SEACAR Oyster Analysis

Last compiled on 09 June, 2023

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 SEACAR_Oyster_ReportRender.R (https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Nekton/SEACAR_Nekton_SpeciesRichness_ReportRender.R).

This script is based off of code originally written by Katie May Laumann, and modified by Stephen Durham.

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

THIS SCRIPT WILL ONLY RUN WITH brms version 2.16.3 or lower. DOES NOT WORK WITH brms 2.17.0

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(Rmisc)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.1.2
```

```
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
library(ggplot2)
library(grid)
library(gridExtra)
library(gtable)
library(here)
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.1.2
```

```
library(mapview)
library(rcompanion)
```

```
## Warning: package 'rcompanion' was built under R version 4.1.2
```

```
library(data.table)
library(brms)
library(modelr)
library(tidybayes)
```

```
## Warning: package 'tidybayes' was built under R version 4.1.2
```

```
library(doFuture)
```

```
## Warning: package 'future' was built under R version 4.1.2
```

```
library(tictoc)
library(doRNG)
library(piecewiseSEM)
library(ggpubr)
options(scipen=999)
knitr::opts_chunk$set(
```

```
   warning=FALSE,
   message=FALSE,
   eval=Run_An,
   dpi=200
)
```

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-2023-Jun-05.txt**

```
oysterraw <- fread(file_in, sep="|", header=TRUE, stringsAsFactors=FALSE,
                   na.strings=c("NULL","","NA"))
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.

```
#Load spatial data files for RCP managed areas, SEACAR sample locations and
#FWC statewide oyster reef layer
fwcoymap <- st_read("data/OysterGIS_files/Oyster_Beds_in_Florida_2021update/Oyster_Beds_in_Florida.shp")
```

```
## Reading layer ‘Oyster_Beds_in_Florida’ from data source
##   ‘C:\Users\jepanzik\Box\R Projects\SEACAR_Trend_Analyses\Oyster\data\OysterGIS_files\Oyster_Beds_in_
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 32693 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -87.21656 ymin: 25.47165 xmax: -80.03876 ymax: 30.7144
## Geodetic CRS:  WGS 84
```

```
aps <- st_read("data/OysterGIS_files/APs/Florida_Aquatic_Preserves.shp")
```

```
## Reading layer 'Florida_Aquatic_Preserves' from data source
##   'C:\Users\jepanzik\Box\R Projects\SEACAR_Trend_Analyses\Oyster\data\OysterGIS_files\APs\Florida_Aqu
##   using driver 'ESRI Shapefile'
## Simple feature collection with 41 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -87.39386 ymin: 24.61092 xmax: -80.07605 ymax: 30.72562
## Geodetic CRS:  WGS 84
```

```
nerrs <- st_read("data/OysterGIS_files/NERRs/Florida_National_Estuarine_Resarch_Reserves__NERR__Boundari
```

```
## Reading layer 'Florida_National_Estuarine_Resarch_Reserves__NERR__Boundaries' from data source
##   'C:\Users\jepanzik\Box\R Projects\SEACAR_Trend_Analyses\Oyster\data\OysterGIS_files\NERRs\Florida_N
##   using driver 'ESRI Shapefile'
## Simple feature collection with 3 features and 21 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -85.22525 ymin: 25.80369 xmax: -81.19734 ymax: 30.21954
## Geodetic CRS:  WGS 84
```

```
GTMnew <- st_read("data/OysterGIS_files/GTM_RB_2016_Merge.kml")
```

```
## Reading layer 'GTM_RB_2016_Merge' from data source
##   'C:\Users\jepanzik\Box\R Projects\SEACAR_Trend_Analyses\Oyster\data\OysterGIS_files\GTM_RB_2016_Mer
## Simple feature collection with 8 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XYZ
## Bounding box:  xmin: -81.38838 ymin: 29.60151 xmax: -81.19733 ymax: 30.16113
## z_range:       zmin: 0 zmax: 0
## Geodetic CRS:  WGS 84
```

```
GTMnew2 <- st_union(GTMnew)
GTMnew <- subset(nerrs, nerrs$SITE_NAME=="Guana Tolomato Matanzas National Estuarine Research Reserve")
GTMnew$geometry <- GTMnew2
othernerrs <- subset(nerrs, nerrs$SITE_NAME != "Guana Tolomato Matanzas National Estuarine Research Rese
oysamplelocs <- st_read("data/OysterGIS_files/SEACAR_SampleLocationMatching/SampleLocations_12jan21/seac
```

```
## Reading layer 'seacar_dbo_SampleLocation_Point' from data source
##   'C:\Users\jepanzik\Box\R Projects\SEACAR_Trend_Analyses\Oyster\data\OysterGIS_files\SEACAR_SampleLo
##   using driver 'ESRI Shapefile'
## replacing null geometries with empty geometries
## Simple feature collection with 351136 features and 5 fields (with 6 geometries empty)
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -97.4537 ymin: -80.1108 xmax: 26.9509 ymax: 38.06
## Geodetic CRS:  WGS 84
```

```
oysterprogs <- unique(oysterraw$ProgramID)
oysamplelocs <- subset(oysamplelocs, oysamplelocs$ProgramID %in% oysterprogs)

#Make sure spatial data are in the same projection
aps_m <- st_transform(aps, 32119)
GTMnew_m <- st_transform(GTMnew, 32119)
othernerrs_m <- st_transform(nerrs, 32119)
fwcoymap_m <- st_transform(fwcoymap, 32119)
oysamplelocs_m <- st_transform(oysamplelocs, 32119)

#Create oyster map file for RCP managed areas
fwcoymap_m_aps <- fwcoymap_m[aps_m, , op=st_intersects]
fwcoymap_m_othernerrs <- fwcoymap_m[othernerrs_m, , op=st_intersects]
fwcoymap_m_GTMnew <- fwcoymap_m[GTMnew_m, , op=st_intersects]
fwcoymap_m_rcp <- unique(rbind(fwcoymap_m_aps, fwcoymap_m_othernerrs))
fwcoymap_m_rcp <- unique(rbind(fwcoymap_m_rcp, fwcoymap_m_GTMnew))

#Create dataframe of oyster sample locations within RCP managed areas
#that will be used to crosswalk reefIDs from different programIDs
reefcrosswalk_aps <- st_join(oysamplelocs_m, aps_m["LONG_NAME"],
                             join=st_intersects)
setnames(reefcrosswalk_aps, "LONG_NAME", "SITE_NAME")
reefcrosswalk_aps <- subset(reefcrosswalk_aps,
                            !is.na(reefcrosswalk_aps$SITE_NAME))
reefcrosswalk_othernerrs <- st_join(oysamplelocs_m, othernerrs_m["SITE_NAME"],
                                    join=st_intersects)
reefcrosswalk_othernerrs <- subset(reefcrosswalk_othernerrs,
                                   !is.na(reefcrosswalk_othernerrs$SITE_NAME))
reefcrosswalk_GTMnew <- st_join(oysamplelocs_m, GTMnew_m["SITE_NAME"],
                                join=st_intersects)
reefcrosswalk_GTMnew <- subset(reefcrosswalk_GTMnew,
                               !is.na(reefcrosswalk_GTMnew$SITE_NAME))
reefcrosswalk_rcp <- unique(rbind(reefcrosswalk_aps, reefcrosswalk_othernerrs))
reefcrosswalk_rcp <- unique(rbind(reefcrosswalk_rcp, reefcrosswalk_GTMnew))

#Need to make sure that samples outside of MA boundaries but taken from reefs
#that are partially within the MA boundaries are included.
reefcrosswalk_oymap <- st_join(oysamplelocs_m, fwcoymap_m_rcp["OBJECTID"],
                               join=st_intersects)
st_geometry(reefcrosswalk_rcp) <- NULL
reefcrosswalk_rcp <- dplyr::left_join(reefcrosswalk_oymap, reefcrosswalk_rcp)

#Create column to record the closest reef to each sample
reefcrosswalk_rcp$closest <- c(1:nrow(reefcrosswalk_rcp))
for(i in seq_len(nrow(reefcrosswalk_rcp))){
      reefcrosswalk_rcp$closest[i] <- fwcoymap_m_rcp[
            which.min(st_distance(fwcoymap_m_rcp,
                                  reefcrosswalk_rcp[i,])),]$OBJECTID
}

#Create match category column to record reef match (or no match) for each sample
reefcrosswalk_rcp$match_cat <- c(1:nrow(reefcrosswalk_rcp))
for(i in seq_len(nrow(reefcrosswalk_rcp))){
```

```
      obj_id <- subset(fwcoymap_m_rcp, fwcoymap_m_rcp$OBJECTID ==
                              reefcrosswalk_rcp$closest[i])
      reefcrosswalk_rcp$match_cat[i] <- ifelse(st_is_within_distance(
            reefcrosswalk_rcp[i,], obj_id, dist=20, sparse=FALSE),
            reefcrosswalk_rcp$closest[i], "no match")
}

#Create a match index column that will provide unique values for
#each sample location (so sampleloc metadata will show correctly on the map)
reefcrosswalk_rcp$match_ind <- rep("1", times=nrow(reefcrosswalk_rcp))
for(i in unique(reefcrosswalk_rcp$match_cat)){
      match <- subset(reefcrosswalk_rcp, reefcrosswalk_rcp$match_cat==i)
      match$match_ind <- NULL

      #need a reference table for match indexes because some samples appear
      #in overlapping managed areas
      match_u <- match[, c(1:6, 8:10)]
      match_u$geometry <- NULL
      match_u <- unique(match_u)

      match_u$match_ind <- rep("1", times=length(match_u$match_cat))

      #create index
      for(j in seq_len(nrow(match_u))){
            match_u$match_ind[j] <- paste0(match_u$match_cat[j], "_", j)
      }

      #use reference index table to add indexes to the full data subset
      #for the match category
      match <- left_join(match, match_u)

      #replace match category data in reef crosswalk table with data updated
      #with match indexes
      everythingelse <- subset(reefcrosswalk_rcp, reefcrosswalk_rcp$match_cat != i)
      reefcrosswalk_rcp <- rbind(everythingelse, match)
}


#Add match category to the FWC oyster map for RCP managed areas
reefcrosswalk_rcp_sum <- reefcrosswalk_rcp %>% dplyr::count(match_cat)
matches <- as.integer(subset(reefcrosswalk_rcp,
                              reefcrosswalk_rcp$match_cat != "no match")$match_cat)
fwcoymap_m_rcp$match <- ifelse(fwcoymap_m_rcp$OBJECTID %in% matches,
                               fwcoymap_m_rcp$OBJECTID, "no match")

#Create crosswalk reef ID column
for(i in seq_len(nrow(reefcrosswalk_rcp))){
      reefcrosswalk_rcp$crosswalk[i] <- ifelse(
            reefcrosswalk_rcp$match_cat[i] != "no match",
            reefcrosswalk_rcp$match_cat[i], reefcrosswalk_rcp$LocationID[i])
}

#manually adjust crosswalk values for some reefs in Estero Bay where the
```

```
#polygons appear to have plotted inaccurately
#samples_to_correct <- c(101957, 101956, 918388, 101955, 918389, 918387, 101945, 918335)
#correct_reef_matches <- c(136121, 136117, 136120, 136119, 136119, 136119, 136064, 136064)
samples_to_correct <- c(918390, 101956, 918388, 101955, 918389, 918387, 918337,
                        918335)
correct_reef_matches <- c(171071, 171067, 171069, 171069, 171069, 171069,
                          171014, 171014)

for(i in 1:length(samples_to_correct)){
      sample_to_correct <- subset(reefcrosswalk_rcp,
                                  reefcrosswalk_rcp$LocationID ==
                                        samples_to_correct[i])
      sample_to_correct$crosswalk <- correct_reef_matches[i]
      allothersamples <- subset(reefcrosswalk_rcp,
                                reefcrosswalk_rcp$LocationID !=
                                      samples_to_correct[i])
      reefcrosswalk_rcp <- rbind(allothersamples, sample_to_correct)
}

#Remove samples that were not either within a managed area or matched to a
#reef that is at least partially within a managed area.
reefcrosswalk_rcp_MA <- subset(reefcrosswalk_rcp,
                               !is.na(reefcrosswalk_rcp$SITE_NAME))
reefcrosswalk_rcp_nMA <- subset(reefcrosswalk_rcp,
                                is.na(reefcrosswalk_rcp$SITE_NAME))
reefcrosswalk_rcp_nMA <- subset(reefcrosswalk_rcp_nMA,
                                !is.na(reefcrosswalk_rcp_nMA$OBJECTID))
reefcrosswalk_rcp <- rbind(reefcrosswalk_rcp_MA, reefcrosswalk_rcp_nMA)

#Fix the special cases where a sample should have been included, but was
#outside both MA and reef boundaries.
samples_to_add <- c(864711, 864856, 918365, 945699, 945698, 78218, 918364,
                    864592)
matches_for_samples <- c("192956", "171697", "168801", "Unknown reef",
                         "Unknown reef", "175231", "168801", "192956")
samples_to_add <- data.frame(samples_to_add, matches_for_samples)
setnames(samples_to_add, c("samples_to_add", "matches_for_samples"),
         c("LocationID", "crosswalk"))
samples_to_add$SITE_NAME <- c(
      "Loxahatchee River-Lake Worth Creek Aquatic Preserve",
      "Loxahatchee River-Lake Worth Creek Aquatic Preserve",
      "Lemon Bay Aquatic Preserve", "St. Martins Marsh Aquatic Preserve",
      "St. Martins Marsh Aquatic Preserve",
      "Guana Tolomato Matanzas National Estuarine Research Reserve",
      "Lemon Bay Aquatic Preserve",
      "Loxahatchee River-Lake Worth Creek Aquatic Preserve")
missedsamps <- subset(reefcrosswalk_oymap,
                      reefcrosswalk_oymap$LocationID %in%
                            samples_to_add$LocationID)
missedsamps$closest <- NA
missedsamps$match_cat <- NA
missedsamps$match_ind <- NA
missedsamps <- dplyr::left_join(missedsamps, samples_to_add, by="LocationID")
```

```
reefcrosswalk_rcp <- rbind(reefcrosswalk_rcp, missedsamps)


#Not sure why, but in the end SA22 ends up with SITE_NAME blank, so I
#correct it manually here.
reefcrosswalk_rcp$SITE_NAME[reefcrosswalk_rcp$ProgramLoc=="SA22"] <-
      "Guana Tolomato Matanzas National Estuarine Research Reserve"



#Parallel version of previous <- THIS FUNCTION WORKS NOW.
registerDoFuture()
no_cores <- availableCores()-1
plan(multisession, workers=no_cores)

tic()
oysterraw$UniversalReefID <- foreach(i=seq_len(nrow(oysterraw)), .packages =
                                           c('data.table')) %dorng% {
                                                 ifelse(oysterraw$ProgramLocationID[i] %in% reefcrosswal
                                                        subset(reefcrosswalk_rcp,
                                                               reefcrosswalk_rcp$ProgramLoc ==
                                                                     oysterraw$ProgramLocationID[i])$cr
                                           }
toc()
```

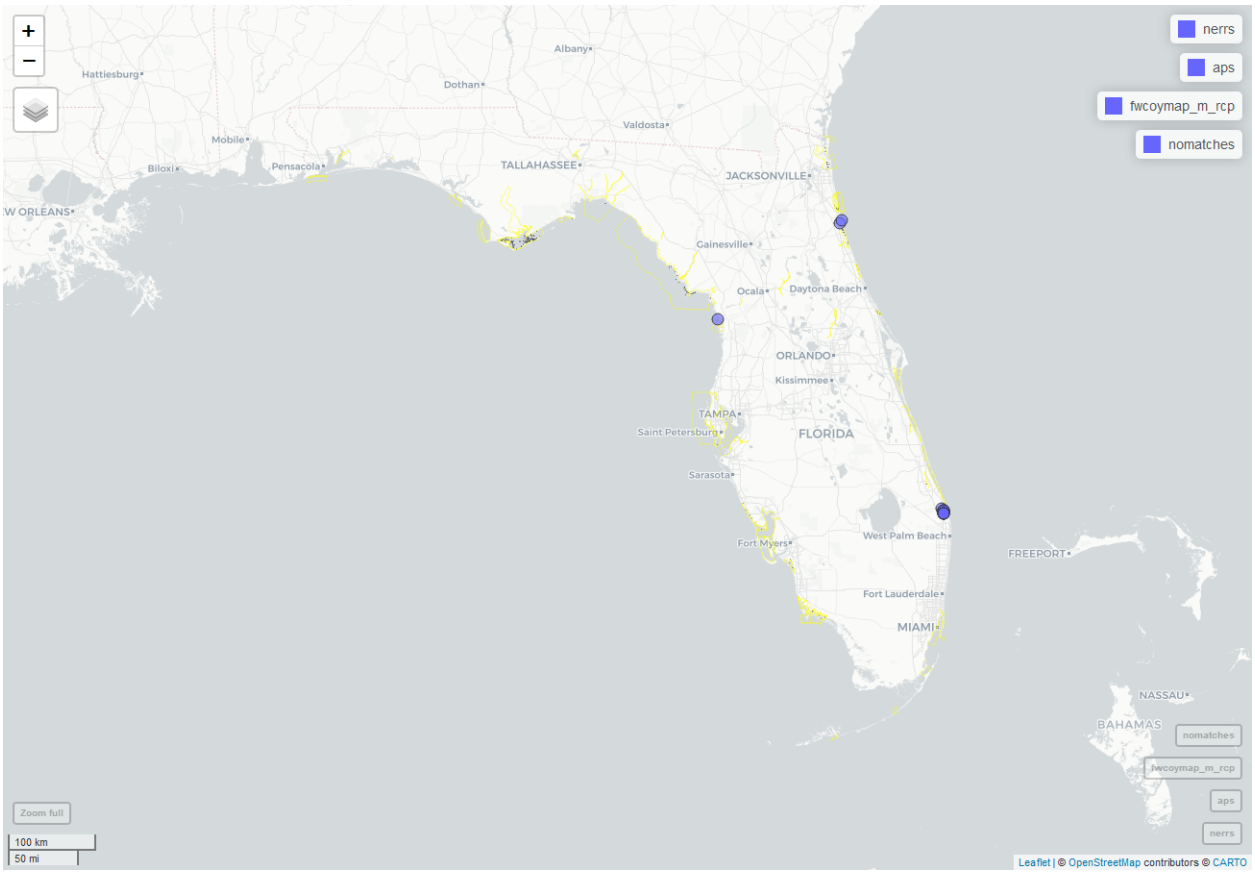
```
## 235.27 sec elapsed
```

```
oysterraw[, UniversalReefID := as.character(UniversalReefID)]


#Create a list of all ProgramLocationIDs in the full oysterraw data table
#(before "no matches" are removed)
shLocations <- unique(oysterraw$ProgramLocationID)

#Verify visually that all "no matches" in oysterraw_sh are from outside RCP
#managed area boundaries. If they are not, then go back to line 454 and
#correct the special cases.
nomatches_u <- unique(subset(oysterraw,
                             oysterraw$UniversalReefID ==
                                   "no match")$ProgramLocationID)
nomatches <- subset(oysamplelocs_m, oysamplelocs_m$ProgramLoc %in% nomatches_u)

mapview(nerrs, color="yellow", alpha.regions=0) +
      mapview(aps, color="yellow", alpha.regions=0) +
      #mapview(fwcoymap_m) +
      mapview(fwcoymap_m_rcp) +
      mapview(nomatches)
```




```
#Remove "no matches" from the oyster program data
oysterraw <- subset(oysterraw, oysterraw$UniversalReefID != "no match")

#Save a .RDS copies to avoid having to run this time consuming code every time
#I need the oysterraw file or crosswalk map
saveRDS(oysterraw, paste0("oysterraw_", Sys.Date(), ".rds"))
saveRDS(othernerrs,
        'data/OysterGIS_files/reefIDcrosswalk_map_files/othernerrs.rds')
saveRDS(GTMnew, 'data/OysterGIS_files/reefIDcrosswalk_map_files/GTMnew.rds')
saveRDS(aps, 'data/OysterGIS_files/reefIDcrosswalk_map_files/aps.rds')
saveRDS(oysamplelocs_m,
        'data/OysterGIS_files/reefIDcrosswalk_map_files/oysamplelocs_m.rds')
saveRDS(fwcoymap_m_rcp,
        'data/OysterGIS_files/reefIDcrosswalk_map_files/fwcoymap_m_rcp.rds')
saveRDS(reefcrosswalk_rcp,
        'data/OysterGIS_files/reefIDcrosswalk_map_files/reefcrosswalk_rcp.rds')


# Assign GTM regions to GTM data ------------------------------------------

#Set up data components to look at GTM reef regions
gtmn <- oysterraw[ManagedAreaName ==
                        "Guana Tolomato Matanzas National Estuarine Research Reserve", ]
#gtmn_regions <- fread(here::here("GTMregions.csv"))

#temporarily load reefcrosswalk_rcp file until UniversalReefID generation code
```

```
#is fixed
reefcrosswalk_rcp <-
      readRDS("data/OysterGIS_files/reefIDcrosswalk_map_files/reefcrosswalk_rcp.rds")

#Used gtmn_regions file to create shape files for the regions within
#GTMNERR boundaries in Google Earth
gtm_oyregions <- st_read("data/OysterGIS_files/GTMNERR_Regions.kml")
```

```
## Reading layer 'GTMNERR Regions' from data source
##   'C:\Users\jepanzik\Box\R Projects\SEACAR_Trend_Analyses\Oyster\data\OysterGIS_files\GTMNERR_Regions
## Simple feature collection with 7 features and 2 fields
## Geometry type: POLYGON
## Dimension:     XYZ
## Bounding box:  xmin: -81.38971 ymin: 29.60125 xmax: -81.20064 ymax: 30.16099
## z_range:       zmin: 0 zmax: 0
## Geodetic CRS:  WGS 84
```

```
gtm_oyregions_m <- st_transform(gtm_oyregions, 32119)

#Assign regions to GTM samples using region polygons
reefcrosswalk_rcp_gtm <- subset(reefcrosswalk_rcp,
                                reefcrosswalk_rcp$SITE_NAME ==
                                      "Guana Tolomato Matanzas National Estuarine Research Reserve")
reefcrosswalk_rcp_gtm_m <- st_transform(reefcrosswalk_rcp_gtm, 32119)
reefcrosswalk_rcp_gtm_m <- st_join(reefcrosswalk_rcp_gtm_m,
                                   gtm_oyregions_m["Name"],
                                   join=st_intersects)
setDT(reefcrosswalk_rcp_gtm_m)
reefcrosswalk_rcp_gtm_m[, ManagedAreaName :=
                            "Guana Tolomato Matanzas National Estuarine Research Reserve"]

rc_reg <- reefcrosswalk_rcp_gtm_m[, c("crosswalk", "Name", "ManagedAreaName")]
setnames(rc_reg, c("crosswalk", "Name"), c("UniversalReefID", "Region"))

#Add region names to the raw data file for the oyster analyses
oysterraw2 <- dplyr::left_join(oysterraw, rc_reg, by=c("ManagedAreaName",
                                                       "UniversalReefID"))
oysterraw2 <- unique(oysterraw2)

#Test that all GTM NERR data rows have a region name (i.e., no "NA"s)
gtmtest <- oysterraw2[ManagedAreaName ==
                          "Guana Tolomato Matanzas National Estuarine Research Reserve", ]
unique(gtmtest$Region.y)
```

```
## [1] "Pellicer Creek" "Tolomato River" "Butler Beach"   "Fort Matanzas"  "Guana River"    "Salt Run"
```

```
#Update oysterraw object and delete oysterraw2
oysterraw <- oysterraw2
rm(oysterraw2)

#Correct Pellicer Flats region name
oysterraw[Region.y=="Pellicer Creek", Region.y := "Pellicer Flats"]
```

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),
                  Region.x=as.character(Region.x),
                  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),
                  DataFileName=as.character(DataFileName),
                  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),
                  Region.y=as.character(Region.y))]

#Fix QuadID and ReefID columns for 2003 data in program 4014
```

```
#***this will not work because the Number_of_Oysters_Counted_Live_Count
#column is no longer populated for this program in the newest combined table.
#I put in a ticket with Claude to fix it.
oysterraw[ProgramID==4014 & Year==2003, `:=`
          (QuadIdentifier=ProgramLocationID, ReefIdentifier =
                     fcase(ProgramLocationID ==
                                 "14", "13",
                           ProgramLocationID=="13", "12",
                           ProgramLocationID=="12", "11",
                           as.numeric(ProgramLocationID) < 12,
                           ProgramLocationID),
                Density_m2 =
                     Number_of_Oysters_Counted_Live_Count/as.numeric(QuadSize_m2))]

#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"]

#Fix multiple spellings of PercentLiveMethod categories
oysterraw[, PercentLiveMethod :=
                fcase(PercentLiveMethod=="Point-Intercept", "Point-intercept",
                      PercentLiveMethod=="percent", "Percent")]



#make sure quadrat identifiers are unique
oysterraw[, QuadIdentifier_old := QuadIdentifier]
oysterraw[, QuadIdentifier := paste(UniversalReefID,
                                    LocationID, Year, Month,
                                    QuadIdentifier_old, sep="_")]
#Note that these QuadIdentifier values DO NOT end up being unique for ReefHeight_mm

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,
                      #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?
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_Den_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_Den_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_Den_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_Den_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_Pct_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_Pct_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_Pct_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_Pct_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.align = 0,
         axis.title.x = element_text(size=10, margin = margin(t = 5, r = 0,
                                                              b = 10, l = 0)),
         axis.title.y = element_text(size=10, margin = margin(t = 0, r = 10,
                                                              b = 0, l = 0)),
         axis.text=element_text(size=10),
         axis.text.x=element_text(angle = -45, hjust = 0))

#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 <- paste0(gsub('\\b(\\pL)\\pL{2,}|.','\\U\\1', managedarea, perl=TRUE),
```

```
            ifelse(str_detect(managedarea, "NERR|National E"), "ERR",
                   ifelse(str_detect(managedarea, "NMS|National M"),
                          "MS", "AP")))
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") ~ ">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"))

    ggsave(paste0("output/Figures/", ind, "_AllDates_GLMM_", ma, "_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 <- paste0(gsub('\\b(\\pL)\\pL{2,}|.','\\U\\1', managedarea, perl=TRUE),
```

```
            ifelse(str_detect(managedarea, "NERR|National E"), "ERR",
                   ifelse(str_detect(managedarea, "NMS|National M"), "MS",
                          "AP")))
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") ~ ">75mm",
                       TRUE ~ "raw")
}


if(ind=="SH"){
  #Marginal effects plot including random effects
  nyrs <- (max(data[!is.na(RelYear), RelYear])+1) -
    (min(data[!is.na(RelYear), RelYear])+1)
  maxyr <- max(data[!is.na(RelYear), RelYear])
  minyr <- min(data[!is.na(RelYear), RelYear]) #+ 1
  if(minyr > 0){
    maxyr <- maxyr-(minyr-1)
    minyr <- minyr-(minyr-1)
  }
  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)
  }
  yrlist <- c(min(data[!is.na(LiveDate), LiveDate]):max(data[!is.na(LiveDate),
                                                             LiveDate]))

  set.seed(987)
  if(length(models)==2){
    liveplot_1 <- plot(conditional_effects(models[[1]],
                                           re_formula=NULL),
                       plot=FALSE)
    histplot_1 <- plot(conditional_effects(models[[2]],
                                           re_formula=NULL),
                       plot=FALSE)
    present <- "Both"
  } else{
```

```
  if(str_detect(models[[1]]$file, "hist")){
    histplot_1 <- plot(conditional_effects(models[[1]],
                                           re_formula=NULL),
                       plot=FALSE)
    present <- "hist"
  } else{
    liveplot_1 <- plot(conditional_effects(models[[1]],
                                           re_formula=NULL),
                       plot=FALSE)
    present <- "live"
  }
}

plot1 <- ggplot() +
  geom_jitter(data=data[!is.na(RelYear) & !is.na(LiveDate), ],
              aes(x=RelYear, y=ShellHeight_mm), shape=21,
              size=3, color="#333333", fill="#cccccc",
              alpha=0.75, inherit.aes=FALSE) +
  {if(class(try(liveplot_1, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__),
                     fill="grey", alpha=0.4),
         geom_line(data=liveplot_1$RelYear$data,
                   aes(x=RelYear, y=estimate__,
                       color="live"), lwd=1))
  }} +
  {if(class(try(histplot_1, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=histplot_1$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__),
                     fill="grey", alpha=0.4,
                     inherit.aes=FALSE),
         geom_line(data=histplot_1$RelYear$data,
                   aes(x=RelYear, y=estimate__,
                       color="hist"),
                   lwd=1, inherit.aes=FALSE))
  }} +
  scale_x_continuous(breaks=breaks, labels=c(yrlist[breaks])) +
  plot_theme +
  theme(legend.position="right"
        #legend.position=ifelse(zoom==TRUE, "none", "right")
  ) +
  labs(subtitle=managedarea,
       subtitle=sizelab,
       x="Year",
       y="Shell height (mm)") +
  scale_color_manual(name="Trendlines",
                     values=c("hist"="red", "live"="#000099"),
                     labels=c("Dead Shells", "Live Oysters")) +
  coord_cartesian(ylim=c(ifelse(size=="25to75", 20,
                                ifelse(size=="35to75", 35, 70)),
                         ifelse(size=="o75", 250, 80)))
```

```
ggsave(paste0("output/Shell_Height/Figures/Oyster_SH_GLMM_", ma, ".png"),
       plot1,
       width=8,
       height=4,
       units="in",
       dpi=200)

if(zoom==TRUE){
  zoomplot <- plot1 +
    {if(length(models)==1)
      geom_boxplot(data=subset(data,
                               data$LiveDate_Qualifier=="Exact" &
                                 !is.na(data$RelYear) &
                                 !is.na(data$LiveDate)),
                   aes(x=RelYear, y=ShellHeight_mm,
                       group=LiveDate_Qualifier),
                   color="blue", alpha=0.5, lwd=1,
                   inherit.aes=FALSE)} +
    coord_cartesian(xlim=c(ifelse(min(data[LiveDate_Qualifier=="Exact" &
                                             !is.na(RelYear) &
                                             !is.na(LiveDate),
                                           RelYear]) -
                                    max(data[LiveDate_Qualifier==
                                               "Estimate" &
                                               !is.na(RelYear) &
                                               !is.na(LiveDate),
                                             RelYear]) > 50,
                                  min(data[LiveDate_Qualifier=="Exact" &
                                             !is.na(RelYear) &
                                             !is.na(LiveDate),
                                           RelYear]-5),
                                  max(data[LiveDate_Qualifier ==
                                             "Estimate" &
                                             !is.na(RelYear) &
                                             !is.na(LiveDate),
                                           RelYear]-5)),
                           ifelse(min(data[LiveDate_Qualifier =="Exact" &
                                             !is.na(RelYear) &
                                             !is.na(LiveDate),
                                           RelYear]) -
                                    max(data[LiveDate_Qualifier ==
                                               "Estimate" &
                                               !is.na(RelYear) &
                                               !is.na(LiveDate),
                                             RelYear]) > 50,
                                  max(data[LiveDate_Qualifier =="Exact" &
                                             !is.na(RelYear) &
                                             !is.na(LiveDate),
                                           RelYear]+5),
                                  max(data[LiveDate_Qualifier ==
                                             "Estimate" &
                                             !is.na(RelYear) &
                                             !is.na(LiveDate),
```

```
                                          RelYear]-5))),

                    ylim=c(ifelse(size=="25to75",
                                  20,
                                  ifelse(size=="35to75", 35,
                                         70)),
                           ifelse(size=="o75", 250, 80))) +
    theme(legend.position="right") +
    labs(title=paste0("post-",
                      yrlist[max(data[LiveDate_Qualifier==
                                        "Estimate" &
                                        !is.na(RelYear) &
                                        !is.na(LiveDate),
                                      RelYear])-5]),
         x="",
         y=NULL)

  cplot <- grid.arrange(grobs=list(plot1, zoomplot), ncol=2)

  ggsave(paste0("output/Shell_Height/Figures/Oyster_SH_GLMM_", ma,
                "_Zoom.png"),
         cplot,
         width=8,
         height=4,
         units="in",
         dpi=200)
}

if("Region.y" %in% if(present=="hist"){names(histplot_1)}
   else{names(liveplot_1)}){
  #Plot of modeled mean shell heights
  meanSH_test_hist <- histplot_1$Region.y$data
  meanSH_test <- liveplot_1$Region.y$data
  meanSH_test_hist$data <- "Historical data"
  meanSH_test$data <- "Real-time data"
  meanSH <- rbind(meanSH_test[, c("effect1__", "estimate__",
                                  "se__", "lower__", "upper__",
                                  "data")],
                  meanSH_test_hist[, c("effect1__",
                                       "estimate__", "se__",
                                       "lower__", "upper__",
                                       "data")])
  setnames(meanSH, c("effect1__"), c("Region"))

  meanshplot <- ggplot(meanSH, aes(x=Region, y=estimate__,
                                   ymin=lower__, ymax=upper__,
                                   fill=data)) +
    geom_pointinterval(
      position=position_jitter(width=0.25, height=0),
      size=3, fatten_point=4, shape=21, color="black") +
    ylab(paste0("ShellHeight_mm | ",
                ifelse(size=="25to75", "trunc(lb=25, ub=75)",
                       ifelse(size=="35to75",
```

```
                                "trunc(lb=35, ub=75)",
                                "trunc(lb=75, ub=250)")))) +
      theme_bw()+
      theme(axis.title=element_text(size=13),
            axis.text=element_text(size=12),
            legend.text=element_text(size=12),
            legend.title=element_text(size=13),
            axis.text.x=element_text(angle=45, hjust=1)) +
      labs(fill=NULL)

    ggsave(paste0("output/Shell_Height/Figures/Oyster_SH_GLMM_", ma,
                  "MeanRes", ".png"),
           meanshplot,
           width=8,
           height=4,
           units="in",
           dpi=200)
  }
}



if(ind=="Den"){
  nyrs <- (max(data$RelYear)+1)-(min(data$RelYear)+1)
  maxyr <- max(data$RelYear)
  minyr <- min(data$RelYear)
  if(grepl("Natural", unique(data$MA_plotlab))==TRUE){
    type <- "Natural"
  } else{
    type <- "Restored"
  }
  if(minyr > 0){
    maxyr <- maxyr-(minyr-1)
    minyr <- minyr-(minyr-1)
  }
  nbreaks <- ifelse(nyrs < 11, nyrs+1, 12)
  breaks <- if(minyr==0){
    c(minyr, round(minyr+c(1:(nbreaks-2))*((nyrs/nbreaks) +
                                             (nyrs/nbreaks)/nbreaks)),
      maxyr)+1
  } else{
    c(minyr, round(minyr+c(1:(nbreaks-2))*((nyrs/nbreaks) +
                                             (nyrs/nbreaks)/nbreaks)),
      maxyr)
  }
  yrlist <- c(min(data$Year):max(data$Year))

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

  plot1 <- ggplot() +
    {if("meanDen_int" %in% colnames(data)){
      geom_jitter(data=data, aes(x=RelYear -
```

```
                                    (min(RelYear)-1),
                                  y=meanDen_int), shape=21,
                  size=2, color="#333333", fill="#cccccc",
                  alpha=1, inherit.aes=FALSE)
    } else{
      geom_jitter(data=data, aes(x=RelYear-(min(RelYear)-1),
                                  y=Density_m2), shape=21,
                  size=2, color="#333333", fill="#cccccc",
                  alpha=1, inherit.aes=FALSE)
    }
    } +
    list(geom_ribbon(data=denplots$RelYear$data,
                     aes(x=RelYear-(min(RelYear)-1), y=Density_m2,
                         ymin=lower__, ymax=upper__),
                     fill="#000099", alpha=0.5, inherit.aes=FALSE),
         geom_line(data=denplots$RelYear$data,
                   aes(x=RelYear-(min(RelYear)-1),
                       y=estimate__),
                   color="#000099", lwd=0.75, inherit.aes=FALSE)) +
    scale_x_continuous(breaks=breaks, labels=c(yrlist[breaks])) +
    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}*')')))



  ggsave(paste0("output/Density/Figures/Oyster_Den_GLMM_", ma, "_", 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$RelYear)+1)-(min(data$RelYear)+1)
maxyr <- max(data$RelYear)
minyr <- min(data$RelYear)
if(minyr > 0){
  maxyr <- maxyr-(minyr-1)
  minyr <- minyr-(minyr-1)
}
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)
}
yrlist <- c(min(data$Year):max(data$Year))

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

plot1 <- ggplot() +
  geom_jitter(data=data, aes(x=RelYear-(min(RelYear)-1),
                             y=100*PercentLive_dec),
              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-(min(RelYear)-1),
                         y=100*PercentLive_dec, ymin=100*lower__,
                         ymax=100*upper__), fill="#000099",
                     alpha=0.5, inherit.aes=FALSE),
         geom_line(data=pctplots$RelYear$data,
                   aes(x=RelYear-(min(RelYear)-1),
                       y=100*estimate__), color="#000099",
                   lwd=0.75, inherit.aes=FALSE))
  } else{
    list(geom_ribbon(data=pctplots$RelYear$data,
                     aes(x=RelYear-(min(RelYear)-1),
                         y=100*LiveObs, ymin=100*lower__,
                         ymax=100*upper__), fill="#000099",
                     alpha=0.5, inherit.aes=FALSE),
         geom_line(data=pctplots$RelYear$data,
                   aes(x=RelYear-(min(RelYear)-1),
                       y=100*estimate__), color="#000099",
                   lwd=0.75, inherit.aes=FALSE))
  }} +
  scale_x_continuous(breaks=breaks, labels=c(yrlist[breaks])) +
  plot_theme +
  labs(title="Oyster Percent Live Cover",
       subtitle=managedarea,
```

```
        x="Year",
        y="Live cover (%)") +
  theme(legend.text=element_text(size=10),
        legend.title=element_text(size=10))

ggsave(paste0("output/Percent_Live/Figures/Oyster_Pct_GLMM_", ma,
              "_raw.png"),
       plot1,
       width=8,
       height=4,
       units="in",
       dpi=200)

#Plot of modeled mean percent live
if("Region.y" %in% names(pctplots)){
  meanPct <- pctplots$Region.y$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))

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

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


  intplot <- RelYrbyRegion +
    geom_point(data=data, aes(x=RelYear-(min(RelYear)-1),
                              y=PercentLive_dec,
                              fill=Region.y),
               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.y, ncol=3, scales="free")

      ggsave(paste0("output/Percent_Live/Figures/Oyster_Pct_GLMM_", ma,
                    "_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 <- paste0(gsub('\\b(\\pL)\\pL{2,}|.','\\U\\1', managedarea, perl=TRUE),
             ifelse(str_detect(managedarea, "NERR|National E"), "ERR",
                    ifelse(str_detect(managedarea, "NMS|National M"),
                           "MS", "AP")))
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") ~ ">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") ~ ">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(25, 300, 50)
y_labs <- seq(25, 300, 50)
y_minor <- seq(25, 300, 25)
ylim_upper <- ceiling(y_max/25)*25

maxyr_hist <- max(data1[!is.na(RelYear) & LiveDate_Qualifier=="Estimate",
                        RelYear],
                  data2[!is.na(RelYear) & LiveDate_Qualifier=="Estimate",
                        RelYear])
minyr_hist <- min(data1[!is.na(RelYear) & LiveDate_Qualifier=="Estimate",
```

```
                        RelYear],
                  data2[!is.na(RelYear) & LiveDate_Qualifier=="Estimate",
                        RelYear])
nyrs_hist <- (maxyr_hist+1)-(minyr_hist+1)
if(minyr_hist > 0){
  maxyr_hist <- maxyr_hist-(minyr_hist-1)
  minyr_hist <- minyr_hist-(minyr_hist-1)
}
nbreaks_hist <- ifelse(nyrs_hist < 11, nyrs_hist+1, 12)
breaks_hist <- if(minyr_hist==0){
  c(minyr_hist, round(minyr_hist+c(1:(nbreaks_hist-2))*
                        ((nyrs_hist/nbreaks_hist)+
                           (nyrs_hist/nbreaks_hist)/nbreaks_hist)),
    maxyr_hist)+1
} else{
  c(minyr_hist, round(minyr_hist+c(1:(nbreaks_hist-2))*
                        ((nyrs_hist/nbreaks_hist)+
                           (nyrs_hist/nbreaks_hist)/nbreaks_hist)),
    maxyr_hist)
}
yrlist_hist <- c(min(data1[!is.na(LiveDate) & LiveDate_Qualifier==
                             "Estimate",LiveDate],
                     data2[!is.na(LiveDate) & LiveDate_Qualifier==
                             "Estimate", LiveDate]):
                   max(data1[!is.na(LiveDate) & LiveDate_Qualifier==
                               "Estimate", LiveDate],
                       data2[!is.na(LiveDate) & LiveDate_Qualifier==
                               "Estimate", LiveDate]))

## Live plot settings
maxyr_live <- max(data1[!is.na(RelYear) & LiveDate_Qualifier=="Exact",
                        RelYear],
                  data2[!is.na(RelYear) & LiveDate_Qualifier=="Exact",
                        RelYear])
minyr_live <- min(data1[!is.na(RelYear) & LiveDate_Qualifier=="Exact",
                        RelYear],
                  data2[!is.na(RelYear) & LiveDate_Qualifier=="Exact",
                        RelYear])
nyrs_live <- (maxyr_live+1)-(minyr_live+1)
nbreaks_live <- ifelse(nyrs_live < 11, nyrs_live+1, 12)
breaks_live <- if(minyr_live==0){
  c(minyr_live, round(minyr_live+c(1:(nbreaks_live-2))*
                        ((nyrs_live/nbreaks_live)+
                           (nyrs_live/nbreaks_live)/nbreaks_live)),
    maxyr_live)+1
} else{
  c(minyr_live, round(minyr_live+c(1:(nbreaks_live-2))*
                        ((nyrs_live/nbreaks_live)+
                           (nyrs_live/nbreaks_live)/nbreaks_live)),
    maxyr_live)
}
yr_breaks_live <- breaks_live-min(breaks_live)+1
```

```
yrlist_live <- c(min(data1[!is.na(LiveDate) & LiveDate_Qualifier=="Exact",
                           LiveDate],
                     data2[!is.na(LiveDate) & LiveDate_Qualifier=="Exact",
                           LiveDate]):
                   max(data1[!is.na(LiveDate) & LiveDate_Qualifier=="Exact",
                             LiveDate],
                       data2[!is.na(LiveDate) & LiveDate_Qualifier=="Exact",
                             LiveDate]))
## 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)==TRUE){
  if(length(models1)==2){
    liveplot_1 <- plot(conditional_effects(models1[[1]],
                                           re_formula=NULL),
                       plot=FALSE)
    histplot_1 <- plot(conditional_effects(models1[[2]],
                                           re_formula=NULL),
                       plot=FALSE)
    present1 <- "Both"
  } else{
    if(str_detect(models1[[1]]$file, "hist")){
      histplot_1 <- plot(conditional_effects(models1[[1]],
                                             re_formula=NULL),
                         plot=FALSE)
      present1 <- "hist"
    } else{
      liveplot_1 <- plot(conditional_effects(models1[[1]],
                                             re_formula=NULL),
                         plot=FALSE)
      present1 <- "live"
    }
  }
}

if(!is.null(models2)==TRUE){
  if(length(models2)==2){
    liveplot_2 <- plot(conditional_effects(models2[[1]],
                                           re_formula=NULL),
                       plot=FALSE)
    histplot_2 <- plot(conditional_effects(models2[[2]],
                                           re_formula=NULL),
                       plot=FALSE)
    present2 <- "Both"
```

```
  } else{
    if(str_detect(models2[[1]]$file, "hist")){
      histplot_2 <- plot(conditional_effects(models2[[1]],
                                             re_formula=NULL),
                         plot=FALSE)
      present2 <- "hist"
    } else{
      liveplot_2 <- plot(conditional_effects(models2[[1]],
                                             re_formula=NULL),
                         plot=FALSE)
      present2 <- "live"
    }
  }
}


# Fixes issue with legend alpha values being added
a_ribb <- 0.5
if(class(try(histplot_1, silent=TRUE)) != "try-error" &
   class(try(liveplot_1, silent=TRUE)) != "try-error"){
  a_ribb <- 0.25
}else if(class(try(histplot_2, silent=TRUE)) != "try-error" &
         class(try(liveplot_2, silent=TRUE)) != "try-error"){
  a_ribb <- 0.25
}

#p_color <- c("size2"="#0094b0", "size1"="#00374f")
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)


plot_leg <- ggplot() +
  {if(class(try(histplot_1, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=histplot_1$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
```

```
                        ymin=lower__, ymax=upper__,
                        fill="size1"), alpha=a_ribb,
                    inherit.aes=FALSE),
        geom_line(data=histplot_1$RelYear$data,
                  aes(x=RelYear, y=estimate__, color="size1"),
                  lwd=0.75, inherit.aes=FALSE))

  }} +
  {if(class(try(histplot_2, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=histplot_2$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__,
                         fill="size2"), alpha=a_ribb,
                     inherit.aes=FALSE),
         geom_line(data=histplot_2$RelYear$data,
                   aes(x=RelYear, y=estimate__, color="size2"),
                   lwd=0.75, inherit.aes=FALSE))
  }} +
  {if(class(try(liveplot_1, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__,
                         fill="size1"), alpha=a_ribb),
         geom_line(data=liveplot_1$RelYear$data,
                   aes(x=RelYear, y=estimate__, color="size1"),
                   lwd=0.75))
  }} +
  {if(class(try(liveplot_2, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=liveplot_2$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size2"),
                     alpha=a_ribb),
         geom_line(data=liveplot_2$RelYear$data,
                   aes(x=RelYear, y=estimate__, color="size2"),
                   lwd=0.75))
  }} +
  geom_jitter(data=data1[!is.na(RelYear) & !is.na(LiveDate), ],
              aes(x=RelYear, y=ShellHeight_mm, shape="size1"),
              size=2, color="#333333", fill="#cccccc",
              alpha=1, inherit.aes=FALSE, width=0.1, height=0.1) +
  geom_jitter(data=data2[!is.na(RelYear) & !is.na(LiveDate), ],
              aes(x=RelYear, y=ShellHeight_mm, shape="size2"),
              size=2, color="#333333", fill="#cccccc",
              alpha=1, inherit.aes=FALSE, width=0.1, height=0.1) +
  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)

plot1 <- ggplot() +
  geom_hline(yintercept=75, size=1, color="grey") +
  {if(n_hist1>0){
    geom_jitter(data=data1[!is.na(RelYear) &
                           !is.na(LiveDate) &
                           LiveDate_Qualifier=="Estimate", ],
                aes(x=RelYear, y=ShellHeight_mm, shape="size1"),
                size=2, color="#333333", fill="#cccccc",
                alpha=1, inherit.aes=FALSE, width=0.1, height=0.1)
  }} +
  {if(n_hist2>0){
    geom_jitter(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
                           LiveDate_Qualifier=="Estimate", ],
                aes(x=RelYear, y=ShellHeight_mm, shape="size2"),
                size=2, color="#333333", fill="#cccccc",
                alpha=1, inherit.aes=FALSE, width=0.1, height=0.1)
  }} +
  {if(class(try(histplot_1, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=histplot_1$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size1"),
                     alpha=0.5, inherit.aes=FALSE),
         geom_line(data=histplot_1$RelYear$data,
                   aes(x=RelYear, y=estimate__, color="size1"),
                   lwd=0.75, inherit.aes=FALSE))
  }} +
  {if(class(try(histplot_2, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=histplot_2$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size2"),
                     alpha=0.5, inherit.aes=FALSE),
         geom_line(data=histplot_2$RelYear$data,
                   aes(x=RelYear, y=estimate__, color="size2"),
                   lwd=0.75, inherit.aes=FALSE))
  }} +
  scale_x_continuous(breaks=breaks_hist,
                     labels=c(yrlist_hist[breaks_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(25, ylim_upper))

plot2 <- ggplot() +
  geom_hline(yintercept=75, size=1, color="grey") +
  {if(n_live1>0){
    geom_jitter(data=data1[!is.na(RelYear) & !is.na(LiveDate) &
                             LiveDate_Qualifier=="Exact", ],
                aes(x=RelYear, y=ShellHeight_mm, shape="size1"),
                size=2, color="#333333", fill="#cccccc",
                alpha=1, inherit.aes=FALSE, width=0.1, height=0.1)
  }} +
  {if(n_live2>0){
    geom_jitter(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
                             LiveDate_Qualifier=="Exact", ],
                aes(x=RelYear, y=ShellHeight_mm, shape="size2"),
                size=2, color="#333333", fill="#cccccc",
                alpha=1, inherit.aes=FALSE, width=0.1, height=0.1)
  }} +
  {if(class(try(liveplot_1, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size1"),
                     alpha=0.5),
         geom_line(data=liveplot_1$RelYear$data,
                   aes(x=RelYear, y=estimate__, color="size1"),
                   lwd=0.75))
  }} +
  {if(class(try(liveplot_2, silent=TRUE)) != "try-error"){
    list(geom_ribbon(data=liveplot_2$RelYear$data,
                     aes(x=RelYear, y=ShellHeight_mm,
                         ymin=lower__, ymax=upper__, fill="size2"),
                     alpha=0.5),
         geom_line(data=liveplot_2$RelYear$data,
                   aes(x=RelYear, y=estimate__, color="size2"),
                   lwd=0.75))
  }} +
  scale_x_continuous(breaks=breaks_live,
                     labels=c(yrlist_live[yr_breaks_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(25, ylim_upper))

  #leg <- get_legend(plot1)

  plot_title <- ggplot()+labs(title="Oyster Size Class",
                              subtitle=managedarea) +
    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))

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


# 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)
}

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[, `:=` (indicator=indicator,
                  managed_area=unique(datafile2$ManagedAreaName),
                  habitat_class=unique(datafile2$HabitatClassification),
                  size_class=sizeclass2,
                  live_date_qual=ifelse(
                    str_detect(modelobj$file, "_hist"),
                    "Estimate", "Exact"),
                  n_programs=if(class(
                    try(datafile2$LiveDate_Qualifier))!=
                    "try-error"){
                    length(
                      unique(
                        datafile2[LiveDate_Qualifier==
                                    ifelse(
                                      str_detect(
                                        modelobj$file,
                                        "_hist"),
                                      "Estimate",
                                      "Exact"),
                                  ProgramID]))
                  } else{length(unique(datafile2[, ProgramID]))},
                  programs=if(class(
                    try(datafile2$LiveDate_Qualifier)) !=
                    "try-error"){
                    list(
                      unique(
                        datafile2[LiveDate_Qualifier==
```

```
                                    ifelse(
                                      str_detect(
                                        modelobj$file,
                                        "_hist"),
                                      "Estimate",
                                      "Exact"),
                                    ProgramID]))
                    } else{list(unique(datafile2[, ProgramID]))},
                    filename=modelobj$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
  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.

```
#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('data/GLMMs/AllDates/Data/ab_sho25_', Sys.Date(), '.rds'))


### ABAP-25 to 75mm ------------------------------------------------------

ab_sh25to75 <- ab_sho25[ShellHeight_mm < 75, ]

saveRDS(ab_sh25to75, paste0('data/GLMMs/AllDates/Data/ab_sh25to75_',
                            Sys.Date(), '.rds'))


# Create model results tables and save diagnostic plots
data1 <- ab_sh25to75
#models1 <- list(ab_sh25to75_glmm_hist)
models1 <- NULL
#modresults(data, models, "Size class")


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

ab_sho75 <- ab_sho25[ShellHeight_mm >= 75, ]

saveRDS(ab_sho75, paste0('data/GLMMs/AllDates/Data/ab_sho75_', Sys.Date(), '.rds'))

ab_sho75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                                me(RelYear, SampleAge_Stdev, gr=QuadIdentifier)+
                                (1 | UniversalReefID),
                          data=subset(ab_sho75,
                                      ab_sho75$LiveDate_Qualifier=="Estimate"),
                          family=gaussian, cores=4,
                          control= list(adapt_delta=0.99, max_treedepth=15),
                          iter=3000, warmup=1000, chains=4, thin=3, seed=1115,
                          backend="cmdstanr", threads=threading(2),
                          file="data/GLMMs/AllDates/ab_sho75_glmm_hist2.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- ab_sho75
models2 <- list(ab_sho75_glmm_hist) #ab_sho75_glmm,
#modresults(data, models, "Size class", meplotzoom=TRUE)
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('data/GLMMs/AllDates/Data/an_sho25_', Sys.Date(), '.rds'))


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

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

saveRDS(an_sh25to75, paste0('data/GLMMs/AllDates/Data/an_sh25to75_',
                            Sys.Date(), '.rds'))

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=4,
                        control=list(adapt_delta=0.99, max_treedepth=15),
                        iter=3000, warmup=1000, chains=4, thin=3, seed=5699,
                        backend="cmdstanr", threads=threading(2),
                        file="data/GLMMs/AllDates/an_sh25to75_glmm4b.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- an_sh25to75
models1 <- list(an_sh25to75_glmm)
#modresults(data, models, "Size class", meplotzoom=TRUE)


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

an_sho75 <- an_sho25[ShellHeight_mm >= 75, ]

saveRDS(an_sho75, paste0('data/GLMMs/AllDates/Data/an_sho75_', Sys.Date(), '.rds'))

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=4,
                     control= list(adapt_delta=0.99, max_treedepth=15),
                     iter=3000, warmup=1000, chains=4, thin=3, seed=3639,
                     backend="cmdstanr", threads=threading(2),
                     file="data/GLMMs/AllDates/an_sho75_glmm4b.rds")

an_sho75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                            me(RelYear, SampleAge_Stdev, gr=QuadIdentifier)+
                            (1 | UniversalReefID),
                          data=subset(an_sho75,
                                      an_sho75$LiveDate_Qualifier=="Estimate"),
                          family=gaussian, cores=4,
                          control=list(adapt_delta=0.99,max_treedepth=15),
                          iter=3000, warmup=1000, chains=4, thin=3, seed=1313,
```

```
                          backend="cmdstanr", threads=threading(2),
                          file="data/GLMMs/AllDates/an_sho75_glmm_hist3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- an_sho75
models2 <- list(an_sho75_glmm, an_sho75_glmm_hist)
#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('data/GLMMs/AllDates/Data/eb_sho25_', Sys.Date(), '.rds'))



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

eb_sh25to75 <- subset(eb_sho25, eb_sho25$ShellHeight_mm < 75)

saveRDS(eb_sh25to75, paste0('data/GLMMs/AllDates/Data/eb_sh25to75_',
                            Sys.Date(), '.rds'))

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=4,
                        control= list(adapt_delta=0.99, max_treedepth=15),
                        iter=3000, warmup=1000, chains=4, thin=3, seed=6881,
                        backend="cmdstanr", threads=threading(2),
                        file="data/GLMMs/AllDates/eb_sh25to75_glmm5.rds")

eb_sh25to75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                                   me(RelYear, SampleAge_Stdev,
                                      gr=QuadIdentifier)+(1 | UniversalReefID),
                             data=subset(eb_sh25to75,
                                         eb_sh25to75$LiveDate_Qualifier=="Estimate"),
                             family=gaussian, cores=4,
                             control=list(adapt_delta=0.99, max_treedepth=20),
                             iter=3000, warmup=1000, chains=4, thin=3, inits=30,
                             seed=6874, backend="cmdstanr", threads=threading(2),
                             file="data/GLMMs/AllDates/eb_sh25to75_glmm_hist3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- eb_sh25to75
models1 <- list(eb_sh25to75_glmm, eb_sh25to75_glmm_hist)
#modresults(data, models, "Size class", meplotzoom=FALSE)
```

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

eb_sho75 <- eb_sho25[ShellHeight_mm >= 75, ]

saveRDS(eb_sho75, paste0('data/GLMMs/AllDates/Data/eb_sho75_', Sys.Date(), '.rds'))

eb_sho75_glmm <- brm(formula=ShellHeight_mm ~
                          RelYear+(1 | UniversalReefID),
                     data=subset(eb_sho75, eb_sho75$LiveDate_Qualifier=="Exact"),
                     family=gaussian, cores=4,
                     control=list(adapt_delta=0.99, max_treedepth=15), iter=3000,
                     warmup=1000, chains=4, thin=3, seed=3138,
                     backend="cmdstanr", threads=threading(2),
                     file="data/GLMMs/AllDates/eb_sho75_glmm4.rds")

eb_sho75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                               me(RelYear, SampleAge_Stdev, gr=QuadIdentifier)+
                               (1 | UniversalReefID),
                          data=subset(eb_sho75,
                                      eb_sho75$LiveDate_Qualifier=="Estimate"),
                          family=gaussian, cores=4,
                          control=list(adapt_delta=0.99, max_treedepth=20),
                          iter=3000, warmup=1000, chains=4, thin=3, seed=4127,
                          backend="cmdstanr", threads=threading(2),
                          file="data/GLMMs/AllDates/eb_sho75_glmm_hist3.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- eb_sho75
models2 <- list(eb_sho75_glmm, eb_sho75_glmm_hist)
#modresults(data, models, "Size class", meplotzoom=FALSE)
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('data/GLMMs/AllDates/Data/grm_sho25_',
                          Sys.Date(), '.rds'))


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

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

saveRDS(grm_sh25to75, paste0('data/GLMMs/AllDates/Data/grm_sh25to75_',
                           Sys.Date(), '.rds'))

grm_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                             RelYear+NumberMeasured_n+(1 | UniversalReefID),
                         data=subset(grm_sh25to75,
```

```
                                    grm_sh25to75$LiveDate_Qualifier=="Exact"),
                          family=gaussian, cores=4,
                          control= list(adapt_delta=0.8, max_treedepth=10),
                          iter=3000, warmup=1000, chains=4, inits=30, thin=3,
                          seed=3457, backend="cmdstanr", threads=threading(2),
                          file="data/GLMMs/AllDates/grm_sh25to75_glmm4.rds")

grm_sh25to75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                                      me(RelYear, SampleAge_Stdev,
                                         gr=QuadIdentifier)+(1 | UniversalReefID),
                              data=subset(grm_sh25to75,
                                          grm_sh25to75$LiveDate_Qualifier=="Estimate"),
                              family=gaussian,
                              prior=c(set_prior("normal(6.25, 7)",
                                                class="meanme",
                                                coef="meRelYear"),
                                      set_prior("normal(15.27, 5)",
                                                class="sdme",
                                                coef="meRelYear"),
                                      set_prior("cauchy(0,2)", class="sd")),
                              cores=4,
                              control= list(adapt_delta=0.99, max_treedepth=15),
                              iter=3000, warmup=1000, chains=4, thin=3, seed=3455,
                              backend="cmdstanr", threads=threading(2),
                              file="data/GLMMs/AllDates/grm_sh25to75_glmm_hist3c.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- grm_sh25to75
models1 <- list(grm_sh25to75_glmm, grm_sh25to75_glmm_hist)
#modresults(data, models, "Size class", meplotzoom=FALSE)


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

grm_sho75 <- grm_sho25[ShellHeight_mm >= 75, ]

saveRDS(grm_sho75, paste0('data/GLMMs/AllDates/Data/grm_sho75_',
                          Sys.Date(), '.rds'))

grm_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                            RelYear+NumberMeasured_n+(1 | UniversalReefID),
                      data=subset(grm_sho75,
                                  grm_sho75$LiveDate_Qualifier=="Exact"),
                      family=gaussian, cores=4,
                      control= list(adapt_delta=0.8, max_treedepth=10),
                      iter=3000, warmup=1000, chains=4, inits=30, thin=3,
                      seed=4352, backend="cmdstanr", threads=threading(2),
                      file="data/GLMMs/AllDates/grm_sho75_glmm4.rds")

grm_sho75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                                   me(RelYear, SampleAge_Stdev,
                                      gr=QuadIdentifier)+
                                   (0+me(RelYear,SampleAge_Stdev,
```

```
                                    gr=QuadIdentifier) | UniversalReefID),
                          data=subset(grm_sho75,
                                      grm_sho75$LiveDate_Qualifier=="Estimate"),
                          family=gaussian, prior=c(set_prior("normal(7.36, 6)",
                                                             class="meanme"),
                                                   set_prior("normal(15.54, 4)",
                                                             class="sdme"),
                                                   set_prior("cauchy(0,2)",
                                                             class="sd")),
                          cores=4, control= list(adapt_delta=0.99,
                                                 max_treedepth=15),
                          iter=3000, warmup=1000, chains=4, thin=3, seed=6784,
                          backend="cmdstanr", threads=threading(2),
                          file="data/GLMMs/AllDates/grm_sho75_glmm_hist4.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- grm_sho75
models2 <- list(grm_sho75_glmm, grm_sho75_glmm_hist)
#modresults(data, models, "Size class", meplotzoom=FALSE)
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('data/GLMMs/AllDates/Data/gtmn_sho25_',
                           Sys.Date(), '.rds'))


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

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

saveRDS(gtmn_sh25to75, paste0('data/GLMMs/AllDates/Data/gtmn_sh25to75_',
                              Sys.Date(), '.rds'))

gtmn_sh25to75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=25, ub=75) ~
                              RelYear+NumberMeasured_n+
                              Region.y+(1 | UniversalReefID),
                          data=subset(gtmn_sh25to75,
                                      gtmn_sh25to75$LiveDate_Qualifier != "Estimate"),
                          family=gaussian, cores=4,
                          control=list(adapt_delta=0.8, max_treedepth=10),
                          iter=3000, warmup=1000, chains=4, inits=30, thin=3,
                          seed=7844, backend="cmdstanr", threads=threading(2),
                          file="data/GLMMs/AllDates/gtmn_sh25to75_glmm5.rds")

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

```
models1 <- list(gtmn_sh25to75_glmm)
#modresults(data, models, "Size class", meplotzoom=FALSE)


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

gtmn_sho75 <- gtmn_sho25[ShellHeight_mm >= 75, ]

saveRDS(gtmn_sho75, paste0('data/GLMMs/AllDates/Data/gtmn_sho75_',
                           Sys.Date(), '.rds'))

gtmn_sho75_glmm <- brm(formula=ShellHeight_mm | trunc(lb=75) ~
                             RelYear+NumberMeasured_n+Region.y+
                             (0+RelYear | UniversalReefID),
                       data=subset(gtmn_sho75,
                                   gtmn_sho75$LiveDate_Qualifier != "Estimate"),
                       family=gaussian, prior=c(set_prior("normal(171,10)",
                                                          class="b",
                                                          coef="RelYear"),
                                                set_prior("cauchy(0,2)")),
                       cores=4, control=list(adapt_delta=0.99, max_treedepth=10),
                       iter=3000, warmup=1000, chains=4, inits=30, thin=3,
                       seed=5332, backend="cmdstanr", threads=threading(2),
                       file="data/GLMMs/AllDates/gtmn_sho75_glmm6.rds")

gtmn_sho75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                                  me(RelYear, SampleAge_Stdev,
                                     gr=QuadIdentifier)+Region.y+
                                  (1+RelYear | UniversalReefID),
                            data=subset(gtmn_sho75,
                                        gtmn_sho75$LiveDate_Qualifier=="Estimate"),
                            family=gaussian,
                            prior=c(set_prior("normal(146,25)",
                                              class="b",
                                              coef="meRelYearSampleAge_StdevgrEQQuadIdentifier")),
                            cores=4,
                            control= list(adapt_delta=0.99, max_treedepth=15),
                            iter=4000, warmup=1000, chains=4, thin=3, seed=4688,
                            backend="cmdstanr", threads=threading(2),
                            file="data/GLMMs/AllDates/gtmn_sho75_glmm_hist22.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- gtmn_sho75
models2 <- list(gtmn_sho75_glmm, gtmn_sho75_glmm_hist)
#modresults(data, models, "Size class", meplotzoom=FALSE)
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('data/GLMMs/AllDates/Data/irvbfp_sho25_',
                             Sys.Date(), '.rds'))


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

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

saveRDS(irvbfp_sh25to75, paste0('data/GLMMs/AllDates/Data/irvbfp_sh25to75_',
                                Sys.Date(), '.rds'))


# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- irvbfp_sh25to75
models1 <- NULL
#modresults(data, models, "Size class", meplotzoom=FALSE)


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

irvbfp_sho75 <- irvbfp_sho25[ShellHeight_mm >= 75, ]

saveRDS(irvbfp_sho75, paste0('data/GLMMs/AllDates/Data/irvbfp_sho75_',
                             Sys.Date(), '.rds'))

irvbfp_sho75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                                    RelYear+(1 | UniversalReefID),
                              data=subset(irvbfp_sho75,
                                          irvbfp_sho75$LiveDate_Qualifier=="Estimate" &
                                                !is.na(irvbfp_sho75$RelYear)),
                              family=gaussian, cores=4,
                              control= list(adapt_delta=0.999, max_treedepth=15),
                              iter=5000, warmup=1000, chains=4, inits=75, thin=3,
                              seed=5334, backend="cmdstanr", threads=threading(2),
                              file="data/GLMMs/AllDates/irvbfp_sho75_glmm_hist6.rds")

# Create model results tables and save diagnostic plots and marginal effects plots
data2 <- irvbfp_sho75
models2 <- list(irvbfp_sho75_glmm_hist)
#modresults(data, models, "Size class", meplotzoom=FALSE)
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('data/GLMMs/AllDates/Data/lb_sho25_',
                         Sys.Date(), '.rds'))
```

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

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

saveRDS(lb_sh25to75, paste0('data/GLMMs/AllDates/Data/lb_sh25to75_', Sys.Date(), '.rds'))


# Create model results tables and save diagnostic plots and marginal effects plots
data1 <- lb_sh25to75
models1 <- NULL
#modresults(data, models, "Size class", meplotzoom=FALSE)


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

lb_sho75 <- lb_sho25[ShellHeight_mm >= 75, ]

saveRDS(lb_sho75, paste0('data/GLMMs/AllDates/Data/lb_sho75_', Sys.Date(), '.rds'))

lb_sho75_glmm_hist <- brm(formula=ShellHeight_mm | trunc(lb=75, ub=250) ~
                                RelYear+(1 | UniversalReefID), data=lb_sho75,
                          family=gaussian, cores=4,
                          control= list(adapt_delta=0.999, max_treedepth=20),
                          iter=5000, warmup=1000, chains=4, inits=75, thin=3,
                          seed=7419, backend="cmdstanr", threads=threading(2),
                          file="data/GLMMs/AllDates/lb_sho75_glmm_hist14.rds")

#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
models2 <- list(lb_sho75_glmm_hist)
#modresults(data, models, "Size class", meplotzoom=FALSE)
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.

```
# #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.x",
                              "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", "Region.y",
                              "MA_plotlab", "Subtidal", "RelYear")]
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.x, SurveyMethod,
                      HabitatClassification, QuadSize_m2, MADup, UniversalReefID,
                      Region.y, 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)

#Calculate estimated Density_m2 values for ProgramID==5074. This line can be deleted after Claude recal
#in the combined table. I couldn't include it at the beginning of the script because I need to use the
#rather than the QuadSize_m2 column which is filled for the whole combined table.
oysterraw_den[ProgramID==5074, Density_m2 :=
                    (Number_of_Oysters_Counted_Total_Count/as.numeric(paste0(QuadSize_m2)))*
                    (Number_of_Oysters_Counted_Live_Count/
                           (Number_of_Oysters_Counted_Live_Count+
                                  Number_of_Oysters_Counted_Dead_Count))]
```

```
#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('data/GLMMs/AllDates/Data/ab_n_', Sys.Date(), '.rds'))

ab_den_glmm <- brm(formula=Density_m2 ~
                           RelYear+(0+RelYear | UniversalReefID), data=ab_n,
                   family=negbinomial, cores=4,
                   control= list(adapt_delta=0.99, max_treedepth=15),
                   iter=3000,
                   warmup=1000, chains=4, inits=0, thin=3, seed=5512,
                   backend="cmdstanr", threads=threading(2),
                   file="data/GLMMs/AllDates/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('data/GLMMs/AllDates/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=4,
                   control= list(adapt_delta=0.99, max_treedepth=15), iter=3000,
                   warmup=1000, chains=4, inits=0, thin=3, seed=4677,
                   backend="cmdstanr", threads=threading(2),
                   file="data/GLMMs/AllDates/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('data/GLMMs/AllDates/Data/eb_n_', Sys.Date(), '.rds'))

eb_den_glmm <- brm(formula=Density_m2 ~
                         RelYear+(1 | UniversalReefID), data=eb_n,
                   family=zero_inflated_negbinomial, cores=4,
                   control= list(adapt_delta=0.99, max_treedepth=15), iter=3000,
                   warmup=1000, chains=4, inits=0, thin=3, seed=1298,
                   backend="cmdstanr", threads=threading(2),
                   file="data/GLMMs/AllDates/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('data/GLMMs/AllDates/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=0.99, max_treedepth=15),
                    iter=3000, warmup=1000, chains=4, inits=0, thin=3,
                    seed=9875, backend="cmdstanr", threads=threading(2),
                    file="data/GLMMs/AllDates/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('data/GLMMs/AllDates/Data/gtmn_n_', Sys.Date(), '.rds'))
```

```
gtmn_den_glmm <- brm(formula=Density_m2 ~
                          RelYear+Region.y+RelYear:Region.y+(1 | UniversalReefID),
                    data=gtmn_n, family=zero_inflated_negbinomial, cores=4,
                    control= list(adapt_delta=0.99, max_treedepth=15),
                    iter=3000, warmup=1000, chains=4, inits=0, thin=3,
                    seed=3647, backend="cmdstanr", threads=threading(2),
                    file="data/GLMMs/AllDates/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)


### 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('data/GLMMs/AllDates/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=0.99, max_treedepth=15),
                   iter=3000, warmup=1000, chains=4, inits=0, thin=3,
                   seed=4612, backend="cmdstanr", threads=threading(2),
                   file="data/GLMMs/AllDates/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('data/GLMMs/AllDates/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=4,
                    control= list(adapt_delta=0.99, max_treedepth=15),
                    iter=3000, warmup=1000, chains=4, inits=0, thin=3,
                    seed=5243, backend="cmdstanr", threads=threading(2),
                    file="data/GLMMs/AllDates/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('data/GLMMs/AllDates/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=4, control= list(adapt_delta=0.99, max_treedepth=15),
                     iter=3000, warmup=1000, chains=4, inits=0, thin=3,
                     seed=8441, backend="cmdstanr", threads=threading(2),
                     file="data/GLMMs/AllDates/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 density
oysterraw_pct <- oysterraw[, c("ProgramID", "ProgramName", "ProgramLocationID",
                              "QuadIdentifier", "ReefIdentifier", "LiveDate",
                              "LiveDate_Qualifier", "SampleDate", "Year",
                              "Month", "ManagedAreaName", "Region.x",
                              "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", "Region.y",
                              "MA_plotlab", "Subtidal", "RelYear")]
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.x, SurveyMethod,
                      PercentLiveMethod, HabitatClassification, QuadSize_m2,
                      MADup, UniversalReefID, Region.y, 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'))


## 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('data/GLMMs/AllDates/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.y=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.y", "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('data/GLMMs/AllDates/Data/grm_p_binom_',
                            Sys.Date(), '.rds'))

grm_pct_glmm <- brm(formula=LiveObs ~ RelYear+(1 | UniversalReefID),
                    data=grm_p_binom, family=bernoulli, cores=4,
                    control= list(adapt_delta=0.99, max_treedepth=15),
                    iter=3000, warmup=1000, chains=4, inits=0, thin=3,
                    seed=4331, backend="cmdstanr", threads=threading(2),
                    file="data/GLMMs/AllDates/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)


## 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('data/GLMMs/AllDates/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),family=Beta,
                   cores=4, control= list(adapt_delta=0.99, max_treedepth=15),
                   iter=3000, warmup=1000, chains=4, inits=0, thin=3, seed=8465,
                   backend="cmdstanr", threads=threading(2),
                   file="data/GLMMs/AllDates/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])
```