SEACAR Discrete Water Quality Analysis: Lab Bottom Colored dissolved organic matter, CDOM

Last compiled on 09 June, 2023

Table of Contents

# Important Notes

The purpose of this script is to create managed area statistics, perform seasonal Kendall Tau analysis, generate summary plots, and create reports in pdf and Word document form for each parameter in Wc Discrete.

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

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

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

This markdown file is designed to be compiled by [SEACAR\_WC\_Discrete\_ReportRender.R](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/WC_Discrete/SEACAR_WC_Discrete_ReportRender.R) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/WC_Discrete/SEACAR_WC_Discrete_ReportRender.R>).

Note: The top 2% of data is excluded when computing mean and standard deviations in plotting sections solely for the purpose of getting y-axis scales. The exclusion of the top 2% is not used in any statistics that are exported.

# Libraries and Settings

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

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

# File Import

Imports file that is determined in the SEACAR\_WC\_Discrete\_ReportRender.R script.

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

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

The latest version of WC Discrete data is available at: <https://usf.box.com/s/fbimxw4hrmazfn5b1d4jbn0addmcsld8>

The file being used for the analysis is: **Combined\_WQ\_WC\_NUT\_Colored\_dissolved\_organic\_matter\_CDOM-2023-Jun-01.txt**

data <- fread(file\_in, sep="|", header=TRUE, stringsAsFactors=FALSE,  
 select=c("ManagedAreaName", "ProgramID", "ProgramName",  
 "ProgramLocationID", "SampleDate", "Year", "Month",  
 "RelativeDepth", "ActivityType", "ParameterName",  
 "ResultValue", "ParameterUnits", "ValueQualifier",  
 "SEACAR\_QAQCFlagCode", "Include"),  
 na.strings=c("NULL","","NA"))  
  
parameter <- unique(data$ParameterName)  
unit <- unique(data$ParameterUnits)  
cat(paste("The data file(s) used:", file\_short, sep="\n"))

## The data file(s) used:  
## Combined\_WQ\_WC\_NUT\_Colored\_dissolved\_organic\_matter\_CDOM-2023-Jun-01.txt

# Data Filtering

Most data filtering is performed on export from the database, and is indicated by the Include variable. Include values of 1 indicate the data should be used for analysis, values of 0 indicate the data should not be used for analysis. Documentation on the database filtering is provided here: [SEACAR Documentation- Analysis Filters and Calculations.pdf](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/SEACAR%20Documentation%20-%20Analysis%20Filters%20and%20Calculations.pdf)

The filtering that is performed by the script at this point removes rows that are missing values for ResultValue, and only keeps data that is measured at the relative depth (surface, bottom, etc.) and activity type (field or sample) of interest. This is partly handled on export with the RelativeDepth variable, but there are some measurements that are considered both surface and bottom based on measurement depth and total depth. By default, these are marked as Surface for RelativeDepth and receive a SEACAR\_QAQCFlag indicator of 12Q. Data passes the filtering the process if it is from the correct depth and has an Include value of 1. The script also only looks at data of the desired ActivityType which indicates whether it was measured in the field (Field) or in the lab (Sample).

After the initial filtering, a second filter variable is created to determine whether enough time is represented in the monitoring location, which is that each monitoring location has 10 year or more of unique year entries and have at least 2 consecutive years of observations with at least 2 repeating months for observations that pass the initial filter. If data passes the first set of filtering criteria and the time criteria, they are used in the analysis.

The function that determines whether a managed area has at least 2 consecutive years of observations with at least 2 repeating months takes the data, creates a list of the managed areas and cycles through each managed area. For each managed area cycle:

1. List the unique years and put them in ascending order
2. If there are fewer than 2 unique years, skip to the next area
3. If there are 2 or more unique years, start a loop that compares adjacent year entries for the area
   * Start with the first two year entries
4. See if the year entries are subsequent years (1 year apart)
   * If not, skip to next pair of years
5. For the two years being compared, get the list of months for each
6. Compare the two lists of months to see what months are the same
   * If there are two or more months that are the same, the managed area passes the criteria and is stored in a variable
7. The list of managed areas that pass the 2 consecutive years with at least 2 repeating months is returned and used to determine if there is sufficient data for analysis.

A data frame is created that stores summary information for each managed area. This information is stored and combined with the results of the Seasonal Kendall Tau analysis and export to a data file once combined.

The sufficient data qualifier is merged with the original data, and a variable Use\_In\_Analysis is created to indicate what data should be used.

# Removes data rows with missing ResultValue  
data <- data[!is.na(data$ResultValue),]  
# Changes "Sample" to "Lab" for ActivityType  
data$ActivityType <- gsub("Sample", "Lab", data$ActivityType)  
  
# Gets data for the specific activity type for Chlorophyll, salinity, TSS,  
# and Turbidity  
if((param\_name=="Chlorophyll\_a\_uncorrected\_for\_pheophytin" |  
 param\_name=="Salinity" | param\_name=="Total\_Suspended\_Solids\_TSS" |  
 param\_name=="Turbidity") & activity!="All"){  
 data <- data[grep(activity, data$ActivityType[!is.na(data$ActivityType)]),]  
}  
  
# Changes RelativeDepth to Bottom for the QAQC flag 12Q that indicates  
# measurements are both surface and bottom if the relative depth is bottom  
if(depth=="Bottom"){  
 data$RelativeDepth[grep("12Q", data$SEACAR\_QAQCFlagCode[  
 data$RelativeDepth=="Surface"])] <- "Bottom"  
}  
# Removes missing RelativeDepth data and data for RelativeDepth not of interest  
# from all parameters except Secchi\_Depth  
if(param\_name!="Secchi\_Depth" & depth!="All"){  
 data <- data[!is.na(data$RelativeDepth),]  
 data <- data[data$RelativeDepth==depth,]  
}  
  
# Removes data rows that have "Blank" as an ActivityType  
if(length(grep("Blank", data$ActivityType))>0){  
 data <- data[-grep("Blank", data$ActivityType),]  
}  
  
# Removes data rows with ResultValue below 0, or -2 for Water\_Temperature  
if(param\_name=="Water\_Temperature"){  
 data <- data[data$ResultValue>=-2,]  
} else{  
 data <- data[data$ResultValue>=0,]  
}  
# Changes Include to be either TRUE or FALSE  
data$Include <- as.logical(data$Include)  
# Changes Include to be TRUE for ProgramID 476 if it had the H value qualifier  
data$Include[grep("H", data$ValueQualifier[data$ProgramID==476])] <- TRUE  
# Change Include to be FALSE for Secchi\_Depth with U value qualifier  
if(param\_name=="Secchi\_Depth"){  
 data$Include[grep("U", data$ValueQualifier)] <- FALSE  
}  
# Gets AreaID for data by merging data with the managed area list  
data <- merge.data.frame(MA\_All[,c("AreaID", "ManagedAreaName")],  
 data, by="ManagedAreaName", all=TRUE)  
# Creates function to checks managed area for at least 2 years of  
# continuous consecutive data  
DiscreteConsecutiveCheck <- function(con\_data){  
 # Gets AreaIDs  
 IDs <- unique(con\_data$AreaID[con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # Loops through each AreaID  
 for(i in 1:length(IDs)) {  
 # Gets list of Years for AreaID  
 Years <- unique(con\_data$Year[con\_data$AreaID==IDs[i] &  
 con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # Puts Years in order  
 Years <- Years[order(Years)]  
 # If there are fewer than 2 years, skip to next AreaID  
 if(length(Years)<2) {  
 next  
 }  
 # Starts loop to make sure there are at least 2 consecutive years  
 # with consecutive months of data  
 for(j in 2:length(Years)) {  
 # If adjacent year entries are not 1 year apart, skip to the  
 # next set of year entries  
 if(Years[j]-Years[j-1]!=1) {  
 next  
 }  
 # Gets the list of months from the first year  
 Months1 <- unique(con\_data$Month[  
 con\_data$AreaID==IDs[i] &  
 con\_data$Year==Years[j-1] &  
 con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # Gets list of months for the second year  
 Months2 <- unique(con\_data$Month[  
 con\_data$AreaID==IDs[i] &  
 con\_data$Year==Years[j] &  
 con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # If there are more than 2 months shared between the two  
 # years, the AreaID passes the check and is stored  
 if(length(intersect(Months1, Months2))>=2) {  
 # Creates variable for stored AreaID if it  
 # doesn't exist  
 if(exists("consecutive")==FALSE){  
 consecutive <- IDs[i]  
 break  
 # Adds to variable for storing AreaID if does exist  
 } else{  
 consecutive <- append(consecutive, IDs[i])  
 break  
 }  
 }  
 }  
 }  
 # After going through all AreaID, return variable with list of all  
 # that pass  
 return(consecutive)  
}  
# Stores the AreaID that pass the consecutive year check  
consMonthIDs <- DiscreteConsecutiveCheck(data)  
  
# Creates data frame with summary for each managed area  
MA\_Summ <- data %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N\_Data=length(ResultValue[Include==TRUE & !is.na(ResultValue)]),  
 N\_Years=length(unique(Year[Include==TRUE & !is.na(Year)])),  
 EarliestYear=min(Year[Include==TRUE & N\_Data!=0]),  
 LatestYear=max(Year[Include==TRUE & N\_Data!=0]),  
 LastSampleDate=max(SampleDate[Include==TRUE]),  
 ConsecutiveMonths=ifelse(unique(AreaID) %in%  
 consMonthIDs==TRUE, TRUE, FALSE),  
 # Determines if monitoring location is sufficient for analysis  
 # based on having more than 0 data entries, more than the  
 # sufficient number of year, and the consecutive month criteria  
 SufficientData=ifelse(N\_Data>0 & N\_Years>=suff\_years &  
 ConsecutiveMonths==TRUE, TRUE, FALSE),  
 Median=median(ResultValue[Include==TRUE & N\_Data!=0], na.rm=TRUE))  
  
MA\_Summ$ConsecutiveMonths <- NULL  
# Creates column in data that determines how many years from the start for each  
# managed area  
data <- data %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 mutate(YearFromStart=Year-min(Year))  
# Adds SufficientData column to data table based on managed area  
data <- merge.data.frame(data, MA\_Summ[,c("ManagedAreaName", "SufficientData")],  
 by="ManagedAreaName")  
# Creates Use\_In\_Analysis column for data that is determined if the row has  
# Include value of TRUE and SufficientData value of TRUE  
data$Use\_In\_Analysis <- ifelse(data$Include==TRUE & data$SufficientData==TRUE,  
 TRUE, FALSE)  
# Rearranges the summary data frame columns to be AreaID, ManagedAreaName,  
# ParameterName RelativeDepth, ActivityType, SufficientData, everything else  
MA\_Summ <- MA\_Summ %>%  
 select(AreaID, ManagedAreaName, ParameterName, RelativeDepth, ActivityType,  
 SufficientData, everything())  
# Puts summary data in order based on managed area  
MA\_Summ <- as.data.frame(MA\_Summ[order(MA\_Summ$ManagedAreaName), ])  
# Put SampleDate as date object  
data$SampleDate <- as.Date(data$SampleDate)  
# Creates character object for Month and Year  
data$YearMonth <- paste0(data$Month, "-", data$Year)  
# Creates variable that puts year and month into a decimal year format  
data$YearMonthDec <- data$Year + ((data$Month-0.5) / 12)  
# Converts ampleDate to a decimal date  
data$DecDate <- decimal\_date(data$SampleDate)  
  
# Get list of and number of managed areas that are to be used in analysis  
MA\_Include <- MA\_Summ$ManagedAreaName[MA\_Summ$SufficientData==TRUE]  
n <- length(MA\_Include)  
# Get list of and number of managed areas that are excluded from analysis  
MA\_Exclude <- MA\_Summ[MA\_Summ$N\_Years<10 & MA\_Summ$N\_Years>0,]  
MA\_Exclude <- MA\_Exclude[,c("ManagedAreaName", "N\_Years")]  
z <- nrow(MA\_Exclude)

# Data Impacted by Specific Value Qualifiers

Reports the amount of data impacted by the H (for dissolved oxygen & pH in program 476), I, Q, S (for Secchi depth), and U value qualifiers. It determines how much of the data for the given ParametetrName, RelativeDepth, and ActivityType is impacted by each value qualifier. Percentages are determined using 100\*(# of value qualifier)/(# of total data)

A variable is also created that determines if scatter plot points should be a different color based on value qualifiers of interest.

A summary data frame is created that determines the amount of data and percentage of data impacted by the value qualifiers for each managed area by year and is written to a csv file in the output directory. Columns with N are the number impacted by the value qualifier, and those with perc are the percent of the data for that managed area and year impacted by the value qualifier. + [WC Discrete Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output>)

# Find out how much total data exists and how much passed the initial filters  
total <- length(data$Include)  
pass\_filter <- length(data$Include[data$Include==TRUE])  
# Get the number and percentage of data entries impacted by value qualifier H  
count\_H <- length(grep("H", data$ValueQualifier[data$ProgramID==476]))  
perc\_H <- 100\*count\_H/length(data$ValueQualifier)  
# Get the number and percentage of data entries impacted by value qualifier I  
count\_I <- length(grep("I", data$ValueQualifier))  
perc\_I <- 100\*count\_I/length(data$ValueQualifier)  
# Get the number and percentage of data entries impacted by value qualifier Q  
count\_Q <- length(grep("Q", data$ValueQualifier))  
perc\_Q <- 100\*count\_Q/length(data$ValueQualifier)  
# Get the number and percentage of data entries impacted by value qualifier S  
count\_S <- length(grep("S", data$ValueQualifier))  
perc\_S <- 100\*count\_S/length(data$ValueQualifier)  
# Get the number and percentage of data entries impacted by value qualifier U  
count\_U <- length(grep("U", data$ValueQualifier))  
perc\_U <- 100\*count\_U/length(data$ValueQualifier)  
# Copy ValueQualifier to a new VQ\_Plot to create codes for plots  
data$VQ\_Plot <- data$ValueQualifier  
# Determine if data with value qualifier H should be included for plots based  
# on the parameter being observed  
inc\_H <- ifelse(param\_name=="pH" | param\_name=="Dissolved\_Oxygen" |  
 param\_name=="Dissolved\_Oxygen\_Saturation", TRUE, FALSE)  
# Loops through conditions to determine what indicators to include in plots.  
# If H should be included  
if (inc\_H==TRUE){  
 # Remove any Value qualifiers that aren't H or U  
 data$VQ\_Plot <- gsub("[^HU]+", "", data$VQ\_Plot)  
 # Standardize order of qualifiers. Puts UH as HU  
 data$VQ\_Plot <- gsub("UH", "HU", data$VQ\_Plot)  
 # Remove anything from ValueQualifier that isn't U from programs and that  
 # aren't ProgramID 476  
 data$VQ\_Plot[na.omit(data$ProgramID!=476)] <-  
 gsub("[^U]+", "", data$VQ\_Plot[na.omit(data$ProgramID!=476)])  
 # Changes blank character strings to NA  
 data$VQ\_Plot[data$VQ\_Plot==""] <- NA  
 # Prints the number and percentage of H, I, Q, U value qualifiers  
 cat(paste0("Number of Measurements: ", total,  
 ", Number Passed Filter: ", pass\_filter, "\n",  
 "Program 476 H Codes: ", count\_H, " (", round(perc\_H, 6), "%)\n",  
 "I Codes: ", count\_I, " (", round(perc\_I, 6), "%)\n",  
 "Q Codes: ", count\_Q, " (", round(perc\_Q, 6), "%)\n",  
 "U Codes: ", count\_U, " (", round(perc\_U, 6), "%)"))  
# If Parameter is Secchi\_Depth  
} else if (param\_name=="Secchi\_Depth") {  
 # Count the number of S ValueQualifier  
 count\_S <- length(grep("S", data$ValueQualifier))  
 # Get percentage of S ValueQualifier  
 perc\_S <- 100\*count\_S/length(data$ValueQualifier)  
 # Remove anything from ValueQualifier that isn't S or U  
 data$VQ\_Plot <- gsub("[^SU]+", "", data$VQ\_Plot)  
 # Change all ValueQualifier that are US to be US, standardizes codes  
 data$VQ\_Plot <- gsub("US", "SU", data$VQ\_Plot)  
 # Sets any blank character ValueQualifier to be NA  
 data$VQ\_Plot[data$VQ\_Plot==""] <- NA  
 # Prints the number and percentage of I, Q, S, U  
 cat(paste0("Number of Measurements: ", total,  
 ", Number Passed Filter: ", pass\_filter, "\n",  
 "I Codes: ", count\_I, " (", round(perc\_I, 6), "%)\n",  
 "Q Codes: ", count\_Q, " (", round(perc\_Q, 6), "%)\n",  
 "S Codes: ", count\_S, " (", round(perc\_S, 6), "%)\n",  
 "U Codes: ", count\_U, " (", round(perc\_U, 6), "%)"))  
# For all other scenarios  
} else{  
 # Remove all ValueQualifier except U  
 data$VQ\_Plot <- gsub("[^U]+", "", data$VQ\_Plot)  
 # Sets any blank character ValueQualifier to be NA  
 data$VQ\_Plot[data$VQ\_Plot==""] <- NA  
 # Prints the number and percentage of I, Q, U  
 cat(paste0("Number of Measurements: ", total,  
 ", Number Passed Filter: ", pass\_filter, "\n",  
 "I Codes: ", count\_I, " (", round(perc\_I, 6), "%)\n",  
 "Q Codes: ", count\_Q, " (", round(perc\_Q, 6), "%)\n",  
 "U Codes: ", count\_U, " (", round(perc\_U, 6), "%)"))  
}

## Number of Measurements: 1677, Number Passed Filter: 1673  
## I Codes: 129 (7.692308%)  
## Q Codes: 19 (1.132976%)  
## U Codes: 53 (3.160405%)

# Creates a data table that summarizes the number and percentage of  
# ValueQualifier H, I, Q, S, and U for each managed area each year  
data\_summ <- data %>%  
 group\_by(AreaID, ManagedAreaName, Year) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N\_Total=length(ResultValue),  
 N\_AnalysisUse=length(ResultValue[Use\_In\_Analysis==TRUE]),  
 N\_H=length(grep("H", ValueQualifier[ProgramID==476])),  
 perc\_H=100\*N\_H/length(ValueQualifier),  
 N\_I=length(grep("I", ValueQualifier)),  
 perc\_I=100\*N\_I/length(ValueQualifier),  
 N\_Q=length(grep("Q", ValueQualifier)),  
 perc\_Q=100\*N\_Q/length(ValueQualifier),  
 N\_S=length(grep("S", ValueQualifier)),  
 perc\_S=100\*N\_S/length(ValueQualifier),  
 N\_U=length(grep("U", ValueQualifier)),  
 perc\_U=100\*N\_U/length(ValueQualifier))  
# Orders the data table rows based on managed area name  
data\_summ <- as.data.table(data\_summ[order(data\_summ$ManagedAreaName,  
 data\_summ$Year), ])  
# Writes the ValueQualifier summary to a csv file  
fwrite(data\_summ, paste0(out\_dir\_param,"/WC\_Discrete\_", param\_abrev, "\_",  
 activity, "\_", depth, "\_VQSummary.csv"), sep=",")  
rm(data\_summ)

# Managed Area Statistics

Gets summary statistics for each managed area. Excluded managed areas are not included into whether the data should be used or not. Uses piping from dplyr package to feed into subsequent steps. The following steps are performed:

1. Take the data variable and only include rows that have a SufficientData value of TRUE
2. 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
3. For each group, provide the following information: Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, and a list of all Program IDs included in these measurements.
4. Sort the data in ascending (A to Z and 0 to 9) order based on ManagedAreaName then Year then Month
5. Write summary stats to a pipe-delimited .txt file in the output directory
   * [WC Discrete Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output>)

# Create summary statistics for each managed area based on Year and Month  
# intervals.  
MA\_YM\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, Year, Month) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N\_Data=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 Programs=paste(sort(unique(ProgramName), decreasing=FALSE),  
 collapse=', '),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
# Puts the data in order based on ManagedAreaName, Year, then Month  
MA\_YM\_Stats <- as.data.table(MA\_YM\_Stats[order(MA\_YM\_Stats$ManagedAreaName,  
 MA\_YM\_Stats$Year,  
 MA\_YM\_Stats$Month), ])  
# Writes summary statistics to file  
fwrite(MA\_YM\_Stats, paste0(out\_dir\_param,"/WC\_Discrete\_", param\_abrev, "\_",  
 activity, "\_", depth, "\_MA\_MMYY\_Stats.txt"), sep="|")  
# Get year from start for each managed area  
MA\_YM\_Stats <- MA\_YM\_Stats %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 mutate(YearFromStart=Year-min(Year))  
# Create decimal value of year and month values  
MA\_YM\_Stats$YearMonthDec <- MA\_YM\_Stats$Year + ((MA\_YM\_Stats$Month-0.5) / 12)  
# Create summary statistics for each managed area based on Year intervals.  
MA\_Y\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, Year) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 Programs=paste(sort(unique(ProgramName), decreasing=FALSE),  
 collapse=', '),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
# Puts the data in order based on ManagedAreaName then Year  
MA\_Y\_Stats <- as.data.table(MA\_Y\_Stats[order(MA\_Y\_Stats$ManagedAreaName,  
 MA\_Y\_Stats$Year), ])  
# Writes summary statistics to file  
fwrite(MA\_Y\_Stats, paste0(out\_dir\_param,"/WC\_Discrete\_", param\_abrev, "\_",  
 activity, "\_", depth, "\_MA\_Yr\_Stats.txt"), sep="|")  
rm(MA\_Y\_Stats)  
# Create summary statistics for each managed area based on Month intervals.  
MA\_M\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, Month) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 Programs=paste(sort(unique(ProgramName), decreasing=FALSE),  
 collapse=', '),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
# Puts the data in order based on ManagedAreaName then Month  
MA\_M\_Stats <- as.data.table(MA\_M\_Stats[order(MA\_M\_Stats$ManagedAreaName,  
 MA\_M\_Stats$Month), ])  
# Writes summary statistics to file  
fwrite(MA\_M\_Stats, paste0(out\_dir\_param,"/WC\_Discrete\_", param\_abrev, "\_",  
 activity, "\_", depth, "\_MA\_Mo\_Stats.txt"), sep="|")  
rm(MA\_M\_Stats)

# Monitoring Location Statistics

Gets monitoring location statistics, which is defined as a unique combination of ManagedAreaName, ProgramID, ProgramAreaName, and ProgramLocationID, using piping from dplyr package. The following steps are performed:

1. Take the data variable and only include rows that have a SufficientData value of TRUE
2. Group data that have the same ManagedAreaName, ProgramID, ProgramName, and ProgramLocationID.
3. For each group, provide the following information: Earliest Sample Date (EarliestSampleDate), Latest Sample Date (LastSampleDate), Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, and Standard Deviation.
4. Sort the data in ascending (A to Z and 0 to 9) order based on ManagedAreaName then ProgramName then ProgramID then ProgramLocationID
5. Write summary stats to a pipe-delimited .txt file in the output directory
   * [WC Discrete Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output>)

# Gets summary statistics for monitoring locations, which are defined as unique  
# combinations of ManagedAreaName, ProgramID, And ProgramLocationID  
Mon\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName,  
 ProgramLocationID) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate),  
 N\_Data=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
# Order data rows by ManagedAreaName, ProgramName, ProgramID, then  
# ProgramLocationID  
Mon\_Stats <- as.data.table(Mon\_Stats[order(Mon\_Stats$ManagedAreaName,  
 Mon\_Stats$ProgramName,  
 Mon\_Stats$ProgramID,   
 Mon\_Stats$ProgramLocationID), ])  
# Write summary statistics to file  
fwrite(Mon\_Stats, paste0(out\_dir\_param,"/WC\_Discrete\_", param\_abrev, "\_",  
 activity, "\_", depth, "\_MonLoc\_Stats.txt"), sep="|")  
rm(Mon\_Stats)

# Seasonal Kendall Tau Analysis

Gets seasonal Kendall Tau statistics using the kendallSeasonalTrendTest from the EnvStats package. The Trend parameter is determined from a user-defined function based on the median, Senn slope, and p values from the data. Analysis modified from code created by Jason Scolaro that performed at The Water Atlas: <https://sarasota.wateratlas.usf.edu/water-quality-trends/#analysis-overview>

The following steps are performed:

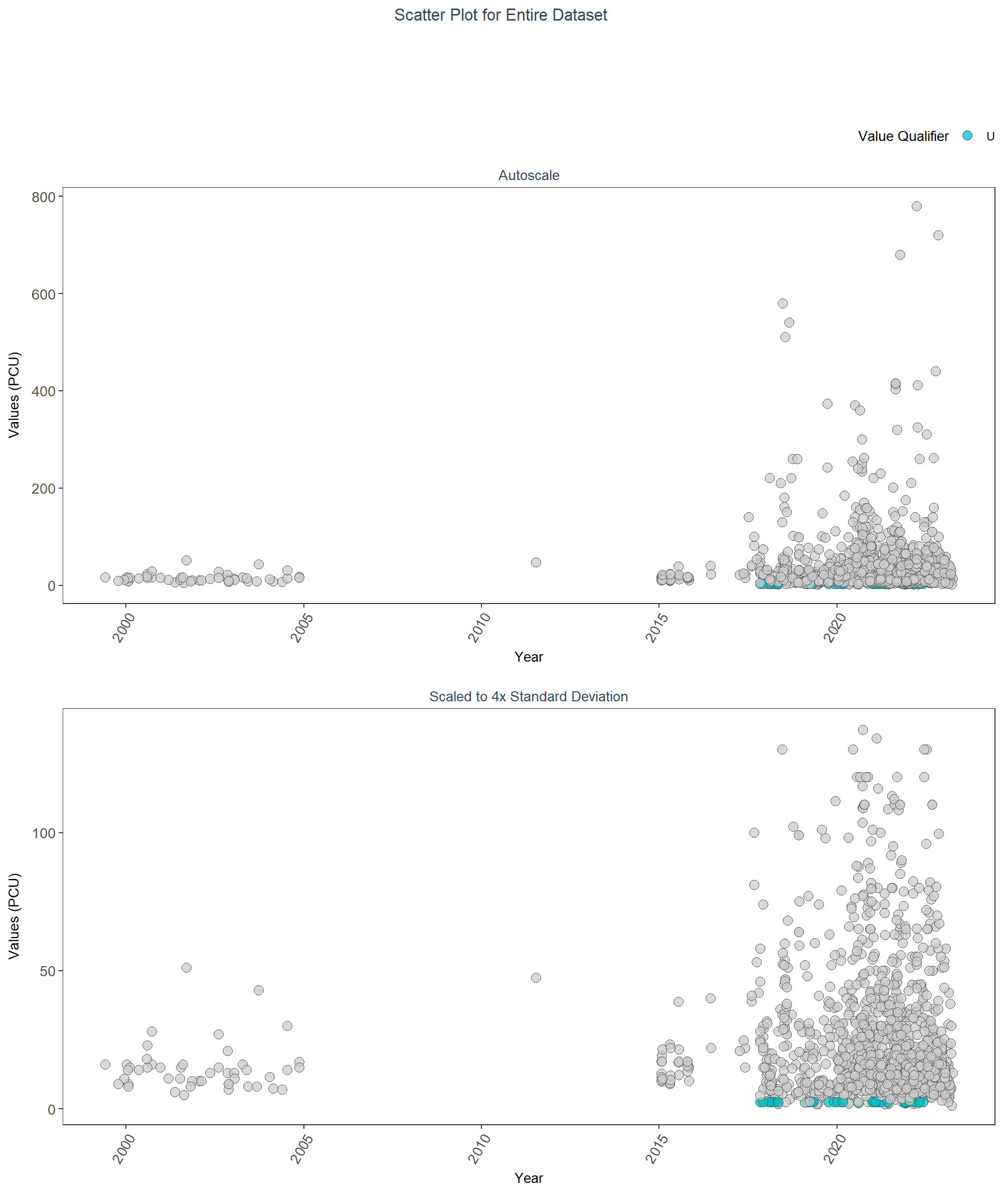
1. Define the functions used in the analysis
2. Check to see if there are any groups to run analysis on.
3. Take the data variable and only include rows that have a SufficientData value of TRUE
4. Group data that have the same ManagedAreaName.
5. For each group, provides the following information: Earliest Sample Date (EarliestSampleDate), Latest Sample Date (LastSampleDate), Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, tau, Senn Slope (SennSlope), Senn Intercept (SennIntercept), and p.
   * The analysis is run with the kendallSeasonalTrendTest function using the Year values for year, and Month as the seasonal qualifier, and Trend.
   * An independent.obs value of TRUE indicates that the data should be treated as not being serially auto-correlated. An independent.obs value of FALSE indicates that it is treated as being serially auto-correlated, but also requires one observation per season per year for the full time of observation.
6. Reformat columns in the data frame from export.
7. Write summary stats to a pipe-delimited .txt file in the output directory
   * [WC Discrete Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Discrete/output>)

# Creates function to get the Kendall Tau statistics  
tauSeasonal <- function(dat, independent, stats.median, stats.minYear,  
 stats.maxYear) {  
 tau <- NULL  
 # Stores results from seasonal Kendall Tau  
 tryCatch({ken <- kendallSeasonalTrendTest(  
 y=dat$Mean,  
 season=dat$Month,  
 year=dat$YearFromStart,  
 independent.obs=independent)  
 # Gets the values of interest from the trend fit  
 tau <- ken$estimate[1]  
 p <- ken$p.value[2]  
 slope <- ken$estimate[2]  
 intercept <- ken$estimate[3]  
 chi\_sq <- ken$statistic[1]  
 p\_chi\_sq <- ken$p.value[1]  
 trend <- trend\_calculator(slope, stats.median, p)  
 rm(ken)  
 # Prints warnings if a fit does not exist and stores values as NA  
 }, warning=function(w) {  
 print(w)  
 }, error=function(e) {  
 print(e)  
 }, finally={  
 if (!exists("tau")) {  
 tau <- NA  
 }  
 if (!exists("p")) {  
 p <- NA  
 }  
 if (!exists("slope")) {  
 slope <- NA  
 }  
 if (!exists("intercept")) {  
 intercept <- NA  
 }  
 if (!exists("trend")) {  
 trend <- NA  
 }  
 })  
 # Puts variables in a vector for the managed area currently being analyzed  
 KT <-c(unique(dat$AreaID),  
 unique(dat$ManagedAreaName),  
 independent,  
 tau,  
 p,  
 slope,  
 intercept,  
 chi\_sq,  
 p\_chi\_sq,  
 trend)  
 # Returns the fit parameters  
 return(KT)  
}  
# Function that determines statistics from data  
runStats <- function(dat, med, minYr, maxYr) {  
 # Get basic stats  
 dat$Mean <- as.numeric(dat$Mean)  
 stats.median <- med  
 stats.minYear <- minYr  
 stats.maxYear <- maxYr  
 # Calculate Kendall Tau and Slope stats assuming they are serially  
 # independent, then store in variable  
 KT <- tauSeasonal(dat, TRUE, stats.median,  
 stats.minYear, stats.maxYear)  
 # If variable returned is empty, run again assuming they are NOT serially  
 # independent  
 if (is.null(KT[9])) {  
 KT <- tauSeasonal(dat, FALSE, stats.median,  
 stats.minYear, stats.maxYear)  
 }  
 # If KT.Stats does not exist, create it and store values  
 if (is.null(KT.Stats)==TRUE) {  
 KT.Stats <- KT  
 # If KT.Stats does exist, add values to it  
 } else{  
 KT.Stats <- rbind(KT.Stats, KT)  
 }  
 return(KT.Stats)  
}  
# Function to determine trend of Kendal Tau  
trend\_calculator <- function(slope, median\_value, p) {  
 # Trend depends on series of conditions  
 trend <-  
 # If the p value is less than 5% and the slope is greater than 10% of the  
 # median value, the trend is large (2).  
 if (p < .05 & abs(slope) > abs(median\_value) / 10.) {  
 if (slope > 0) {  
 2  
 }  
 else {  
 -2  
 }  
 }  
 # If the p value is less than 5% and the slope is less than 10% of the  
 # median value, there is a trend (1).  
 else if (p < .05 & abs(slope) < abs(median\_value) / 10.) {  
 if (slope > 0) {  
 1  
 }  
 else {  
 -1  
 }  
 # Otherwise, there is no trend (0)  
 } else  
 0  
 return(trend)  
}  
# Creates a null data frame for storing kendall tau results  
KT.Stats <- NULL  
# List for column names  
c\_names <- c("AreaID", "ManagedAreaName", "Independent", "tau", "p",  
 "SennSlope", "SennIntercept", "ChiSquared", "pChiSquared", "Trend")  
# Determines if there are any monitoring locations to analyze  
if(n==0){  
 # Creates data frame to store analysis values in  
 KT.Stats <- data.frame(matrix(ncol=length(c\_names),  
 nrow=length(MA\_Summ$ManagedAreaName)))  
 colnames(KT.Stats) <- c\_names  
   
 KT.Stats[, c("AreaID", "ManagedAreaName")] <-  
 MA\_Summ[, c("AreaID", "ManagedAreaName")]  
} else{  
 # Starts cycling through managed areas to determine seasonal Kendall Tau  
 for (i in 1:n) {  
 # Gets the number of rows of data for the managed area  
 x <- nrow(MA\_YM\_Stats[MA\_YM\_Stats$ManagedAreaName==MA\_Include[i], ])  
 # Perform analysis if there is more than 1 row  
 if (x>0) {  
 # Store the managed area summary statistics to be used in trend analysis  
 SKT.med <- MA\_Summ$Median[MA\_Summ$ManagedAreaName==MA\_Include[i]]  
 SKT.minYr <- MA\_Summ$EarliestYear[MA\_Summ$ManagedAreaName==  
 MA\_Include[i]]  
 SKT.maxYr <- MA\_Summ$LatestYear[MA\_Summ$ManagedAreaName==MA\_Include[i]]  
 # Get seasonal Kendall Tau statistics by running data for managed areas  
 # through the functions  
 KT.Stats <- runStats(MA\_YM\_Stats[MA\_YM\_Stats$ManagedAreaName==  
 MA\_Include[i], ],  
 SKT.med, SKT.minYr, SKT.maxYr)  
 }  
 }  
 # Stores as data frame  
 KT.Stats <- as.data.frame(KT.Stats)  
 # If there was only one location, it is stored as a column vector. Change to  
 # row vector  
 if(dim(KT.Stats)[2]==1){  
 KT.Stats <- as.data.frame(t(KT.Stats))  
 }  
 # Sets column and row names for KT.Stats  
 colnames(KT.Stats) <- c\_names  
 rownames(KT.Stats) <- seq(1:nrow(KT.Stats))  
 # Sets variables to proper format and rounds values if necessary  
 KT.Stats$tau <- round(as.numeric(KT.Stats$tau), digits=4)  
 KT.Stats$p <- round(as.numeric(KT.Stats$p), digits=4)  
 KT.Stats$SennSlope <- as.numeric(KT.Stats$SennSlope)  
 KT.Stats$SennIntercept <- as.numeric(KT.Stats$SennIntercept)  
 KT.Stats$ChiSquared <- round(as.numeric(KT.Stats$ChiSquared), digits=4)  
 KT.Stats$pChiSquared <- round(as.numeric(KT.Stats$pChiSquared), digits=4)  
 KT.Stats$Trend <- as.integer(KT.Stats$Trend)  
}  
# Combines the KT.Stats with MA\_Summ  
KT.Stats <- merge.data.frame(MA\_Summ, KT.Stats,  
 by=c("AreaID", "ManagedAreaName"), all=TRUE)  
  
KT.Stats <- as.data.table(KT.Stats[order(KT.Stats$ManagedAreaName), ])  
# Writes combined statistics to file  
fwrite(KT.Stats, paste0(out\_dir\_param,"/WC\_Discrete\_", param\_abrev, "\_",  
 activity, "\_", depth, "\_KendallTau\_Stats.txt"),  
 sep="|")  
# Removes data rows for managed areas with no ResultValue  
data <- data[!is.na(data$ResultValue),]  
# Gets x and y values for starting point for trendline  
KT.Plot <- KT.Stats %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 summarize(x=EarliestYear,  
 y=SennIntercept)  
# Gets x and y values for ending point for trendline  
KT.Plot2 <- KT.Stats %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 summarize(x=decimal\_date(LastSampleDate),  
 y=(x-EarliestYear)\*SennSlope+SennIntercept)  
# Combines the starting and endpoints for plotting the trendline  
KT.Plot <- bind\_rows(KT.Plot, KT.Plot2)  
rm(KT.Plot2)  
KT.Plot <- as.data.table(KT.Plot[order(KT.Plot$ManagedAreaName), ])  
KT.Plot <- KT.Plot[!is.na(KT.Plot$y),]

# Appendix I: Scatter Plot of Entire Dataset

This part will create a scatter plot of the all data that passed initial filtering criteria with points colored based on specific value qualifiers. The values determined at the beginning (year\_lower, year\_upper, min\_RV, mn\_RV, x\_scale, and y\_scale) are solely for use by the plotting functions and are not output as part of the computed statistics.

# Defines standard plot theme: black and white, no major or minor grid lines,  
# Arial font. Title is centered, size 12, and blue (hex coded). Subtitle is  
# centered, size 10, and blue (hex coded). Legend title is size 10 and the  
# legend is left-justified. X-axis title is size 10 and the margins are padded  
# at the top and bottom to give more space for angled axis labels. Y-axis title  
# is size 10 and margins are padded on the right side to give more space for  
# axis labels. Axis labels are size 10 and the x-axis labels are rotated -45  
# degrees with a horizontal justification that aligns them with the tick mark  
plot\_theme <- theme\_bw() +  
 theme(panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 text=element\_text(family="Arial"),  
 plot.title=element\_text(hjust=0.5, size=12, color="#314963"),  
 plot.subtitle=element\_text(hjust=0.5, size=10, color="#314963"),  
 legend.title=element\_text(size=10),  
 legend.text.align = 0,  
 axis.title.x = element\_text(size=10, margin = margin(t = 5, r = 0,  
 b = 10, l = 0)),  
 axis.title.y = element\_text(size=10, margin = margin(t = 0, r = 10,  
 b = 0, l = 0)),  
 axis.text=element\_text(size=10),  
 axis.text.x=element\_text(angle = 60, hjust = 1))  
# Gets first and most recent years from data set  
year\_lower <- min(data$Year)  
year\_upper <- max(data$Year)  
# Gets minimum, mean, and standard deviation of ResultValue for setting y-axis  
# scale  
min\_RV <- min(data$ResultValue)  
mn\_RV <- mean(data$ResultValue[data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
sd\_RV <- sd(data$ResultValue[data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
x\_scale <- ifelse(year\_upper - year\_lower > 30, 10, 5)  
y\_scale <- mn\_RV + 4 \* sd\_RV  
  
# Create plot object for auto-scaled y-axis plot  
p1 <- ggplot(data=data[data$Include==TRUE,],  
 aes(x=SampleDate, y=ResultValue, fill=VQ\_Plot)) +  
 geom\_point(shape=21, size=3, color="#333333", alpha=0.75) +  
 labs(subtitle="Autoscale",  
 x="Year", y=paste0("Values (", unit, ")"),  
 fill="Value Qualifier") +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal",  
 legend.justification="right") +  
 scale\_x\_date(labels=date\_format("%Y")) +  
 {if(inc\_H==TRUE){  
 scale\_fill\_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",  
 "HU"="#7CAE00"), na.value="#cccccc")  
 } else if(param\_name=="Secchi\_Depth"){  
 scale\_fill\_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",  
 "SU"="#7CAE00"), na.value="#cccccc")  
 } else {  
 scale\_fill\_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")  
 }}  
# Create plot object for y-axis scaled plot  
p2 <- ggplot(data=data[data$Include==TRUE,],  
 aes(x=SampleDate, y=ResultValue, fill=VQ\_Plot)) +  
 geom\_point(shape=21, size=3, color="#333333", alpha=0.75) +  
 ylim(min\_RV, y\_scale) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 plot\_theme +  
 theme(legend.position="none") +  
 scale\_x\_date(labels=date\_format("%Y")) +  
 {if(inc\_H==TRUE){  
 scale\_fill\_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",  
 "HU"="#7CAE00"), na.value="#cccccc")  
 } else if(param\_name=="Secchi\_Depth"){  
 scale\_fill\_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",  
 "SU"="#7CAE00"), na.value="#cccccc")  
 } else {  
 scale\_fill\_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")  
 }}  
# Create legend object  
leg <- get\_legend(p1)  
# Arrange plots and legend  
pset <- ggarrange(leg, p1 + theme(legend.position="none"), p2,  
 ncol=1, heights=c(0.1, 1, 1))  
# Create title object  
p0 <- ggplot() + labs(title="Scatter Plot for Entire Dataset") +  
 plot\_theme + theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(),  
 axis.line=element\_blank())  
# Arrange and print title with plots  
ggarrange(p0, pset, ncol=1, heights=c(0.1, 1))



# Appendix II: Dataset Summary Box Plots

Box plots are created by using the entire data set and excludes any data that has been previously filtered out. The scripts that create plots follow this format

1. Use the data set that only has SufficientData of TRUE
2. Set what values are to be used for the x-axis, y-axis, and the variable that should determine groups for the box plots
3. Set the plot type as a box plot with the size of the outlier points
4. Create the title, x-axis, y-axis, and color fill labels
5. Set the y and x limits
6. Make the axis labels bold
7. Plot the arrangement as a set of panels

This set of box plots are grouped by year.

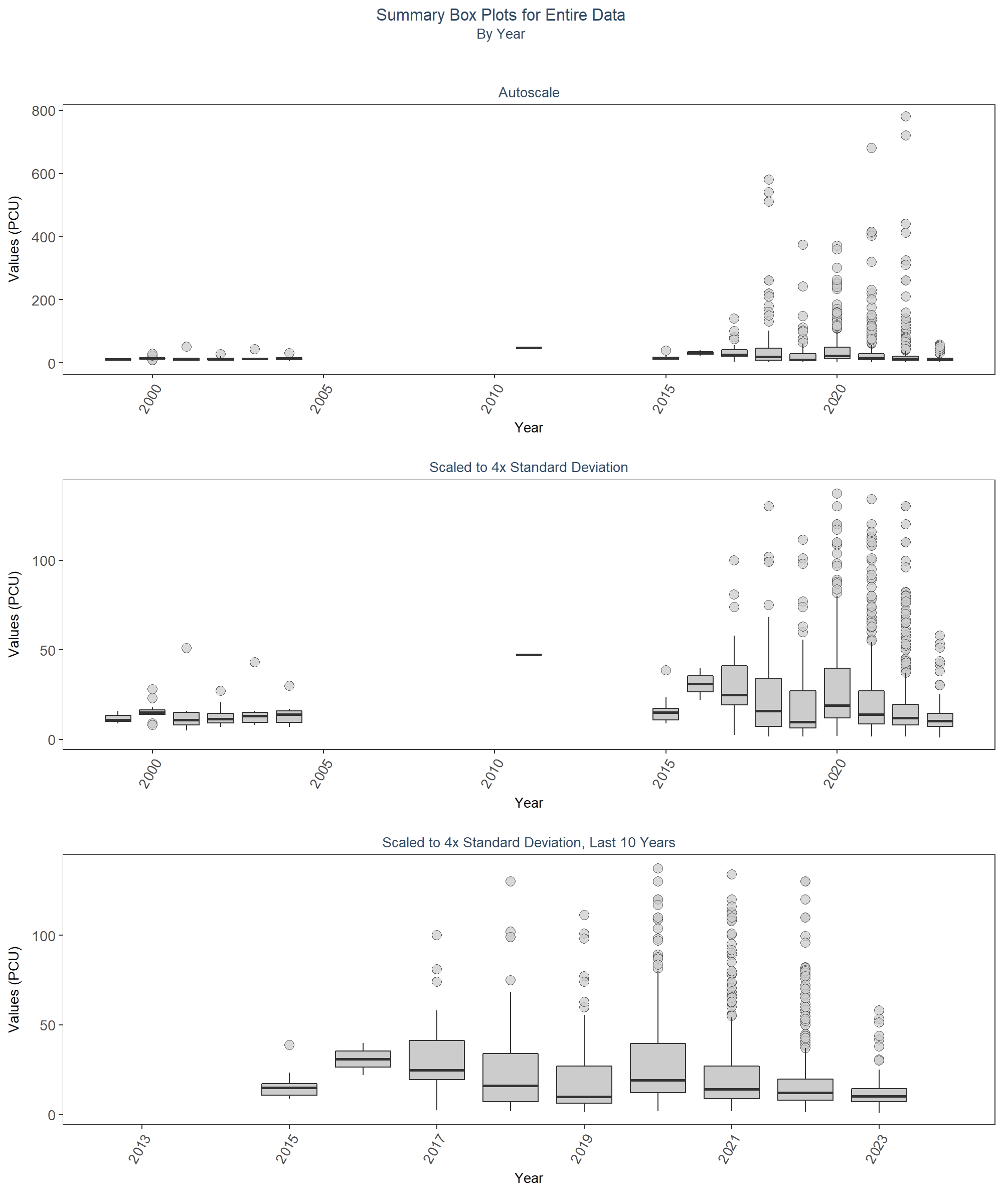
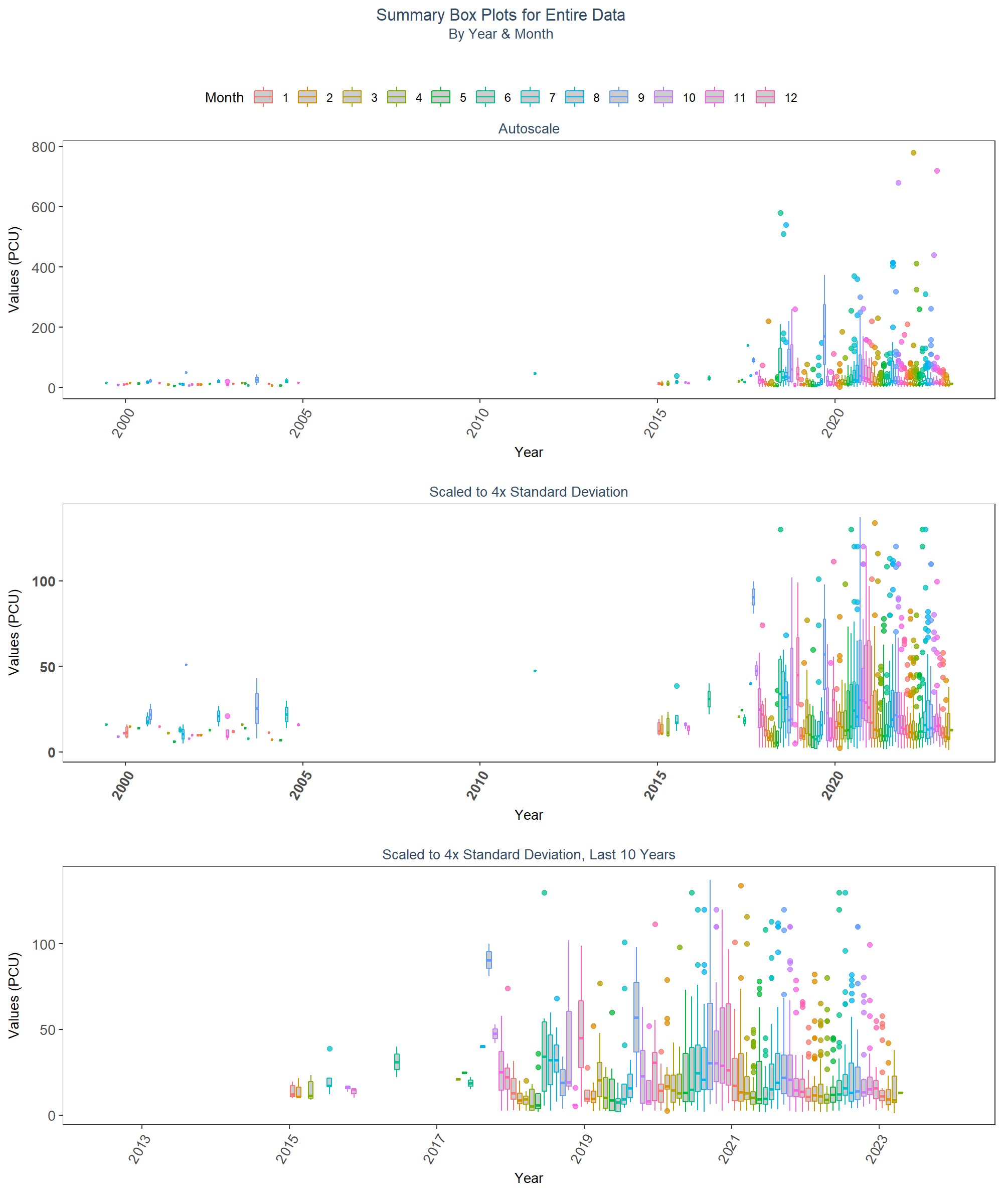
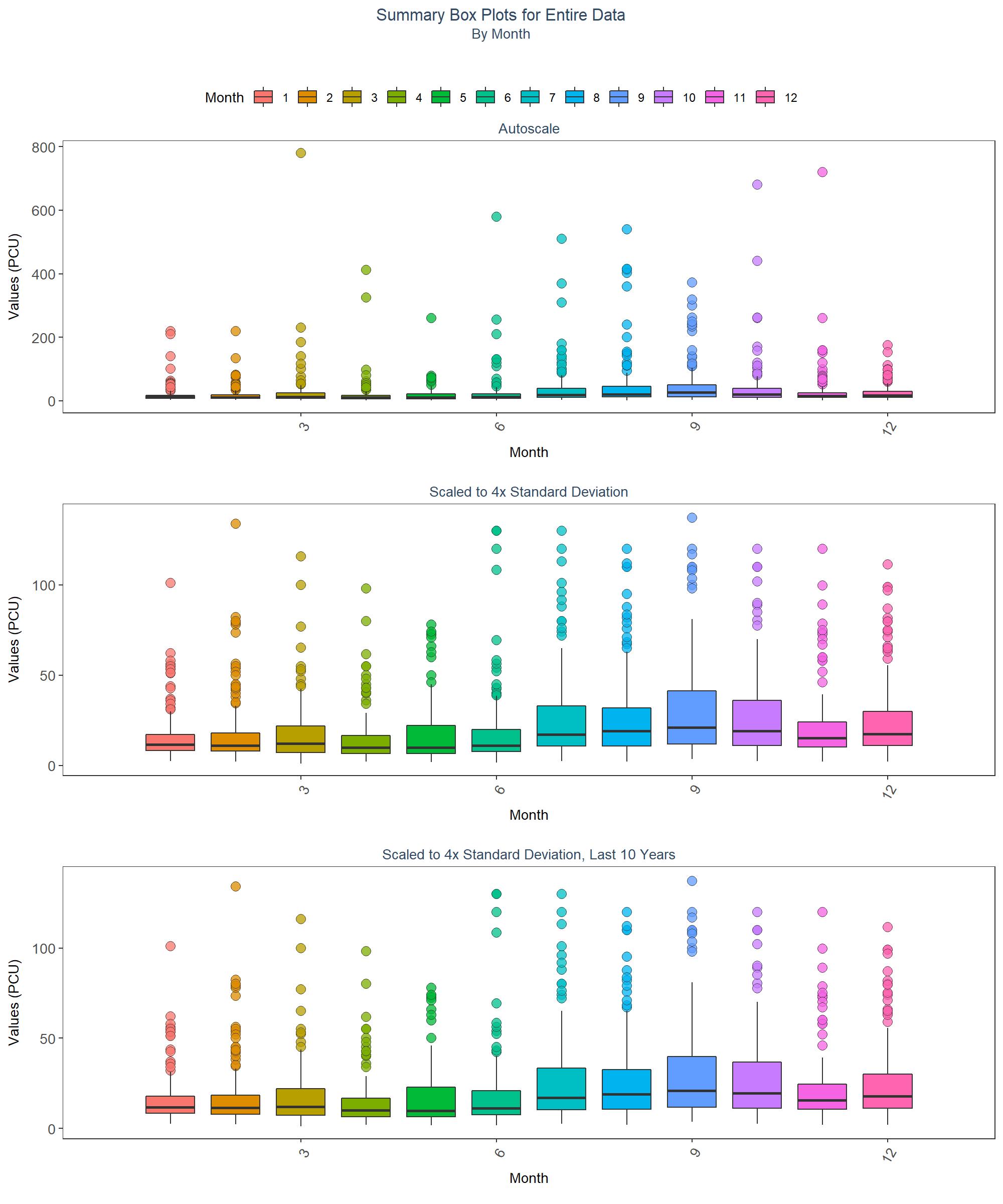
# Get minimum, mean, and standard deviation of the data  
min\_RV <- min(data$ResultValue[data$Include==TRUE])  
mn\_RV <- mean(data$ResultValue[data$Include==TRUE &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
sd\_RV <- sd(data$ResultValue[data$Include==TRUE &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
# Sets y scale based on data  
y\_scale <- mn\_RV + 4 \* sd\_RV  
# Create plot object for auto-scaled y-axis plot  
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")")) +  
 plot\_theme  
# Create plot object for y-axis scaled plot  
p2 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation", x="Year",  
 y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 plot\_theme  
# Create plot object for y-axis scaled plot for past 10 years  
p3 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=as.integer(Year), y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+1),  
 breaks=seq(max(data$Year) - 10, max(data$Year), 2)) +  
 plot\_theme  
# Arrange plot objects  
set <- ggarrange(p1, p2, p3, ncol=1)  
# Create title object for plots  
p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",  
 subtitle="By Year") + plot\_theme +  
 theme(panel.border=element\_blank(), panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
# Arrange title on plots  
Yset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))

This set of box plots are grouped by year and month with the color being related to the month.

# Create plot object for auto-scaled y-axis plot  
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")"), color="Month") +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(color=guide\_legend(nrow=1))  
# Create plot object for y-axis scaled plot  
p2 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 plot\_theme +  
 theme(legend.position="none", axis.text.x=element\_text(face="bold"),  
 axis.text.y=element\_text(face="bold"))  
# Create plot object for y-axis scaled plot for past 10 years  
p3 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+1),  
 breaks=seq(max(data$Year) - 10, max(data$Year), 2)) +  
 plot\_theme +  
 theme(legend.position="none")  
# Create legend item  
leg <- get\_legend(p1)  
# Arrange plots and legend  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
# Create plot title object  
p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",  
 subtitle="By Year & Month") + plot\_theme +  
 theme(panel.border=element\_blank(), panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
# Arrange plots and title  
YMset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))

The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

# Create plot object for auto-scaled y-axis plot  
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Month",  
 y=paste0("Values (", unit, ")"), fill="Month") +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(fill=guide\_legend(nrow=1))  
# Create plot object for y-axis scaled plot  
p2 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Month", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
# Create plot object for y-axis scaled plot for past 10 years  
p3 <- ggplot(data=data[data$Include==TRUE &  
 data$Year >= max(data$Year) - 10, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Month", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
# Create legend object  
leg <- get\_legend(p1)  
# Arrange plots and legend  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
# Create title object for plots  
p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",  
 subtitle="By Month") + plot\_theme +  
 theme(panel.border=element\_blank(), panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
# Arrange plots and title  
Mset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))

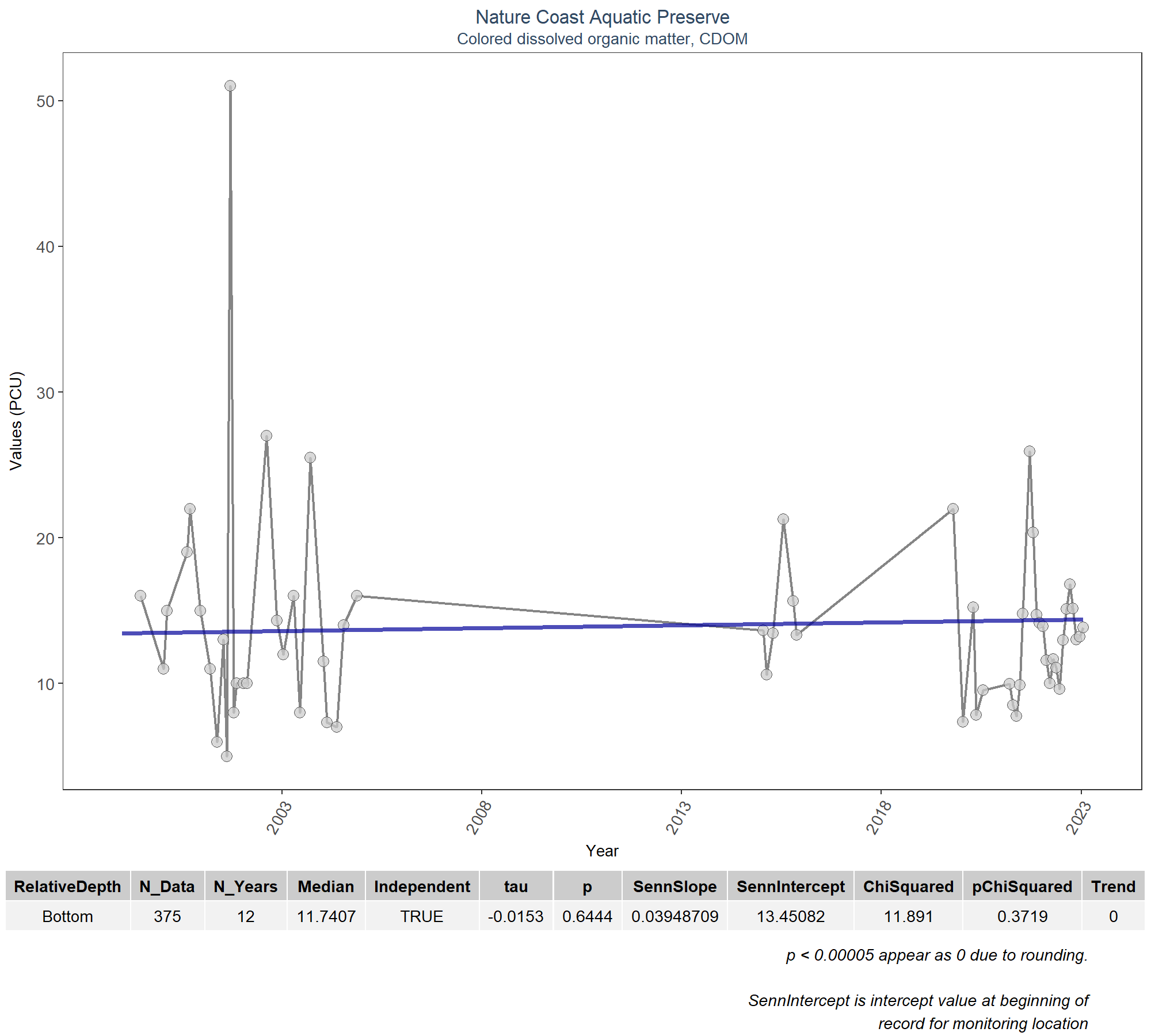
  

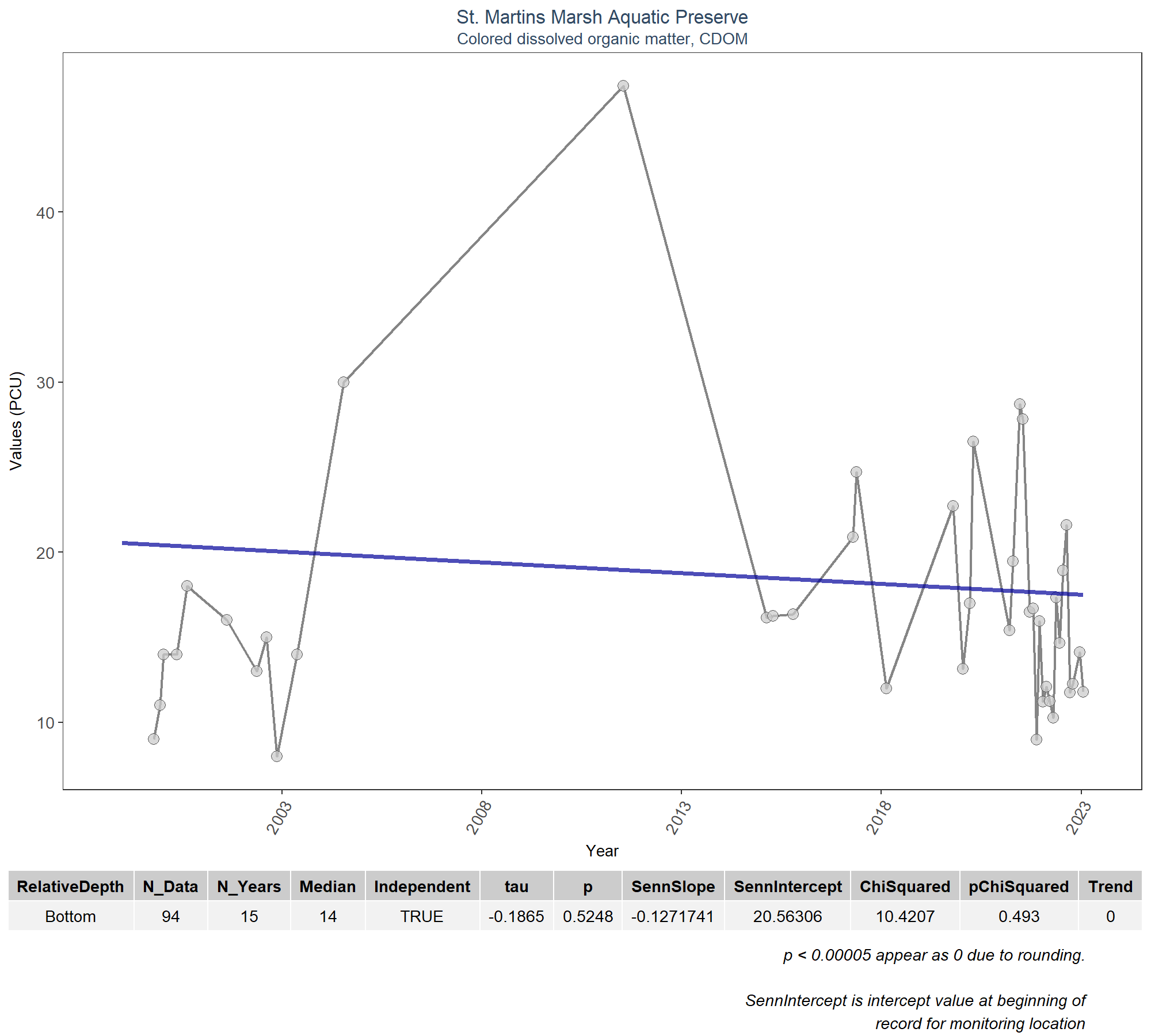
# Appendix III: Managed Area Trendlines

The plots created in this section are designed to show the general trend of the data. Data is taken and grouped by ManagedAreaName. The trendlines on the plots are created using the Senn slope and intercept from the seasonal Kendall Tau analysis. The scripts that create plots follow this format

1. Use the data set that only has SufficientData of TRUE for the desired managed area
2. Determine the earliest and latest year of the data to create x-axis scale and intervals
3. Determine the minimum, mean, and standard deviation for the data to be used for y-axis scales
   * Excludes the top 2% of values to reduce the impact of extreme outliers on the y-axis scale
4. Set what values are to be used for the x-axis, y-axis, and the variable that should determine groups for the plots
5. Set the plot type as a point plot with the size of the points
6. Add the linear trend
7. Create the title, x-axis, y-axis, and color fill labels
8. Set the y and x limits
9. Make the axis labels bold
10. Plot the arrangement as a set of panels

# Determines whether analyzed managed areas exist. If they do, begins  
# looping through them  
if(n==0){  
 print("There are no managed areas that qualify.")  
} else {  
 # Begins looping through each managed area  
 for (i in 1:n) {  
 # Gets data to be used in plot for managed area  
 plot\_data <- MA\_YM\_Stats[MA\_YM\_Stats$ManagedAreaName==MA\_Include[i],]  
 # Gets trendline data for managed area  
 KT.plot\_data <- KT.Plot[KT.Plot$ManagedAreaName==MA\_Include[i],]  
 #Determine max and min time (Year) for plot x-axis  
 t\_min <- min(plot\_data$Year)  
 t\_max <- max(plot\_data$YearMonthDec)  
 t\_max\_brk <- as.integer(round(t\_max, 0))  
 t <- t\_max-t\_min  
 min\_RV <- min(plot\_data$Mean)  
   
 # Sets break intervals based on the number of years spanned by data  
 if(t>=30){  
 brk <- -10  
 }else if(t<30 & t>=10){  
 brk <- -5  
 }else if(t<10 & t>=4){  
 brk <- -2  
 }else if(t<4){  
 brk <- -1  
 }  
   
 # Create plot object with data and trendline  
 p1 <- ggplot(data=plot\_data,  
 aes(x=YearMonthDec, y=Mean)) +  
 geom\_line(size=0.75, color="#333333", alpha=0.6) +  
 geom\_point(shape=21, size=3, color="#333333", fill="#cccccc",  
 alpha=0.75) +  
 geom\_line(data=KT.plot\_data, aes(x=x, y=y),  
 color="#000099", size=1.2, alpha=0.7) +  
 labs(title=paste0(MA\_Include[i]),  
 subtitle=parameter,  
 x="Year", y=paste0("Values (", unit, ")")) +  
 scale\_x\_continuous(limits=c(t\_min-0.25, t\_max+0.25),  
 breaks=seq(t\_max\_brk, t\_min, brk)) +  
 plot\_theme   
 # Creates ResultTable to display statistics below plot  
 ResultTable <- KT.Stats[KT.Stats$ManagedAreaName==MA\_Include[i], ] %>%  
 select(RelativeDepth, N\_Data, N\_Years, Median, Independent, tau, p,  
 SennSlope, SennIntercept, ChiSquared, pChiSquared, Trend)  
 # Create table object  
 t1 <- ggtexttable(ResultTable, rows=NULL,  
 theme=ttheme(base\_size=10)) %>%  
 tab\_add\_footnote(text="p < 0.00005 appear as 0 due to rounding.\n  
 SennIntercept is intercept value at beginning of  
 record for monitoring location",  
 size=10, face="italic")  
 # Arrange and display plot and statistic table  
 print(ggarrange(p1, t1, ncol=1, heights=c(0.85, 0.15)))  
 # Add extra space at the end to prevent the next figure from being too  
 # close.  
 cat("\n \n \n")   
 rm(plot\_data)  
 rm(KTset, leg)  
 rm(plot\_data)  
 rm(KTset, leg)  
 }  
}





# Appendix IV: Managed Area Summary Box Plots

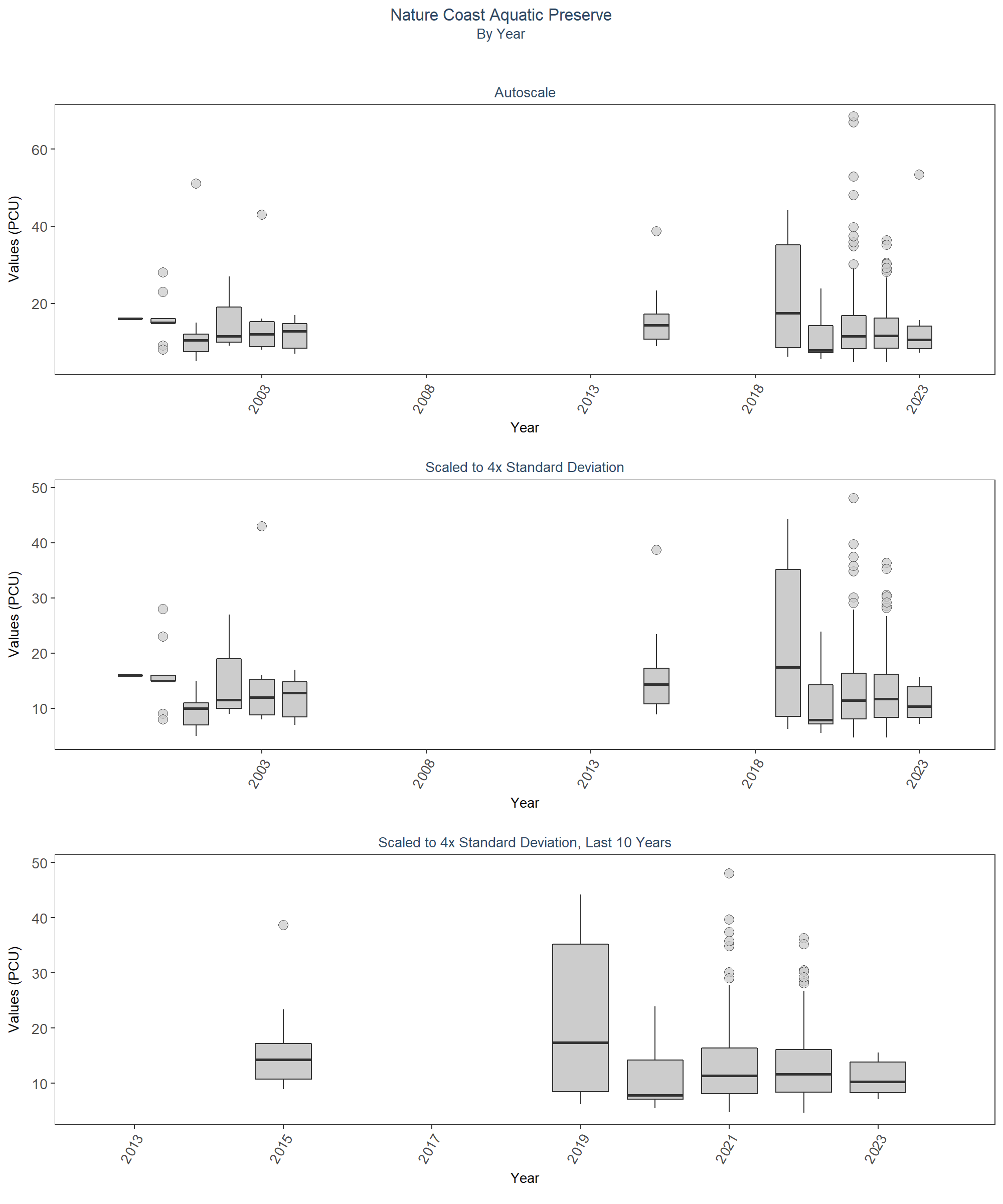
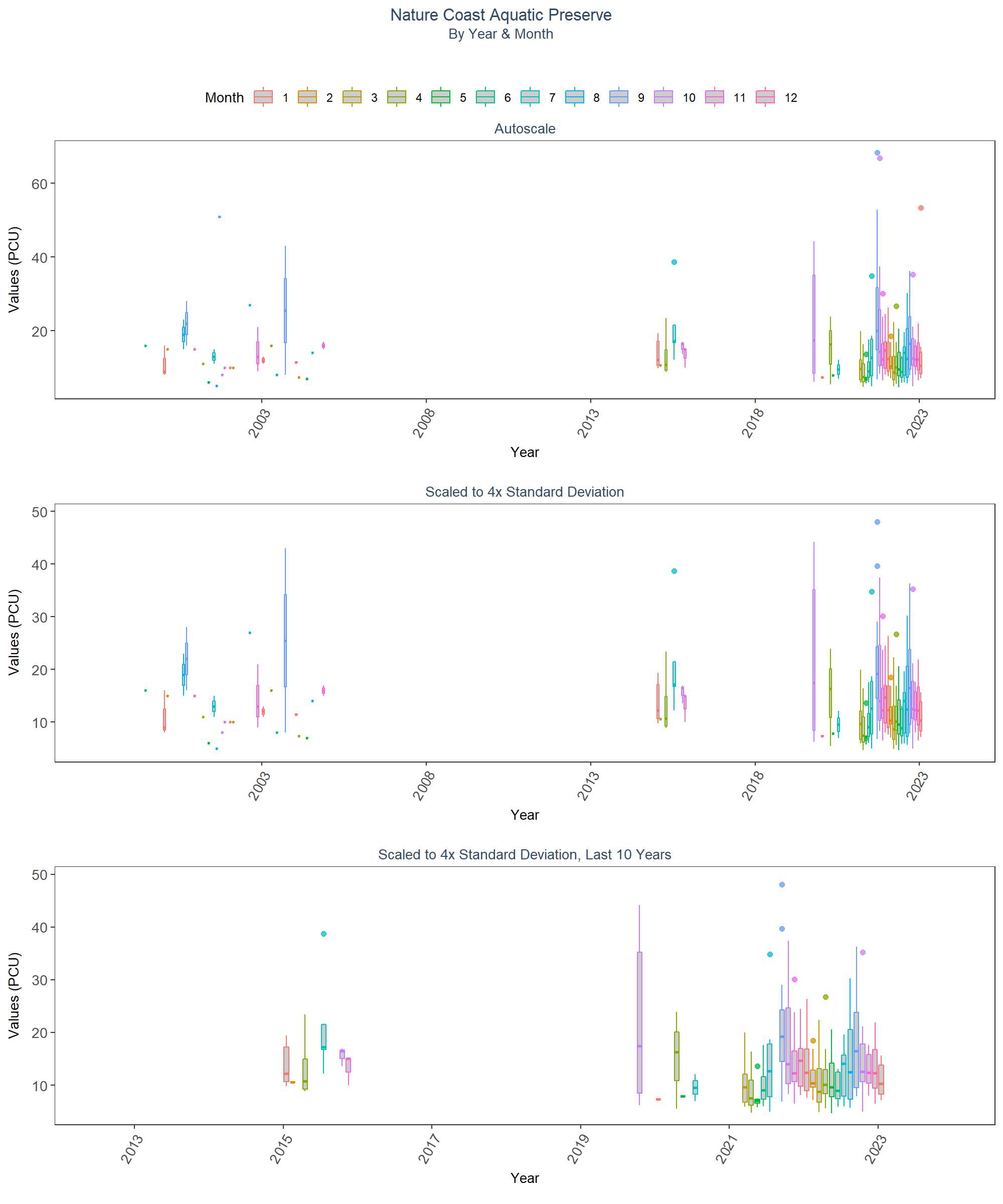
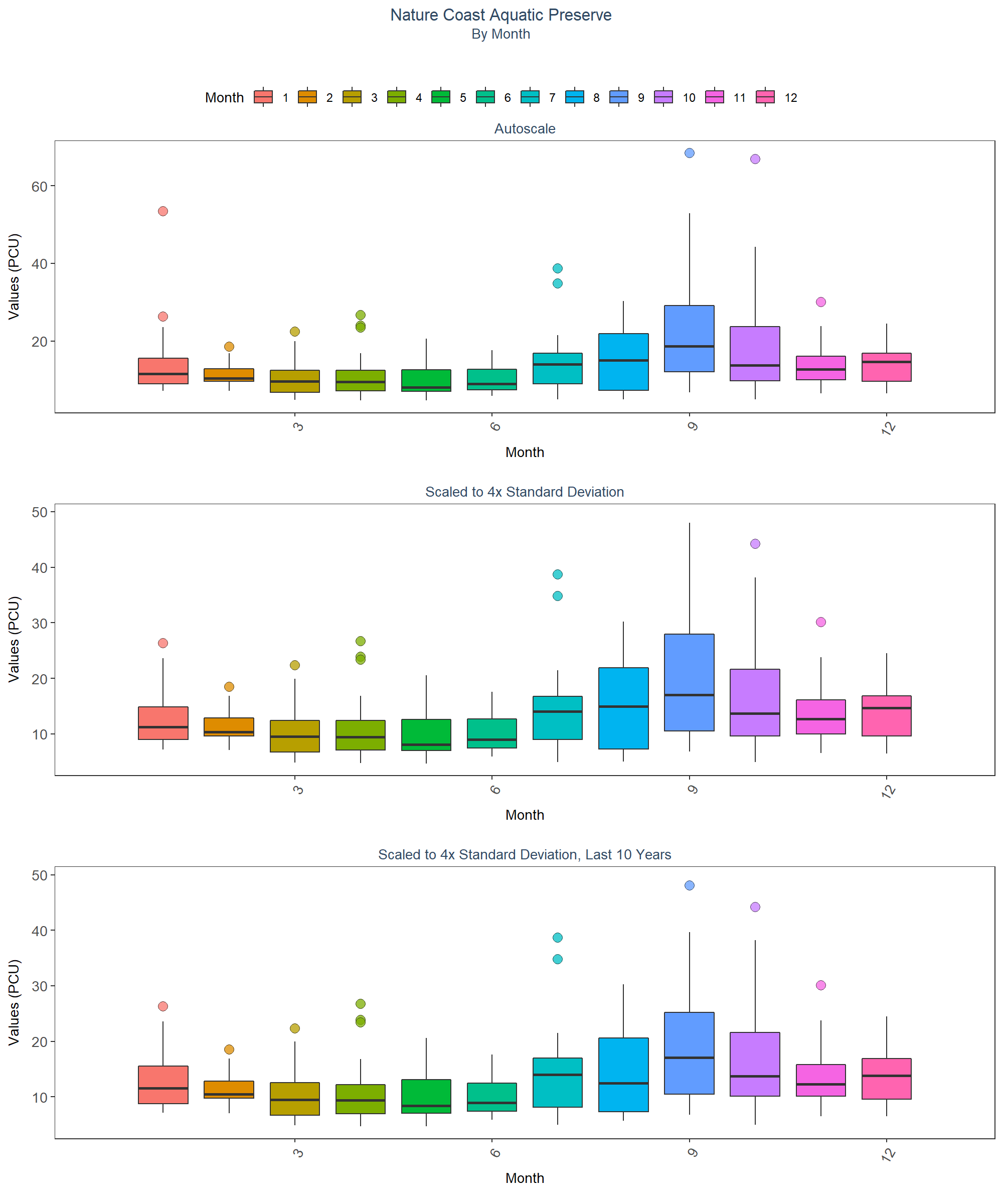
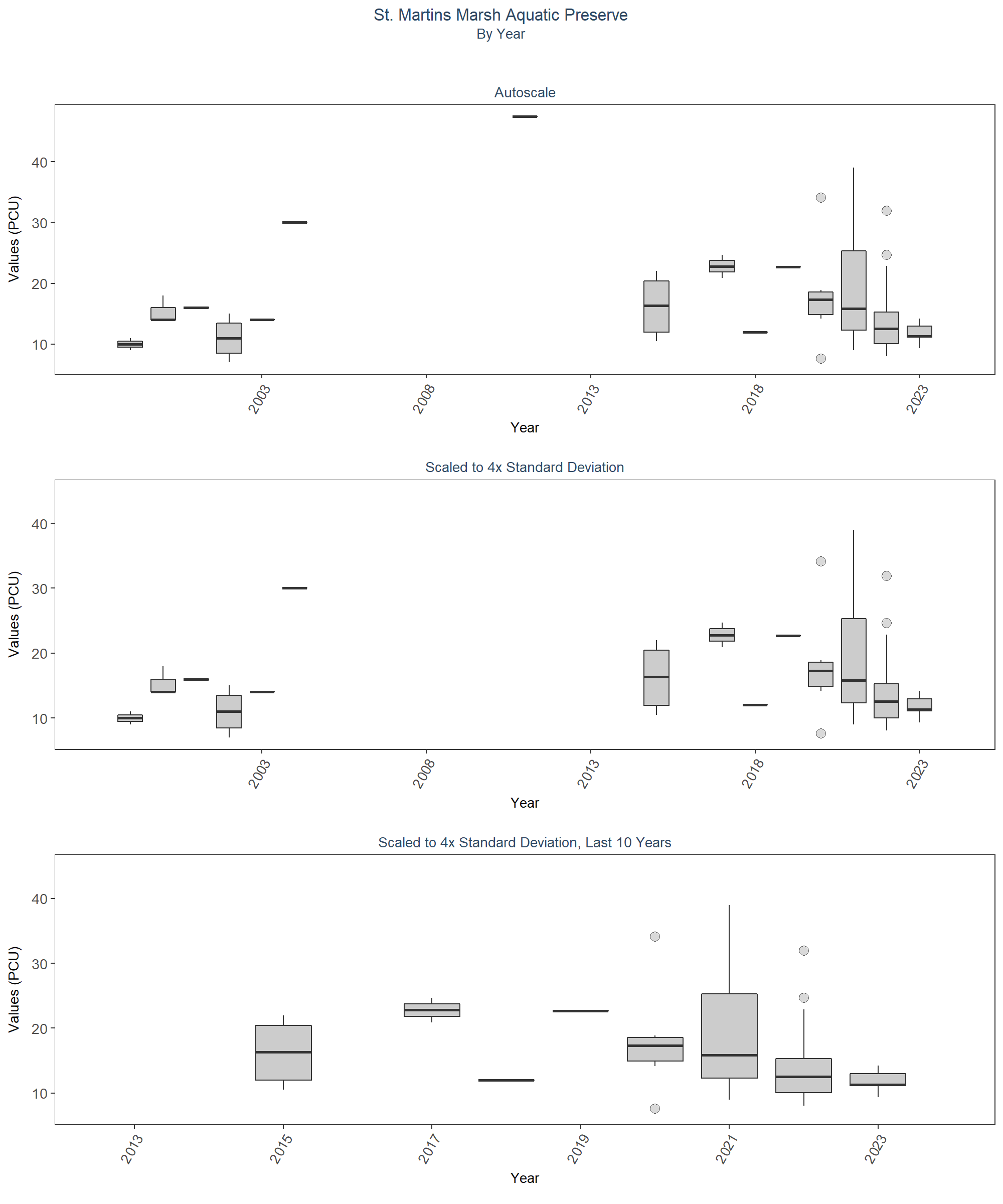
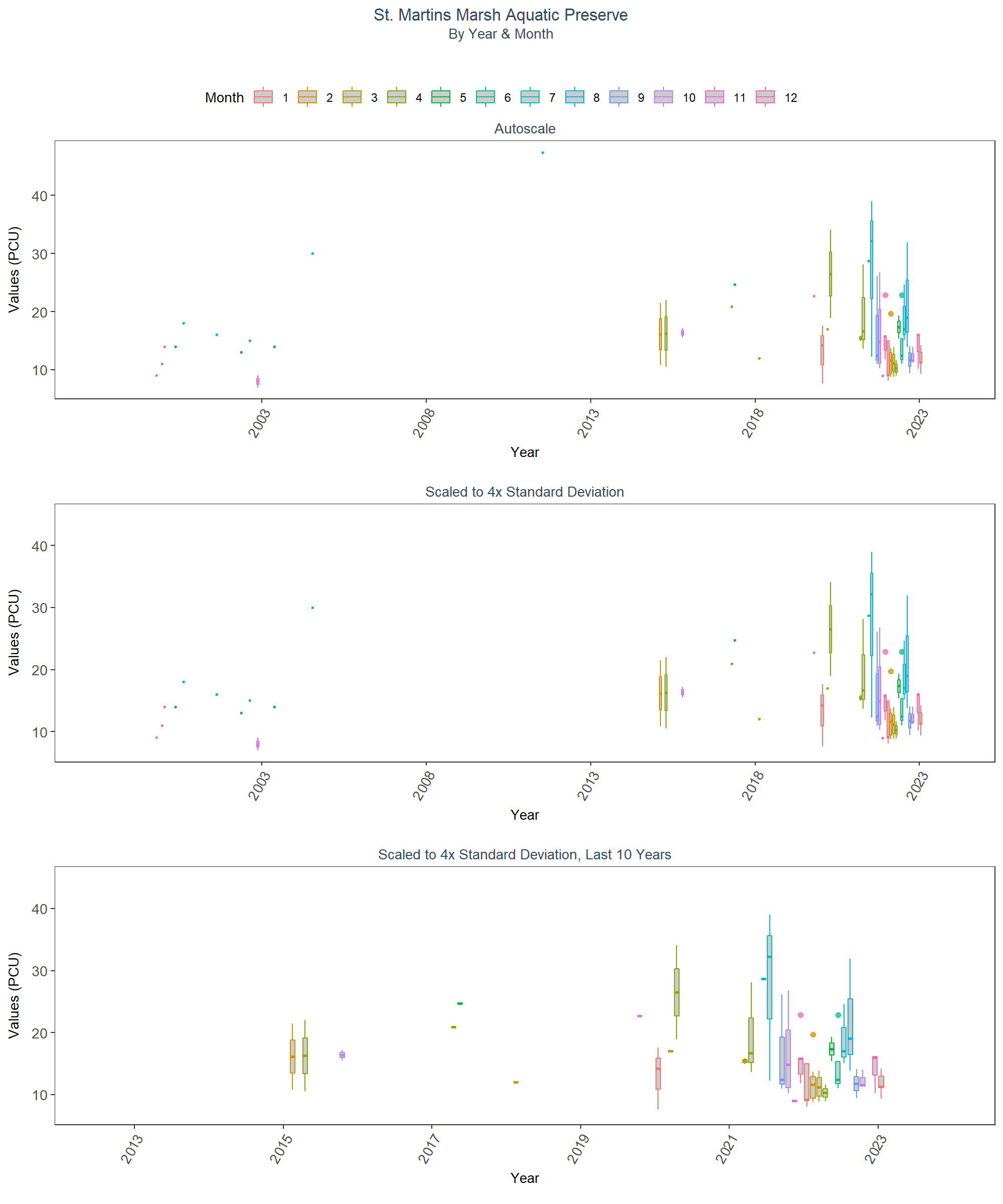
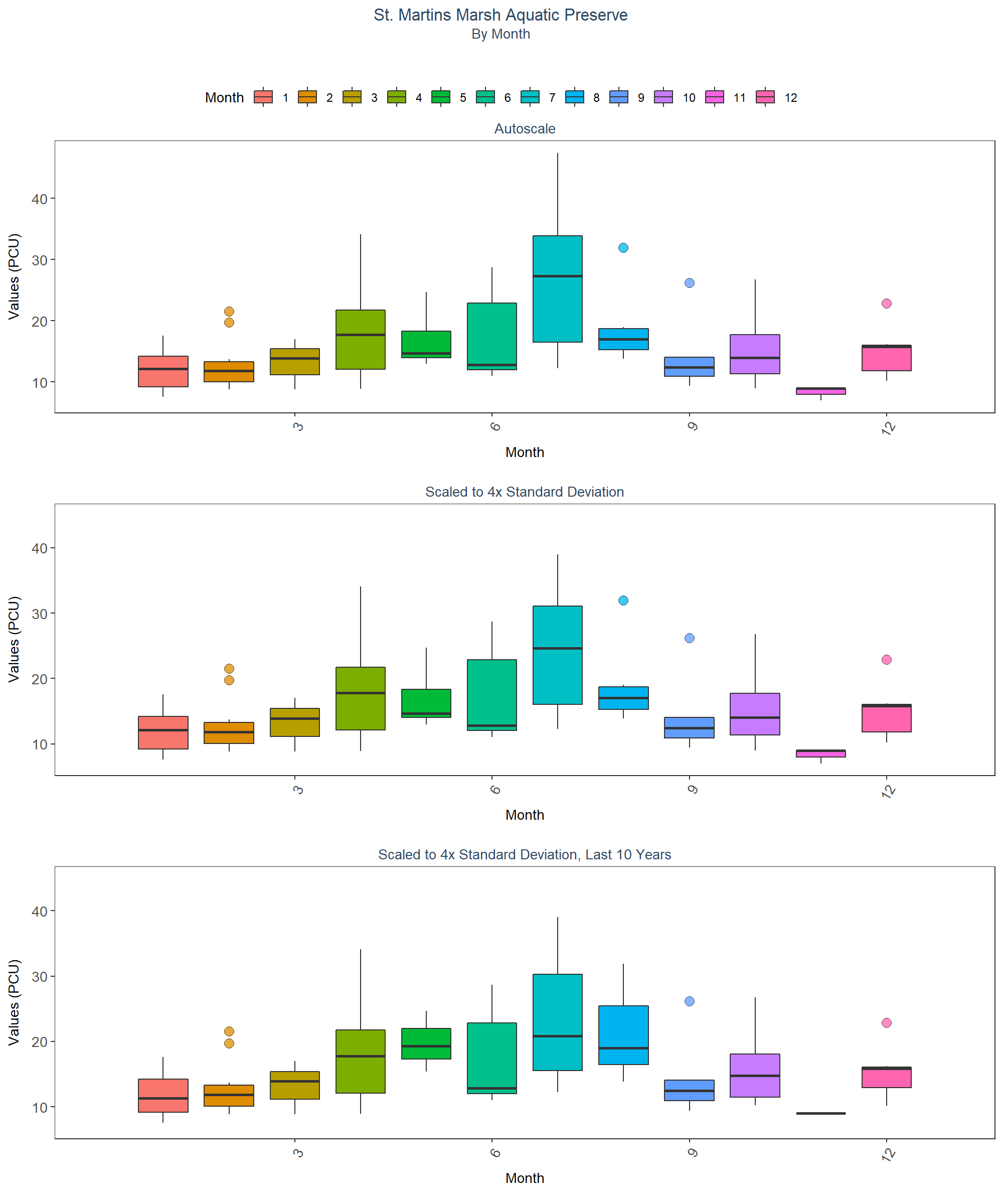
Data is taken and grouped by ManagedAreaName. The scripts that create plots follow this format

1. Use the data set that only has SufficientData of TRUE for the desired managed area
2. Determine the earliest and latest year of the data to create x-axis scale and intervals
3. Determine the minimum, mean, and standard deviation for the data to be used for y-axis scales
   * Excludes the top 2% of values to reduce the impact of extreme outliers on the y-axis scale
4. Set what values are to be used for the x-axis, y-axis, and the variable that should determine groups for the box plots
5. Set the plot type as a box plot with the size of the outlier points
6. Create the title, x-axis, y-axis, and color fill labels
7. Set the y and x limits
8. Make the axis labels bold
9. Plot the arrangement as a set of panels

The following plots are arranged by ManagedAreaName with data grouped by Year, then Year and Month, then finally Month only. Each managed area will have 3 sets of plots, each with 3 panels in them. Each panel goes as follows:

1. Y-axis autoscaled
2. Y-axis set to be mean + 4 times the standard deviation
3. Y-axis set to be mean + 4 times the standard deviation for most recent 10 years of data

# Determines whether analyzed managed area exist. If they do, begins  
# looping through them  
if(n==0){  
 print("There are no managed areas that qualify.")  
} else {  
 # Begin looping through managed area  
 for (i in 1:n) {  
 # Determine upper and lower bounds of time for x-axis  
 plot\_data <- data[data$SufficientData==TRUE &  
 data$ManagedAreaName==MA\_Include[i],]  
 year\_lower <- min(plot\_data$Year)  
 year\_upper <- max(plot\_data$Year)  
 # Determine upper and lower bounds of ResultValue for y-axis  
 min\_RV <- min(plot\_data$ResultValue)  
 mn\_RV <- mean(plot\_data$ResultValue[plot\_data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
 sd\_RV <- sd(plot\_data$ResultValue[plot\_data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
 # Sets x- and y-axis scale  
 x\_scale <- ifelse(year\_upper - year\_lower > 30, 10, 5)  
 y\_scale <- mn\_RV + 4 \* sd\_RV  
   
 ##Year plots  
 # Create plot object for auto-scaled y-axis plot  
 p1 <- ggplot(data=plot\_data,  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme  
 # Create plot object for y-axis scaled plot  
 p2 <- ggplot(data=plot\_data,  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme  
 # Create plot object for y-axis scaled plot for past 10 years  
 p3 <- ggplot(data=plot\_data[plot\_data$Year >= year\_upper - 10, ],  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_upper - 10.5, year\_upper + 1),  
 breaks=rev(seq(year\_upper, year\_upper - 10,-2))) +  
 plot\_theme  
 # Arrange plot objects  
 Yset <- ggarrange(p1, p2, p3, ncol=1)  
 # Create plot title object  
 p0 <- ggplot() + labs(title=paste0(MA\_Include[i]),  
 subtitle="By Year") +  
 plot\_theme + theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(),  
 axis.line=element\_blank())  
   
   
 ## Year & Month Plots  
 # Create plot object for auto-scaled y-axis plot  
 p4 <- ggplot(data=plot\_data,  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Autoscale",  
 x="Year", y=paste0("Values (", unit, ")"), color="Month") +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme +  
 theme(legend.position="none")  
 # Create plot object for y-axis scaled plot  
 p5 <- ggplot(data=plot\_data,  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")"), color="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(color=guide\_legend(nrow=1))  
 # Create plot object for y-axis scaled plot for past 10 years  
 p6 <- ggplot(data=plot\_data[plot\_data$Year >= year\_upper - 10, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")"), color="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_upper - 10.5, year\_upper + 1),  
 breaks=rev(seq(year\_upper, year\_upper - 10,-2))) +  
 plot\_theme +  
 theme(legend.position="none")  
 # Create legend object  
 leg1 <- get\_legend(p5)  
 # Arrange plots and legend  
 YMset <- ggarrange(leg1, p4, p5 + theme(legend.position="none"), p6,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
 # Create plot title object  
 p00 <- ggplot() + labs(title=paste0(MA\_Include[i]),  
 subtitle="By Year & Month") + plot\_theme +  
 theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
   
 ## Month Plots  
 # Create plot object for auto-scaled y-axis plot  
 p7 <- ggplot(data=plot\_data,  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale",  
 x="Month", y=paste0("Values (", unit, ")"), fill="Month") +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
 # Create plot object for y-axis scaled plot  
 p8 <- ggplot(data=plot\_data,  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Month", y=paste0("Values (", unit, ")"), fill="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(fill=guide\_legend(nrow=1))  
 # Create plot object for y-axis scaled plot for past 10 years  
 p9 <- ggplot(data=plot\_data[plot\_data$Year >= year\_upper - 10, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Month", y=paste0("Values (", unit, ")"), fill="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
 # Create legend object  
 leg2 <- get\_legend(p8)  
 # Arrange plots and legend  
 Mset <- ggarrange(leg2, p7, p8 + theme(legend.position="none"), p9,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
 # Create title object  
 p000 <- ggplot() + labs(title=paste0(MA\_Include[i]),  
 subtitle="By Month") + plot\_theme +  
 theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
 # Arrange and display plots with titles for all combinations  
 print(ggarrange(p0, Yset, ncol=1, heights=c(0.07, 1)))  
 print(ggarrange(p00, YMset, ncol=1, heights=c(0.07, 1)))  
 print(ggarrange(p000, Mset, ncol=1, heights=c(0.07, 1, 0.7)))  
   
 rm(plot\_data)  
 rm(p1, p2, p3, p4, p5, p6, p7, p8, p9, p0, p00, p000, leg1, leg2,  
 Yset, YMset, Mset)  
 }  
}

# Appendix V: Excluded Managed Areas

Scatter plots of data values are created for managed areas that have fewer than 10 separate years of data entries. Data points are colored based on specific value qualifiers of interest.

# Determines whether excluded managed areas exist. If they do, begins  
# looping through them  
if(z==0){  
 print("There are no managed areas that qualify.")  
} else {  
 for(i in 1:z){  
 # Create scatter plot with data  
 p1<-ggplot(data=data[data$ManagedAreaName==MA\_Exclude$ManagedAreaName[i]&  
 data$Include==TRUE, ],  
 aes(x=SampleDate, y=ResultValue, fill=VQ\_Plot)) +  
 geom\_point(shape=21, size=3, color="#333333", alpha=0.75) +  
 labs(title=paste0(MA\_Exclude$ManagedAreaName[i], " (",  
 MA\_Exclude$N\_Years[i], " Unique Years)"),  
 subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")"), fill="Value Qualifier") +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal",  
 legend.justification="right") +  
 scale\_x\_date(labels=date\_format("%m-%Y")) +  
 {if(inc\_H==TRUE){  
 scale\_fill\_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",  
 "HU"="#7CAE00"), na.value="#cccccc")  
 } else if(param\_name=="Secchi\_Depth"){  
 scale\_fill\_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",  
 "SU"="#7CAE00"), na.value="#cccccc")  
 } else {  
 scale\_fill\_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")  
 }}  
 print(p1)  
 }  
}

