
modresultssh(data1, models1, data2, models2, "Size class", meplotzoom=FALSE)

## Lemon Bay Aquatic Preserve_Natural ---------------------------------------

lb_sho25 <- oysterraw[!is.na(ShellHeight_mm) &
                        ShellHeight_mm >= 25 &
                        MA_plotlab=="Lemon Bay Aquatic Preserve_Natural", ]

saveRDS(lb_sho25, paste0('output/model_results/data/lb_sho25_',
                         Sys.Date(), '.rds'))

### LBAP-25 to 75mm ----------------------------------------------------------

lb_sh25to75 <- subset(lb_sho25, lb_sho25$ShellHeight_mm < 75)

saveRDS(lb_sh25to75, paste0('output/model_results/data/lb_sh25to75_', Sys.Date(), '.rds'))

if(suff_years(subset(lb_sh25to75, lb_sh25to75$LiveDate_Qualifier != "Estimate"))){
  lb_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                            RelYear+(1 | UniversalReefID),
                          data=subset(lb_sh25to75,
                                      lb_sh25to75$LiveDate_Qualifier != "Estimate"),
                          family=gaussian, cores=ncores,
                          control=list(adapt_delta=0.8, max_treedepth=20),
                          iter=iter, warmup=warmup, chains=nchains, thin=3,
                          seed=7844, backend="rstan", threads=threading(threads),
                          file="output/model_results/GLMMs/lb_sh25to75_glmm.rds")

  models1 <- list(lb_sh25to75_glmm)
} else {models1 <- NULL}

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- lb_sh25to75

### LBAP->75mm ---------------------------------------------------------------

lb_sho75 <- lb_sho25[ShellHeight_mm >= 75, ]

saveRDS(lb_sho75, paste0('output/model_results/data/lb_sho75_', Sys.Date(), '.rds'))

if(suff_years(subset(lb_sho75, lb_sho75$LiveDate_Qualifier != "Estimate"))){
  lb_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                            RelYear+(1 | UniversalReefID),
                          data=subset(lb_sho75,
                                      lb_sho75$LiveDate_Qualifier != "Estimate"),
                          family=gaussian, cores=ncores,
                          control=list(adapt_delta=0.8, max_treedepth=20),
                          iter=iter, warmup=warmup, chains=nchains, thin=3,
                          seed=7844, backend="rstan", threads=threading(threads),
                          file="output/model_results/GLMMs/lb_sho75_glmm.rds")

  models2 <- list(lb_sho75_glmm)
```

```
} else {models2 <- NULL}

#Important: note that time-averaging is not accounted for in the model fit for
#the data on shell height >75mm. The measurement error approach I was taking
#did not result in any models that converged, possibly because the combination
#of the data and degree of measurement error leads to multiple possible
#solutions. This means the model reported in this section makes the unrealistic
#assumption that the estimated sample ages are exactly correct.

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- lb_sho75
modresultssh(data1, models1, data2, models2, "Size class", meplotzoom=FALSE)
```

Density Analysis

Subsets data for that which is related to density. The data is further subset for each managed area. Appropriate GLMMs are called from file for each shell size class, or are created if they don't exist. Data and models are then sent to the modresults function to create figures and save data.

```
oysterraw$YearDiff <- oysterraw$LiveDate-oysterraw$RelYear
# #Make a collapsed version of the oysterraw table for density
oysterraw_den <- oysterraw[, c("ProgramID", "ProgramName", "LocationID",
                               "ProgramLocationID", "QuadIdentifier",
                               "ReefIdentifier", "LiveDate",
                               "LiveDate_Qualifier", "SampleDate", "Year",
                               "Month", "ManagedAreaName", "Region",
                               "SurveyMethod", "HabitatClassification",
                               "QuadSize_m2", "MADup", "Density_m2",
                               "Number_of_Oysters_Counted_Total_Count",
                               "Number_of_Oysters_Counted_Live_Count",
                               "Number_of_Oysters_Counted_Dead_Count",
                               "ObsIndex", "UniversalReefID",
                               "MA_plotlab", "Subtidal", "RelYear", "YearDiff")]
oysterraw_den[!is.na(Density_m2), DensIndex := ObsIndex]
oysterraw_den[!is.na(Number_of_Oysters_Counted_Total_Count), NTotIndex := ObsIndex]
oysterraw_den[!is.na(Number_of_Oysters_Counted_Live_Count), NLiveIndex := ObsIndex]
oysterraw_den[!is.na(Number_of_Oysters_Counted_Dead_Count), NDeadIndex := ObsIndex]
oysterraw_den[, ObsIndex := NULL]

oysterraw_den <- unique(oysterraw_den)
oysterraw_den <- oysterraw_den %>%
  dplyr::group_by(ProgramID, ProgramName, LocationID, ProgramLocationID,
                  QuadIdentifier, ReefIdentifier, LiveDate,
                  LiveDate_Qualifier, SampleDate, Year, Month,
                  ManagedAreaName, Region, SurveyMethod,
                  HabitatClassification, QuadSize_m2, MADup, UniversalReefID,
                  MA_plotlab, Subtidal) %>%
  tidyr::fill(Density_m2, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
              Number_of_Oysters_Counted_Dead_Count,
              DensIndex, NTotIndex, NLiveIndex, NDeadIndex) %>%
  tidyr::fill(Density_m2, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
```

```
              Number_of_Oysters_Counted_Dead_Count,
              DensIndex, NTotIndex, NLiveIndex, NDeadIndex,
              .direction='up') %>%
  dplyr::distinct()

oysterraw_den <- subset(oysterraw_den, !is.na(oysterraw_den$Density_m2) |
                          !is.na(oysterraw_den$Number_of_Oysters_Counted_Total_Count) |
                          !is.na(oysterraw_den$Number_of_Oysters_Counted_Live_Count) |
                          !is.na(oysterraw_den$Number_of_Oysters_Counted_Dead_Count) |
                          !is.na(oysterraw_den$DensIndex) |
                          !is.na(oysterraw_den$NTotIndex) |
                          !is.na(oysterraw_den$NLiveIndex) |
                          !is.na(oysterraw_den$NDeadIndex))
setDT(oysterraw_den)

#Remove NAs in Density_m2 column
oysterraw_den <- subset(oysterraw_den, !is.na(oysterraw_den$Density_m2))

#Summarize density data by managed area
den_all_sum <- summarySE(oysterraw_den, measurevar='Density_m2',
                         groupvars=c('ManagedAreaName', 'Year'))

## Raw density results ------------------------------------------------------


### Apalachicola Bay Aquatic Preserve_Natural ---------------------------------------

ab_n <- subset(oysterraw_den,
               oysterraw_den$MA_plotlab==
                 "Apalachicola Bay Aquatic Preserve_Natural")
ab_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(ab_n, paste0('output/model_results/data/ab_n_', Sys.Date(), '.rds'))

ab_den_glmm <- brm(formula=Density_m2 ~
                     RelYear+(0+RelYear | UniversalReefID), data=ab_n,
                   family=negbinomial, cores=ncores,
                   control= list(adapt_delta=0.995, max_treedepth=20),
                   iter=iter,
                   warmup=warmup, chains=nchains, init=0, thin=3, seed=5512,
                   backend="rstan", threads=threading(threads),
                   file="output/model_results/GLMMs/ab_den_glmm9.rds")

# ab_den_glmm <- update(ab_den_glmm,
#                       newdata = ab_n,
#                       family=negbinomial, cores=ncores,
#                       control= list(adapt_delta=0.995, max_treedepth=20),
#                       iter=iter,
#                       warmup=warmup, chains=nchains, init=0, thin=3, seed=5512,
#                       backend="rstan", threads=threading(threads),
#                       file="output/model_results/GLMMs/ab_den_glmm9.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- ab_n
```

```
models <- list(ab_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)


### Apalachicola National Estuarine Research Reserve_Natural ----------------------------------------

an_n <- subset(oysterraw_den,
               oysterraw_den$MA_plotlab==
                 "Apalachicola National Estuarine Research Reserve_Natural")
an_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(an_n, paste0('output/model_results/data/an_n_', Sys.Date(), '.rds'))

an_den_glmm <- brm(formula=Density_m2 ~
                     RelYear+Subtidal+(0+RelYear | UniversalReefID),
                   data=an_n, family=zero_inflated_negbinomial, cores=ncores,
                   control= list(adapt_delta=0.995, max_treedepth=20), iter=iter,
                   warmup=warmup, chains=nchains, init=0, thin=3, seed=4677,
                   backend="rstan", threads=threading(threads),
                   file="output/model_results/GLMMs/an_den_glmm11.rds")

# an_den_glmm <- update(an_den_glmm,
#                       newdata = an_n,
#                       family=zero_inflated_negbinomial, cores=ncores,
#                       control= list(adapt_delta=0.995, max_treedepth=20), iter=iter,
#                       warmup=warmup, chains=nchains, init=0, thin=3, seed=4677,
#                       backend="rstan", threads=threading(threads),
#                       file="output/model_results/GLMMs/an_den_glmm11.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- an_n
models <- list(an_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)

### Estero Bay Aquatic Preserve_Natural ----------------------------------------

eb_n <- subset(oysterraw_den,
               oysterraw_den$MA_plotlab=="Estero Bay Aquatic Preserve_Natural")
eb_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(eb_n, paste0('output/model_results/data/eb_n_', Sys.Date(), '.rds'))

eb_den_glmm <- brm(formula=Density_m2 ~
                     RelYear+(1 | UniversalReefID), data=eb_n,
                   family=zero_inflated_negbinomial, cores=ncores,
                   control= list(adapt_delta=0.995, max_treedepth=20), iter=iter,
                   warmup=warmup, chains=nchains, init=0, thin=3, seed=1298,
                   backend="rstan", threads=threading(threads),
                   file="output/model_results/GLMMs/eb_den_glmm10.rds")

# eb_den_glmm <- update(eb_den_glmm,
#                       newdata = eb_n,
#                       family=zero_inflated_negbinomial, cores=ncores,
#                       control= list(adapt_delta=0.995, max_treedepth=20), iter=iter,
#                       warmup=warmup, chains=nchains, init=0, thin=3, seed=1298,
```

```
#                       backend="rstan", threads=threading(threads),
#                       file="output/model_results/GLMMs/eb_den_glmm10.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- eb_n
models <- list(eb_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)

### Guana River Marsh Aquatic Preserve_Natural ---------------------------------------

grm_n <- subset(oysterraw_den,
                oysterraw_den$MA_plotlab==
                  "Guana River Marsh Aquatic Preserve_Natural")
grm_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(grm_n, paste0('output/model_results/data/grm_n_',
                      Sys.Date(), '.rds'))

grm_den_glmm <- brm(formula=Density_m2 ~
                      RelYear+(1 | UniversalReefID), data=grm_n,
                    family=zero_inflated_negbinomial, cores=2,
                    control= list(adapt_delta=0.995, max_treedepth=20),
                    iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                    seed=9875, backend="rstan", threads=threading(threads),
                    file="output/model_results/GLMMs/grm_den_glmm6.rds")

# grm_den_glmm <- update(grm_den_glmm,
#                        newdata = grm_n,
#                        family=zero_inflated_negbinomial, cores=2,
#                        control= list(adapt_delta=0.995, max_treedepth=20),
#                        iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                        seed=9875, backend="rstan", threads=threading(threads),
#                        file="output/model_results/GLMMs/grm_den_glmm6.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- grm_n
models <- list(grm_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)

### Guana Tolomato Matanzas National Estuarine Research Reserve_Natural ------------------------------

gtmn_n <- subset(oysterraw_den,
                 oysterraw_den$MA_plotlab==
                   "Guana Tolomato Matanzas National Estuarine Research Reserve_Natural")
gtmn_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(gtmn_n, paste0('output/model_results/data/gtmn_n_', Sys.Date(), '.rds'))

gtmn_den_glmm <- brm(formula=Density_m2 ~
                       RelYear+(1 | UniversalReefID),
                     data=gtmn_n, family=zero_inflated_negbinomial, cores=ncores,
                     control= list(adapt_delta=0.995, max_treedepth=20),
                     iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                     seed=3647, backend="rstan", threads=threading(threads),
                     file="output/model_results/GLMMs/gtmn_den_glmm18.rds")
```

```
# gtmn_den_glmm <- update(gtmn_den_glmm,
#                         newdata = gtmn_n,
#                         family=zero_inflated_negbinomial, cores=ncores,
#                         control= list(adapt_delta=0.995, max_treedepth=20),
#                         iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                         seed=3647, backend="rstan", threads=threading(threads),
#                         file="output/model_results/GLMMs/gtmn_den_glmm18.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- gtmn_n
models <- list(gtmn_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)

### Indian River-Vero Beach to Ft. Pierce Aquatic Preserve_Natural ----------------------------------

irvb_n <- subset(oysterraw_den,
                 oysterraw_den$MA_plotlab==
                   "Indian River-Vero Beach to Ft. Pierce Aquatic Preserve_Natural")
irvb_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(irvb_n, paste0('output/model_results/data/irvb_n_',
                       Sys.Date(), '.rds'))

irvb_den_glmm <- brm(formula=Density_m2 ~
                       RelYear+(0+RelYear | UniversalReefID),
                     data=irvb_n,
                     family=negbinomial, cores=ncores,
                     control= list(adapt_delta=0.995, max_treedepth=20),
                     iter=iter,
                     warmup=warmup, chains=nchains, init=0, thin=3, seed=5512,
                     backend="rstan", threads=threading(threads),
                     file="output/model_results/GLMMs/irvb_den_glmm9.rds")

# irvb_den_glmm <- update(irvb_den_glmm,
#                         newdata = irvb_n,
#                         family=negbinomial, cores=ncores,
#                         control= list(adapt_delta=0.995, max_treedepth=20),
#                         iter=iter,
#                         warmup=warmup, chains=nchains, init=0, thin=3, seed=5512,
#                         backend="rstan", threads=threading(threads),
#                         file="output/model_results/GLMMs/irvb_den_glmm9.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- irvb_n
models <- list(irvb_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)

### Jensen Beach to Jupiter Inlet Aquatic Preserve_Natural ----------------------------------------

jbji_n <- subset(oysterraw_den,
                 oysterraw_den$MA_plotlab==
                   "Jensen Beach to Jupiter Inlet Aquatic Preserve_Natural")
jbji_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(jbji_n, paste0('output/model_results/data/jbji_n_',
```

```
                        Sys.Date(), '.rds'))

jbji_den_glmm <- brm(formula=Density_m2 ~
                        RelYear+(0+RelYear | UniversalReefID),
                      data=jbji_n,
                      family=negbinomial, cores=ncores,
                      control= list(adapt_delta=0.995, max_treedepth=20),
                      iter=iter,
                      warmup=warmup, chains=nchains, init=0, thin=3, seed=5512,
                      backend="rstan", threads=threading(threads),
                      file="output/model_results/GLMMs/jbji_den_glmm9.rds")

# jbji_den_glmm <- update(irvb_den_glmm,
#                         newdata = jbji_n,
#                         family=negbinomial, cores=ncores,
#                         control= list(adapt_delta=0.995, max_treedepth=20),
#                         iter=iter,
#                         warmup=warmup, chains=nchains, init=0, thin=3, seed=5512,
#                         backend="rstan", threads=threading(threads),
#                         file="output/model_results/GLMMs/jbji_den_glmm9.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- jbji_n
models <- list(jbji_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)

### Lemon Bay Aquatic Preserve_Natural ----------------------------------------

lb_n <- subset(oysterraw_den,
               oysterraw_den$MA_plotlab=="Lemon Bay Aquatic Preserve_Natural")
lb_n[, Density_m2 := as.integer(round(Density_m2))]
saveRDS(lb_n, paste0('output/model_results/data/lb_n_', Sys.Date(), '.rds'))

lb_den_glmm <- brm(formula=Density_m2 ~
                      RelYear+(1 | ReefIdentifier), data=lb_n,
                    family=zero_inflated_negbinomial, cores=2,
                    control= list(adapt_delta=0.995, max_treedepth=20),
                    iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                    seed=4612, backend="rstan", threads=threading(threads),
                    file="output/model_results/GLMMs/lb_den_glmm6.rds")

# lb_den_glmm <- update(lb_den_glmm,
#                       newdata = lb_n,
#                       family=zero_inflated_negbinomial, cores=2,
#                       control= list(adapt_delta=0.995, max_treedepth=20),
#                       iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                       seed=4612, backend="rstan", threads=threading(threads),
#                       file="output/model_results/GLMMs/lb_den_glmm6.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- lb_n
models <- list(lb_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)
```

```
### Pine Island Sound Aquatic Preserve_Natural ----------------------------------------

oysterraw_den[str_detect(MA_plotlab, "Pine Island Sound"), `:=`
              (MA_plotlab=ifelse(str_detect(ProgramLocationID, "Reference") |
                                   str_detect(ProgramLocationID, "Control"),
                                 "Pine Island Sound Aquatic Preserve_Natural",
                                 "Pine Island Sound Aquatic Preserve_Restored"),
                HabitatClassification=ifelse(str_detect(ProgramLocationID,
                                                        "Reference") |
                                               str_detect(ProgramLocationID,
                                                          "Control"),
                                             "Natural", "Restored"))]
pis_n <- subset(oysterraw_den,
                oysterraw_den$MA_plotlab==
                  "Pine Island Sound Aquatic Preserve_Natural")
pis_n[, `:=` (Density_m2=as.integer(round(Density_m2)),
              Treatment=ifelse(UniversalReefID==170711,
                               "Reference", "Control"))]
saveRDS(pis_n, paste0('output/model_results/data/pis_n_', Sys.Date(), '.rds'))

pis_den_glmm <- brm(formula=Density_m2 ~
                      RelYear+(0+RelYear | UniversalReefID),
                    data=pis_n, family=zero_inflated_negbinomial, cores=ncores,
                    control= list(adapt_delta=0.995, max_treedepth=20),
                    iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                    seed=5243, backend="rstan", threads=threading(threads),
                    file="output/model_results/GLMMs/pis_den_glmm9.rds")

# pis_den_glmm <- update(pis_den_glmm,
#                        newdata = pis_n,
#                        family=zero_inflated_negbinomial, cores=ncores,
#                        control= list(adapt_delta=0.995, max_treedepth=20),
#                        iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                        seed=5243, backend="rstan", threads=threading(threads),
#                        file="output/model_results/GLMMs/pis_den_glmm9.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- pis_n
models <- list(pis_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)


### Pine Island Sound Aquatic Preserve_Restored ----------------------------------------

pisr_n <- subset(oysterraw_den,
                 oysterraw_den$MA_plotlab==
                   "Pine Island Sound Aquatic Preserve_Restored")
pisr_n[, `:=` (Density_m2=as.integer(round(Density_m2)),
               Treatment=ifelse(UniversalReefID==170711,
                                "Reference", "Control"))]
saveRDS(pisr_n, paste0('output/model_results/data/pisr_n_', Sys.Date(), '.rds'))

pisr_den_glmm <- brm(formula=Density_m2 ~
```

```
                        RelYear+QuadSize_m2, data=pisr_n,
                      family=zero_inflated_negbinomial,
                      prior=set_prior("uniform(0,5)", class="b", lb=0, ub=5),
                      cores=ncores, control= list(adapt_delta=0.995, max_treedepth=20),
                      iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                      seed=8441, backend="rstan", threads=threading(threads),
                      file="output/model_results/GLMMs/pisr_den_glmm12.rds")

# pisr_den_glmm <- update(pisr_den_glmm,
#                         newdata = pisr_n,
#                         family=zero_inflated_negbinomial,
#                         prior=set_prior("uniform(0,5)", class="b", lb=0, ub=5),
#                         cores=ncores, control= list(adapt_delta=0.995, max_treedepth=20),
#                         iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                         seed=8441, backend="rstan", threads=threading(threads),
#                         file="output/model_results/GLMMs/pisr_den_glmm12.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- pisr_n
models <- list(pisr_den_glmm)
modresults(data, models, "Density", meplotzoom=FALSE)

PI_R <- nrow(subset(oysterresults,
                    oysterresults$managed_area==
                      "Pine Island Sound Aquatic Preserve" &
                      oysterresults$indicator=="Density" &
                      oysterresults$habitat_class=="Restored"))

oysterresults$group[is.na(oysterresults$group)] <- NA

if(PI_R>0){
  oysterresults$group[oysterresults$managed_area==
                        "Pine Island Sound Aquatic Preserve" &
                        oysterresults$indicator=="Density" &
                        oysterresults$habitat_class=="Restored"] <-
    c(NA, NA, NA)

  oysterresults$term[oysterresults$managed_area==
                       "Pine Island Sound Aquatic Preserve" &
                       oysterresults$indicator=="Density" &
                       oysterresults$habitat_class=="Restored"] <-
    c("(Intercept)", "RelYear", "QuadSize_m2")
}
```

Percent Live Analysis

Subsets data for that which is related to percent live. The data is further subset for each managed area. Appropriate GLMMs are called from file for each shell size class, or are created if they don't exist. Data and models are then sent to the modresults function to create figures and save data.

```
#Make a collapsed version of the oysterraw table for percent live
oysterraw_pct <- oysterraw[, c("ProgramID", "ProgramName", "ProgramLocationID",
                               "QuadIdentifier", "ReefIdentifier", "LiveDate",
```

```
                              "LiveDate_Qualifier", "SampleDate", "Year",
                              "Month", "ManagedAreaName", "Region",
                              "SurveyMethod", "PercentLiveMethod",
                              "HabitatClassification", "QuadSize_m2", "MADup",
                              "PercentLive_pct",
                              "Number_of_Oysters_Counted_Total_Count",
                              "Number_of_Oysters_Counted_Live_Count",
                              "Number_of_Oysters_Counted_Dead_Count",
                              "ObsIndex", "UniversalReefID",
                              "MA_plotlab", "Subtidal", "RelYear", "YearDiff")]
oysterraw_pct[!is.na(PercentLive_pct), PctIndex := ObsIndex]
oysterraw_pct[!is.na(Number_of_Oysters_Counted_Total_Count),
              NTotIndex := ObsIndex]
oysterraw_pct[!is.na(Number_of_Oysters_Counted_Live_Count),
              NLiveIndex := ObsIndex]
oysterraw_pct[!is.na(Number_of_Oysters_Counted_Dead_Count),
              NDeadIndex := ObsIndex]
oysterraw_pct[, ObsIndex := NULL]

oysterraw_pct <- unique(oysterraw_pct)
oysterraw_pct <- oysterraw_pct %>%
  dplyr::group_by(ProgramID, ProgramName, ProgramLocationID, QuadIdentifier,
                  ReefIdentifier, LiveDate, LiveDate_Qualifier, SampleDate,
                  Year, Month, ManagedAreaName, Region, SurveyMethod,
                  PercentLiveMethod, HabitatClassification, QuadSize_m2,
                  MADup, UniversalReefID, MA_plotlab, Subtidal,
                  RelYear) %>%
  tidyr::fill(PercentLive_pct, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
              Number_of_Oysters_Counted_Dead_Count,
              PctIndex, NTotIndex, NLiveIndex, NDeadIndex) %>%
  tidyr::fill(PercentLive_pct, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
              Number_of_Oysters_Counted_Dead_Count,
              PctIndex, NTotIndex, NLiveIndex, NDeadIndex,
              .direction='up') %>%
  dplyr::distinct()

oysterraw_pct <- subset(oysterraw_pct, !is.na(oysterraw_pct$PercentLive_pct) |
                          !is.na(oysterraw_pct$Number_of_Oysters_Counted_Total_Count) |
                          !is.na(oysterraw_pct$Number_of_Oysters_Counted_Live_Count) |
                          !is.na(oysterraw_pct$Number_of_Oysters_Counted_Dead_Count) |
                          !is.na(oysterraw_pct$PctIndex) |
                          !is.na(oysterraw_pct$NTotIndex) |
                          !is.na(oysterraw_pct$NLiveIndex) |
                          !is.na(oysterraw_pct$NDeadIndex))
setDT(oysterraw_pct)

#Calculate PercentLive_pct values for some ProgramIDs where it is missing.
#Couldn't include at the start of the script because need to use the counts columns
#rather than the QuadSize_m2 column which is filled for the whole combined table.
oysterraw_pct[ProgramID==972 | ProgramID==4014 | ProgramID==4044,
              PercentLive_pct :=
```

```
                (Number_of_Oysters_Counted_Live_Count/
                   (Number_of_Oysters_Counted_Live_Count+
                      Number_of_Oysters_Counted_Dead_Count) * 100)]

#Filter NAs for PercentLive_pct (these are related to 1) programs that do
#counts to measure density, but do not estimate percent live and
#2) Programs that are listed as measuring percent live by a Point-intercept
#method, which cannot be calculated from counts.
oysterraw_pct <- oysterraw_pct[!is.na(PercentLive_pct), ]

#Add column of decimal versions of percent live values
oysterraw_pct[, PercentLive_dec := PercentLive_pct/100]

#Summarize percent live values
pct_all_sum <- summarySE(oysterraw_pct, measurevar='PercentLive_pct',
                         groupvars=c('ManagedAreaName', 'Year', 'PercentLiveMethod'))


## Apalachicola Bay Aquatic Preserve_Natural ----------------------------------------

abap_p <- subset(oysterraw_pct,
                 oysterraw_pct$MA_plotlab==
                   "Apalachicola Bay Aquatic Preserve_Natural")
saveRDS(abap_p, paste0('output/model_results/data/abap_p_', Sys.Date(), '.rds'))

abap_p_binom <- data.table(ProgramID=character(), ProgramLocationID=character(),
                           QuadIdentifier=character(), Year=integer(),
                           ManagedAreaName=character(),
                           PercentLiveMethod=character(),
                           UniversalReefID=factor(), Region=character(),
                           MA_plotlab=character(), RelYear=integer(),
                           PercentLive_pct=numeric(), LiveObs=logical())
for(i in 1:nrow(abap_p)){
  dat_i <- abap_p[i, c("ProgramID", "ProgramLocationID", "QuadIdentifier",
                       "Year", "ManagedAreaName", "PercentLiveMethod",
                       "UniversalReefID", "Region", "MA_plotlab", "RelYear",
                       "PercentLive_pct")]
  dat_l <- purrr::map_dfr(seq_len(round(dat_i$PercentLive_pct[1], digits=0)),
                          ~dat_i[, LiveObs := 1])
  dat_nl <- purrr::map_dfr(seq_len((100-round(dat_i$PercentLive_pct[1],
                                              digits=0))),
                           ~dat_i[, LiveObs := 0])
  dat <- rbind(dat_l, dat_nl)
  abap_p_binom <- rbind(abap_p_binom, dat)
}
saveRDS(abap_p_binom,
        paste0('output/model_results/data/abap_p_binom_',
               Sys.Date(), '.rds'))

abap_pct_glmm <- brm(formula=LiveObs ~ RelYear+(1 | UniversalReefID),
                     data=abap_p_binom, family=bernoulli, cores=ncores,
                     control= list(adapt_delta=0.995,
                                   max_treedepth=20),
```

```
                    iter=iter, warmup=warmup, chains=nchains, init=0,
                    thin=3,
                    seed=4331, backend="rstan",
                    threads=threading(threads),
                    file="output/model_results/GLMMs/abap_pct_glmm3.rds")

# abap_pct_glmm <- update(abap_pct_glmm,
#                         newdata = abap_p_binom,
#                         family=bernoulli, cores=ncores,
#                         control= list(adapt_delta=0.995,
#                                       max_treedepth=20),
#                         iter=iter, warmup=warmup, chains=nchains, init=0,thin=3,
#                         seed=4331, backend="rstan", threads=threading(threads),
#                         file="output/model_results/GLMMs/abap_pct_glmm3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- abap_p
models <- list(abap_pct_glmm)
modresults(data, models, "Percent live", meplotzoom=FALSE)

## Apalachicola National Estuarine Research Reserve_Natural ------------------------------------

anerr_p <- subset(oysterraw_pct,
                  oysterraw_pct$MA_plotlab==
                    "Apalachicola National Estuarine Research Reserve_Natural")
saveRDS(anerr_p, paste0('output/model_results/data/anerr_p_', Sys.Date(), '.rds'))

anerr_p_binom <- data.table(ProgramID=character(), ProgramLocationID=character(),
                            QuadIdentifier=character(), Year=integer(),
                            ManagedAreaName=character(),
                            PercentLiveMethod=character(),
                            UniversalReefID=factor(), Region=character(),
                            MA_plotlab=character(), RelYear=integer(),
                            PercentLive_pct=numeric(), LiveObs=logical())
for(i in 1:nrow(anerr_p)){
  dat_i <- anerr_p[i, c("ProgramID", "ProgramLocationID", "QuadIdentifier",
                        "Year", "ManagedAreaName", "PercentLiveMethod",
                        "UniversalReefID", "Region", "MA_plotlab", "RelYear",
                        "PercentLive_pct")]
  dat_l <- purrr::map_dfr(seq_len(round(dat_i$PercentLive_pct[1], digits=0)),
                          ~dat_i[, LiveObs := 1])
  dat_nl <- purrr::map_dfr(seq_len((100-round(dat_i$PercentLive_pct[1],
                                              digits=0))),
                           ~dat_i[, LiveObs := 0])
  dat <- rbind(dat_l, dat_nl)
  anerr_p_binom <- rbind(anerr_p_binom, dat)
}
saveRDS(anerr_p_binom,
        paste0('output/model_results/data/anerr_p_binom_',
               Sys.Date(), '.rds'))

anerr_pct_glmm <- brm(formula=LiveObs ~ RelYear+(1 | UniversalReefID),
                      data=anerr_p_binom, family=bernoulli, cores=ncores,
```

```
                      control= list(adapt_delta=0.995,
                                    max_treedepth=20),
                      iter=iter, warmup=warmup, chains=nchains, init=0,
                      thin=3,
                      seed=4331, backend="rstan",
                      threads=threading(threads),
                      file="output/model_results/GLMMs/anerr_pct_glmm3.rds")

# anerr_pct_glmm <- update(anerr_pct_glmm,
#                          newdata = anerr_p_binom,
#                          family=bernoulli, cores=ncores,
#                          control= list(adapt_delta=0.995,
#                                        max_treedepth=20),
#                          iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                          seed=4331, backend="rstan", threads=threading(threads),
#                          file="output/model_results/GLMMs/anerr_pct_glmm3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- anerr_p
models <- list(anerr_pct_glmm)
modresults(data, models, "Percent live", meplotzoom=FALSE)

## Guana River Marsh Aquatic Preserve_Natural ----------------------------------------

grm_p <- subset(oysterraw_pct,
                oysterraw_pct$MA_plotlab==
                  "Guana River Marsh Aquatic Preserve_Natural")
saveRDS(grm_p, paste0('output/model_results/data/grm_p_', Sys.Date(), '.rds'))

grm_p_binom <- data.table(ProgramID=character(), ProgramLocationID=character(),
                          QuadIdentifier=character(), Year=integer(),
                          ManagedAreaName=character(),
                          PercentLiveMethod=character(),
                          UniversalReefID=factor(), Region=character(),
                          MA_plotlab=character(), RelYear=integer(),
                          PercentLive_pct=numeric(), LiveObs=logical())
for(i in 1:nrow(grm_p)){
  dat_i <- grm_p[i, c("ProgramID", "ProgramLocationID", "QuadIdentifier",
                      "Year", "ManagedAreaName", "PercentLiveMethod",
                      "UniversalReefID", "Region", "MA_plotlab", "RelYear",
                      "PercentLive_pct")]
  dat_l <- purrr::map_dfr(seq_len(round(dat_i$PercentLive_pct[1], digits=0)),
                          ~dat_i[, LiveObs := 1])
  dat_nl <- purrr::map_dfr(seq_len((100-round(dat_i$PercentLive_pct[1],
                                              digits=0))),
                           ~dat_i[, LiveObs := 0])
  dat <- rbind(dat_l, dat_nl)
  grm_p_binom <- rbind(grm_p_binom, dat)
}
saveRDS(grm_p_binom, paste0('output/model_results/data/grm_p_binom_',
                            Sys.Date(), '.rds'))

grm_pct_glmm <- brm(formula=LiveObs ~ RelYear+(1 | UniversalReefID),
```

```
                    data=grm_p_binom, family=bernoulli, cores=ncores,
                    control= list(adapt_delta=0.995, max_treedepth=20),
                    iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                    seed=4331, backend="rstan", threads=threading(threads),
                    file="output/model_results/GLMMs/grm_pct_glmm3.rds")

# grm_pct_glmm <- update(grm_pct_glmm,
#                        newdata = grm_p_binom,
#                        family=bernoulli, cores=ncores,
#                        control= list(adapt_delta=0.995, max_treedepth=20),
#                        iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                        seed=4331, backend="rstan", threads=threading(threads),
#                        file="output/model_results/GLMMs/grm_pct_glmm3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- grm_p
models <- list(grm_pct_glmm)
modresults(data, models, "Percent live", meplotzoom=FALSE)

## Guana Tolomato Matanzas National Estuarine Research
## Reserve_Natural ---------------------------------------

gtm_p <- subset(oysterraw_pct,
                oysterraw_pct$MA_plotlab==
                  "Guana Tolomato Matanzas National Estuarine Research Reserve_Natural")
saveRDS(gtm_p, paste0('output/model_results/data/gtm_p_', Sys.Date(), '.rds'))

gtm_p_binom <- data.table(ProgramID=character(), ProgramLocationID=character(),
                          QuadIdentifier=character(), Year=integer(),
                          ManagedAreaName=character(),
                          PercentLiveMethod=character(),
                          UniversalReefID=factor(), Region=character(),
                          MA_plotlab=character(), RelYear=integer(),
                          PercentLive_pct=numeric(), LiveObs=logical())
for(i in 1:nrow(gtm_p)){
  dat_i <- gtm_p[i, c("ProgramID", "ProgramLocationID", "QuadIdentifier",
                      "Year", "ManagedAreaName", "PercentLiveMethod",
                      "UniversalReefID", "Region", "MA_plotlab", "RelYear",
                      "PercentLive_pct")]
  dat_l <- purrr::map_dfr(seq_len(round(dat_i$PercentLive_pct[1], digits=0)),
                          ~dat_i[, LiveObs := 1])
  dat_nl <- purrr::map_dfr(seq_len((100-round(dat_i$PercentLive_pct[1],
                                              digits=0))),
                           ~dat_i[, LiveObs := 0])
  dat <- rbind(dat_l, dat_nl)
  gtm_p_binom <- rbind(gtm_p_binom, dat)
}
saveRDS(gtm_p_binom,
        paste0('output/model_results/data/gtm_p_binom_',
               Sys.Date(), '.rds'))

gtm_pct_glmm <- brm(formula=LiveObs ~ RelYear+(1 | UniversalReefID),
                    data=gtm_p_binom, family=bernoulli, cores=ncores,
```

```
                    control= list(adapt_delta=0.995,
                                  max_treedepth=20),
                    iter=iter, warmup=warmup, chains=nchains, init=0,
                    thin=3,
                    seed=4331, backend="rstan",
                    threads=threading(threads),
                    file="output/model_results/GLMMs/gtm_pct_glmm3.rds")

# gtm_pct_glmm <- update(gtm_pct_glmm,
#                        newdata = gtm_p_binom,
#                        family=bernoulli, cores=ncores,
#                        control= list(adapt_delta=0.995,
#                                      max_treedepth=20),
#                        iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                        seed=4331, backend="rstan", threads=threading(threads),
#                        file="output/model_results/GLMMs/gtm_pct_glmm3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- gtm_p
models <- list(gtm_pct_glmm)
modresults(data, models, "Percent live", meplotzoom=FALSE)


## Indian River-Vero Beach to Ft. Pierce Aquatic Preserve_Natural ------------------------------------

irvb_p <- subset(oysterraw_pct,
                 oysterraw_pct$MA_plotlab==
                   "Indian River-Vero Beach to Ft. Pierce Aquatic Preserve_Natural")
saveRDS(irvb_p, paste0('output/model_results/data/irvb_p_', Sys.Date(), '.rds'))

irvb_p_binom <- data.table(ProgramID=character(), ProgramLocationID=character(),
                           QuadIdentifier=character(), Year=integer(),
                           ManagedAreaName=character(),
                           PercentLiveMethod=character(),
                           UniversalReefID=factor(), Region=character(),
                           MA_plotlab=character(), RelYear=integer(),
                           PercentLive_pct=numeric(), LiveObs=logical())
for(i in 1:nrow(irvb_p)){
  dat_i <- irvb_p[i, c("ProgramID", "ProgramLocationID", "QuadIdentifier",
                       "Year", "ManagedAreaName", "PercentLiveMethod",
                       "UniversalReefID", "Region", "MA_plotlab", "RelYear",
                       "PercentLive_pct")]
  dat_l <- purrr::map_dfr(seq_len(round(dat_i$PercentLive_pct[1], digits=0)),
                          ~dat_i[, LiveObs := 1])
  dat_nl <- purrr::map_dfr(seq_len((100-round(dat_i$PercentLive_pct[1],
                                              digits=0))),
                           ~dat_i[, LiveObs := 0])
  dat <- rbind(dat_l, dat_nl)
  irvb_p_binom <- rbind(irvb_p_binom, dat)
}
saveRDS(irvb_p_binom, paste0('output/model_results/data/irvb_p_binom_',
                             Sys.Date(), '.rds'))
```

```
irvb_pct_glmm <- brm(formula=LiveObs ~ RelYear+(1 | UniversalReefID),
                    data=irvb_p_binom, family=bernoulli, cores=ncores,
                    control= list(adapt_delta=0.995, max_treedepth=20),
                    iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                    seed=4331, backend="rstan", threads=threading(threads),
                    file="output/model_results/GLMMs/irvb_pct_glmm3.rds")

# irvb_pct_glmm <- update(irvb_pct_glmm,
#                         newdata = irvb_p_binom,
#                         family=bernoulli, cores=ncores,
#                         control= list(adapt_delta=0.995, max_treedepth=20),
#                         iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                         seed=4331, backend="rstan", threads=threading(threads),
#                         file="output/model_results/GLMMs/irvb_pct_glmm3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- irvb_p
models <- list(irvb_pct_glmm)
modresults(data, models, "Percent live", meplotzoom=FALSE)


## Jensen Beach to Jupiter Inlet Aquatic Preserve_Natural ----------------------------------------

jbji_p <- subset(oysterraw_pct,
                 oysterraw_pct$MA_plotlab==
                   "Jensen Beach to Jupiter Inlet Aquatic Preserve_Natural")
saveRDS(jbji_p, paste0('output/model_results/data/jbji_p_', Sys.Date(), '.rds'))

jbji_p_binom <- data.table(ProgramID=character(), ProgramLocationID=character(),
                           QuadIdentifier=character(), Year=integer(),
                           ManagedAreaName=character(),
                           PercentLiveMethod=character(),
                           UniversalReefID=factor(), Region=character(),
                           MA_plotlab=character(), RelYear=integer(),
                           PercentLive_pct=numeric(), LiveObs=logical())
for(i in 1:nrow(jbji_p)){
  dat_i <- jbji_p[i, c("ProgramID", "ProgramLocationID", "QuadIdentifier",
                       "Year", "ManagedAreaName", "PercentLiveMethod",
                       "UniversalReefID", "Region", "MA_plotlab", "RelYear",
                       "PercentLive_pct")]
  dat_l <- purrr::map_dfr(seq_len(round(dat_i$PercentLive_pct[1], digits=0)),
                          ~dat_i[, LiveObs := 1])
  dat_nl <- purrr::map_dfr(seq_len((100-round(dat_i$PercentLive_pct[1],
                                              digits=0))),
                           ~dat_i[, LiveObs := 0])
  dat <- rbind(dat_l, dat_nl)
  jbji_p_binom <- rbind(jbji_p_binom, dat)
}
saveRDS(jbji_p_binom, paste0('output/model_results/data/jbji_p_binom_',
                             Sys.Date(), '.rds'))

jbji_pct_glmm <- brm(formula=LiveObs ~ RelYear+(1 | UniversalReefID),
                     data=jbji_p_binom, family=bernoulli, cores=ncores,
```

```
                      control= list(adapt_delta=0.995, max_treedepth=20),
                      iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
                      seed=4331, backend="rstan", threads=threading(threads),
                      file="output/model_results/GLMMs/jbji_pct_glmm3.rds")

# jbji_pct_glmm <- update(jbji_pct_glmm,
#                         newdata = jbji_p_binom,
#                         family=bernoulli, cores=ncores,
#                         control= list(adapt_delta=0.995, max_treedepth=20),
#                         iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
#                         seed=4331, backend="rstan", threads=threading(threads),
#                         file="output/model_results/GLMMs/jbji_pct_glmm3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- jbji_p
models <- list(jbji_pct_glmm)
modresults(data, models, "Percent live", meplotzoom=FALSE)

## Lemon Bay Aquatic Preserve_Natural ----------------------------------------

lb_p <- subset(oysterraw_pct,
               oysterraw_pct$MA_plotlab=="Lemon Bay Aquatic Preserve_Natural")
lb_p[, PercentLive_dec := PercentLive_pct/100]
#PercentLiveMethod=="Percent" for Lemon Bay program(s) with sufficient data,
#so cannot be modeled as binomial
saveRDS(lb_p, paste0('output/model_results/data/lb_p_', Sys.Date(), '.rds'))

lb_pct_glmm <- brm(formula=PercentLive_dec ~
                     RelYear+(0+RelYear | ReefIdentifier),
                   data=subset(lb_p, lb_p$PercentLive_dec > 0 & lb_p$PercentLive_dec < 1),family=Beta,
                   cores=ncores, control= list(adapt_delta=0.995, max_treedepth=20),
                   iter=iter, warmup=warmup, chains=nchains, init=0, thin=3, seed=8465,
                   backend="rstan", threads=threading(threads),
                   file="output/model_results/GLMMs/lb_pct_glmm6.rds")

# lb_pct_glmm <- update(lb_pct_glmm,
#                       newdata = subset(lb_p, lb_p$PercentLive_dec > 0),
#                       family=Beta,
#                       cores=ncores, control= list(adapt_delta=0.995, max_treedepth=20),
#                       iter=iter, warmup=warmup, chains=nchains, init=0, thin=3, seed=8465,
#                       backend="rstan", threads=threading(threads),
#                       file="output/model_results/GLMMs/lb_pct_glmm6.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data <- lb_p
models <- list(lb_pct_glmm)
modresults(data, models, "Percent live", meplotzoom=FALSE)
```

Save & Export Results

The compiled model results variable is saved to a csv and rds file. The model results are evaluated to see if any models need to be reconsidered based on the rhat convergence being over 1.05. rhat values are written to file.

```
fwrite(oysterresults, paste0("output/GLMM_AllDates_ModelResults.csv"), sep=",")
saveRDS(oysterresults, paste0("output/GLMM_AllDates_ModelResults.rds"))

#Get Rhat values for all models to check which ones may need to be reparameterized
model_list <- unique(oysterresults$filename)
rhats_all <- data.table(filename=character(),
                        term=character(),
                        rhat=numeric())
rhats_sum <- data.table(filename=character(),
                        rhat=numeric())

for(mod in model_list){
  mod_i <- readRDS(mod)
  allrhat_i <- rhat(mod_i)
  sumrhat_i <- c(summary(mod_i)$fixed$Rhat, summary(mod_i)$spec_pars$Rhat)
  allrhat_model_i <- data.table(filename=mod,
                                term=names(allrhat_i),
                                rhat=allrhat_i)
  sumrhat_model_i <- data.table(filename=mod,
                                rhat=sumrhat_i)
  rhats_all <- rbind(rhats_all, allrhat_model_i)
  rhats_sum <- rbind(rhats_sum, sumrhat_model_i)
}

rhats_all[, rhat_r := round(rhat, 2)]
rhats_sum[, rhat_r := round(rhat, 2)]

saveRDS(rhats_all, paste0("output/rhats_all_", Sys.Date(), ".rds"))
saveRDS(rhats_sum, paste0("output/rhats_sum_", Sys.Date(), ".rds"))

models_to_check_allrhat <- unique(rhats_all[rhat_r > 1.05, filename])
models_to_check_sumrhat <- unique(rhats_sum[rhat_r > 1.05, filename])

# Zip all figures
for(p in c("Density", "Percent_Live", "Shell_Height")){
  out_dir <- paste0("output/", p)
  fig_list <- list.files(paste0(out_dir, "/Figures"), pattern=".png", full=FALSE)
  filename <- paste0("Oyster", gsub("_", "", p), "Figures")
  setwd(paste0(out_dir, "/Figures"))
  zip(filename, files=fig_list)
  setwd(wd)
}
```

Oyster Figures

Density

Apalachicola Bay Aquatic Preserve

Natural

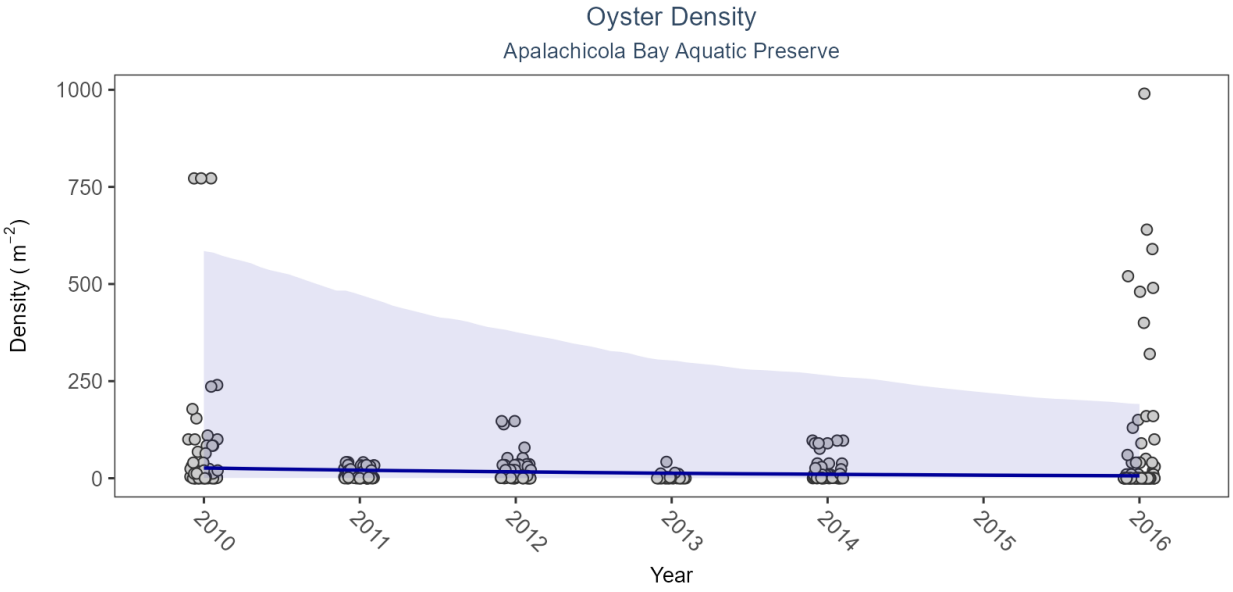
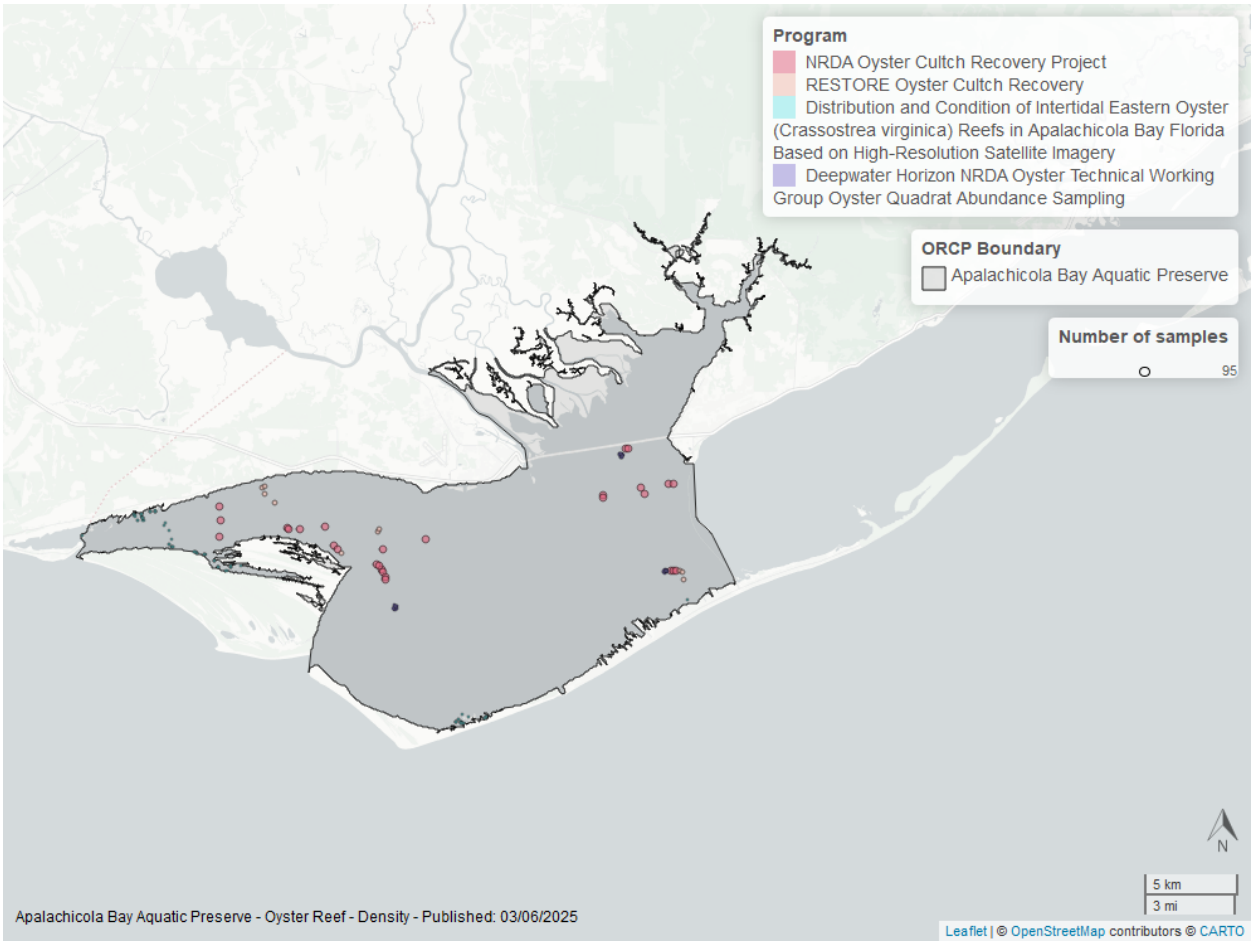


Figure 1: Figure for Oyster Density in Apalachicola Bay Aquatic Preserve

Table 1: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.24	0.11	-0.47 to -0.02


Program
NRDA Oyster Cultch Recovery Project
RESTORE Oyster Cultch Recovery
Distribution and Condition of Intertidal Eastern Oyster (Crassostrea virginica) Reefs in Apalachicola Bay Florida Based on High-Resolution Satellite Imagery
Deepwater Horizon NRDA Oyster Technical Working Group Oyster Quadrat Abundance Sampling
ORCP Boundary
Apalachicola Bay Aquatic Preserve
Number of samples
95
N
5 km
3 mi
Apalachicola Bay Aquatic Preserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Apalachicola National Estuarine Research Reserve

Natural

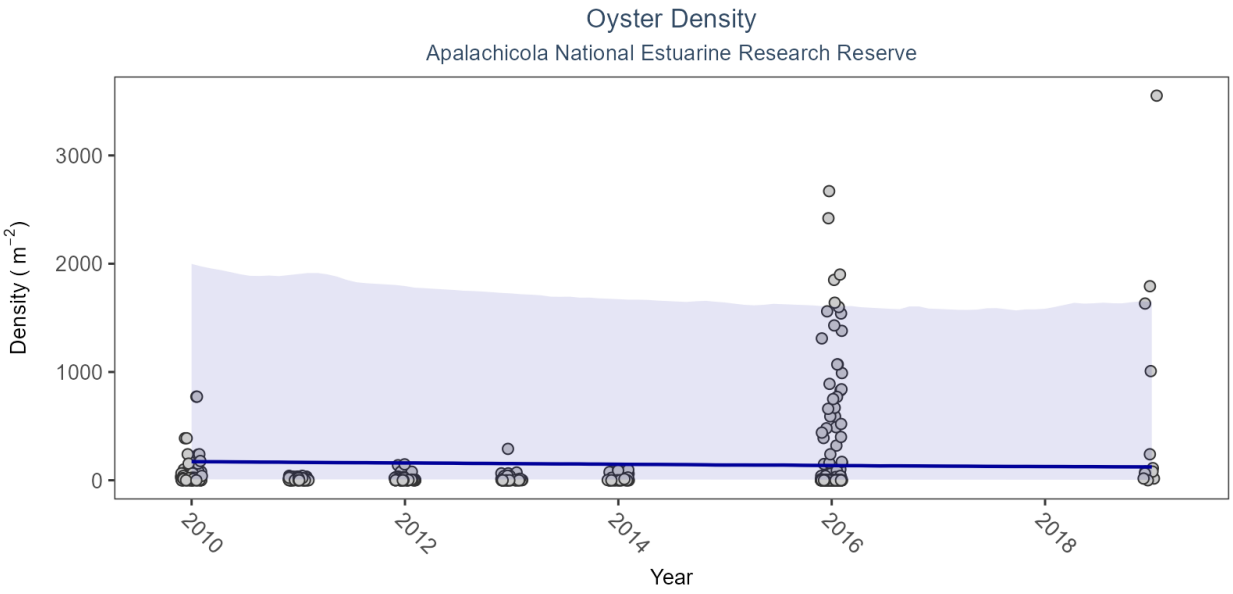
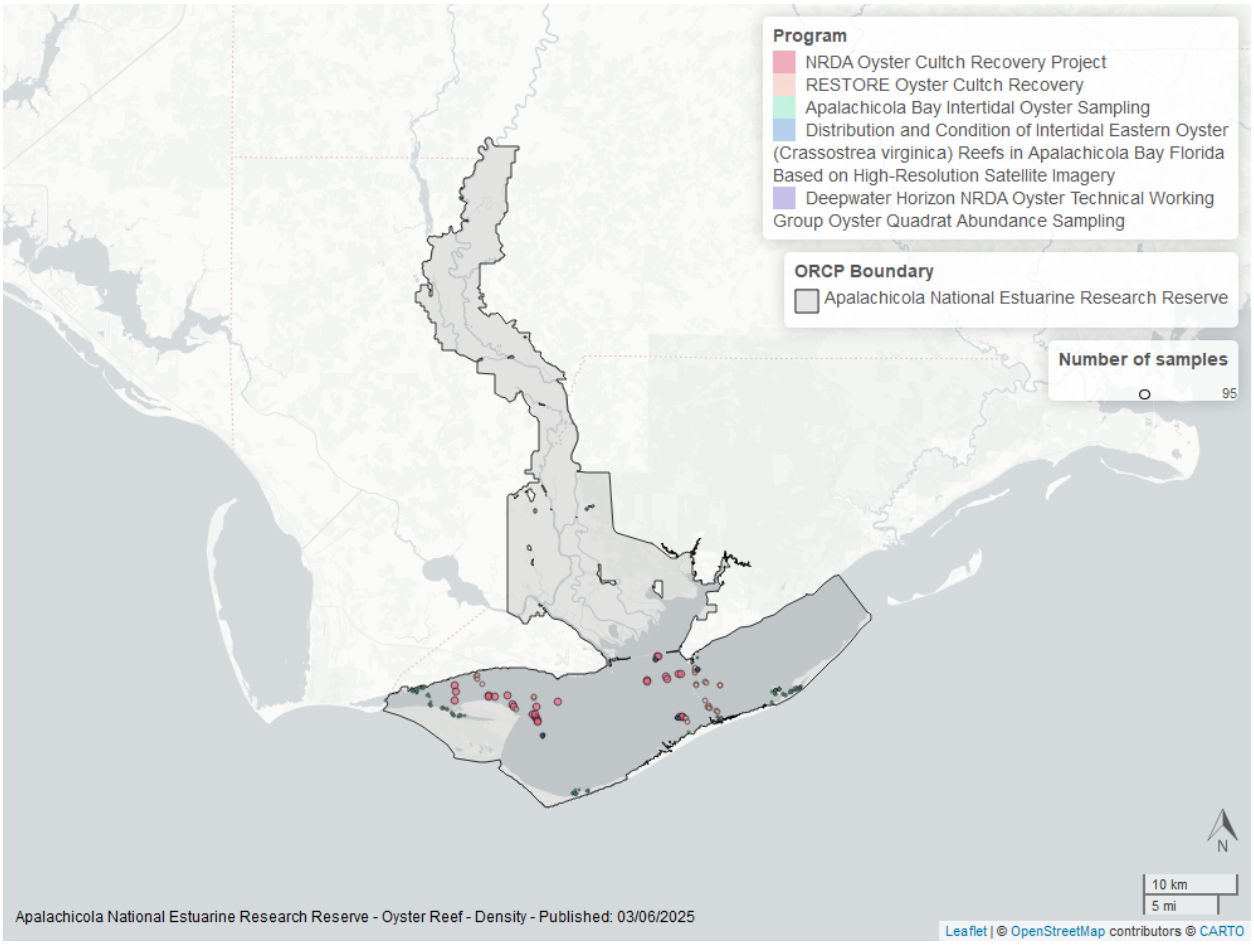


Figure 2: Figure for Oyster Density in Apalachicola National Estuarine Research Reserve

Table 2: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	No significant change	-0.04	0.1	-0.23 to 0.16


Program
NRDA Oyster Cultch Recovery Project
RESTORE Oyster Cultch Recovery
Apalachicola Bay Intertidal Oyster Sampling
Distribution and Condition of Intertidal Eastern Oyster (Crassostrea virginica) Reefs in Apalachicola Bay Florida Based on High-Resolution Satellite Imagery
Deepwater Horizon NRDA Oyster Technical Working Group Oyster Quadrat Abundance Sampling
ORCP Boundary
Apalachicola National Estuarine Research Reserve
Number of samples
95
N
10 km
5 mi
Apalachicola National Estuarine Research Reserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Estero Bay Aquatic Preserve

Natural

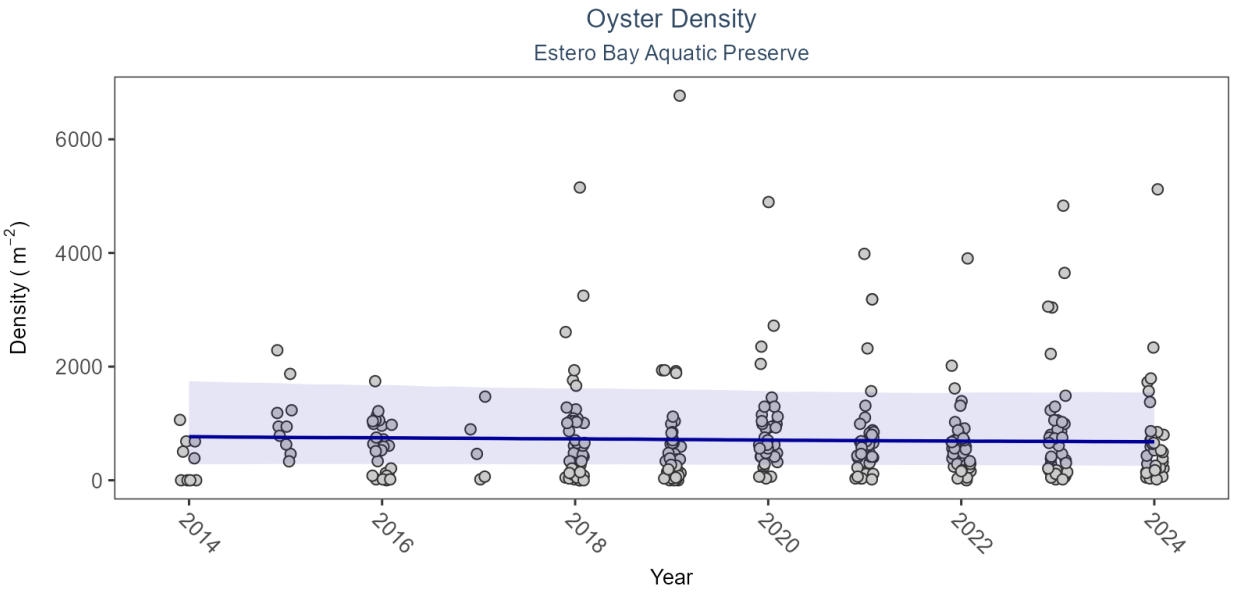
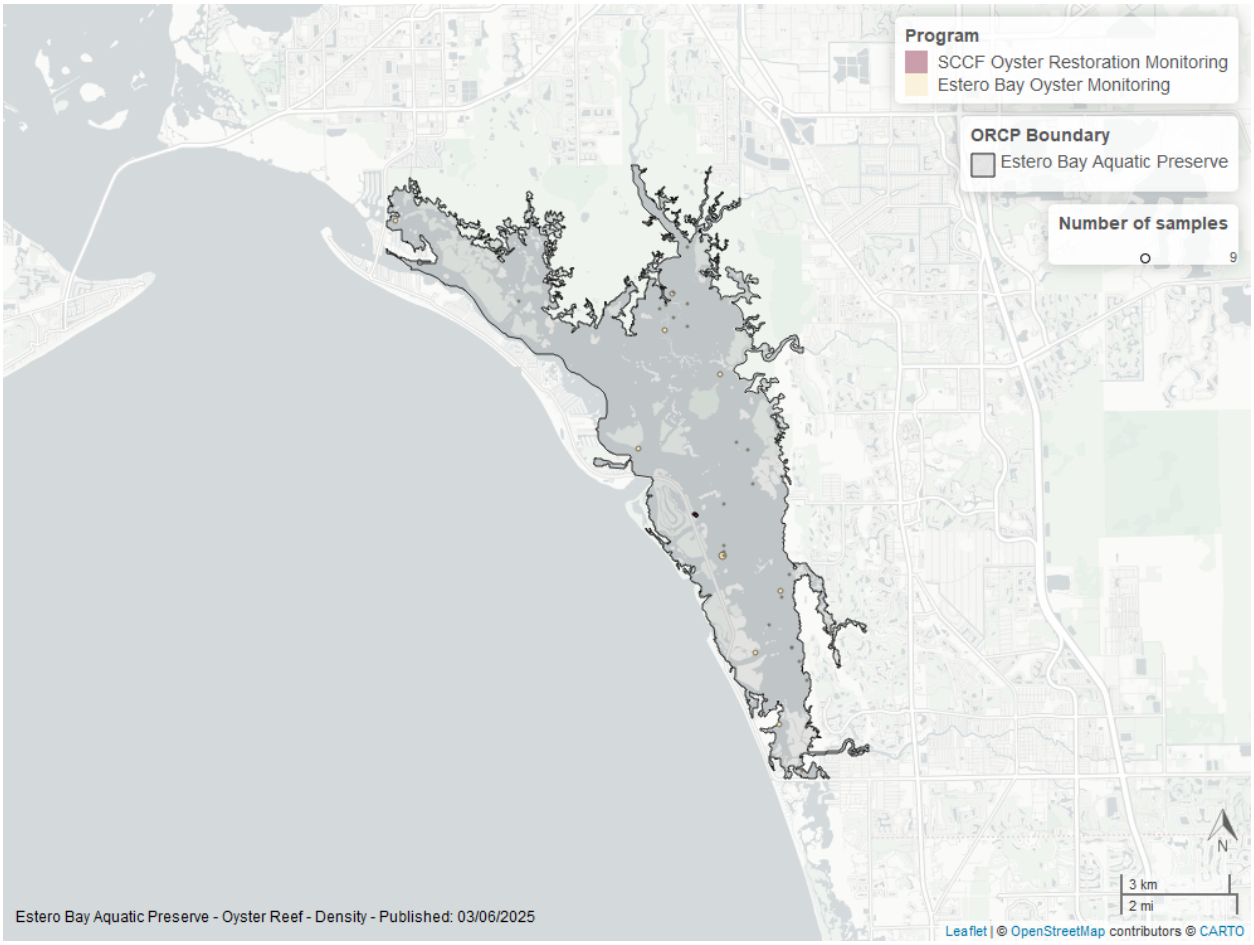


Figure 3: Figure for Oyster Density in Estero Bay Aquatic Preserve

Table 3: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	No significant change	-0.01	0.02	-0.06 to 0.03


Program
SCCF Oyster Restoration Monitoring
Estero Bay Oyster Monitoring
ORCP Boundary
Estero Bay Aquatic Preserve
Number of samples
9
N
3 km
2 mi
Estero Bay Aquatic Preserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Guana River Marsh Aquatic Preserve

Natural

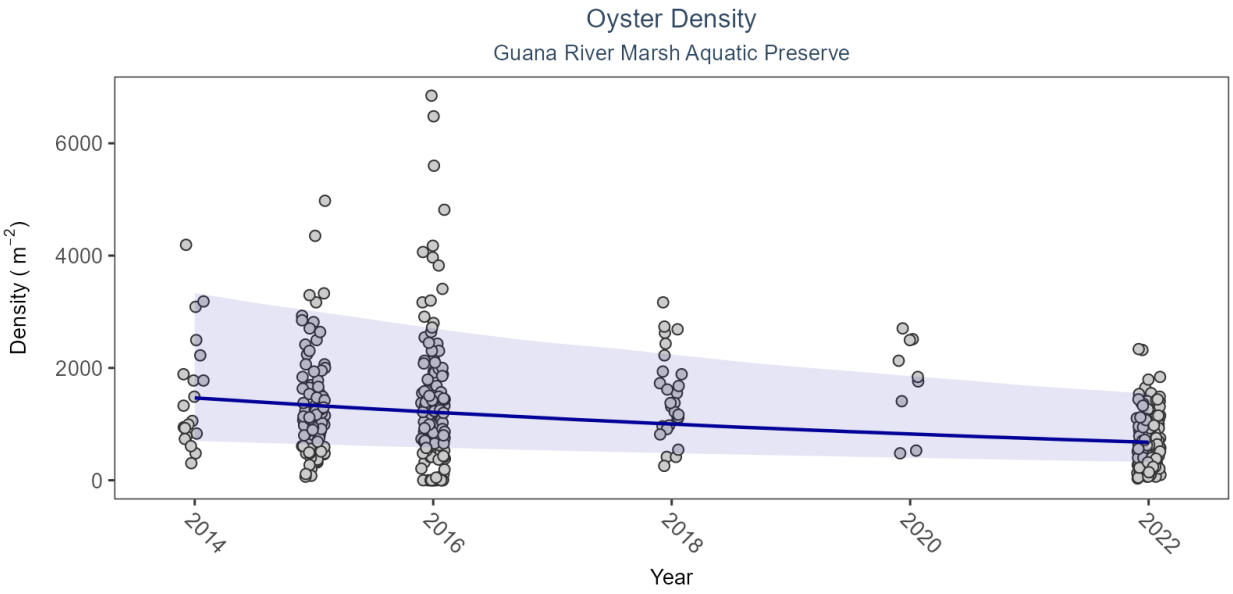


Figure 4: Figure for Oyster Density in Guana River Marsh Aquatic Preserve

Table 4: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.1	0.01	-0.12 to -0.07


Program
Guana Tolomato Matanzas NERR Oyster monitoring
Northern Coastal Basins and Mosquito Lagoon Oyster Condition Assessment
ORCP Boundary
Guana River Marsh Aquatic Preserve
Number of samples
18
N
3 km
2 mi
Guana River Marsh Aquatic Preserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Guana Tolomato Matanzas National Estuarine Research Reserve

Natural

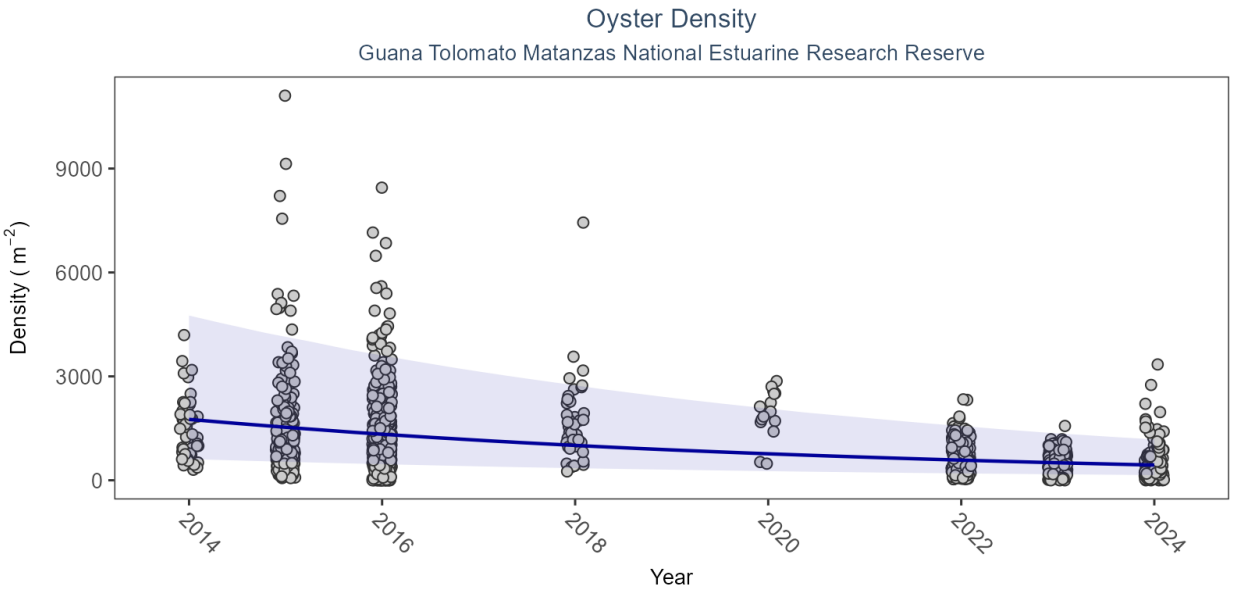
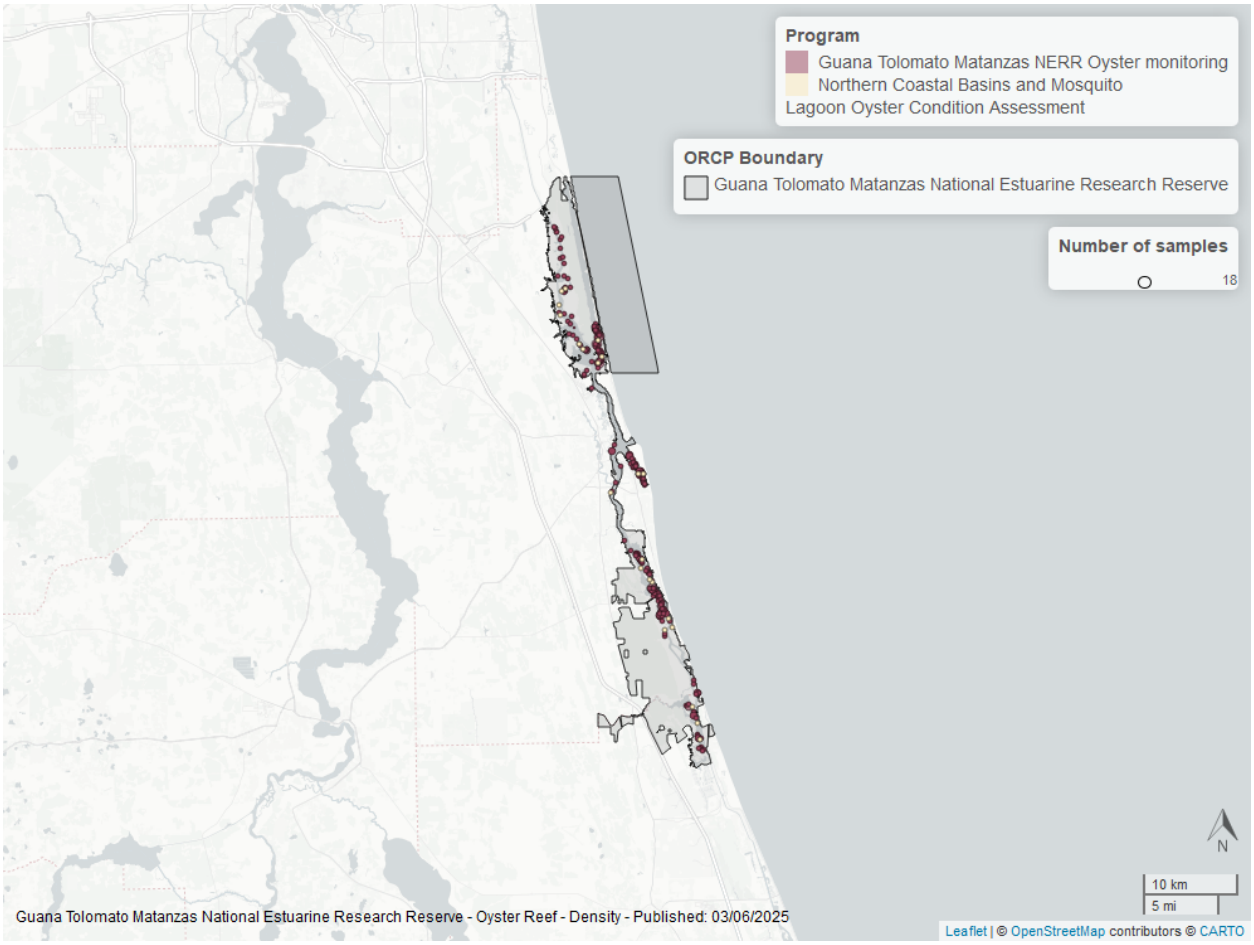


Figure 5: Figure for Oyster Density in Guana Tolomato Matanzas National Estuarine Research Reserve

Table 5: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.14	0.01	-0.15 to -0.12


Program
Guana Tolomato Matanzas NERR Oyster monitoring
Northern Coastal Basins and Mosquito Lagoon Oyster Condition Assessment
ORCP Boundary
Guana Tolomato Matanzas National Estuarine Research Reserve
Number of samples
18
N
10 km
5 mi
Guana Tolomato Matanzas National Estuarine Research Reserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Natural

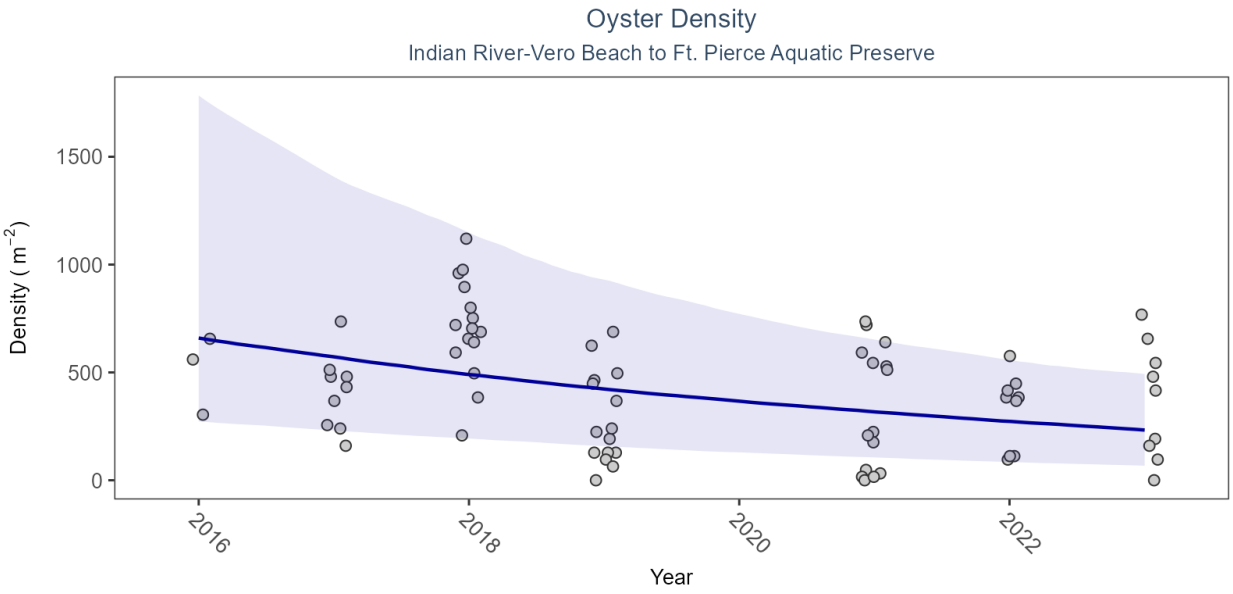
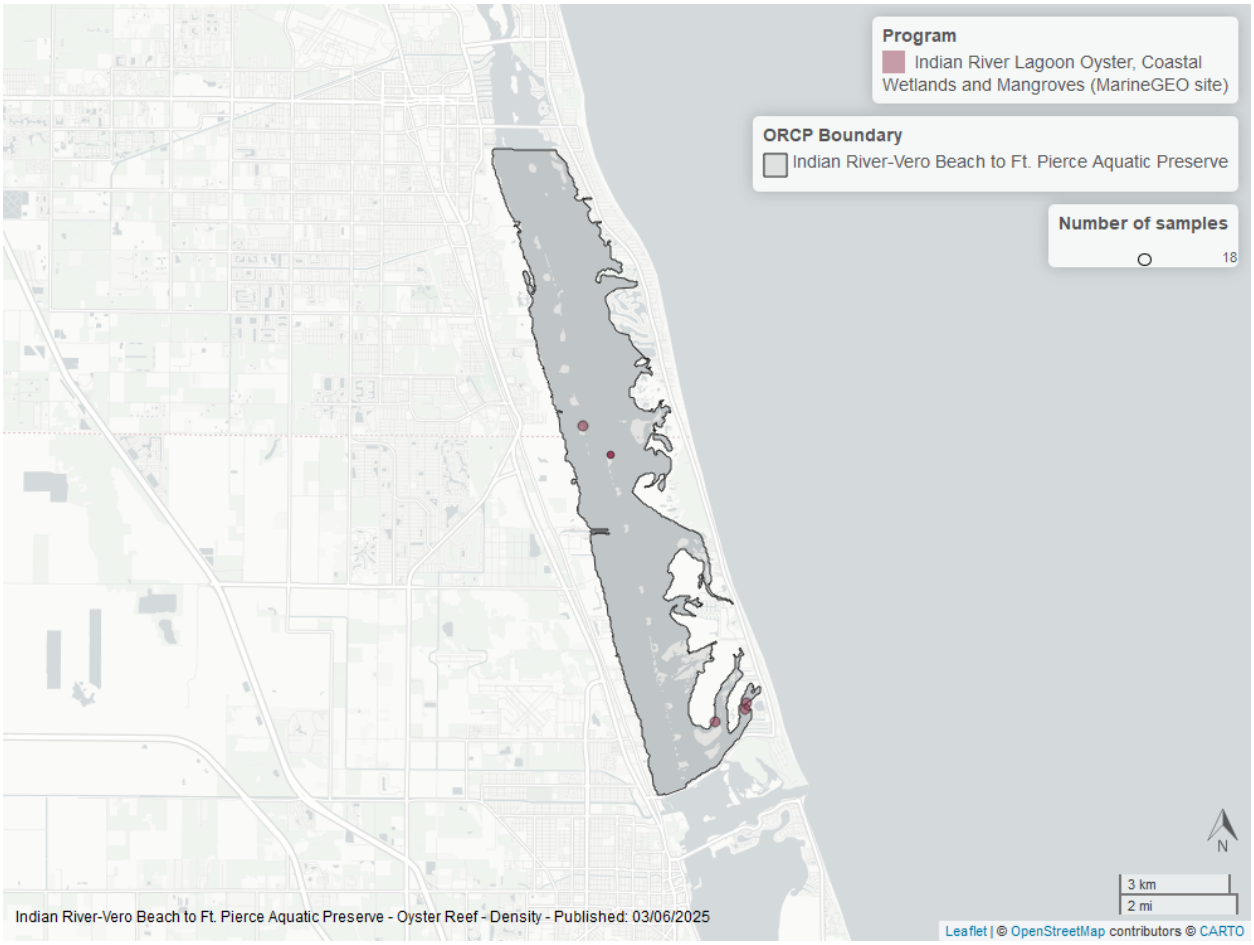


Figure 6: Figure for Oyster Density in Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Table 6: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.17	0.06	-0.29 to -0.05


Program
Indian River Lagoon Oyster, Coastal Wetlands and Mangroves (MarineGEO site)
ORCP Boundary
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve
Number of samples
18
N
3 km
2 mi
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Lemon Bay Aquatic Preserve

Natural

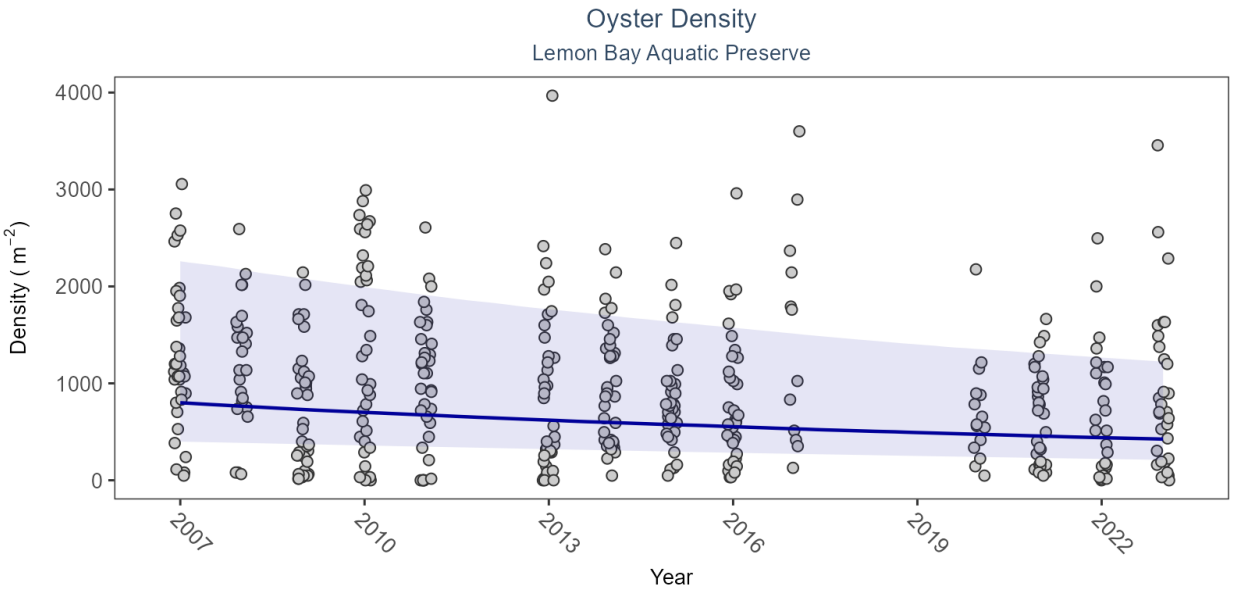
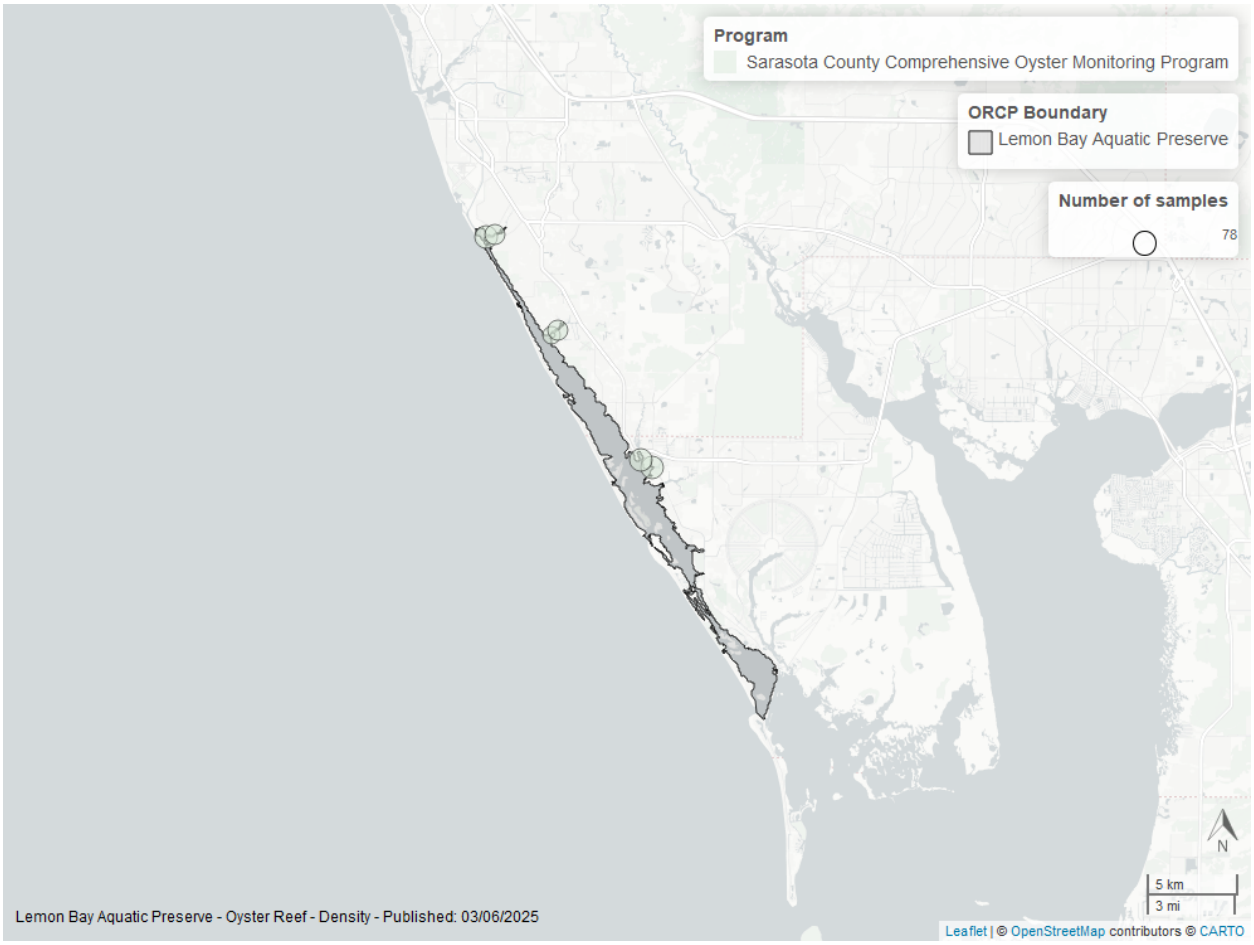


Figure 7: Figure for Oyster Density in Lemon Bay Aquatic Preserve

Table 7: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.04	0.01	-0.05 to -0.03


Program
Sarasota County Comprehensive Oyster Monitoring Program
ORCP Boundary
Lemon Bay Aquatic Preserve
Number of samples
78
N
5 km
3 mi
Lemon Bay Aquatic Preserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Pine Island Sound Aquatic Preserve

Natural

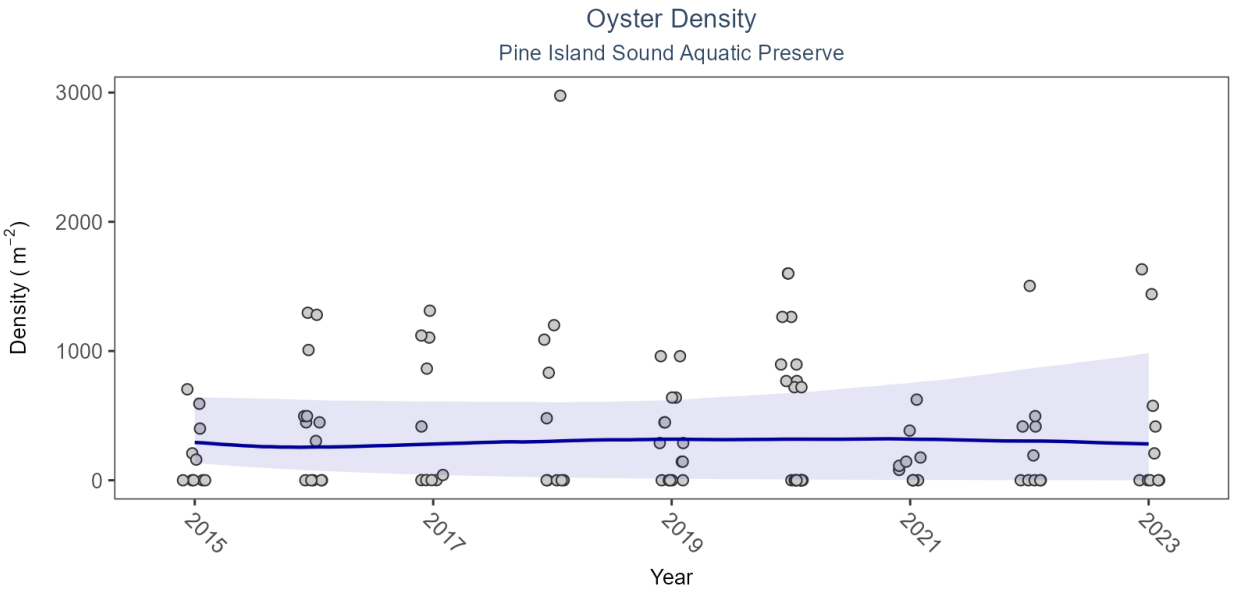


Figure 8: Figure for Oyster Density in Pine Island Sound Aquatic Preserve

Table 8: Model results for Oyster Density - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	No significant change	-0.27	0.47	-1.22 to 0.72

Restored

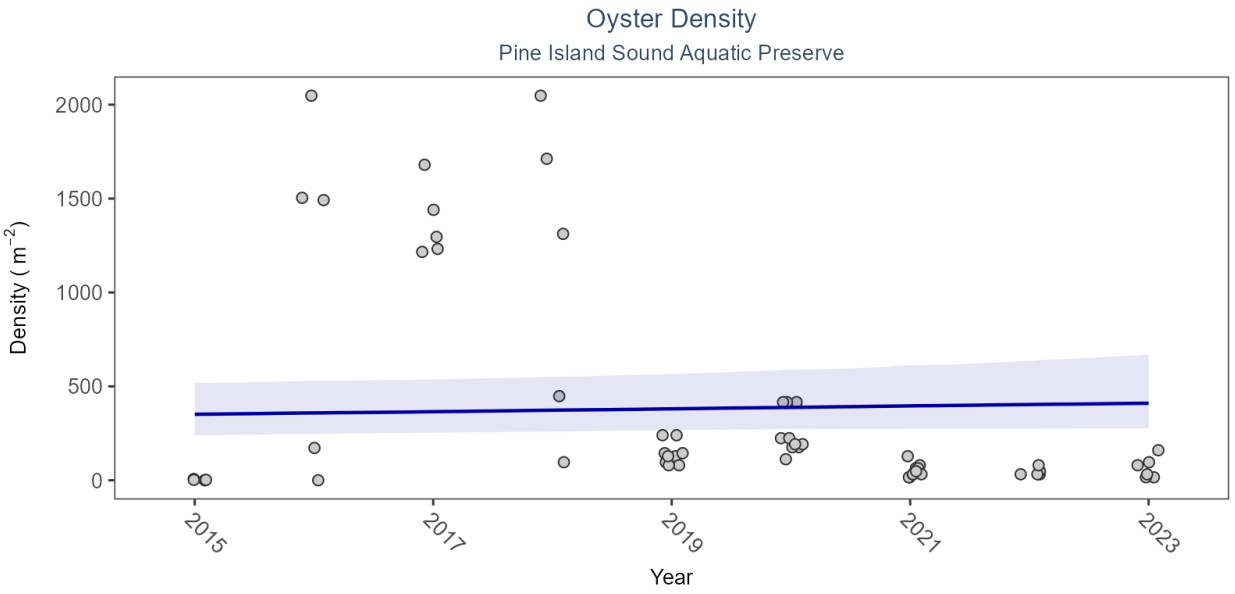
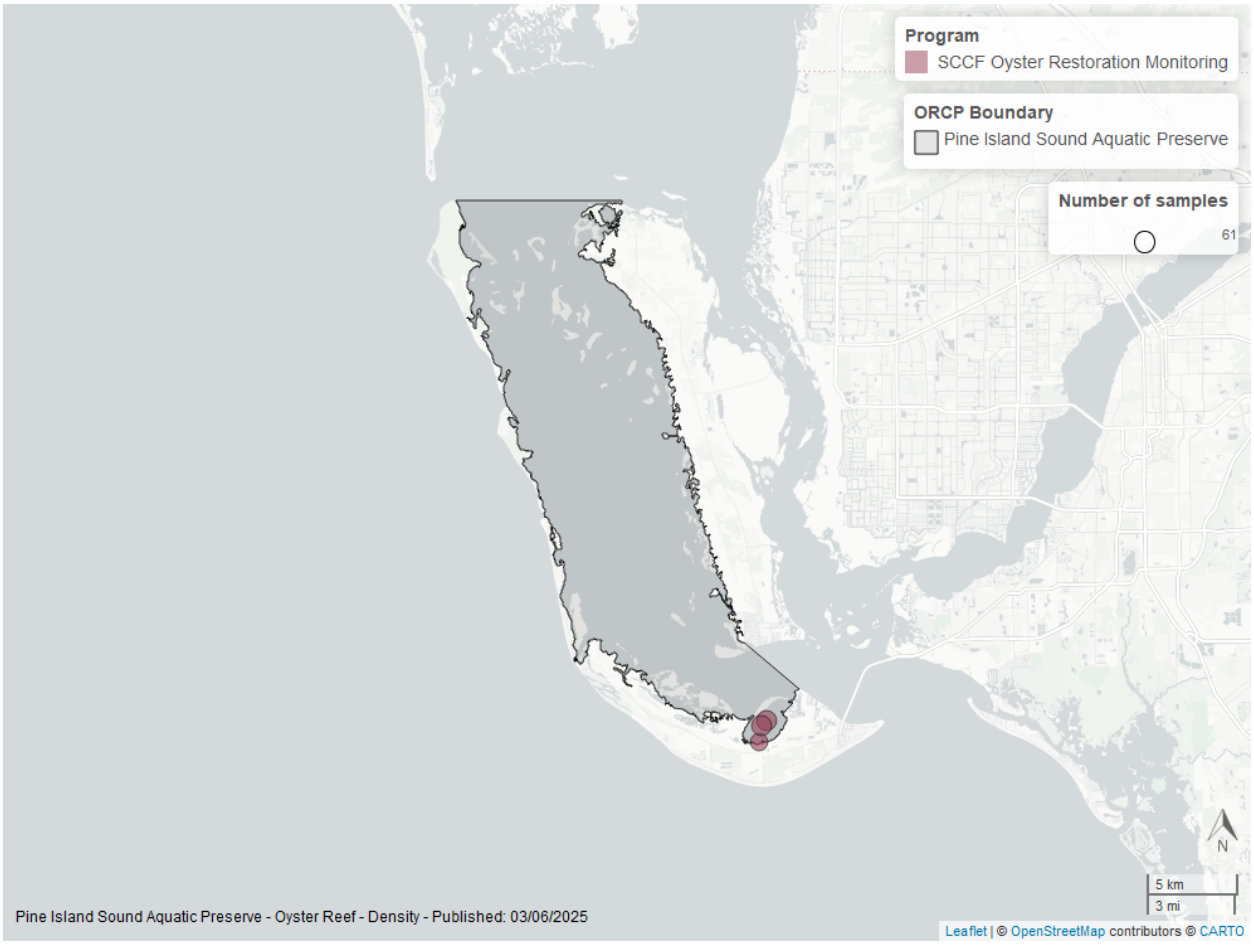


Figure 9: Figure for Oyster Density in Pine Island Sound Aquatic Preserve

Table 9: Model results for Oyster Density - Restored

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Restored	Significantly increasing trend	0.02	0.02	0 to 0.08


Program
SCCF Oyster Restoration Monitoring
ORCP Boundary
Pine Island Sound Aquatic Preserve
Number of samples
61
N
5 km
3 mi
Pine Island Sound Aquatic Preserve - Oyster Reef - Density - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Percent Live

Apalachicola Bay Aquatic Preserve

Natural

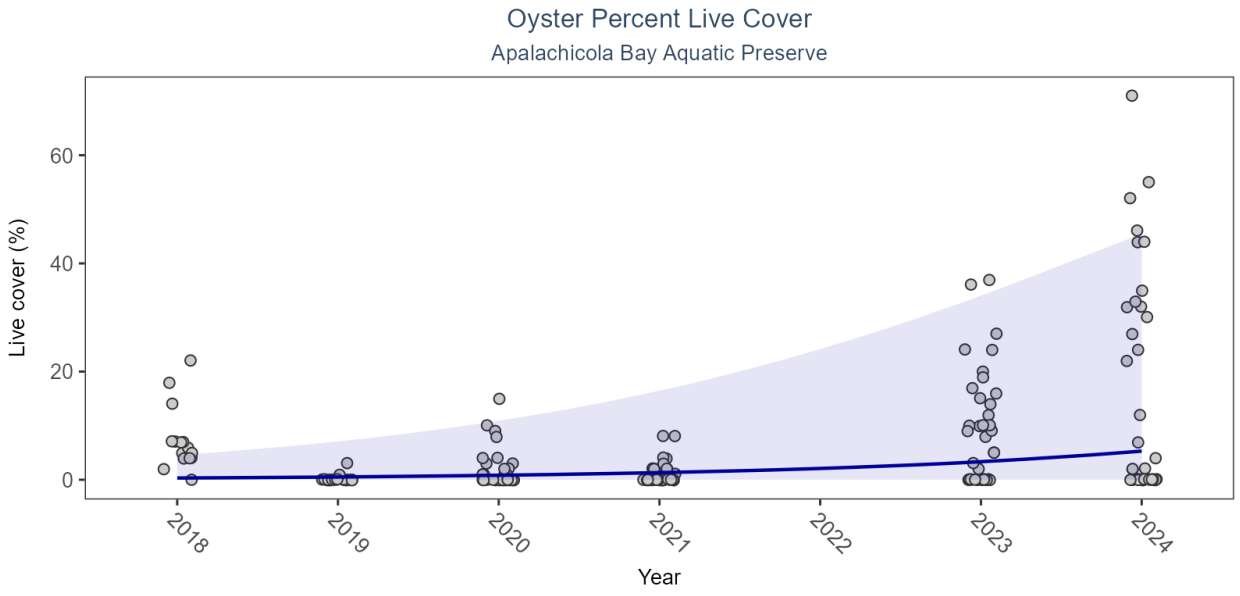
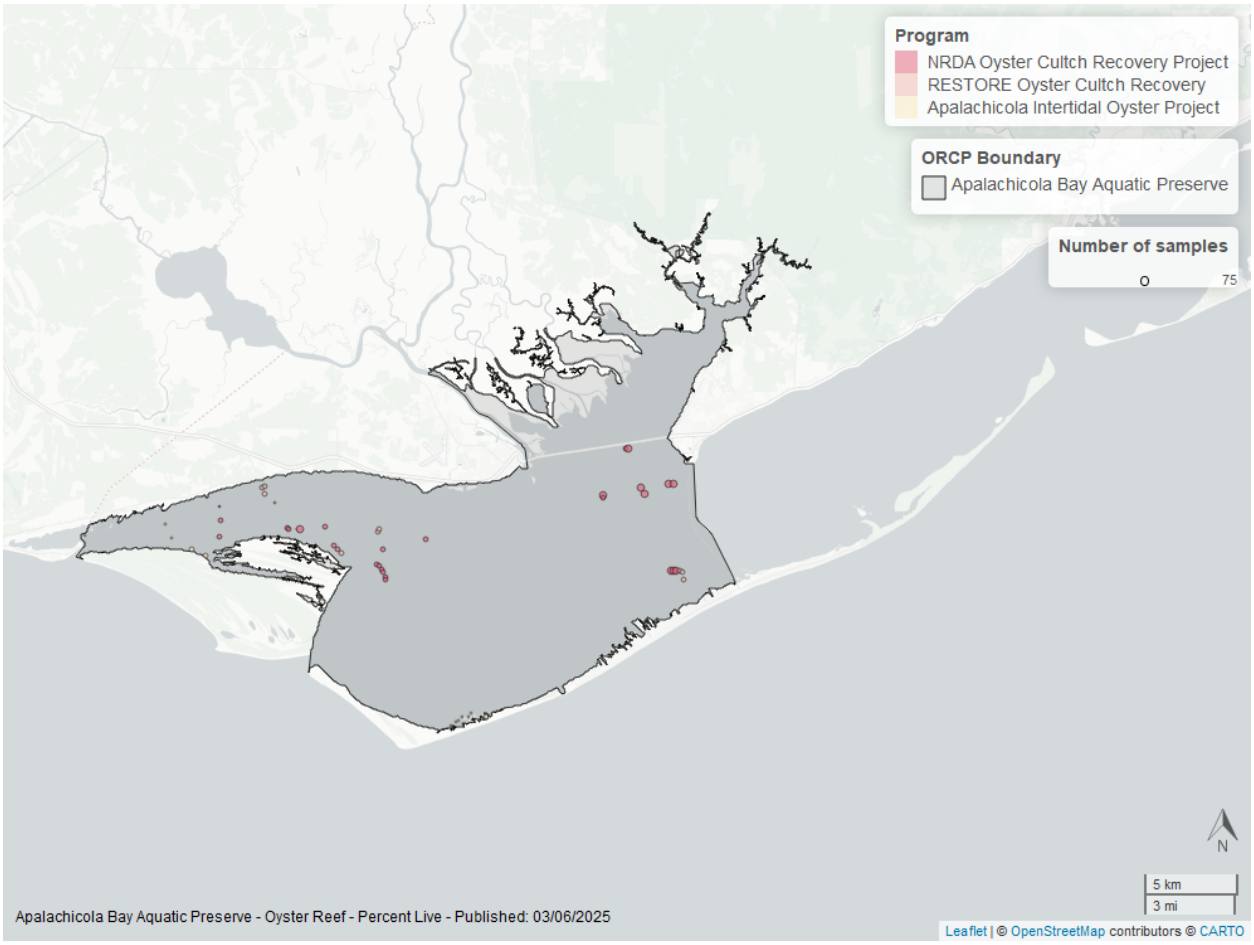


Figure 10: Figure for Oyster Percent Live in Apalachicola Bay Aquatic Preserve

Table 10: Model results for Oyster Percent Live - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly increasing trend	0.48	0.02	0.44 to 0.53


Program
NRDA Oyster Cultch Recovery Project
RESTORE Oyster Cultch Recovery
Apalachicola Intertidal Oyster Project
ORCP Boundary
Apalachicola Bay Aquatic Preserve
Number of samples
0
75
N
5 km
3 mi
Apalachicola Bay Aquatic Preserve - Oyster Reef - Percent Live - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Apalachicola National Estuarine Research Reserve

Natural

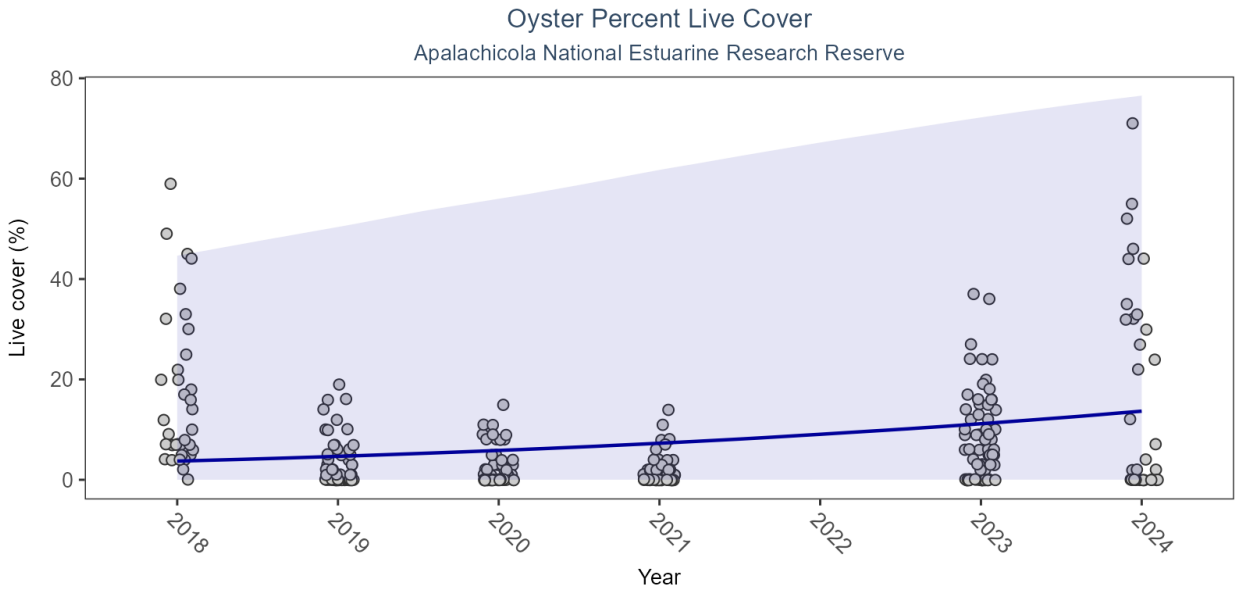
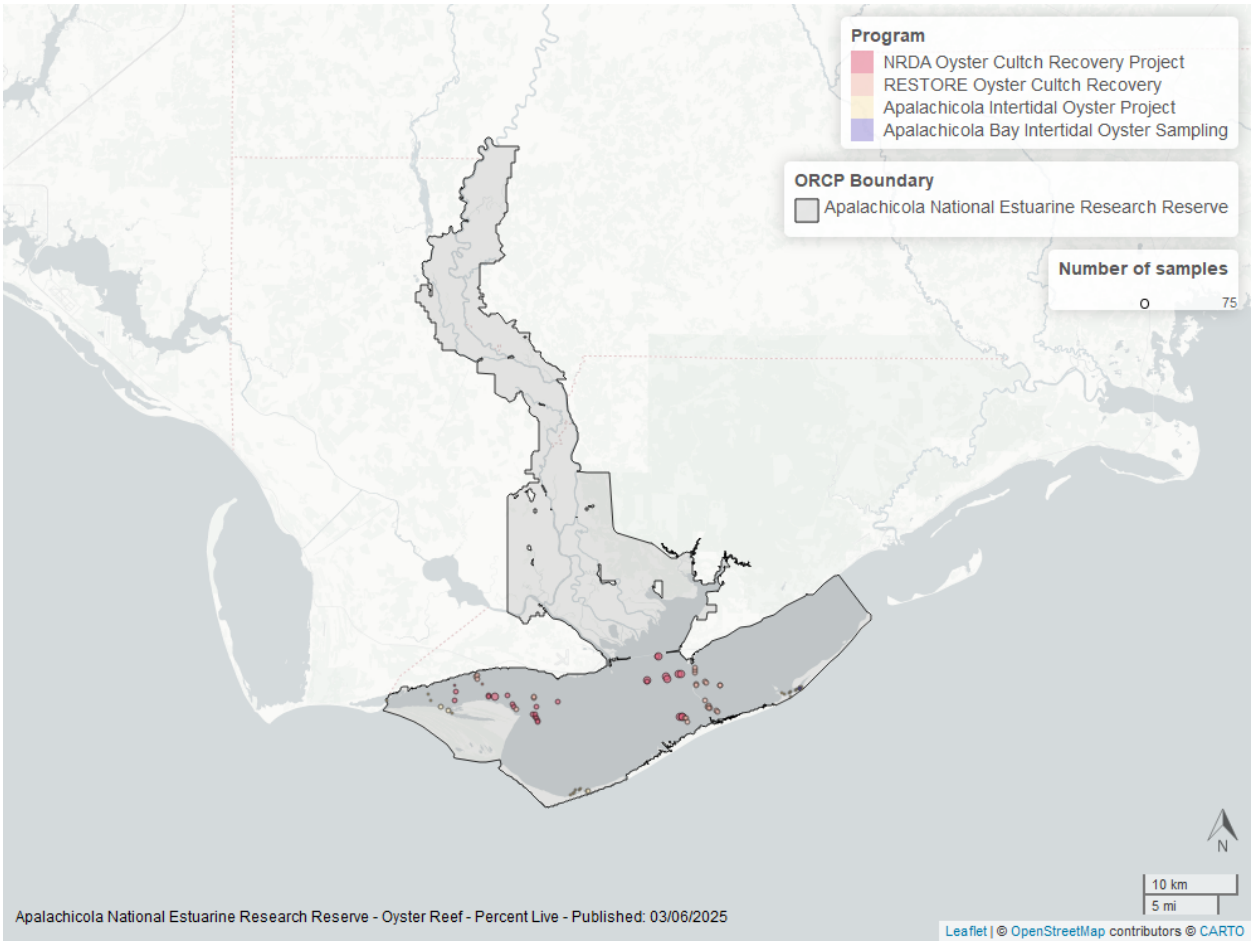


Figure 11: Figure for Oyster Percent Live in Apalachicola National Estuarine Research Reserve

Table 11: Model results for Oyster Percent Live - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly increasing trend	0.23	0.01	0.21 to 0.26


Program
NRDA Oyster Cultch Recovery Project
RESTORE Oyster Cultch Recovery
Apalachicola Intertidal Oyster Project
Apalachicola Bay Intertidal Oyster Sampling
ORCP Boundary
Apalachicola National Estuarine Research Reserve
Number of samples
0
75
N
10 km
5 mi
Apalachicola National Estuarine Research Reserve - Oyster Reef - Percent Live - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Guana River Marsh Aquatic Preserve

Natural

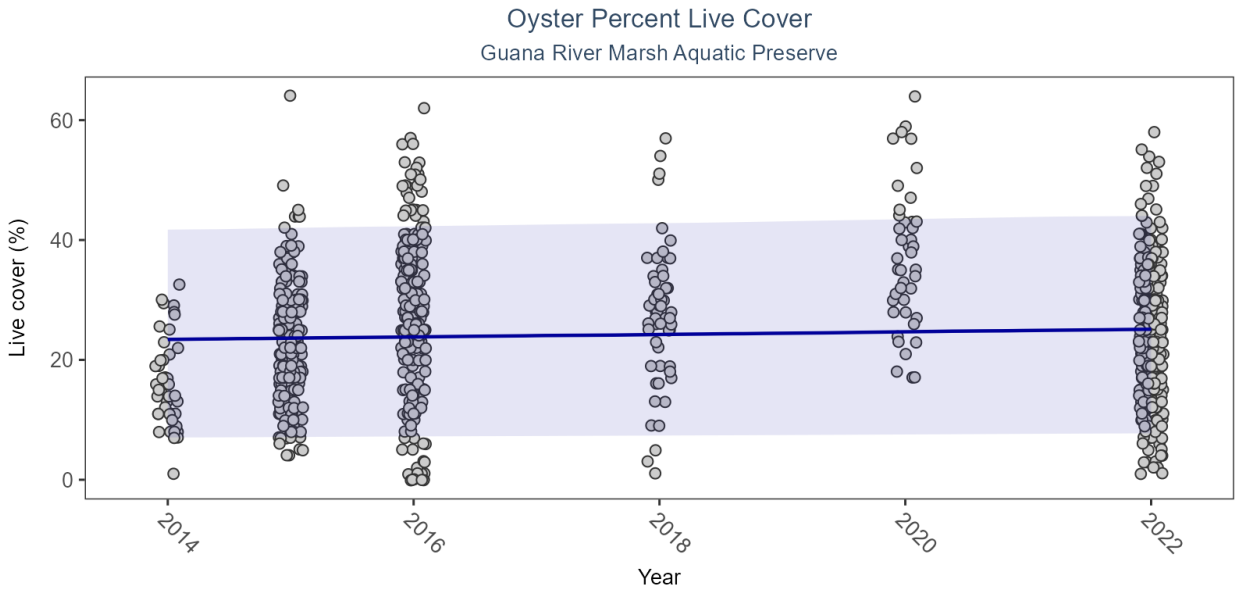


Figure 12: Figure for Oyster Percent Live in Guana River Marsh Aquatic Preserve

Table 12: Model results for Oyster Percent Live - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly increasing trend	0.01	0	0 to 0.02


Program
Guana Tolomato Matanzas NERR Oyster monitoring
Northern Coastal Basins and Mosquito Lagoon Oyster Condition Assessment
ORCP Boundary
Guana River Marsh Aquatic Preserve
Number of samples
42
4
N
3 km
2 mi
Guana River Marsh Aquatic Preserve - Oyster Reef - Percent Live - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Guana Tolomato Matanzas National Estuarine Research Reserve

Natural

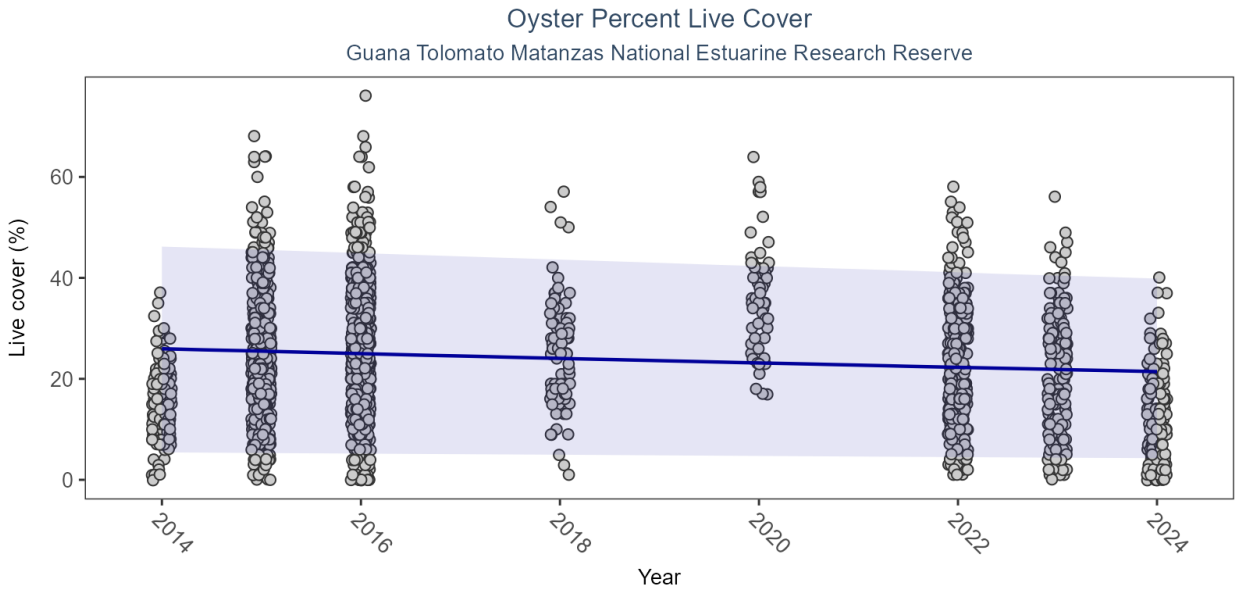
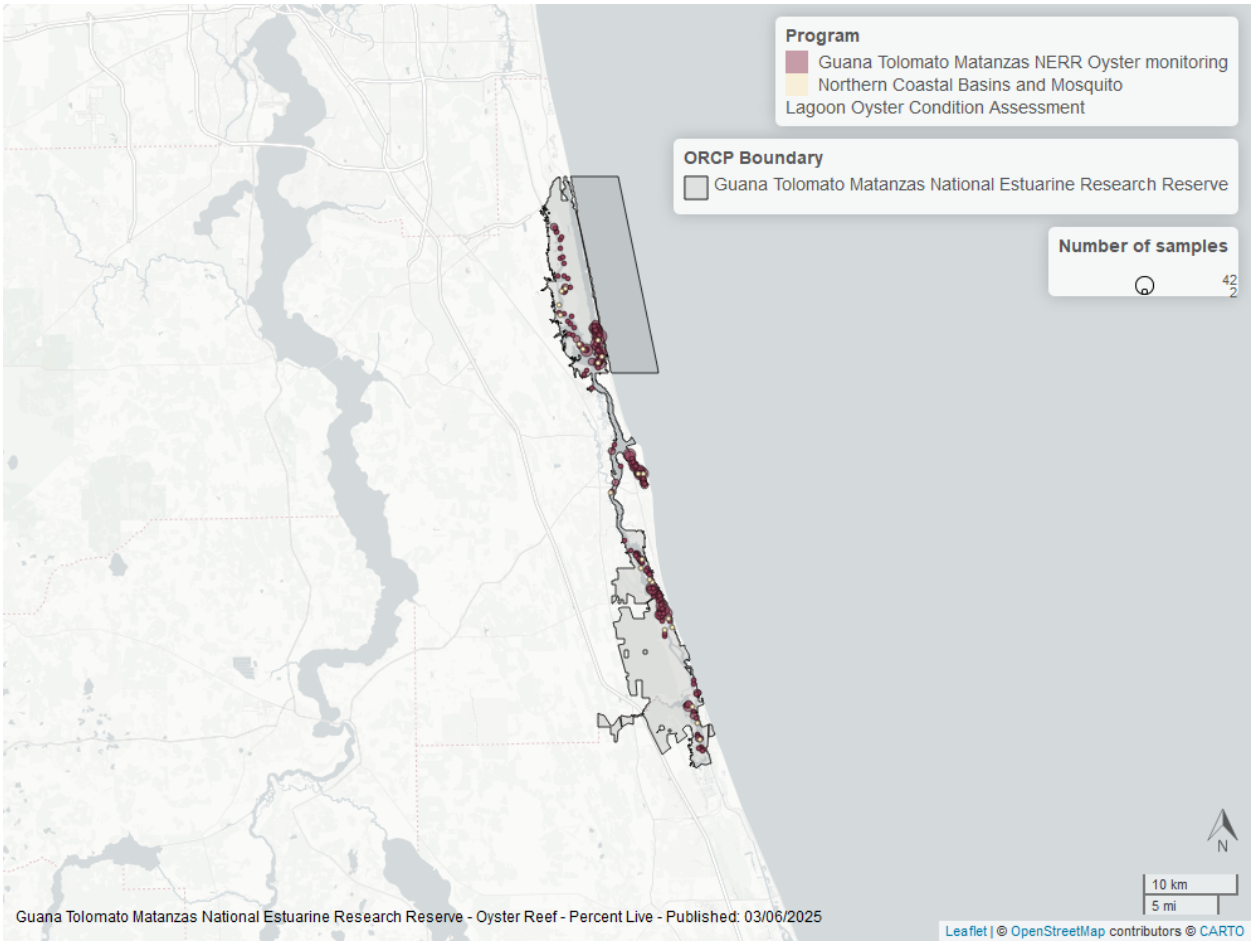


Figure 13: Figure for Oyster Percent Live in Guana Tolomato Matanzas National Estuarine Research Reserve

Table 13: Model results for Oyster Percent Live - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.02	0	-0.03 to -0.02


Program
Guana Tolomato Matanzas NERR Oyster monitoring
Northern Coastal Basins and Mosquito Lagoon Oyster Condition Assessment
ORCP Boundary
Guana Tolomato Matanzas National Estuarine Research Reserve
Number of samples
42
2
N
10 km
5 mi
Guana Tolomato Matanzas National Estuarine Research Reserve - Oyster Reef - Percent Live - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Natural

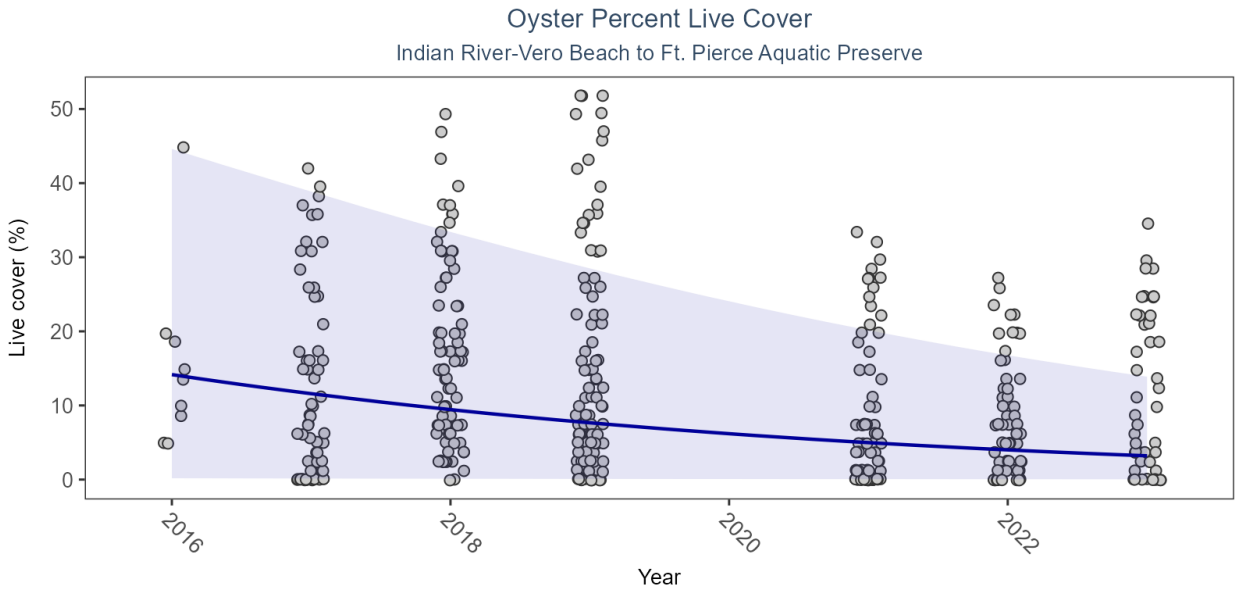
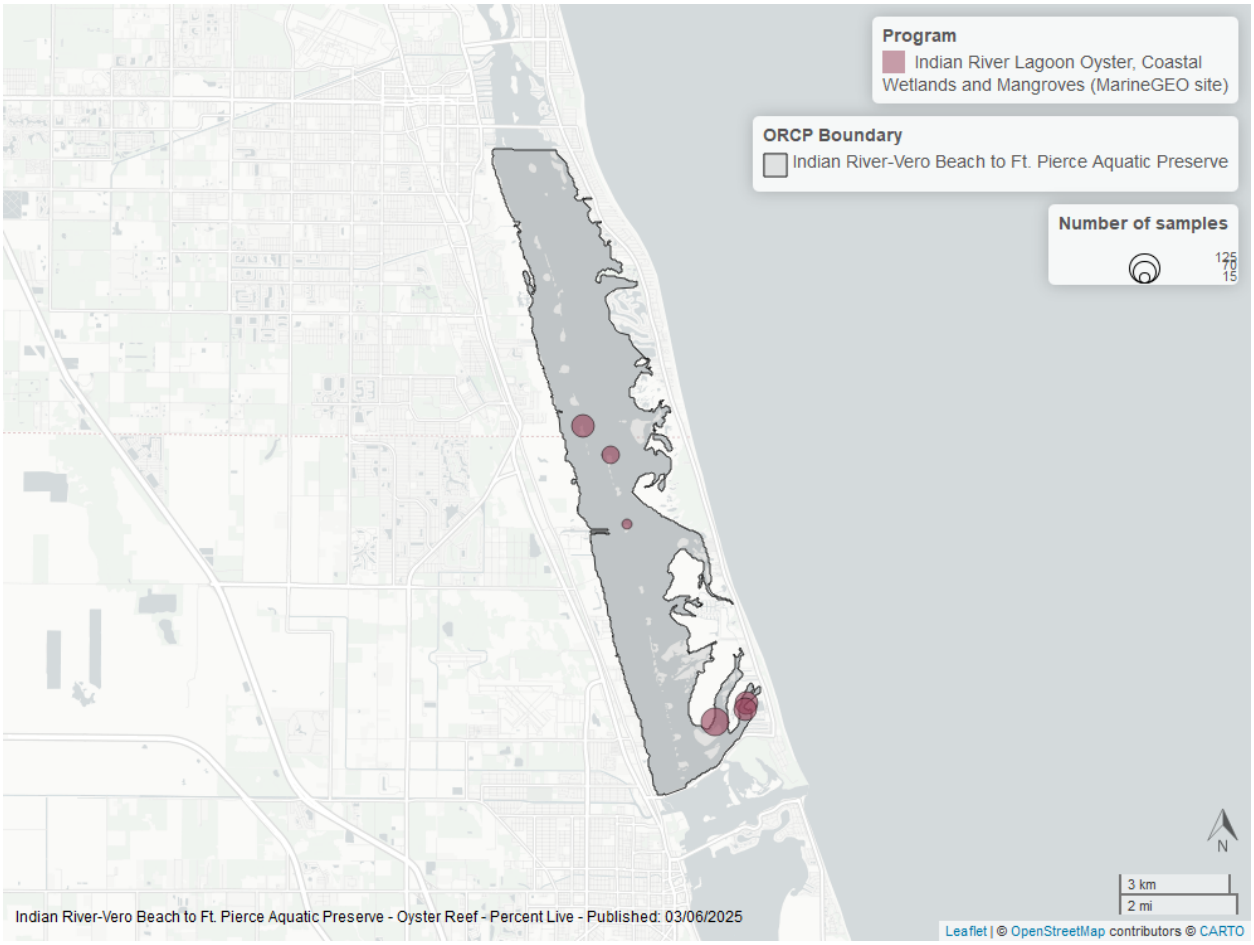


Figure 14: Figure for Oyster Percent Live in Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Table 14: Model results for Oyster Percent Live - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.23	0.01	-0.25 to -0.21


Program
Indian River Lagoon Oyster, Coastal Wetlands and Mangroves (MarineGEO site)
ORCP Boundary
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve
Number of samples
125
70
15
N
3 km
2 mi
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve - Oyster Reef - Percent Live - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Jensen Beach to Jupiter Inlet Aquatic Preserve

Natural

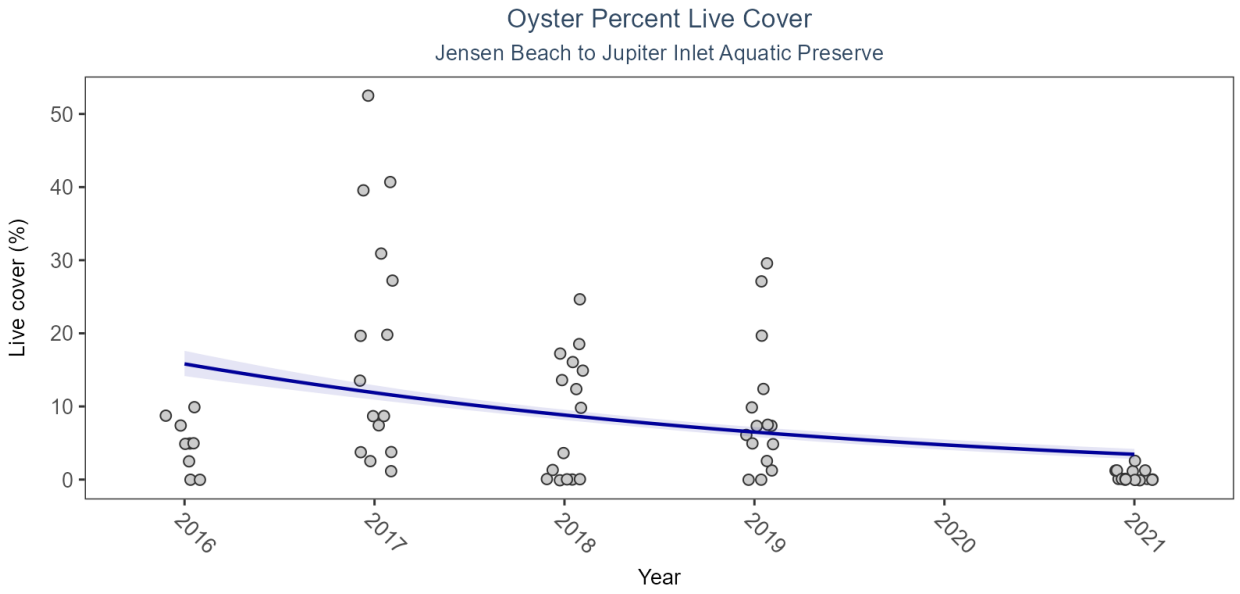
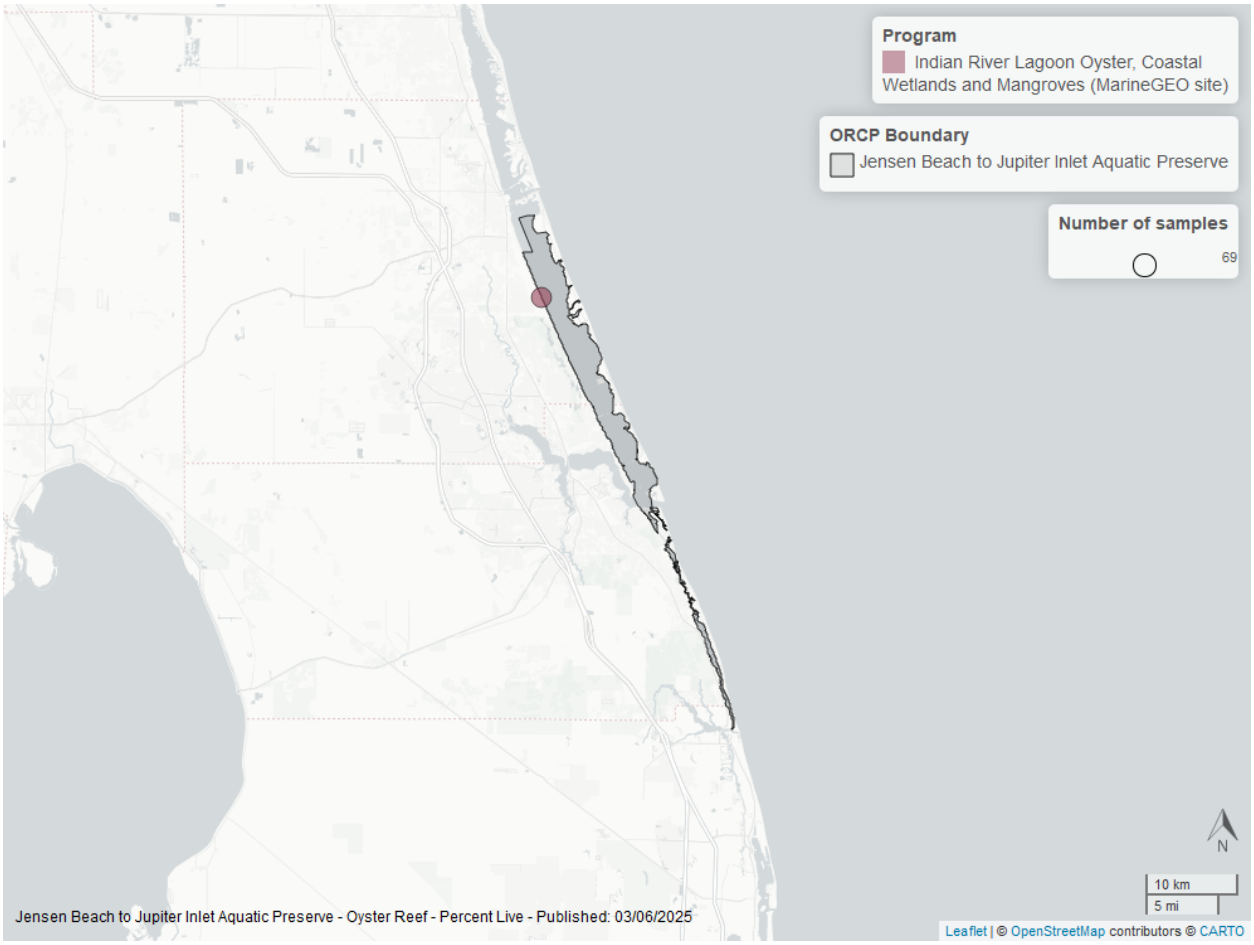


Figure 15: Figure for Oyster Percent Live in Jensen Beach to Jupiter Inlet Aquatic Preserve

Table 15: Model results for Oyster Percent Live - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	Significantly decreasing trend	-0.33	0.03	-0.39 to -0.28


Program
Indian River Lagoon Oyster, Coastal Wetlands and Mangroves (MarineGEO site)
ORCP Boundary
Jensen Beach to Jupiter Inlet Aquatic Preserve
Number of samples
69
N
10 km
5 mi
Jensen Beach to Jupiter Inlet Aquatic Preserve - Oyster Reef - Percent Live - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Lemon Bay Aquatic Preserve

Natural

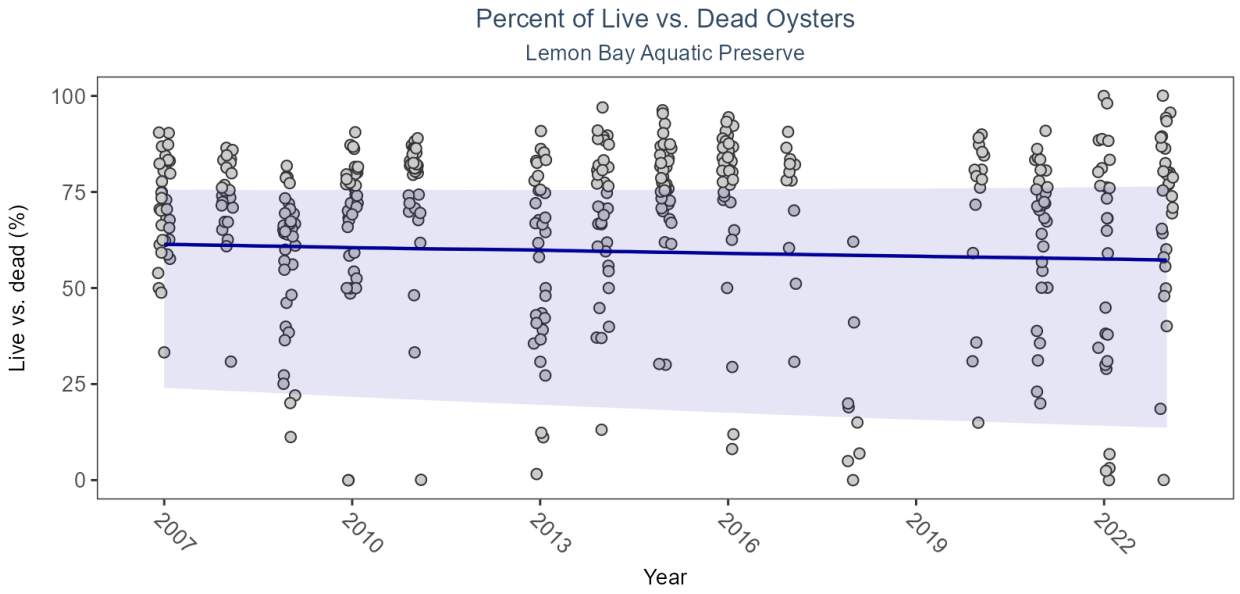
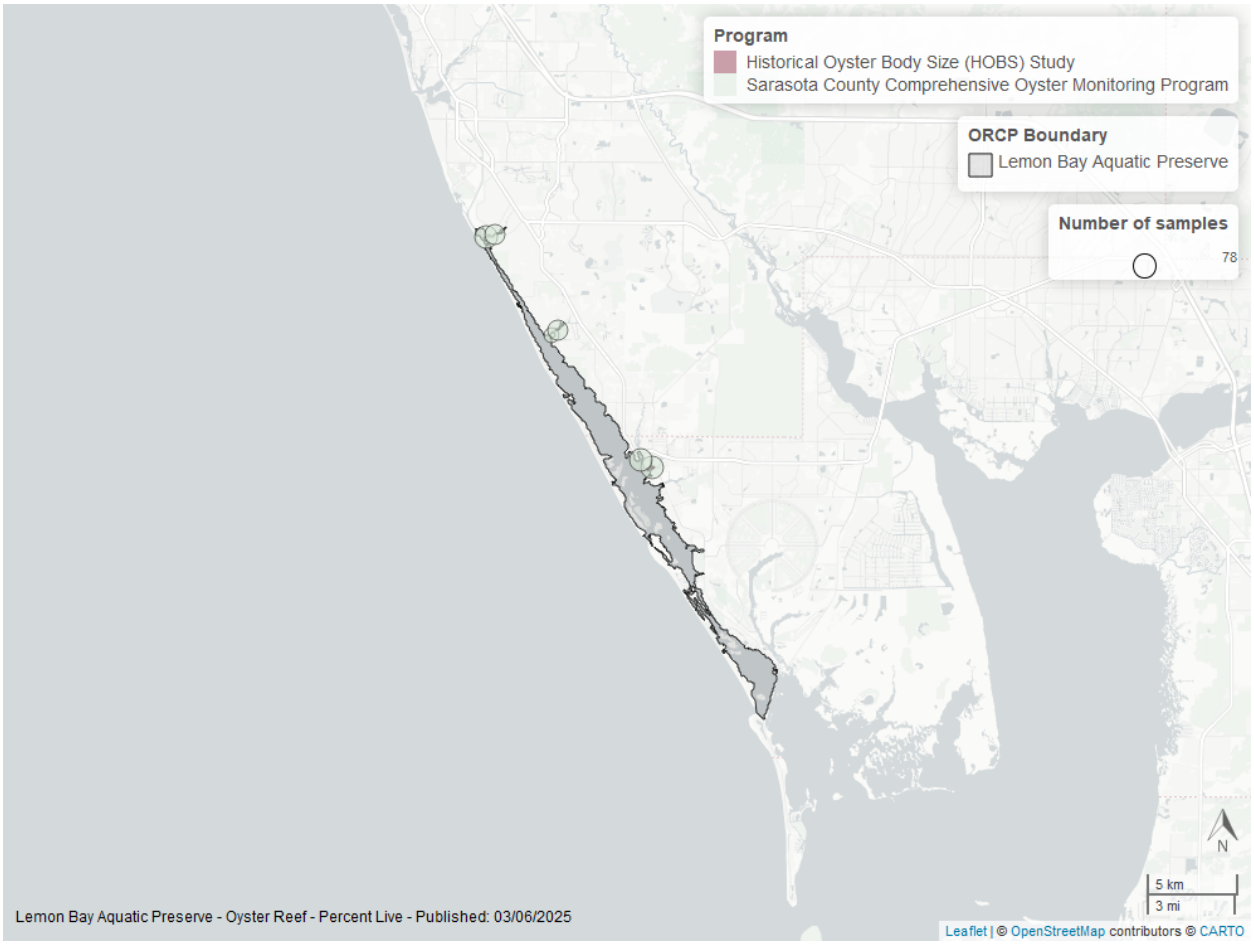


Figure 16: Figure for Oyster Percent Live in Lemon Bay Aquatic Preserve

Table 16: Model results for Oyster Percent Live - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Live Oyster Shells	Natural	No significant change	-0.01	0.01	-0.03 to 0.01


Program
Historical Oyster Body Size (HOBS) Study
Sarasota County Comprehensive Oyster Monitoring Program
ORCP Boundary
Lemon Bay Aquatic Preserve
Number of samples
78
N
5 km
3 mi
Lemon Bay Aquatic Preserve - Oyster Reef - Percent Live - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Shell Height

Apalachicola National Estuarine Research Reserve

Natural

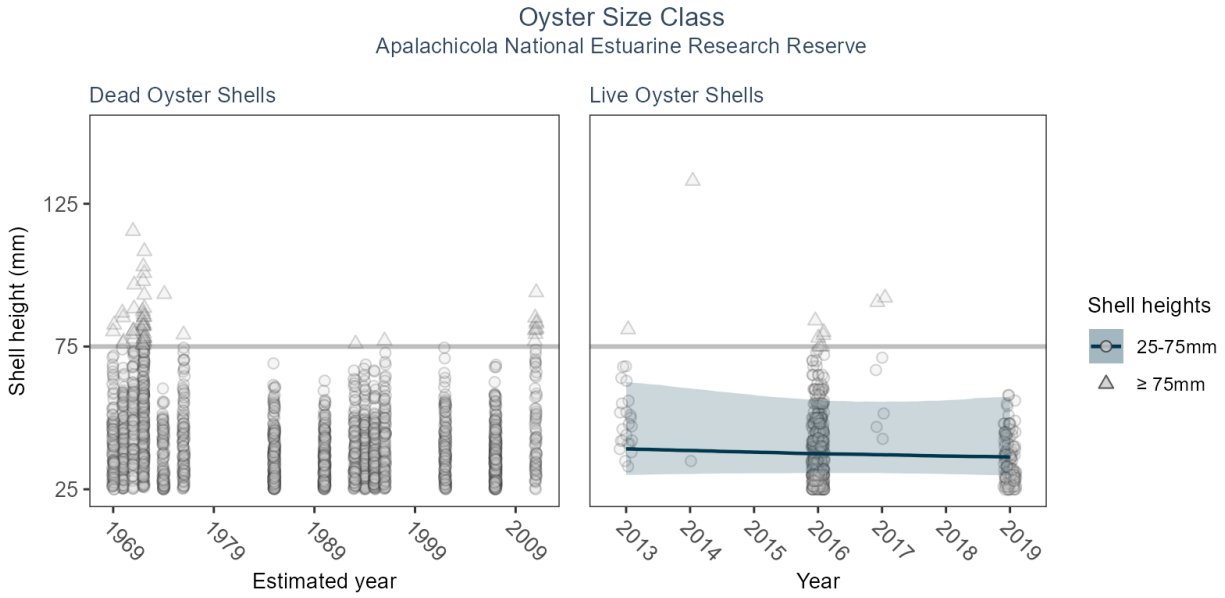
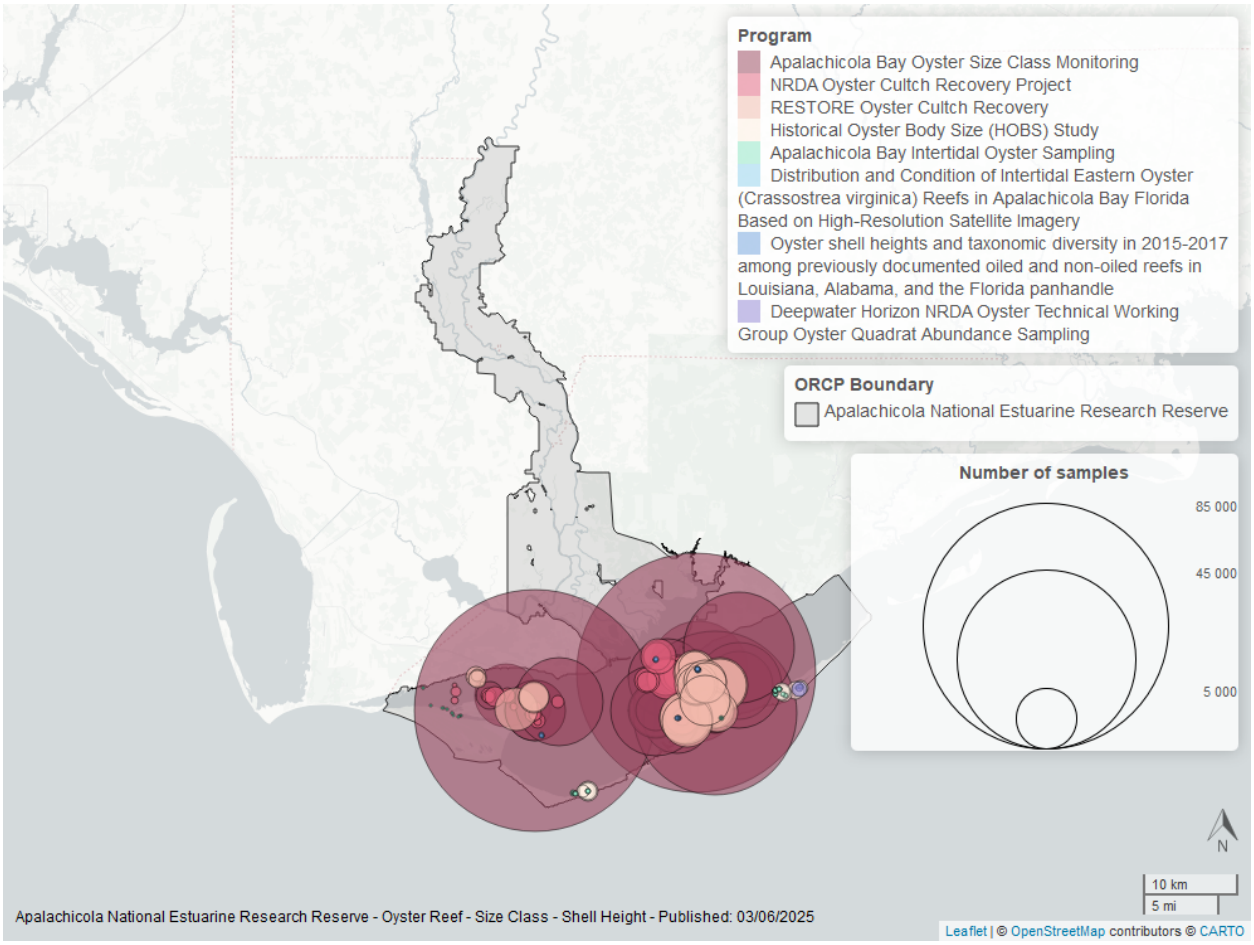


Figure 17: Figure for Oyster Shell Height in Apalachicola National Estuarine Research Reserve

Table 17: Model results for Oyster Shell Height - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Live Oyster Shells	Natural	-	-	-	NA to NA
Live Oyster Shells	Natural	No significant change	-1.21	4.52	-10.4 to 7.72
Live Oyster Shells	Natural	-	-	-	NA to NA


Program
Apalachicola Bay Oyster Size Class Monitoring
NRDA Oyster Cultch Recovery Project
RESTORE Oyster Cultch Recovery
Historical Oyster Body Size (HOBS) Study
Apalachicola Bay Intertidal Oyster Sampling
Distribution and Condition of Intertidal Eastern Oyster (Crassostrea virginica) Reefs in Apalachicola Bay Florida Based on High-Resolution Satellite Imagery
Oyster shell heights and taxonomic diversity in 2015-2017 among previously documented oiled and non-oiled reefs in Louisiana, Alabama, and the Florida panhandle
Deepwater Horizon NRDA Oyster Technical Working Group Oyster Quadrat Abundance Sampling
ORCP Boundary
Apalachicola National Estuarine Research Reserve
Number of samples
85 000
45 000
5 000
N
10 km
5 mi
Apalachicola National Estuarine Research Reserve - Oyster Reef - Size Class - Shell Height - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Estero Bay Aquatic Preserve

Natural

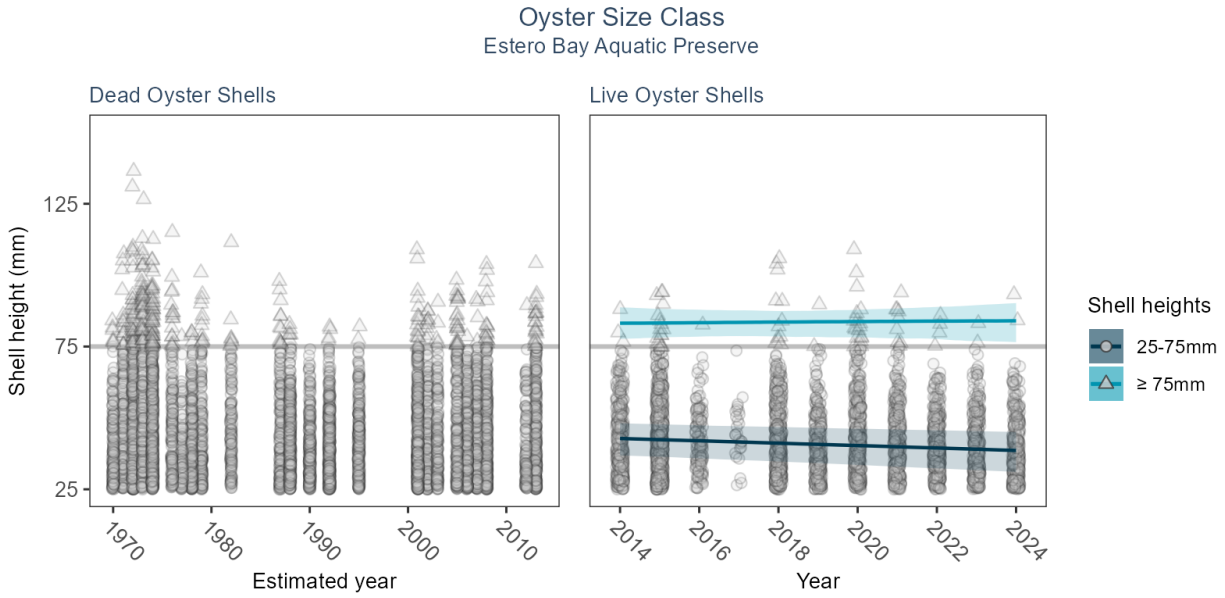
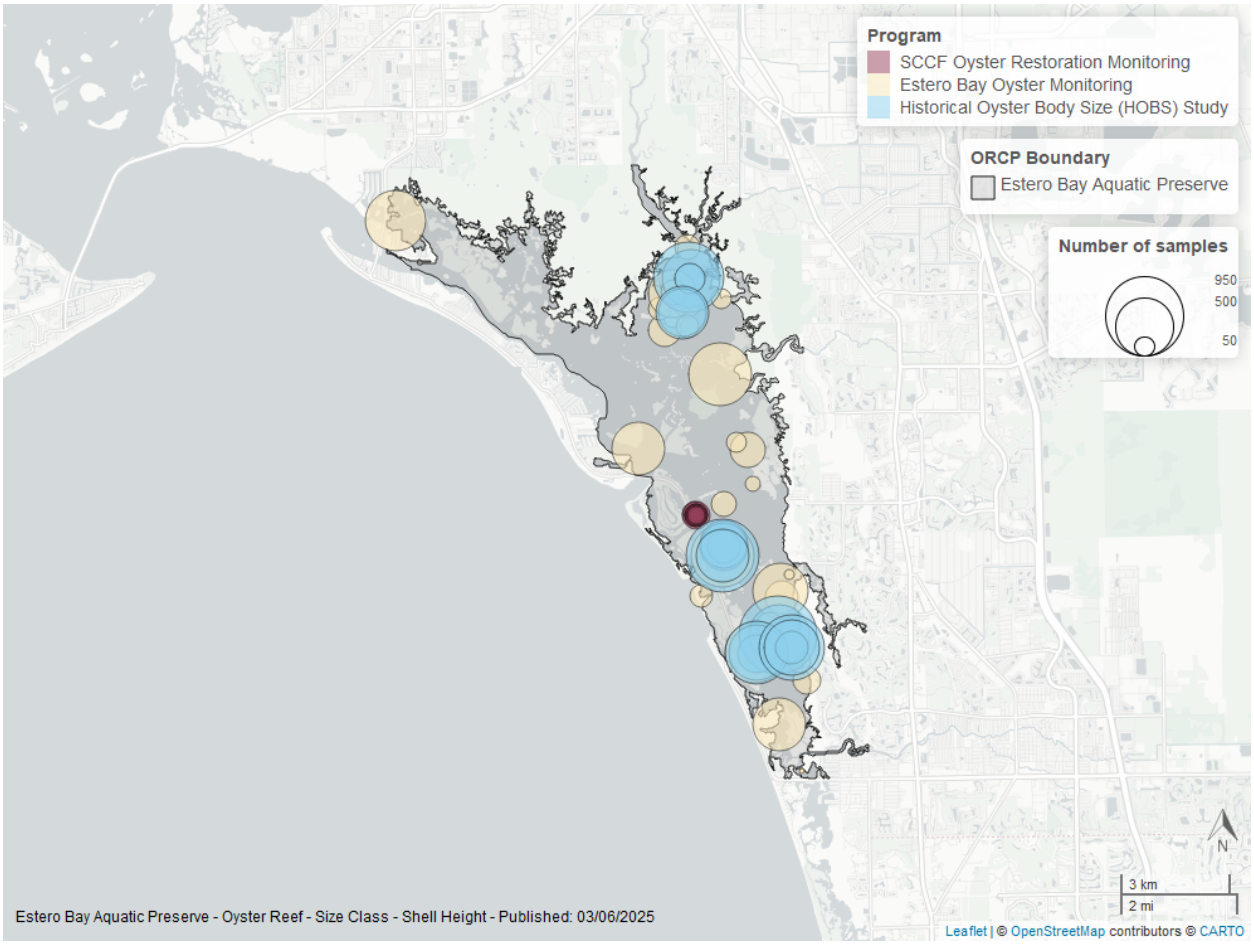


Figure 18: Figure for Oyster Shell Height in Estero Bay Aquatic Preserve

Table 18: Model results for Oyster Shell Height - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Live Oyster Shells	Natural	No significant change	0.07	0.42	-0.75 to 0.89
Live Oyster Shells	Natural	Significantly decreasing trend	-0.43	0.10	-0.64 to -0.22
Live Oyster Shells	Natural	-	-	-	NA to NA


Program
SCCF Oyster Restoration Monitoring
Estero Bay Oyster Monitoring
Historical Oyster Body Size (HOBS) Study
ORCP Boundary
Estero Bay Aquatic Preserve
Number of samples
950
500
50
N
3 km
2 mi
Estero Bay Aquatic Preserve - Oyster Reef - Size Class - Shell Height - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Guana River Marsh Aquatic Preserve

Natural

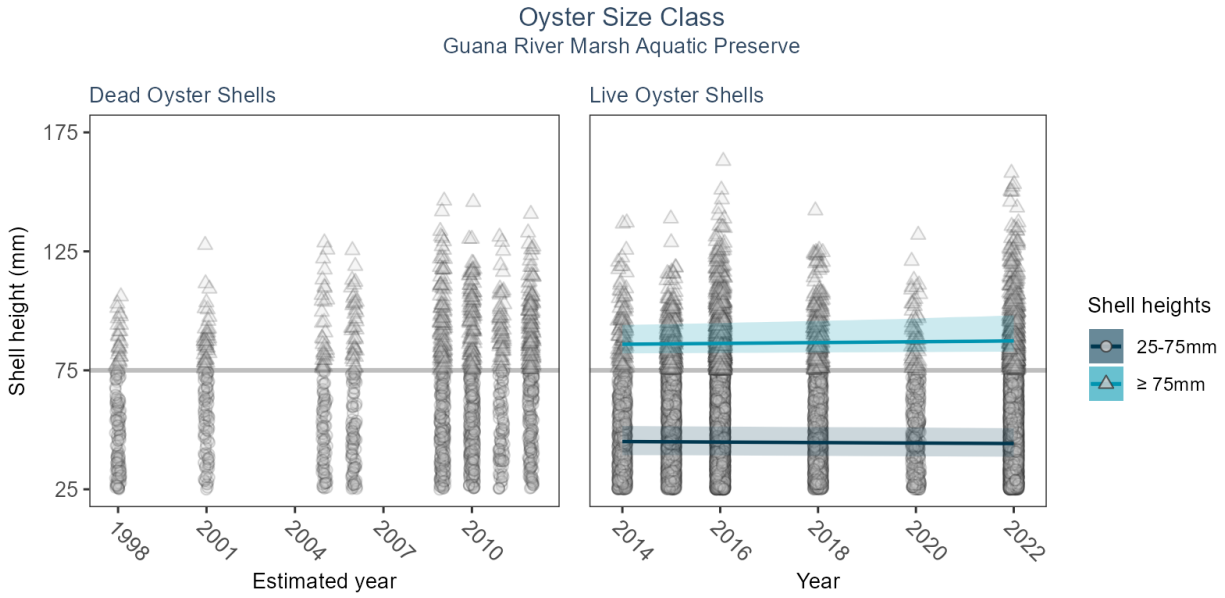
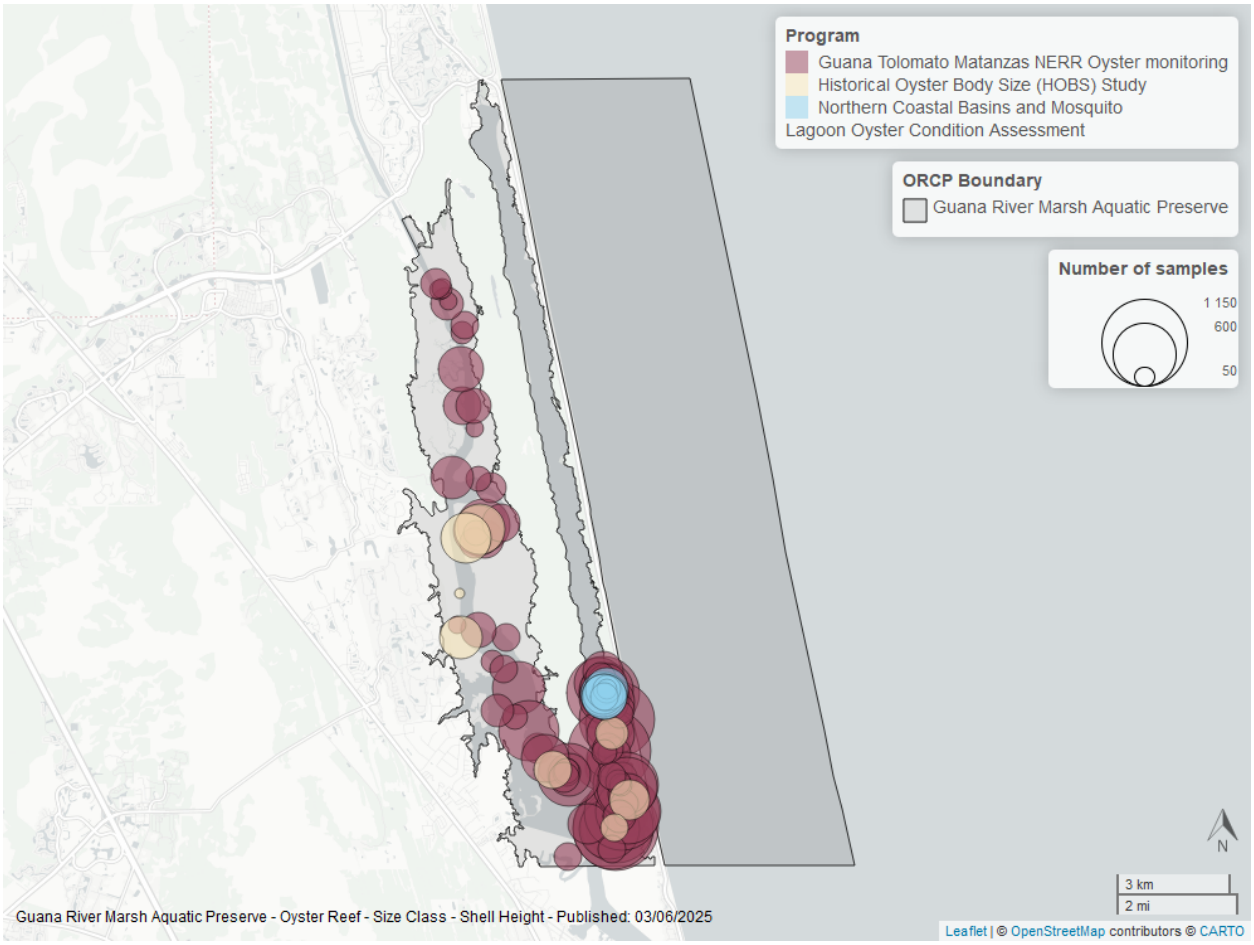


Figure 19: Figure for Oyster Shell Height in Guana River Marsh Aquatic Preserve

Table 19: Model results for Oyster Shell Height - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Live Oyster Shells	Natural	Significantly increasing trend	1.16	0.54	0.13 to 2.25
Live Oyster Shells	Natural	No significant change	-0.91	0.56	-2.04 to 0.14
Live Oyster Shells	Natural	-	-	-	NA to NA


Program
Guana Tolomato Matanzas NERR Oyster monitoring
Historical Oyster Body Size (HOBS) Study
Northern Coastal Basins and Mosquito Lagoon Oyster Condition Assessment
ORCP Boundary
Guana River Marsh Aquatic Preserve
Number of samples
1 150
600
50
N
3 km
2 mi
Guana River Marsh Aquatic Preserve - Oyster Reef - Size Class - Shell Height - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Guana Tolomato Matanzas National Estuarine Research Reserve

Natural

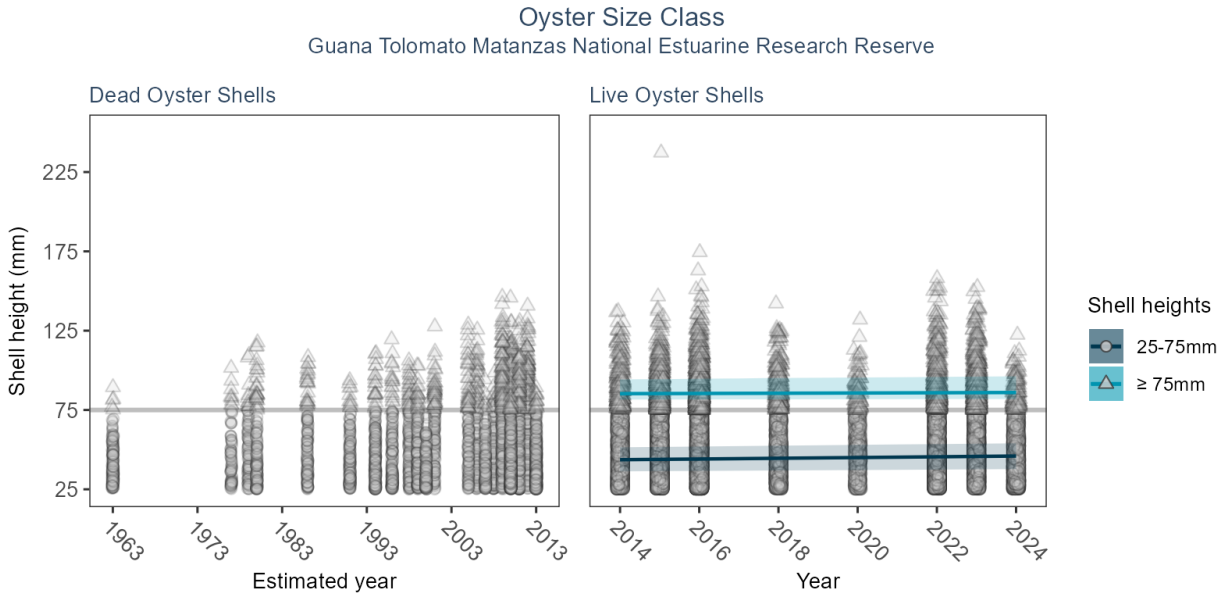
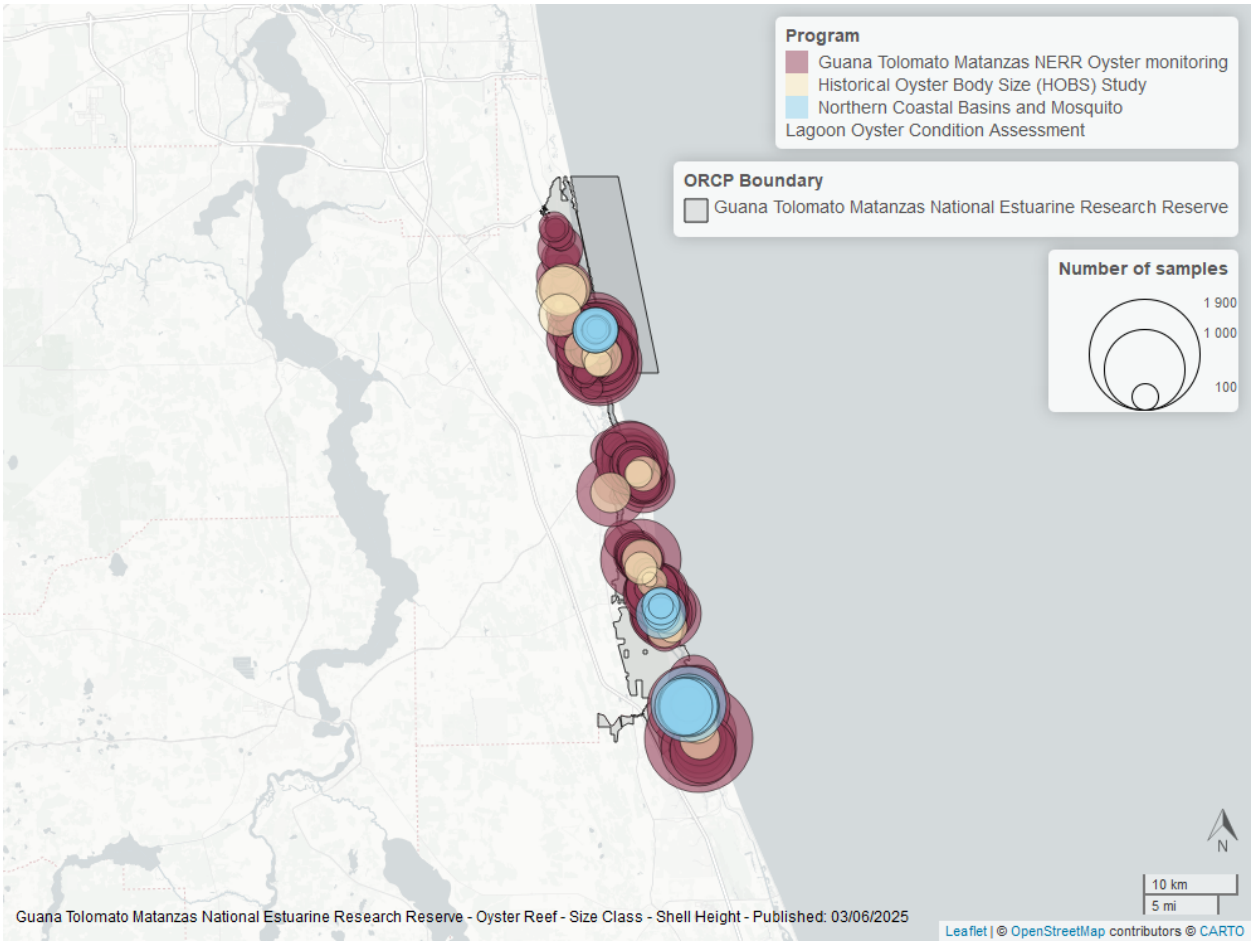


Figure 20: Figure for Oyster Shell Height in Guana Tolomato Matanzas National Estuarine Research Reserve

Table 20: Model results for Oyster Shell Height - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Live Oyster Shells	Natural	No significant change	0.57	0.37	-0.14 to 1.26
Live Oyster Shells	Natural	Significantly increasing trend	1.55	0.25	1.07 to 2.07
Live Oyster Shells	Natural	-	-	-	NA to NA


Program
Guana Tolomato Matanzas NERR Oyster monitoring
Historical Oyster Body Size (HOBS) Study
Northern Coastal Basins and Mosquito Lagoon Oyster Condition Assessment
ORCP Boundary
Guana Tolomato Matanzas National Estuarine Research Reserve
Number of samples
1 900
1 000
100
N
10 km
5 mi
Guana Tolomato Matanzas National Estuarine Research Reserve - Oyster Reef - Size Class - Shell Height - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO

Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Natural

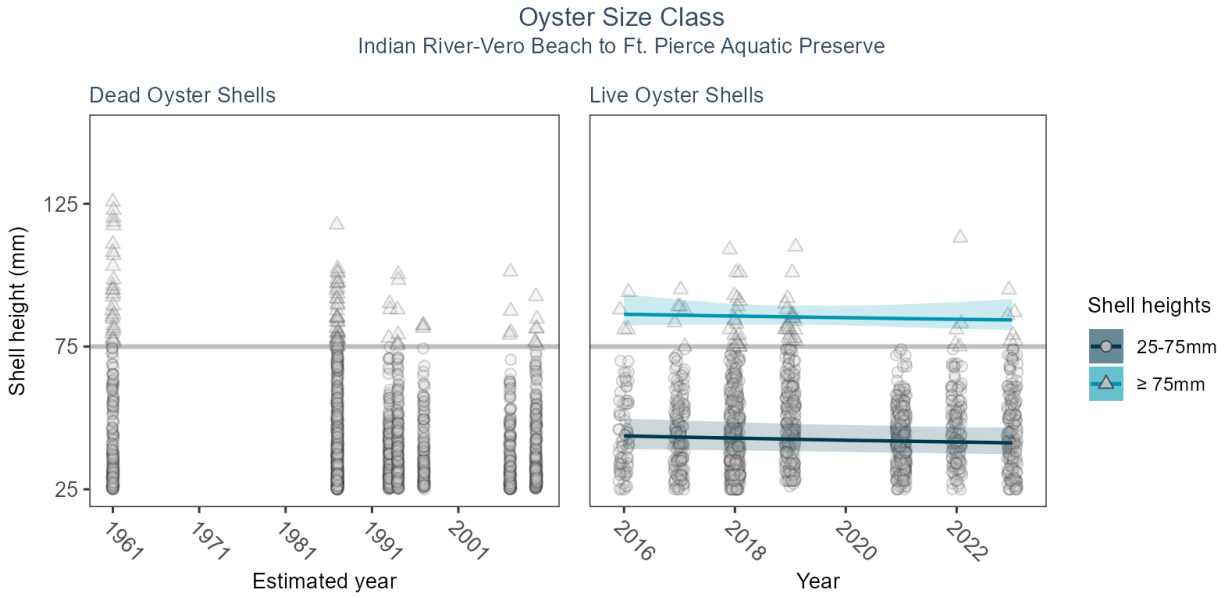
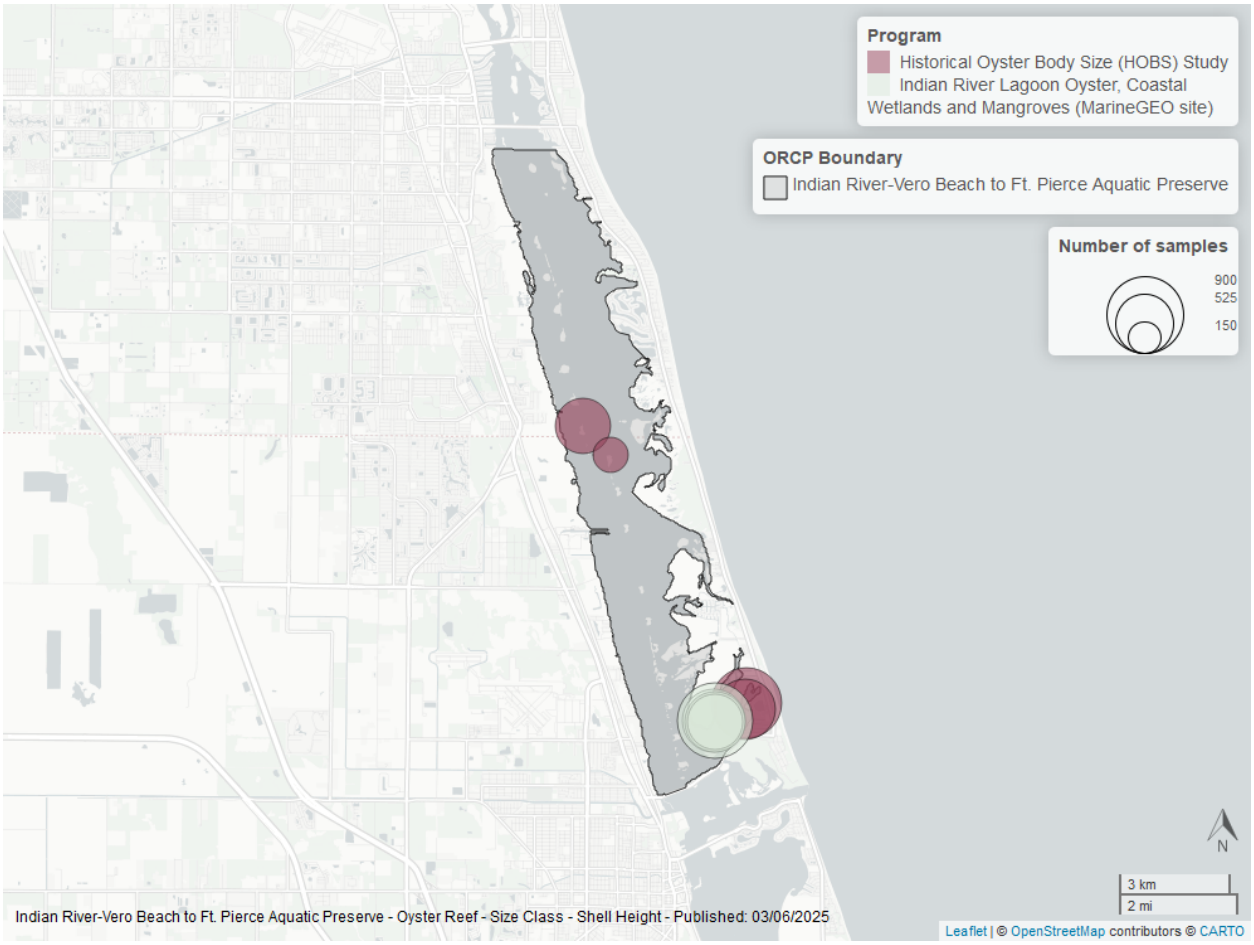


Figure 21: Figure for Oyster Shell Height in Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Table 21: Model results for Oyster Shell Height - Natural

Shell Type	*Habitat Type*	*Trend Status*	*Estimate*	*Standard Error*	*Credible Interval*
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Dead Oyster Shells	Natural	-	-	-	NA to NA
Live Oyster Shells	Natural	No significant change	-1.21	2.98	-7.4 to 3.64
Live Oyster Shells	Natural	Significantly decreasing trend	-0.95	0.46	-1.9 to -0.07
Live Oyster Shells	Natural	-	-	-	NA to NA


Program
Historical Oyster Body Size (HOBS) Study
Indian River Lagoon Oyster, Coastal Wetlands and Mangroves (MarineGEO site)
ORCP Boundary
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve
Number of samples
900
525
150
N
3 km
2 mi
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve - Oyster Reef - Size Class - Shell Height - Published: 03/06/2025
Leaflet | © OpenStreetMap contributors © CARTO