SEACAR Oyster Analysis

Last compiled on 06 June, 2023

Table of Contents

# Important Notes

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/Oyster/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](jepanzik@usf.edu) 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)  
library(tidyverse)  
library(ggplot2)  
library(grid)  
library(gridExtra)  
library(gtable)  
library(here)  
library(sf)  
library(mapview)  
library(rcompanion)  
library(data.table)  
library(brms)  
library(modelr)  
library(tidybayes)  
library(doFuture)  
library(tictoc)  
library(doRNG)  
library(piecewiseSEM)  
library(ggpubr)  
options(scipen=999)  
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="")  
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\_Florida\_2021update\Oyster\_Beds\_in\_Florida.shp'   
## 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\_Aquatic\_Preserves.shp'   
## 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\_\_Boundaries.shp")

## 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\_National\_Estuarine\_Resarch\_Reserves\_\_NERR\_\_Boundaries.shp'   
## 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\_Merge.kml'   
## using driver `KML'  
## 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

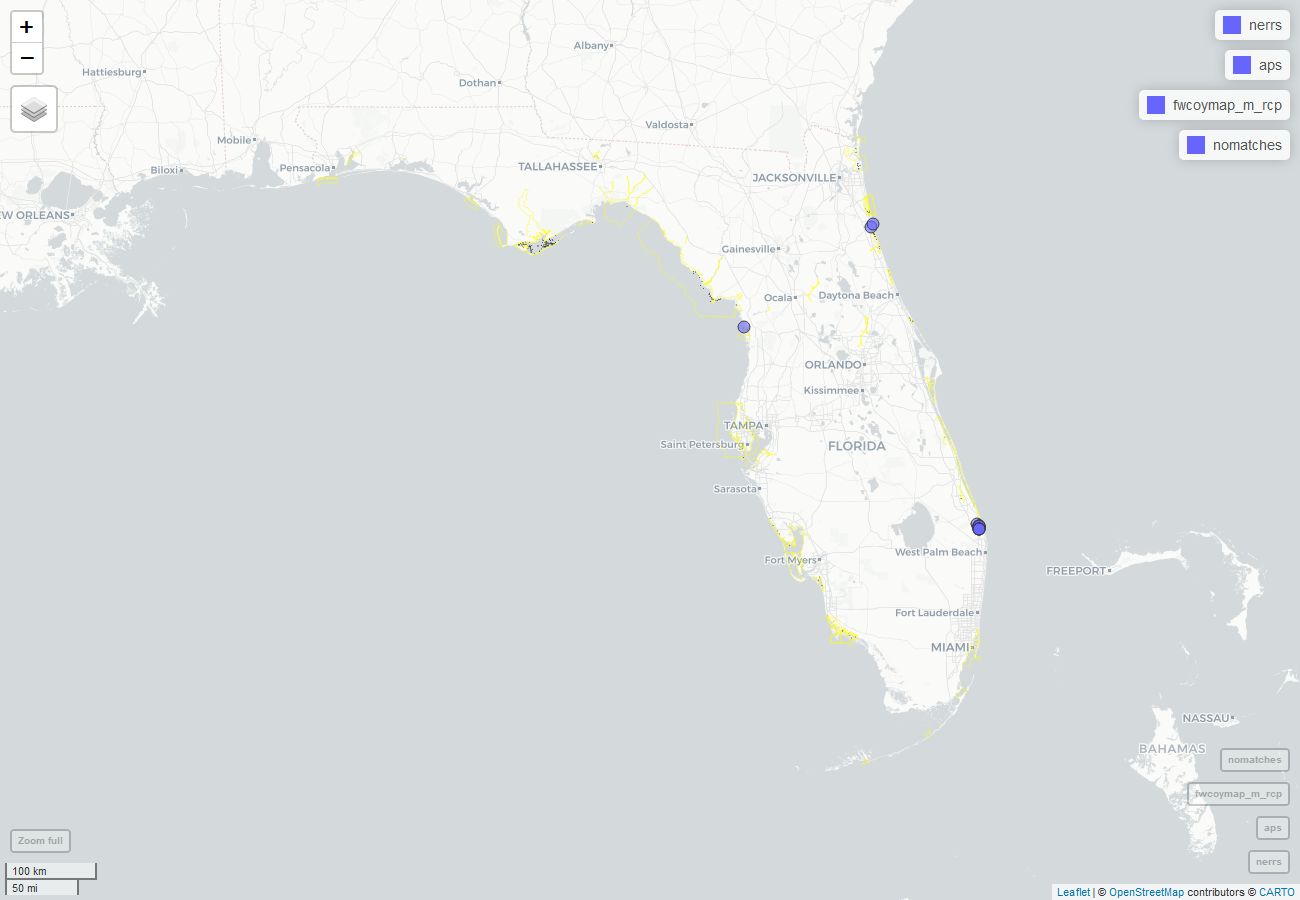
GTMnew2 <- st\_union(GTMnew)  
GTMnew <- subset(nerrs, nerrs$SITE\_NAME=="Guana Tolomato Matanzas National Estuarine Research Reserve")  
GTMnew$geometry <- GTMnew2  
othernerrs <- subset(nerrs, nerrs$SITE\_NAME != "Guana Tolomato Matanzas National Estuarine Research Reserve")  
oysamplelocs <- st\_read("data/OysterGIS\_files/SEACAR\_SampleLocationMatching/SampleLocations\_12jan21/seacar\_dbo\_SampleLocation\_Point.shp")

## Reading layer `seacar\_dbo\_SampleLocation\_Point' from data source   
## `C:\Users\jepanzik\Box\R Projects\SEACAR\_Trend\_Analyses\Oyster\data\OysterGIS\_files\SEACAR\_SampleLocationMatching\SampleLocations\_12jan21\seacar\_dbo\_SampleLocation\_Point.shp'   
## 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% reefcrosswalk\_rcp$ProgramLoc,   
 subset(reefcrosswalk\_rcp,  
 reefcrosswalk\_rcp$ProgramLoc ==  
 oysterraw$ProgramLocationID[i])$crosswalk, 'no match')  
 }  
toc()

## 262.58 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.kml'   
## using driver `KML'  
## 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"   
## [6] "Salt Run" "St. Augustine"

#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](https://github.com/FloridaSEACAR/SEACAR_Panzik/blob/main/SEACAR%20Documentation%20-%20Analysis%20Filters%20and%20Calculations.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, and Month.
   * Second summary statistics do not use the Month grouping and are only for ManagedAreaName and Year.
   * Third summary statistics do not use Year grouping and are only for ManagedAreaName and Month
   * Fourth summary statistics are only grouped based on ManagedAreaName
     + Determines the years that the minimum and maximum species richness occurred
2. For each group, provide the following information: Parameter Name (ParameterName), Number of Entries (N\_Data), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, and a list of all Programs included in these measurements.
3. Sort the data in ascending (A to Z and 0 to 9) order based on ManagedAreaName then Year then Month
4. Write summary stats to a pipe-delimited .txt file in the output directory
   * [Oyster Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Oyster/output) (<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)

# Data Summary

Creates and outputs a summary data file with basic statistics for each combination of managed area, parameter, shell type, shell size, and habitat classification. Column names and the shell type values are updated.

# Oyster Data Summary -----------------------------------------------------  
  
oyster\_sh\_summ <- oysterraw %>%  
 subset(nyrpar=="ShellHeight\_mm" & !is.na(SizeClass)) %>%  
 group\_by(ManagedAreaName, nyrpar, LiveDate\_Qualifier, SizeClass,  
 HabitatClassification) %>%  
 dplyr::summarize(N\_Data=length(ShellHeight\_mm[!is.na(ShellHeight\_mm)]),  
 N\_Years=length(unique(LiveDate[!is.na(LiveDate) &  
 !is.na(ShellHeight\_mm)])),  
 EarliestLiveDate=min(LiveDate[!is.na(ShellHeight\_mm)]),  
 LatestLiveDate=max(LiveDate[!is.na(ShellHeight\_mm)]),  
 LastSampleDate=max(SampleDate),  
 Mean=mean(ShellHeight\_mm[!is.na(ShellHeight\_mm)]),  
 Median=median(ShellHeight\_mm[!is.na(ShellHeight\_mm)]))  
  
oyster\_den\_summ <- oysterraw %>%  
 subset(nyrpar=="Density\_m2") %>%  
 group\_by(ManagedAreaName, nyrpar, LiveDate\_Qualifier, SizeClass,  
 HabitatClassification) %>%  
 dplyr::summarize(N\_Data=length(Density\_m2[!is.na(Density\_m2)]),  
 N\_Years=length(unique(LiveDate[!is.na(LiveDate) &  
 !is.na(Density\_m2)])),  
 EarliestLiveDate=min(LiveDate[!is.na(Density\_m2)]),  
 LatestLiveDate=max(LiveDate[!is.na(Density\_m2)]),  
 LastSampleDate=max(SampleDate),  
 Mean=mean(Density\_m2[!is.na(Density\_m2)]),  
 Median=median(Density\_m2[!is.na(Density\_m2)]))  
  
oyster\_pct\_summ <- oysterraw %>%  
 subset(nyrpar=="PercentLive\_pct") %>%  
 group\_by(ManagedAreaName, nyrpar, LiveDate\_Qualifier, SizeClass,  
 HabitatClassification) %>%  
 dplyr::summarize(N\_Data=length(PercentLive\_pct[!is.na(PercentLive\_pct)]),  
 N\_Years=length(unique(LiveDate[!is.na(LiveDate) &  
 !is.na(PercentLive\_pct)])),  
 EarliestLiveDate=min(LiveDate[!is.na(PercentLive\_pct)]),  
 LatestLiveDate=max(LiveDate[!is.na(PercentLive\_pct)]),  
 LastSampleDate=max(SampleDate),  
 Mean=mean(PercentLive\_pct[!is.na(PercentLive\_pct)]),  
 Median=median(PercentLive\_pct[!is.na(PercentLive\_pct)]))  
  
oyster\_summ <- bind\_rows(oyster\_sh\_summ, oyster\_den\_summ, oyster\_pct\_summ)  
  
#rm(oyster\_sh\_summ, oyster\_den\_summ, oyster\_pct\_summ)  
  
setnames(oyster\_summ, c("nyrpar", "LiveDate\_Qualifier",  
 "HabitatClassification"),  
 c("ParameterName", "ShellType", "HabitatType"))  
  
oyster\_summ$ShellType[oyster\_summ$ShellType=="Exact"] <- "Live Oyster Shells"  
oyster\_summ$ShellType[oyster\_summ$ShellType=="Estimate"] <- "Dead Oyster Shells"  
  
  
fwrite(oyster\_summ, file="output/OysterSummary.txt", sep="|")  
  
# 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

1. meplots

* Creates and saves plots for density and live percent

1. 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

1. meplotssh

* Creates and saves plots for shell height

1. 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 recalculates   
#in the combined table. I couldn't include it at the beginning of the script because I need to use the counts columns  
#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])