SEACAR Discrete Water Quality Analysis: Sample Bottom Total Suspended Solids, TSS

Last compiled on 29 June, 2022

# Important Notes

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

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

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

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(knitr)  
library(data.table)  
library(plyr)  
library(dplyr)  
library(lubridate)  
library(ggplot2)  
library(ggpubr)  
library(scales)  
library(EnvStats)  
library(tidyr)  
library(stringr)  
library(kableExtra)  
  
windowsFonts(`Segoe UI`=windowsFont('Segoe UI'))  
options(scipen=999)  
opts\_chunk$set(warning=FALSE, message=FALSE, dpi=200)

# File Import

Imports file that is determined in the WC\_Discrete\_parameter\_ReportCompile.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, units of the parameter, sets the SampleDate as a date object, and creates various scales of the date to be used by plotting functions.

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="")  
  
parameter <- unique(data$ParameterName)  
unit <- unique(data$ParameterUnits)  
data$SampleDate <- as.Date(data$SampleDate)  
data$YearMonth <- paste0(data$Month, "-", data$Year)  
data$YearMonthDec <- data$Year + ((data$Month-0.5) / 12)  
data$DecDate <- decimal\_date(data$SampleDate)  
data[, `:=` (relyear=Year - min(Year), relyear\_dd=DecDate - min(DecDate)),  
 by="ManagedAreaName"]  
data <- data[ParameterName==parameter & str\_detect(ActivityType, activity) &  
 RelativeDepth==depth & Include==1, ]

# Data Filtering and Data Impacted by Specific Value Qualifiers

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.docx](https://github.com/FloridaSEACAR/SEACAR_Panzik/blob/main/SEACAR%20Documentation%20-%20Analysis%20Filters%20and%20Calculations.docx)

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 managed area, which is that each managed area has 10 year or more of unique year entries for observation that pass the initial filter. If data passes the first set of filtering criteria and the time criteria, they are used in the analysis.

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

# param\_name <- "Water\_Temperature"  
# out\_dir <- here::here("WQ\_Discrete/output/by\_parameter/")  
# APP\_Plots <- TRUE  
  
if(depth=="Bottom"){  
 data$RelativeDepth[grep("12Q", data$SEACAR\_QAQCFlagCode[  
 data$RelativeDepth=="Surface"])] <- "Bottom"  
}  
  
data$Include <- as.logical(data$Include)  
data$Include[grep("H", data$ValueQualifier[data$ProgramID==476])] <- TRUE  
data <- data[!is.na(data$ResultValue),]  
  
if(param\_name!="Secchi\_Depth"){  
 data <- data[!is.na(data$RelativeDepth),]  
 data <- data[data$RelativeDepth==depth,]  
}  
  
if(length(grep("Blank", data$ActivityType))>0){  
 data <- data[-grep("Blank", data$ActivityType),]  
}  
  
if(param\_name=="Chlorophyll\_a\_uncorrected\_for\_pheophytin" |  
 param\_name=="Salinity" | param\_name=="Turbidity"){  
 data <- data[grep(activity, data$ActivityType[!is.na(data$ActivityType)]),]  
}  
  
if(param\_name=="Water\_Temperature"){  
 data <- data[data$ResultValue>=-2,]  
} else{  
 data <- data[data$ResultValue>=0,]  
}  
  
  
data <- merge.data.frame(MA\_All[,c("AreaID", "ManagedAreaName")],  
 data, by="ManagedAreaName", all=TRUE)  
  
MA\_Summ <- data %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 dplyr::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]),  
 LatestYear=max(Year[Include==TRUE]),  
 SufficientData=ifelse(N\_Data>0 & N\_Years>=10, TRUE, FALSE))  
  
data <- merge.data.frame(data, MA\_Summ[,c("ManagedAreaName", "SufficientData")],  
 by="ManagedAreaName")  
  
data$Use\_In\_Analysis <- ifelse(data$Include==TRUE & data$SufficientData==TRUE,  
 TRUE, FALSE)  
  
MA\_Summ <- MA\_Summ %>%  
 select(AreaID, ManagedAreaName, ParameterName, RelativeDepth, ActivityType,  
 SufficientData, everything())  
MA\_Summ <- as.data.frame(MA\_Summ[order(MA\_Summ$ManagedAreaName), ])  
  
  
total <- length(data$Include)  
pass\_filter <- length(data$Include[data$Include==TRUE])  
  
count\_H <- length(grep("H", data$ValueQualifier[data$ProgramID==476]))  
perc\_H <- 100\*count\_H/length(data$ValueQualifier)  
  
count\_I <- length(grep("I", data$ValueQualifier))  
perc\_I <- 100\*count\_I/length(data$ValueQualifier)  
  
count\_Q <- length(grep("Q", data$ValueQualifier))  
perc\_Q <- 100\*count\_Q/length(data$ValueQualifier)  
  
count\_S <- length(grep("S", data$ValueQualifier))  
perc\_S <- 100\*count\_S/length(data$ValueQualifier)  
  
count\_U <- length(grep("U", data$ValueQualifier))  
perc\_U <- 100\*count\_U/length(data$ValueQualifier)  
  
  
data$VQ\_Plot <- data$ValueQualifier  
  
inc\_H <- ifelse(param\_name=="pH" | param\_name=="Dissolved\_Oxygen" |  
 param\_name=="Dissolved\_Oxygen\_Saturation", TRUE, FALSE)  
  
if (inc\_H==TRUE){  
 data$VQ\_Plot <- gsub("[^HU]+", "", data$VQ\_Plot)  
 data$VQ\_Plot <- gsub("UH", "HU", data$VQ\_Plot)  
 data$VQ\_Plot[na.omit(data$ProgramID!=476)] <- gsub("[^U]+", "",  
 data$VQ\_Plot[  
 na.omit(data$ProgramID!=476)])  
   
 data$VQ\_Plot[data$VQ\_Plot==""] <- NA  
   
 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), "%)"))  
} else if (param\_name=="Secchi\_Depth") {  
 count\_S <- length(grep("S", data$ValueQualifier))  
 perc\_S <- 100\*count\_S/length(data$ValueQualifier)  
 data$VQ\_Plot <- gsub("[^SU]+", "", data$VQ\_Plot)  
 data$VQ\_Plot <- gsub("US", "SU", data$VQ\_Plot)  
 data$VQ\_Plot[data$VQ\_Plot==""] <- NA  
 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), "%)"))  
} else{  
 data$VQ\_Plot <- gsub("[^U]+", "", data$VQ\_Plot)  
 data$VQ\_Plot[data$VQ\_Plot==""] <- NA  
 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: 4726, Number Passed Filter: 4726  
## I Codes: 786 (16.631401%)  
## Q Codes: 43 (0.90986%)  
## U Codes: 114 (2.412188%)

data\_summ <- data %>%  
 group\_by(AreaID, ManagedAreaName) %>%  
 dplyr::summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N\_Total=length(ResultValue),  
 N\_AnalysisUse=length(ResultValue[SufficientData==TRUE]),  
 N\_H=length(grep("H", data$ValueQualifier[data$ProgramID==476])),  
 perc\_H=100\*N\_H/length(data$ValueQualifier),  
 N\_I=length(grep("I", data$ValueQualifier)),  
 perc\_I=100\*N\_I/length(data$ValueQualifier),  
 N\_Q=length(grep("Q", data$ValueQualifier)),  
 perc\_Q=100\*N\_Q/length(data$ValueQualifier),  
 N\_S=length(grep("S", data$ValueQualifier)),  
 perc\_S=100\*N\_S/length(data$ValueQualifier),  
 N\_U=length(grep("U", data$ValueQualifier)),  
 perc\_U=100\*N\_U/length(data$ValueQualifier))  
  
data\_summ <- as.data.table(data\_summ[order(data\_summ$ManagedAreaName), ])  
fwrite(data\_summ, paste0(out\_dir,"/", param\_name, "\_", activity, "\_", depth,  
 "\_DataSummary.csv"), sep=",")  
rm(data\_summ)  
MA\_Include <- MA\_Summ$ManagedAreaName[MA\_Summ$SufficientData==TRUE &  
 MA\_Summ$N\_Data<2000000]  
n <- length(MA\_Include)  
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)  
setDT(data)

# 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
   * [Click this text to open Git directory with output files](https://github.com/FloridaSEACAR/SEACAR_Panzik/tree/main/output/WQ)

MA\_YM\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, Year, Month) %>%  
 dplyr::summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N\_Data=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
MA\_YM\_Stats <- as.data.table(MA\_YM\_Stats[order(MA\_YM\_Stats$ManagedAreaName,  
 MA\_YM\_Stats$Year,  
 MA\_YM\_Stats$Month), ])  
fwrite(MA\_YM\_Stats, paste0(out\_dir,"/", param\_name, "\_", activity, "\_", depth,  
 "\_ManagedArea\_YearMonth\_Stats.txt"), sep="|")  
rm(MA\_YM\_Stats)  
  
MA\_Y\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, Year) %>%  
 dplyr::summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
MA\_Y\_Stats <- as.data.table(MA\_Y\_Stats[order(MA\_Y\_Stats$ManagedAreaName,  
 MA\_Y\_Stats$Year), ])  
fwrite(MA\_Y\_Stats, paste0(out\_dir,"/", param\_name, "\_", activity, "\_", depth,  
 "\_ManagedArea\_Year\_Stats.txt"), sep="|")  
rm(MA\_Y\_Stats)  
  
MA\_M\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, Month) %>%  
 dplyr::summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 N=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue),  
 ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),  
 collapse=', '))  
MA\_M\_Stats <- as.data.table(MA\_M\_Stats[order(MA\_M\_Stats$ManagedAreaName,  
 MA\_M\_Stats$Month), ])  
fwrite(MA\_M\_Stats, paste0(out\_dir,"/", param\_name, "\_", activity, "\_", depth,  
 "\_ManagedArea\_Month\_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
   * [Click this text to open Git directory with output files](https://github.com/FloridaSEACAR/SEACAR_Panzik/tree/main/output/WQ)

Mon\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName,  
 ProgramLocationID) %>%  
 dplyr::summarize(ParameterName=parameter,  
 RelativeDepth=depth,  
 ActivityType=activity,  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate),  
 N=length(ResultValue),  
 Min=min(ResultValue),  
 Max=max(ResultValue),  
 Median=median(ResultValue),  
 Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
  
Mon\_Stats <- as.data.table(Mon\_Stats[order(Mon\_Stats$ManagedAreaName,  
 Mon\_Stats$ProgramName,  
 Mon\_Stats$ProgramID,   
 Mon\_Stats$ProgramLocationID), ])  
fwrite(Mon\_Stats, paste0(out\_dir,"/", param\_name, "\_", activity, "\_", depth,  
 "\_MonitoringLoc\_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:

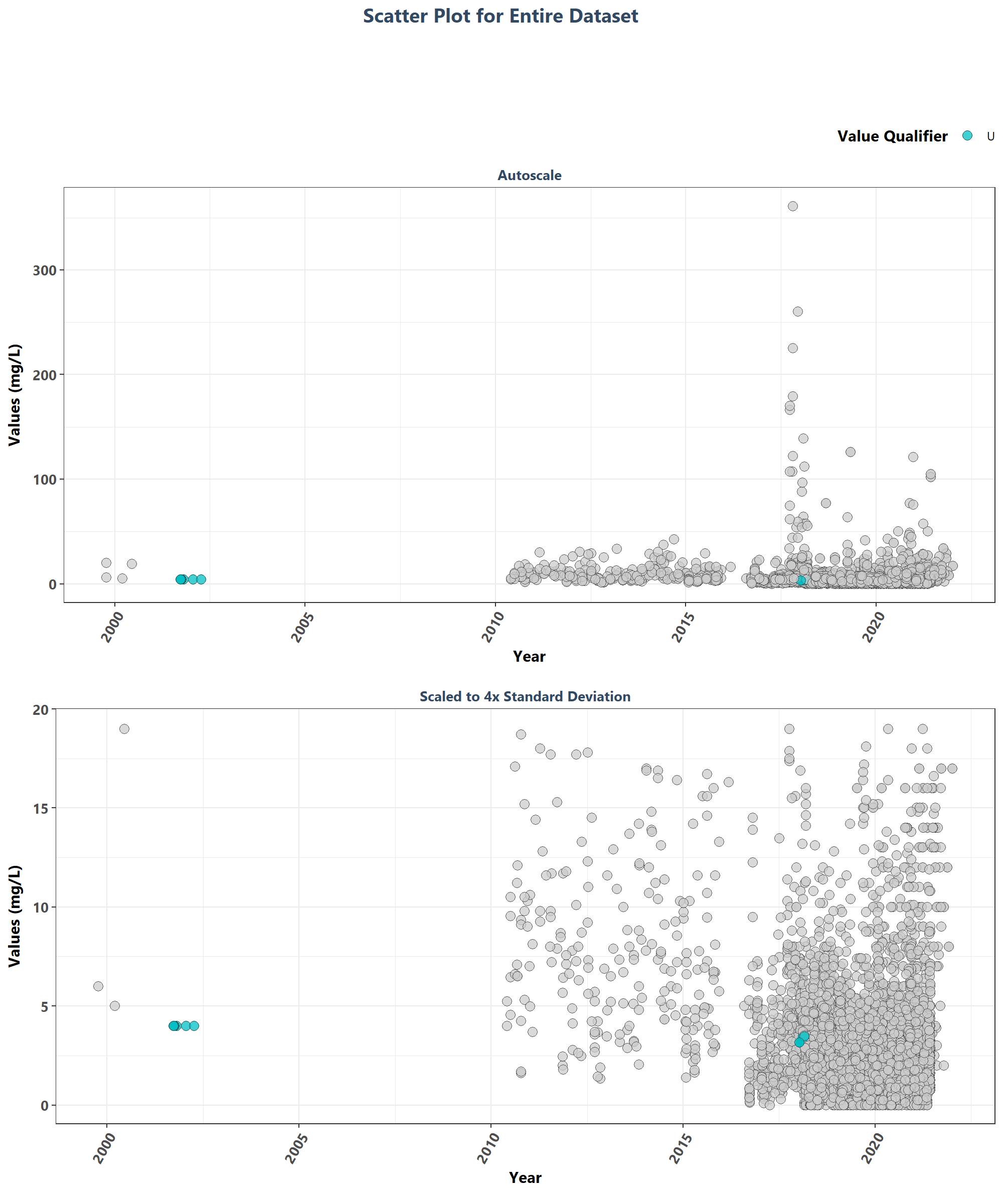
1. Define the functions used in the analysis
2. Check to see if there are any groups to run analysis on.
3. Take the data variable and only include rows that have a SufficientData value of TRUE
4. Group data that have the same ManagedAreaName.
5. For each group, provides the following information: Earliest 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
   * [Click this text to open Git directory with output files](https://github.com/FloridaSEACAR/SEACAR_Panzik/tree/main/output/WQ)

tauSeasonal <- function(data, independent, stats.median, stats.minYear,  
 stats.maxYear, seasondata=MA\_M\_Stats[  
 MA\_M\_Stats$ManagedAreaName==MA\_Include[i]]) {  
 setDT(data)   
 tau <- NULL  
 tryCatch({ken <- kendallSeasonalTrendTest(  
 y=data$ResultValue,  
 season=data$Month,  
 year=data$relyear,  
 independent.obs=independent)  
   
 tau <- ken$estimate[1]  
 z <- ken$statistic[2]  
 p\_z <- ken$p.value[2]  
 chi\_sq <- ken$statistic[1]  
 p\_chi\_sq <- ken$p.value[1]  
 slope <- ken$estimate[2]  
 intercept <- ken$estimate[3]  
 trend <- trend\_calculator(slope, stats.median, p\_z)  
   
 seasonresults <- as.data.table(ken$seasonal.estimates)  
 rm(ken)  
 }, warning=function(w) {  
 print(w)  
 }, error=function(e) {  
 print(e)  
 }, finally={  
 if (!exists("tau")) {  
 tau <- NA  
 }  
 if (!exists("z")) {  
 z <- NA  
 }  
 if (!exists("p\_z")) {  
 p\_z <- NA  
 }  
 if (!exists("chi\_sq")) {  
 chi\_sq <- NA  
 }  
 if (!exists("p\_chi\_sq")) {  
 p\_chi\_sq <- NA  
 }  
 if (!exists("slope")) {  
 slope <- NA  
 }  
 if (!exists("intercept")) {  
 intercept <- NA  
 }  
 if (!exists("trend")) {  
 trend <- NA  
 }  
 })  
 KT <-data.table(AreaID=unique(data$AreaID),  
 ManagedAreaName=unique(data$ManagedAreaName),  
 season="All",  
 stats.median=stats.median,  
 independent=independent,  
 tau=tau,  
 z=z,  
 p\_z=p\_z,  
 chi\_sq=chi\_sq,  
 p\_chi\_sq=p\_chi\_sq,  
 slope=slope,  
 intercept=intercept,  
 trend=trend)  
   
 seasonresults[, `:=` (AreaID=unique(data$AreaID),  
 ManagedAreaName=unique(data$ManagedAreaName),  
 season=unique(data$Month),  
 stats.median=as.numeric(NA),  
 independent=independent,  
 z=as.numeric(NA),  
 p\_z=as.numeric(NA),  
 chi\_sq=as.numeric(NA),  
 p\_chi\_sq=as.numeric(NA),  
 trend=as.integer(NA))]  
   
 for(s in as.integer(unique(seasonresults$season))){  
 seasondat\_s <- data[Month==s, ]  
   
 if(nrow(seasondat\_s) < 3 | length(unique(seasondat\_s$Year)) < 3 |  
 is.na(seasonresults[season==s, tau])){  
 next  
   
 } else{  
 if(!is.na(unique(seasondat\_s$Month))){  
 trend\_s <- trend\_calculator(seasonresults[season==s, slope],  
 seasondata[Month==s, Median], p\_z)  
 ken\_s <- kendallTrendTest(ResultValue ~ relyear, data=seasondat\_s)  
 seasonresults[season==s, `:=` (stats.median=unique(seasondata[  
 Month==s, Median]),  
 z=ken\_s$statistic,   
 p\_z=ken\_s$p.value,  
 chi\_sq=NA,  
 p\_chi\_sq=NA,  
 trend=trend\_s)]  
 } else{  
 next  
 }  
 }  
 }  
   
 seasonresults[, season := as.character(season)]  
   
 KT <- rbind(KT, seasonresults)  
 KT[, season := factor(season, levels=c("All", seq(1:12)), ordered=TRUE)]  
   
 return(KT)  
}  
runStats <- function(data, MA\_M\_Stats) {  
 data$Index <- as.Date(data$SampleDate) # , "%Y-%m-%d")  
 data$ResultValue <- as.numeric(data$ResultValue)  
 # Calculate basic stats  
 stats.median <- median(data$ResultValue, na.rm=TRUE)  
 stats.minYear <- min(data$relyear, na.rm=TRUE)  
 stats.maxYear <- max(data$relyear, na.rm=TRUE)  
 # Calculate Kendall Tau and Slope stats,  
 # then update appropriate columns and table  
 seasondata <- MA\_M\_Stats[MA\_M\_Stats$ManagedAreaName==MA\_Include[i]]  
 KT <- tauSeasonal(data, TRUE, stats.median,  
 stats.minYear, stats.maxYear, seasondata)  
 # if (is.null(KT[9])) {  
 if (is.na(KT[season=="All", trend])) {  
 KT <- tauSeasonal(data, FALSE, stats.median,  
 stats.minYear, stats.maxYear, seasondata)  
 }  
 if (is.null(KT.Stats)==TRUE) {  
 KT.Stats <- KT  
 } else{  
 KT.Stats <- rbind(KT.Stats, KT)  
 }  
 return(KT.Stats)  
}  
trend\_calculator <- function(slope, median\_value, p) {  
 trend <-  
 if (p < .05 & abs(slope) > abs(median\_value) / 10.) {  
 if (slope > 0) {  
 2  
 }  
 else {  
 -2  
 }  
 }  
 else if (p < .05 & abs(slope) < abs(median\_value) / 10.) {  
 if (slope > 0) {  
 1  
 }  
 else {  
 -1  
 }  
 }  
 else  
 0  
 return(trend)  
}  
KT.Stats <- NULL  
# Loop that goes through each managed area.  
# List of managed areas stored in MA\_Years$ManagedAreaName  
c\_names <- c("AreaID", "ManagedAreaName", "Season", "Median", "Independent",   
 "tau", "z", "p\_z", "chi\_sq", "p\_chi\_sq", "SennSlope",  
 "SennIntercept", "Trend")  
if(n==0){  
 KT.Stats <- data.frame(matrix(ncol=length(c\_names),  
 nrow=length(MA\_Summ$ManagedAreaName)))  
 colnames(KT.Stats) <- c\_names  
 # KT.Stats[, c("AreaID", "ManagedAreaName")] <-  
 # MA\_Summ[, c("AreaID", "ManagedAreaName")]  
} else{  
 for (i in 1:n) {  
 x <- nrow(data[data$Use\_In\_Analysis==TRUE &  
 data$ManagedAreaName==MA\_Include[i], ])  
 if (x>0) {  
 KT.Stats <- runStats(data[data$Use\_In\_Analysis==TRUE &  
 data$ManagedAreaName ==  
 MA\_Include[i], ], MA\_M\_Stats)  
 }  
 }  
 KT.Stats <- as.data.frame(KT.Stats)  
   
 if(dim(KT.Stats)[2]==1){  
 KT.Stats <- as.data.frame(t(KT.Stats))  
 }  
 colnames(KT.Stats) <- c\_names  
 rownames(KT.Stats) <- seq(1:nrow(KT.Stats))  
 KT.Stats$tau <- round(as.numeric(KT.Stats$tau), digits=4)  
 KT.Stats$z <- round(as.numeric(KT.Stats$z), digits=4)  
 KT.Stats$p\_z <- round(as.numeric(KT.Stats$p\_z), digits=4)  
 KT.Stats$chi\_sq <- round(as.numeric(KT.Stats$chi\_sq), digits=4)  
 KT.Stats$p\_chi\_sq <- round(as.numeric(KT.Stats$p\_chi\_sq), digits=4)  
 KT.Stats$SennSlope <- as.numeric(KT.Stats$SennSlope)  
 KT.Stats$SennIntercept <- as.numeric(KT.Stats$SennIntercept)  
 KT.Stats$Trend <- as.integer(KT.Stats$Trend)  
}  
  
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$ManagedAreaName,  
 KT.Stats$Season), ])  
KT.Stats2 <- copy(KT.Stats)  
KT.Stats[, `:=` (RelativeDepth=depth, Units=unit)]  
KT.Stats\_all <- rbind(KT.Stats\_all, KT.Stats)  
  
KT.Stats2$MonitoringID <- NULL  
fwrite(KT.Stats2, paste0(out\_dir,"/", param\_name, "\_", activity, "\_", depth,  
 "\_KendallTau\_Stats.txt"), sep="|")  
rm(KT.Stats2)  
data <- data[!is.na(data$ResultValue),]  
  
plot\_theme <- theme\_bw() +  
 theme(text=element\_text(family="Segoe UI"),  
 title=element\_text(face="bold"),  
 plot.title=element\_text(hjust=0.5, size=14, color="#314963"),  
 plot.subtitle=element\_text(hjust=0.5, size=10, color="#314963"),  
 axis.title.x=element\_text(margin=margin(t=5, r=0,  
 b=10, l=0)),  
 axis.title.y=element\_text(margin=margin(t=0, r=10,  
 b=0, l=0)),  
 axis.text=element\_text(size=10),  
 axis.text.x=element\_text(face="bold", angle=60, hjust=1),  
 axis.text.y=element\_text(face="bold"))

# 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.

plot\_theme <- theme\_bw() +  
 theme(text=element\_text(family="Segoe UI"),  
 title=element\_text(face="bold"),  
 plot.title=element\_text(hjust=0.5, size=14, color="#314963"),  
 plot.subtitle=element\_text(hjust=0.5, size=10, color="#314963"),  
 axis.title.x=element\_text(margin=margin(t=5, r=0,  
 b=10, l=0)),  
 axis.title.y=element\_text(margin=margin(t=0, r=10,  
 b=0, l=0)),  
 axis.text=element\_text(size=10),  
 axis.text.x=element\_text(face="bold", angle=60, hjust=1),  
 axis.text.y=element\_text(face="bold"))  
  
year\_lower <- min(data$Year)  
year\_upper <- max(data$Year)  
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  
  
  
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")  
 }}  
  
  
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")  
 }}  
  
leg <- get\_legend(p1)  
pset <- ggarrange(leg, p1 + theme(legend.position="none"), p2,  
 ncol=1, heights=c(0.1, 1, 1))  
  
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())  
  
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.

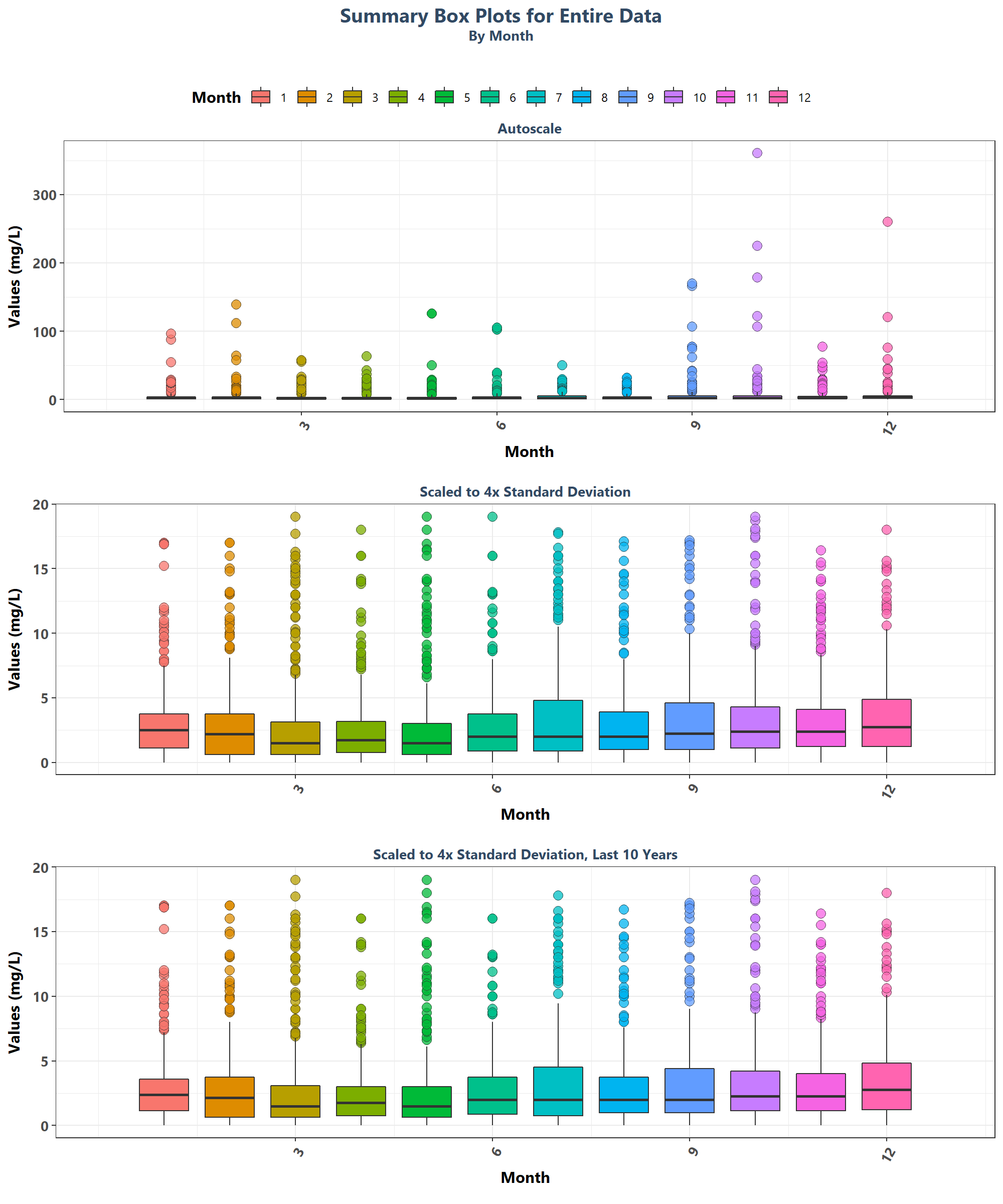
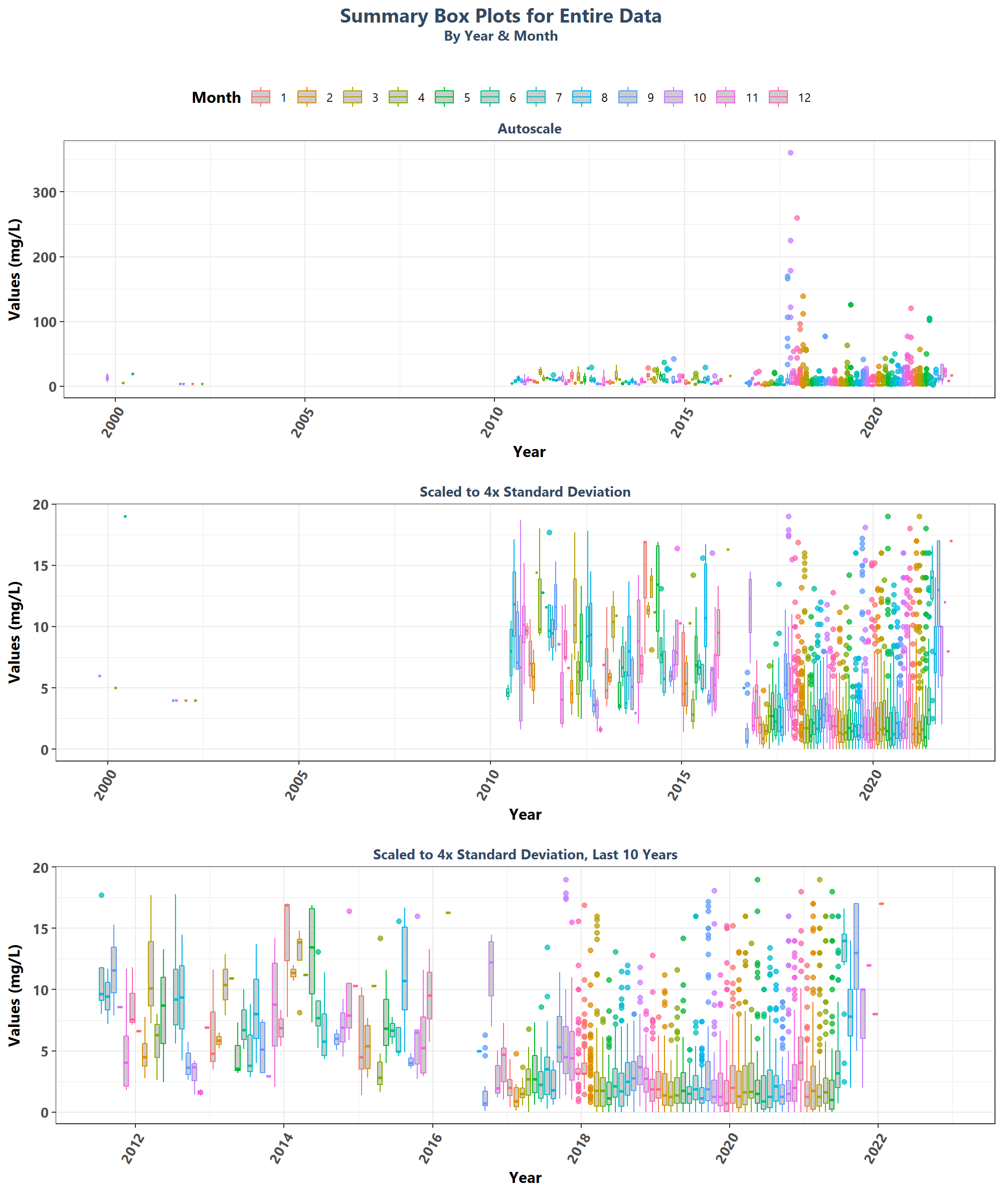
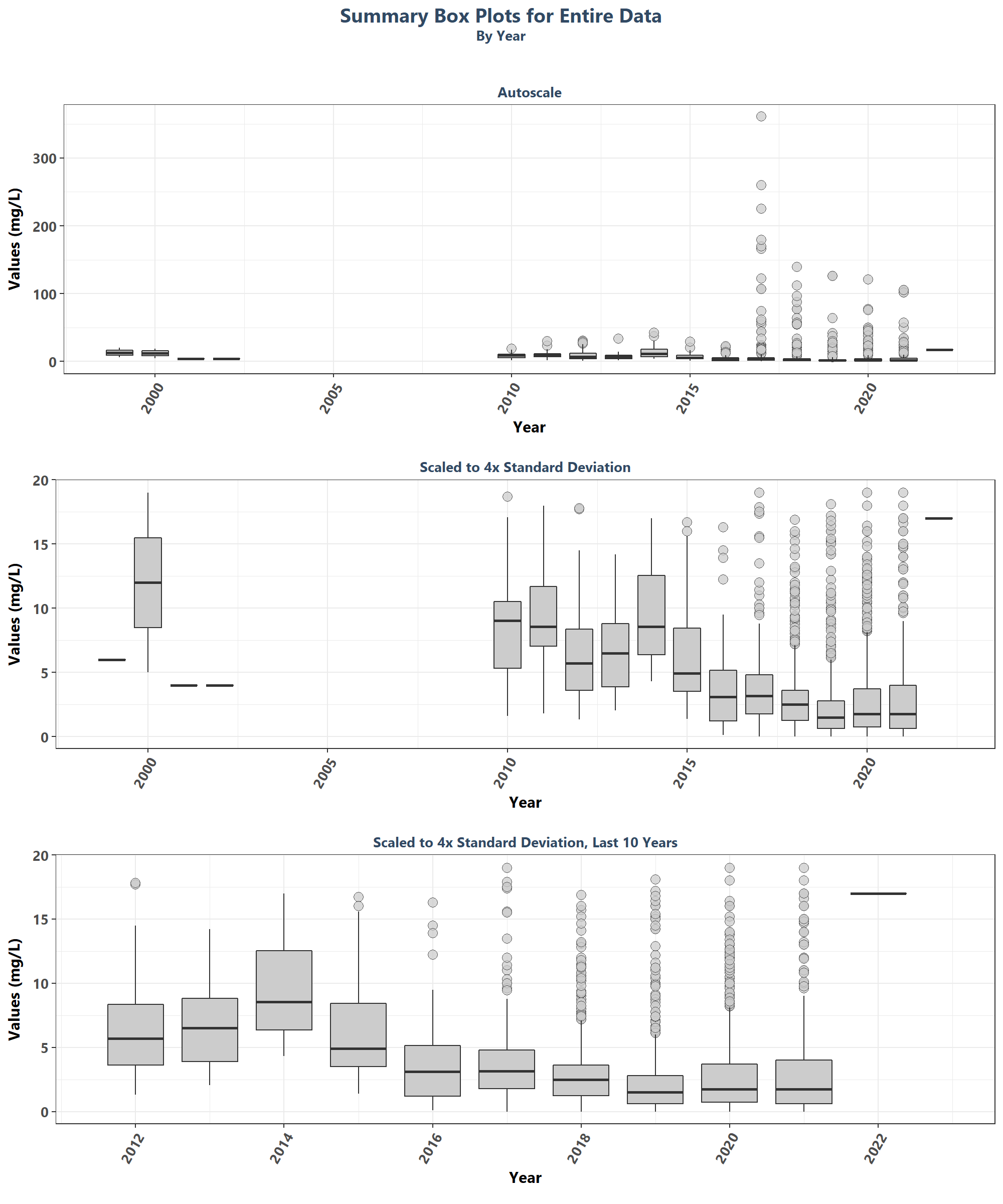
min\_RV <- min(data$ResultValue[data$Include==TRUE])  
mn\_RV <- mean(data$ResultValue[data$Include==TRUE &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
sd\_RV <- sd(data$ResultValue[data$Include==TRUE &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
y\_scale <- mn\_RV + 4 \* sd\_RV  
  
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  
  
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  
  
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  
  
set <- ggarrange(p1, p2, p3, ncol=1)  
  
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())  
  
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.

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))  
  
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"))  
  
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")  
leg <- get\_legend(p1)  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
  
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())  
  
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.

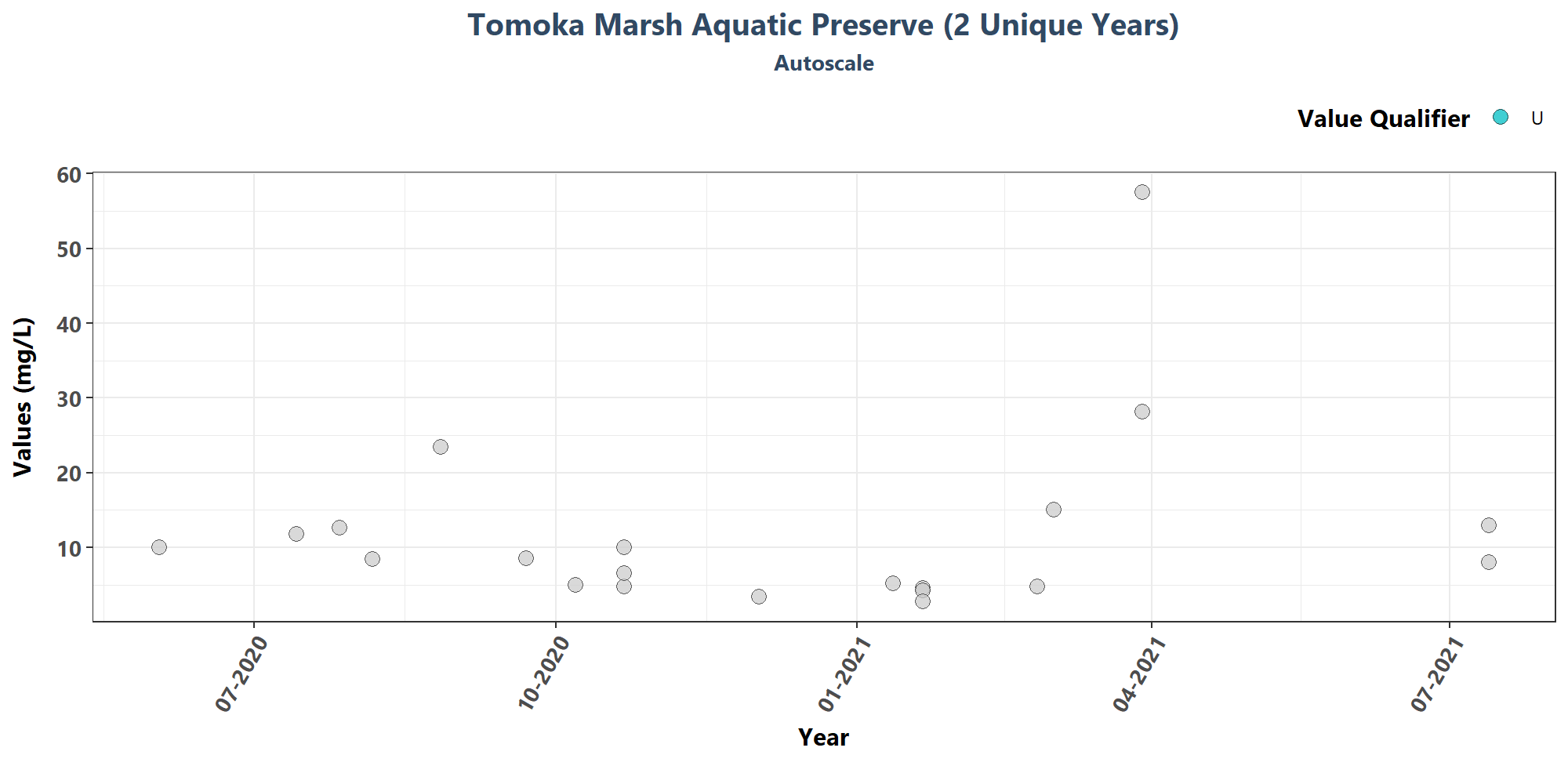
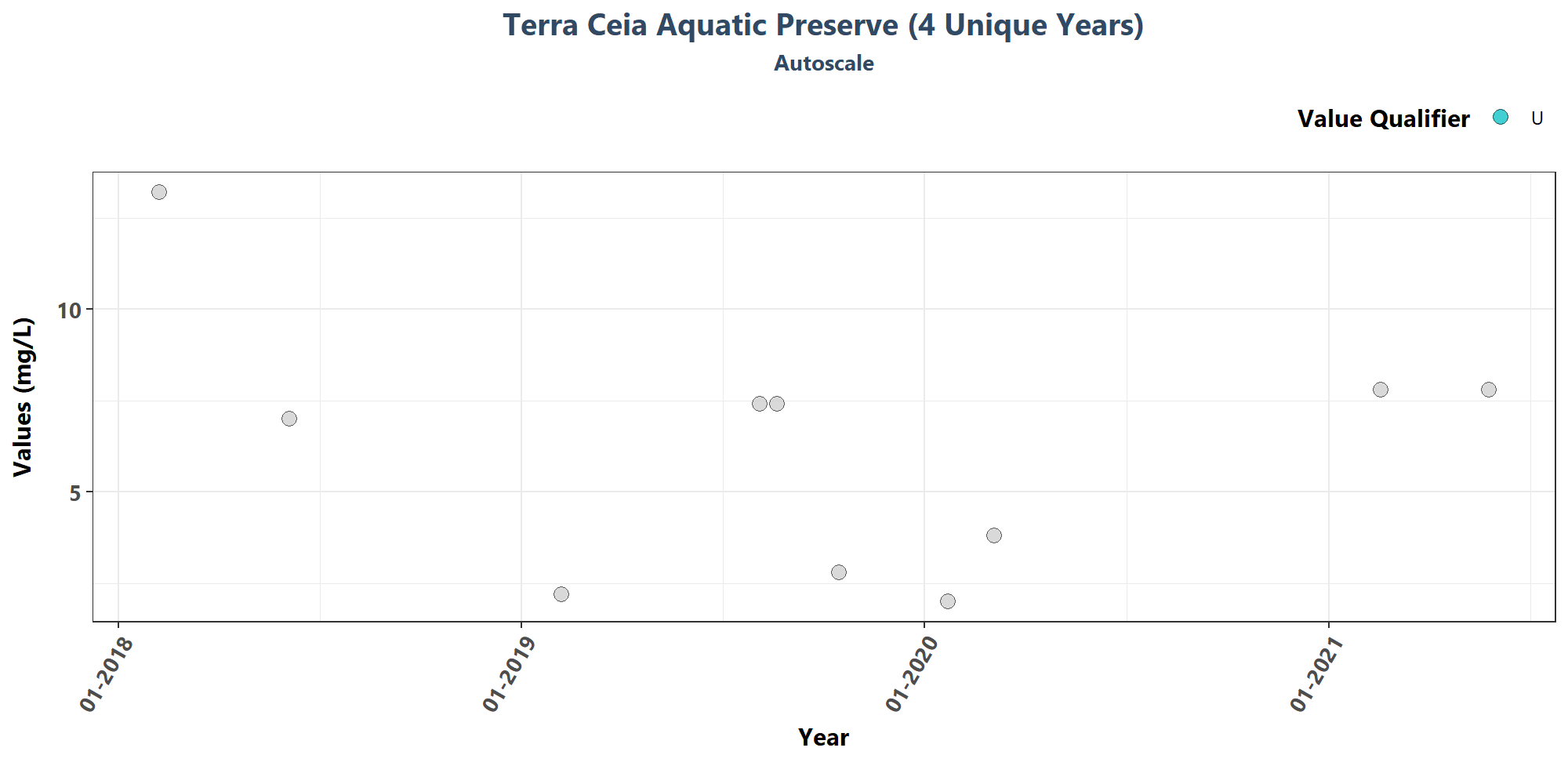
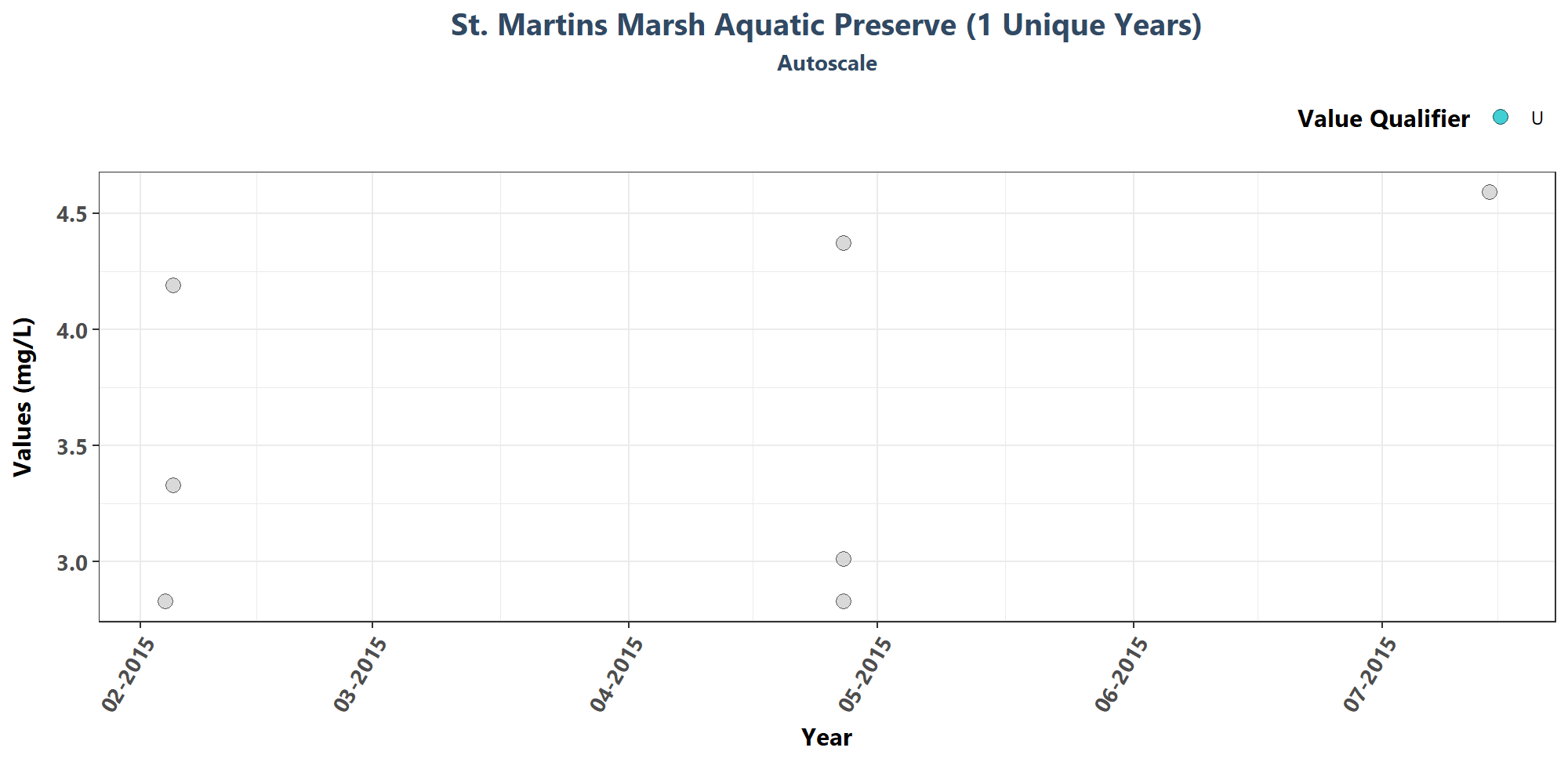
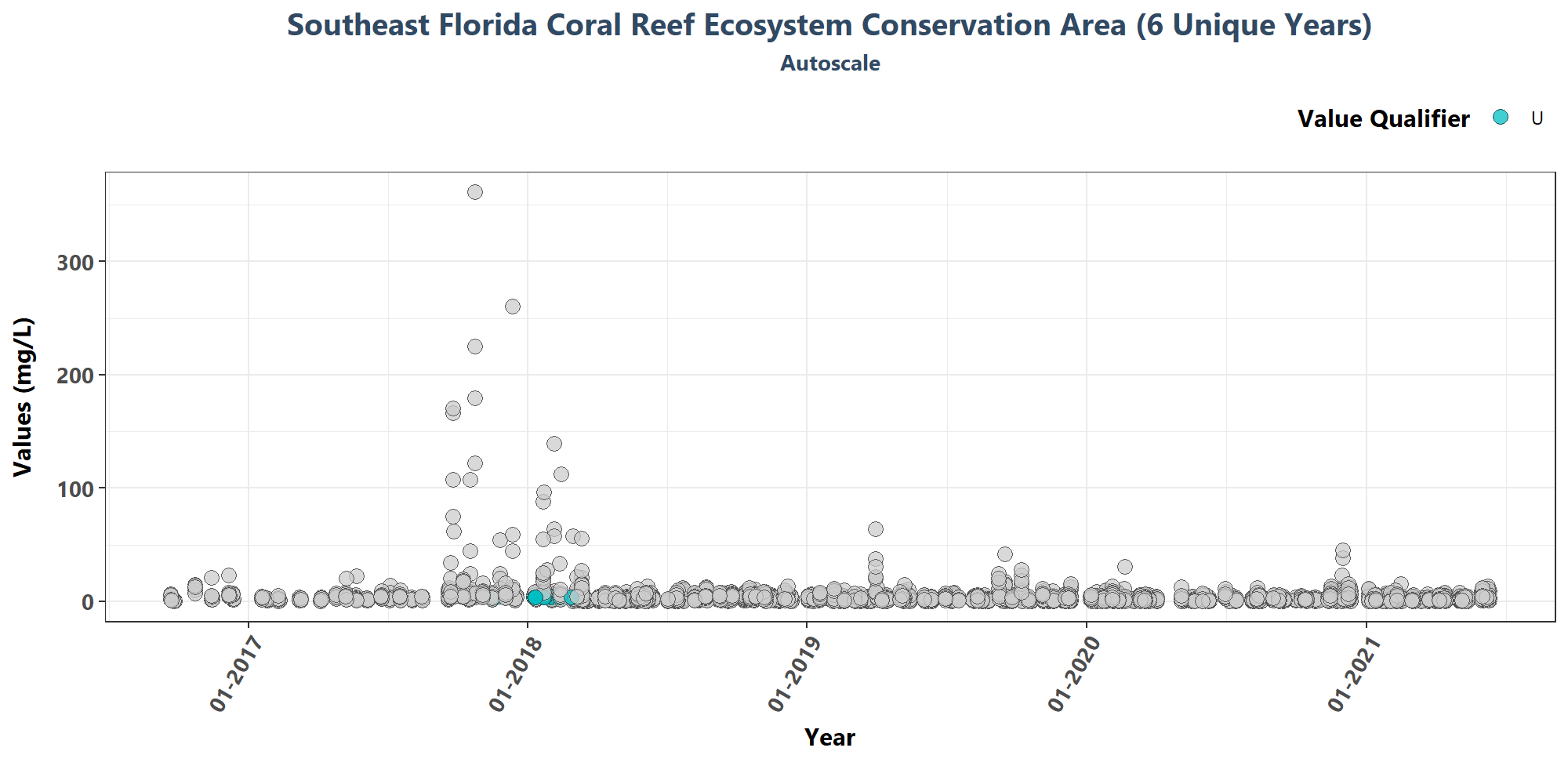
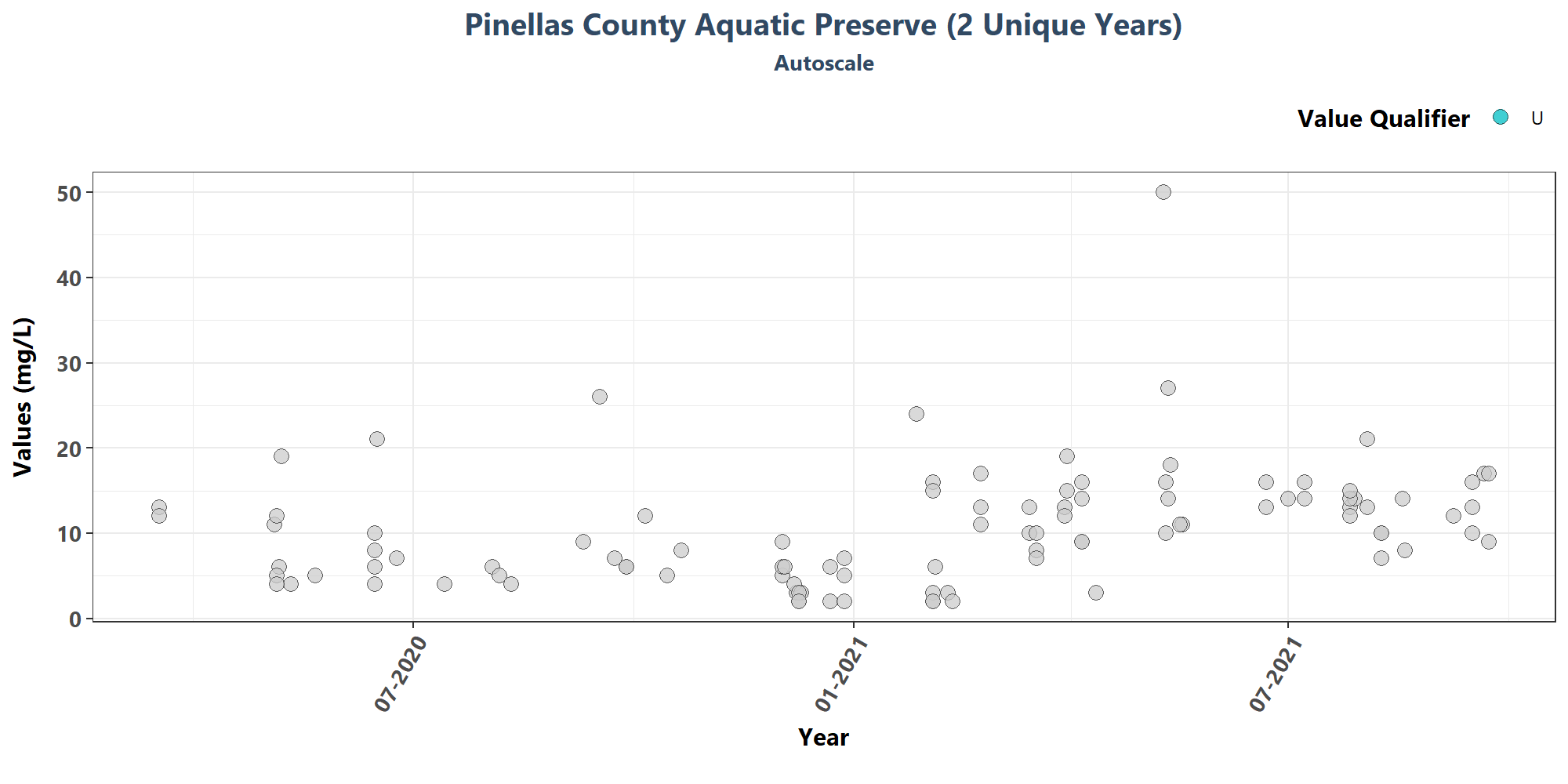
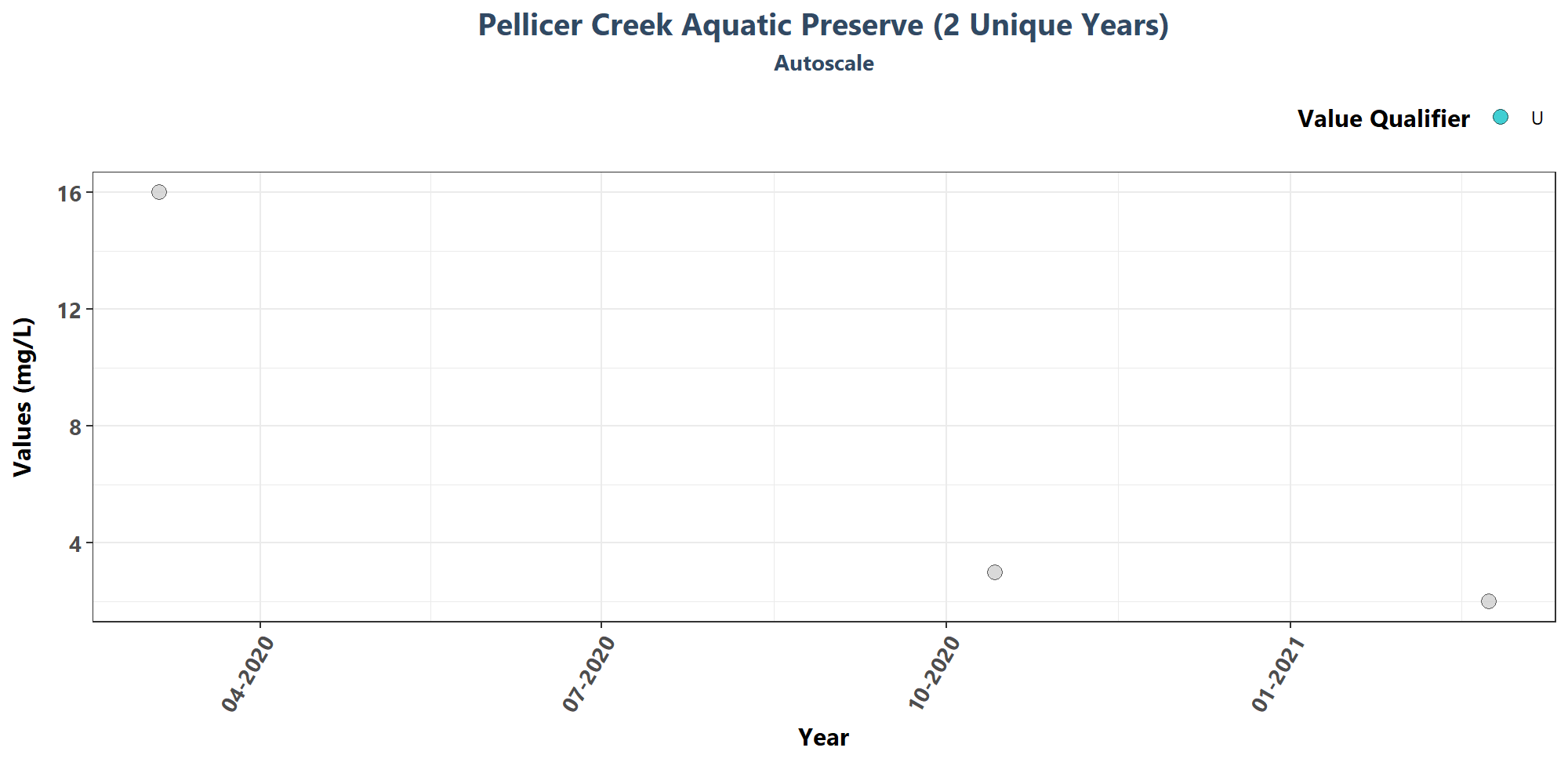
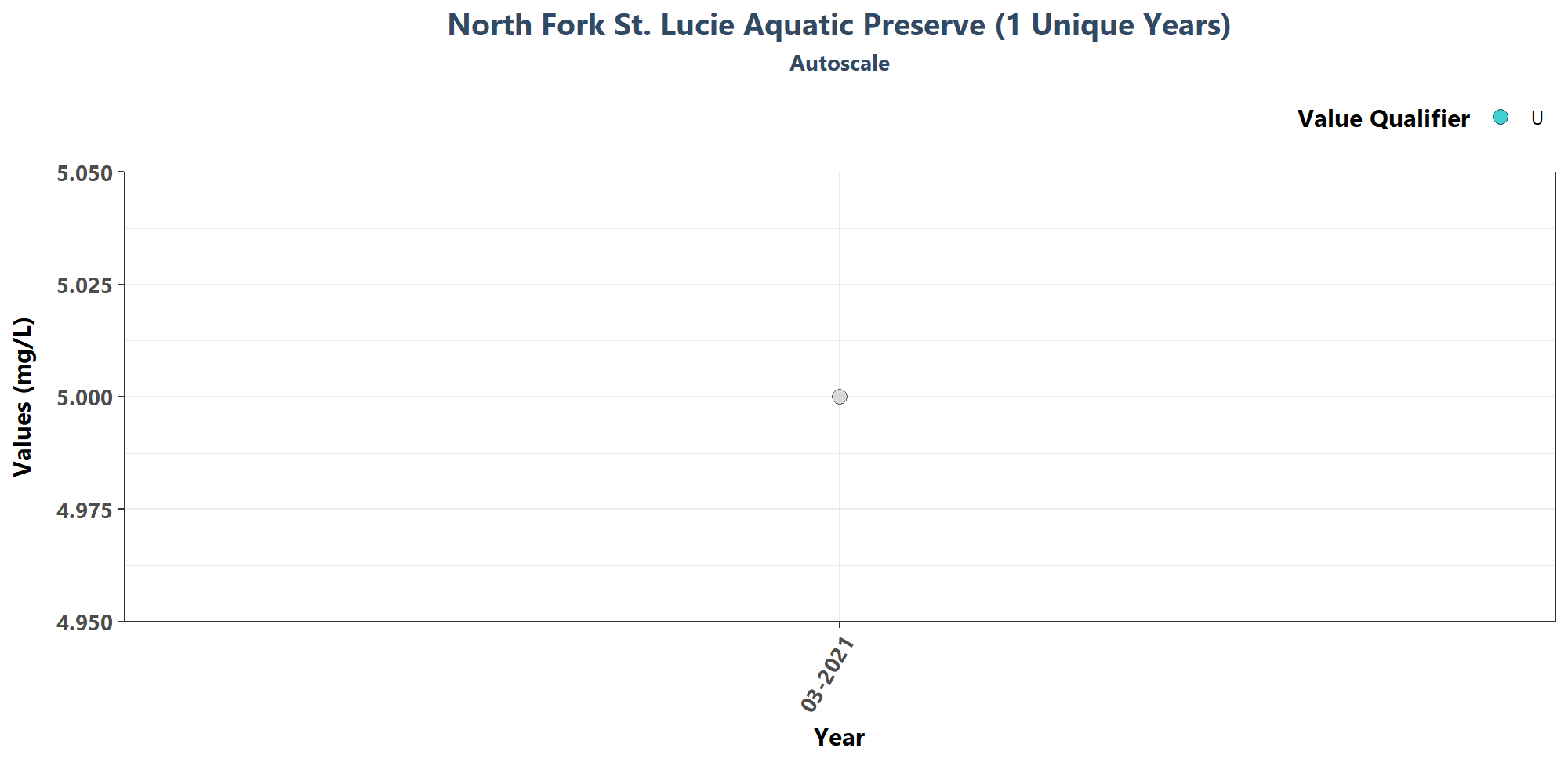
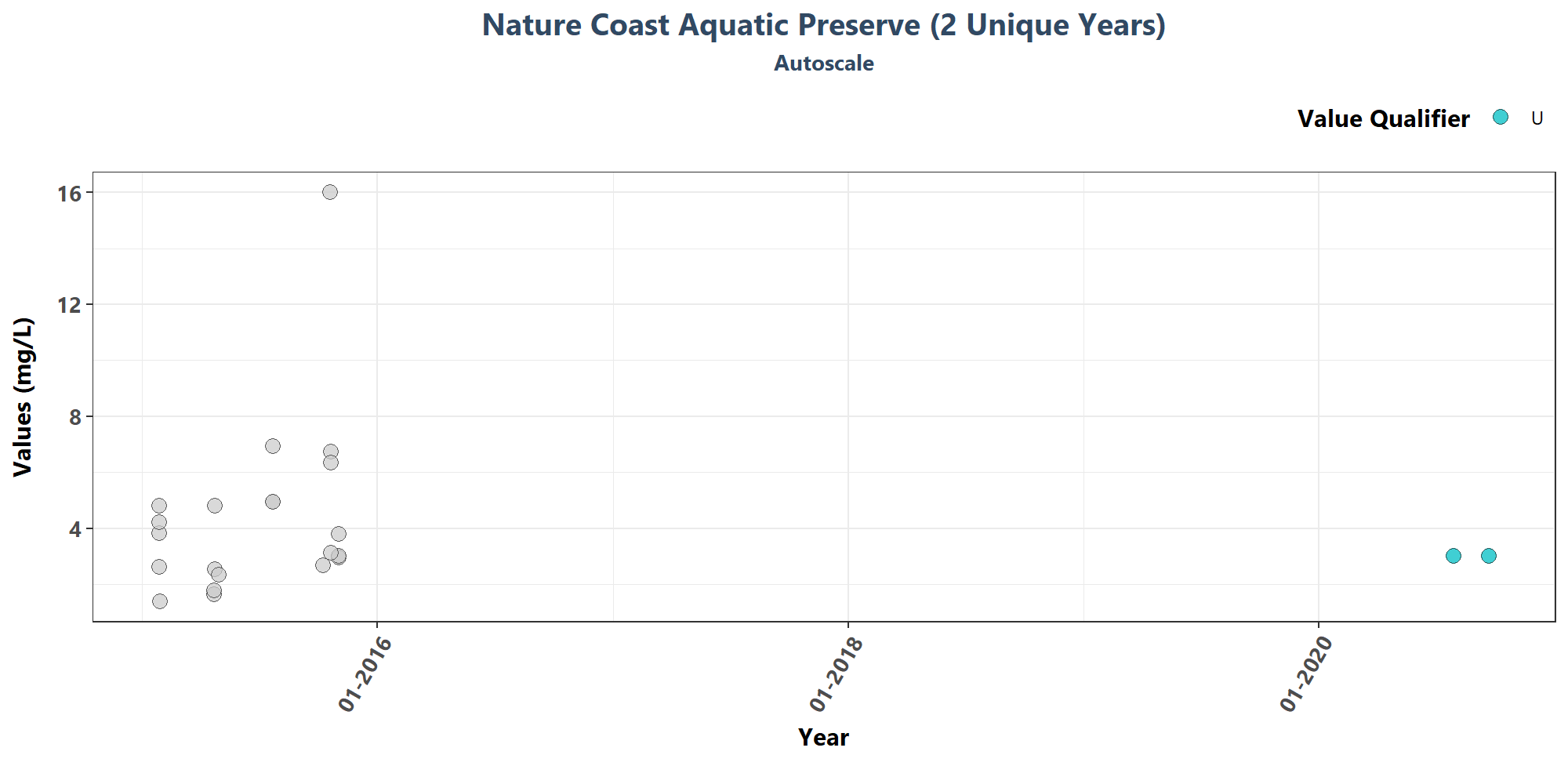
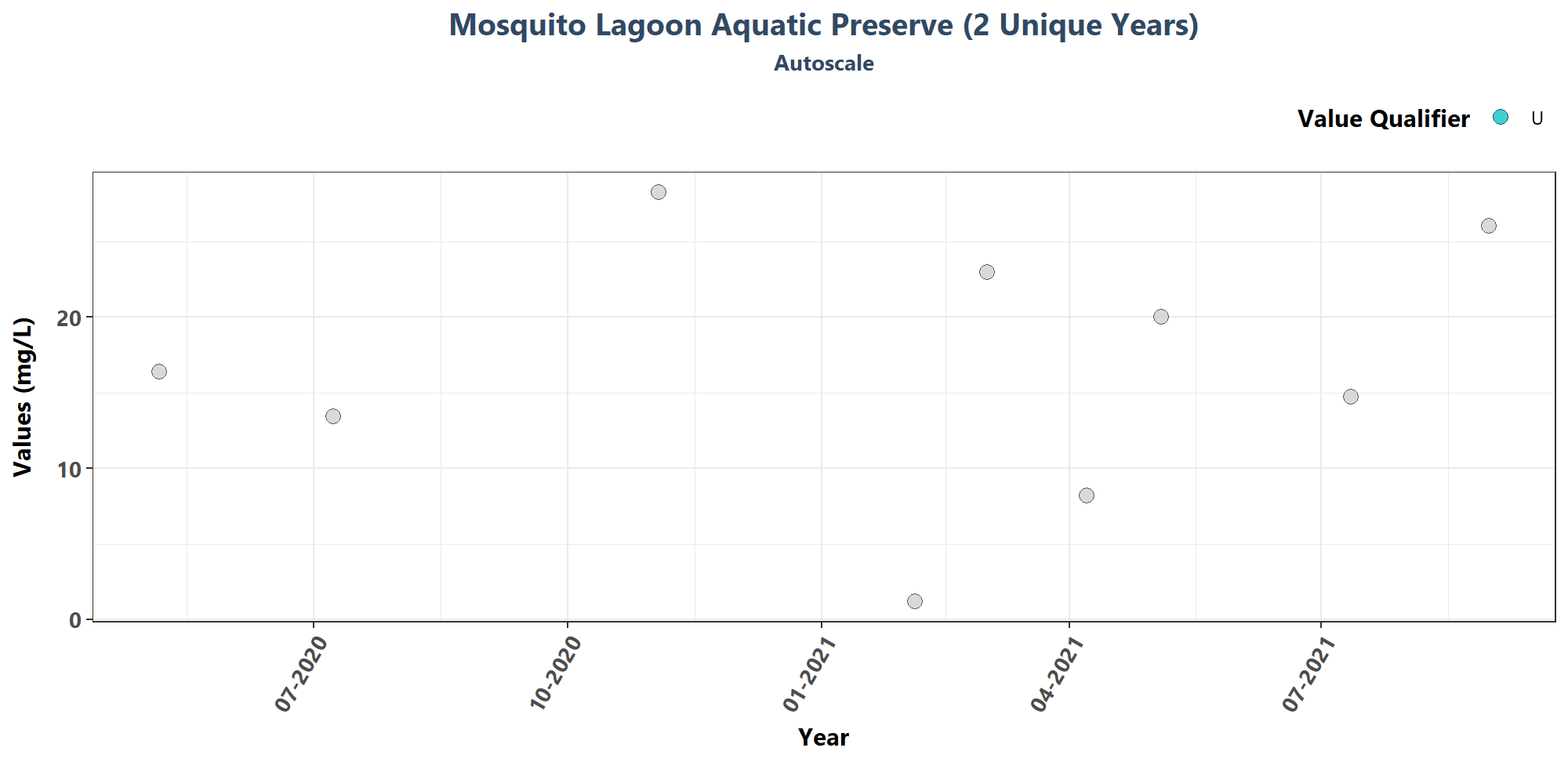
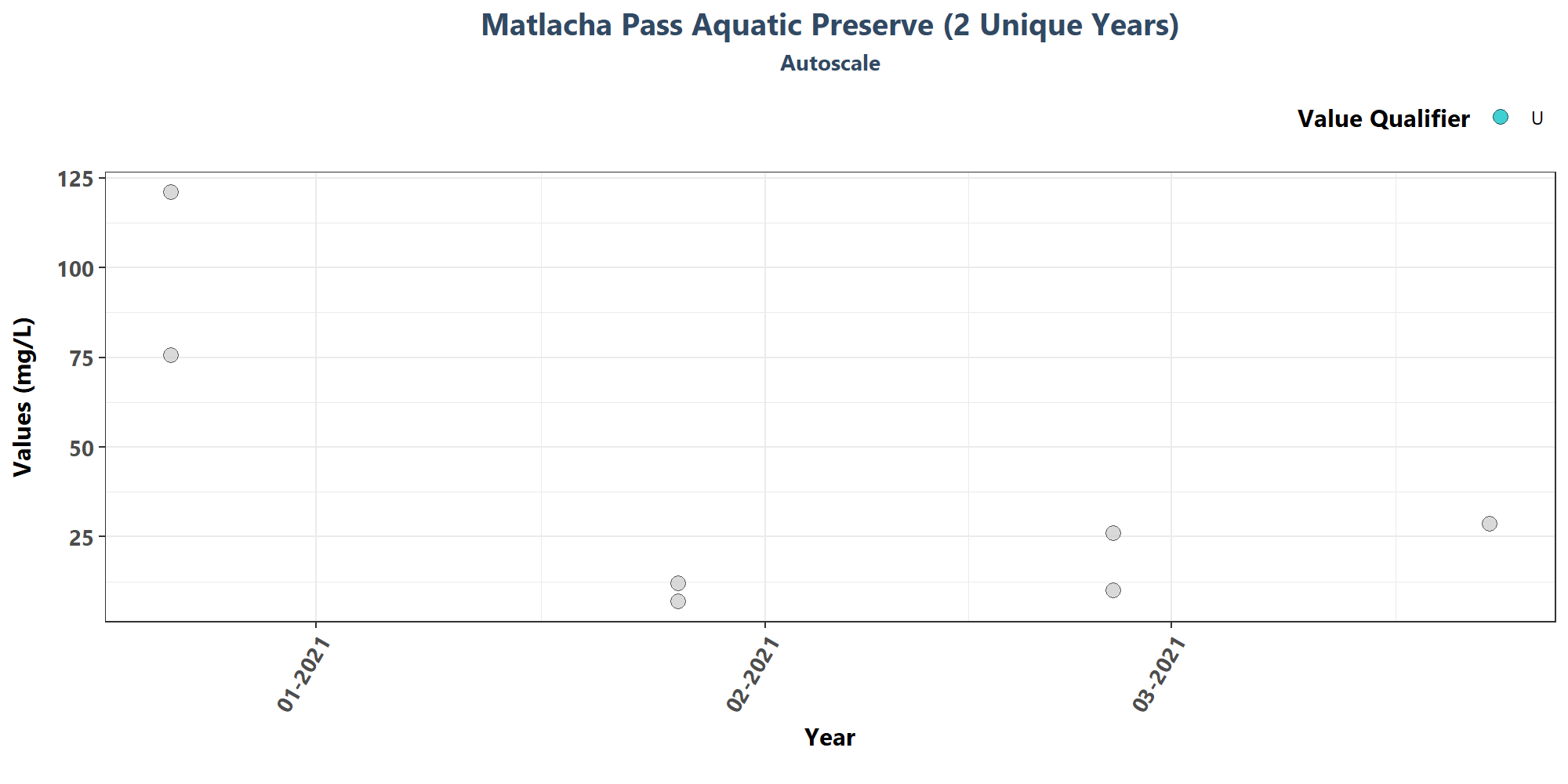
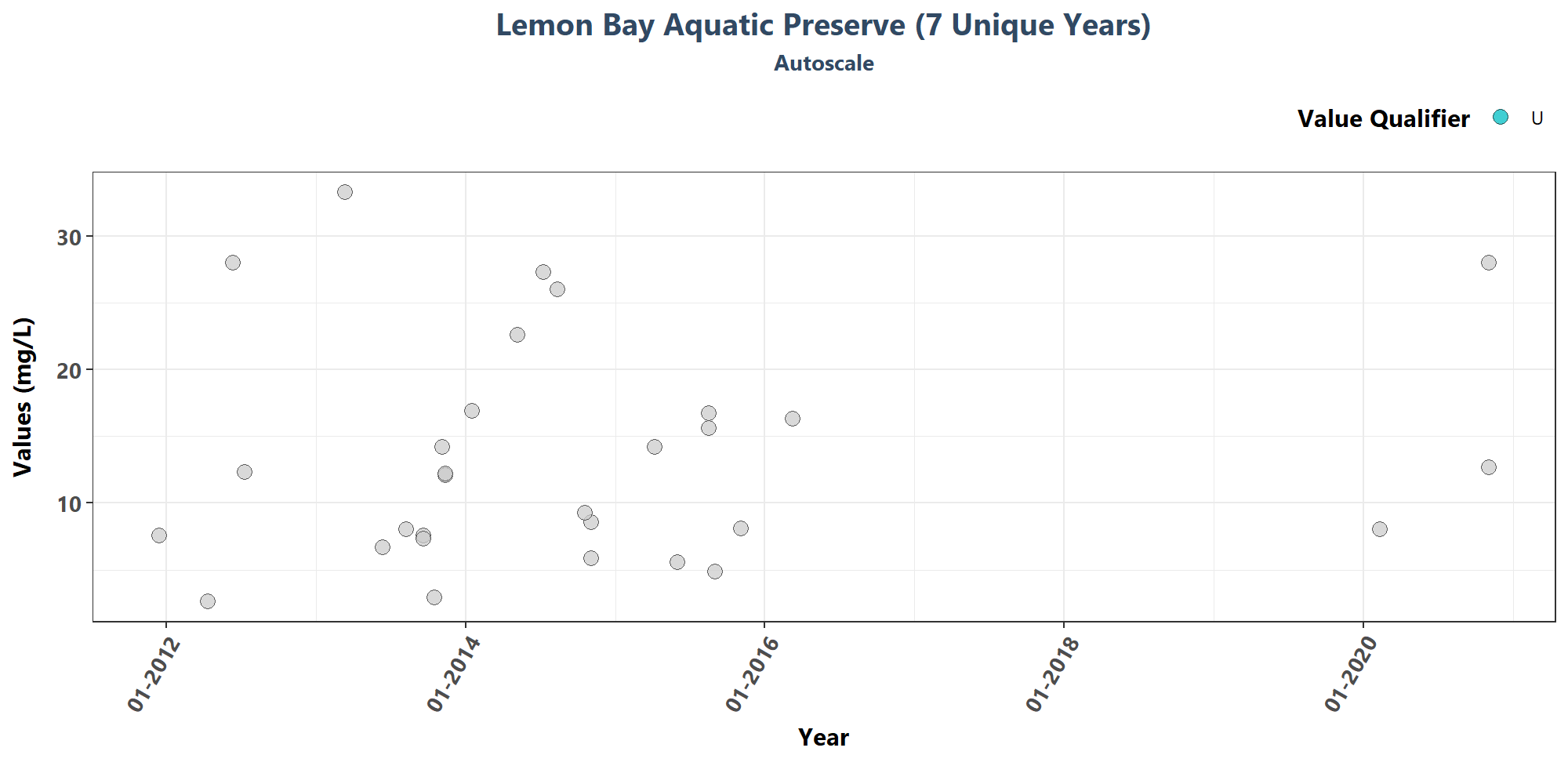
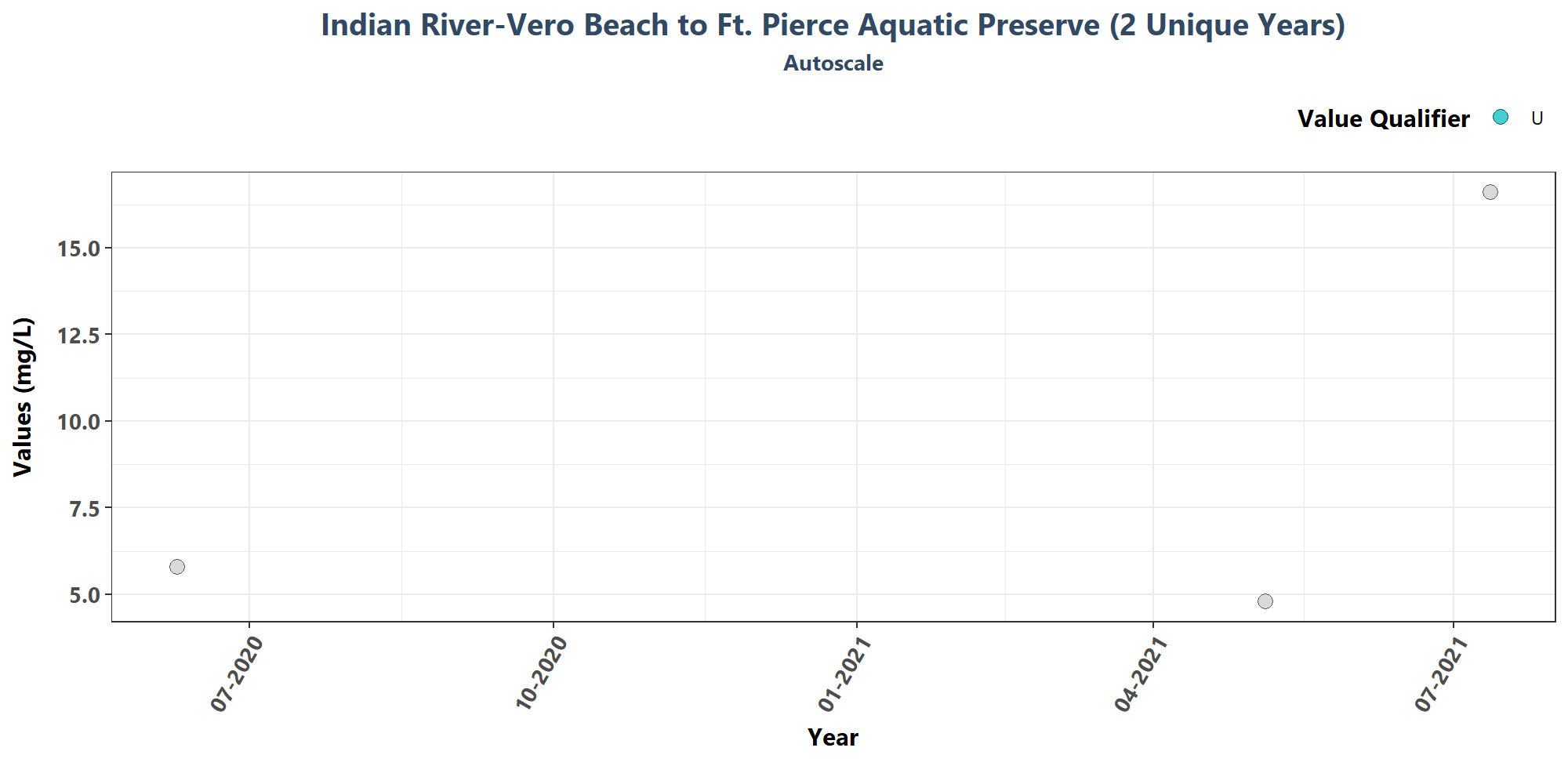
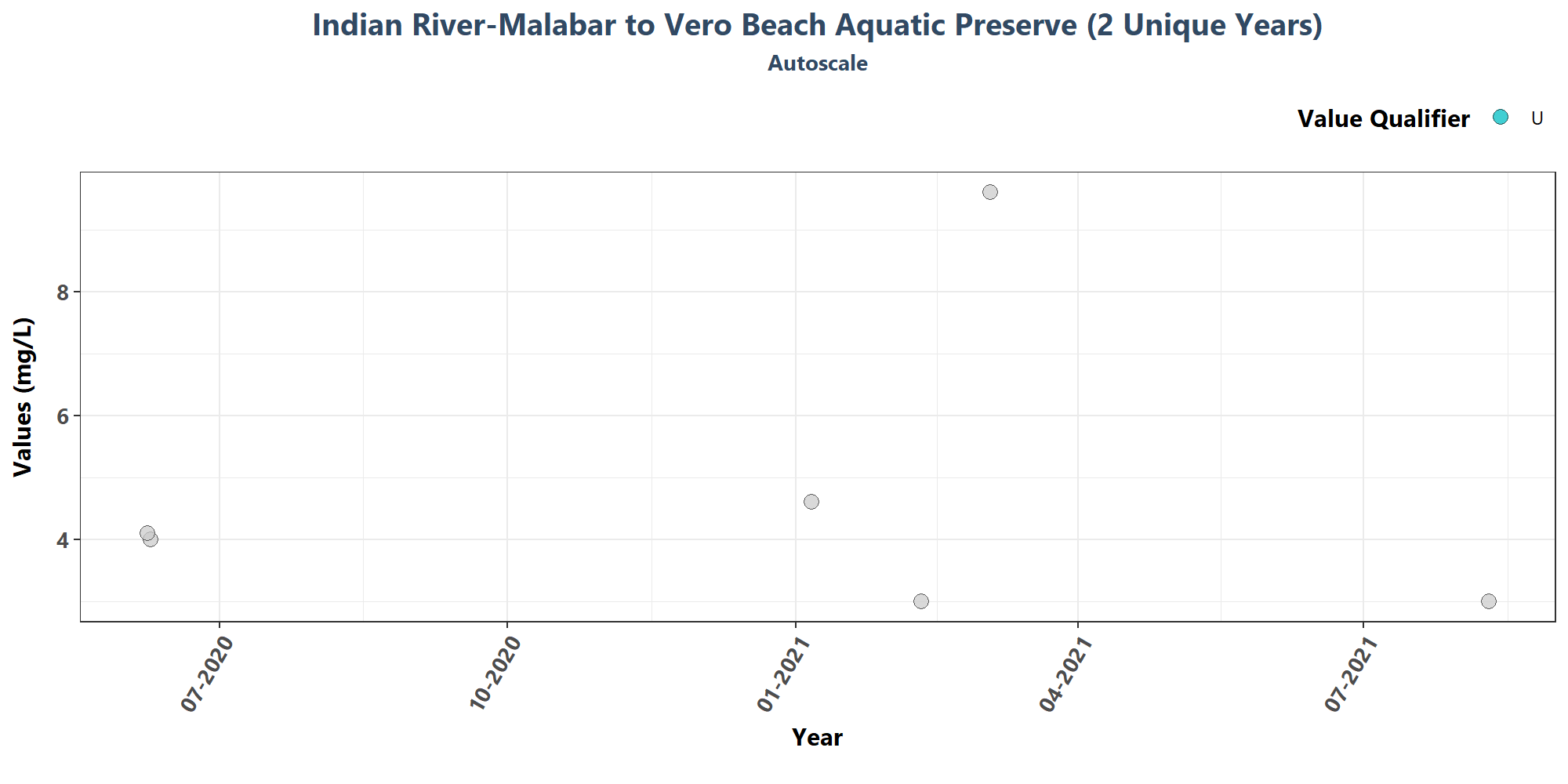
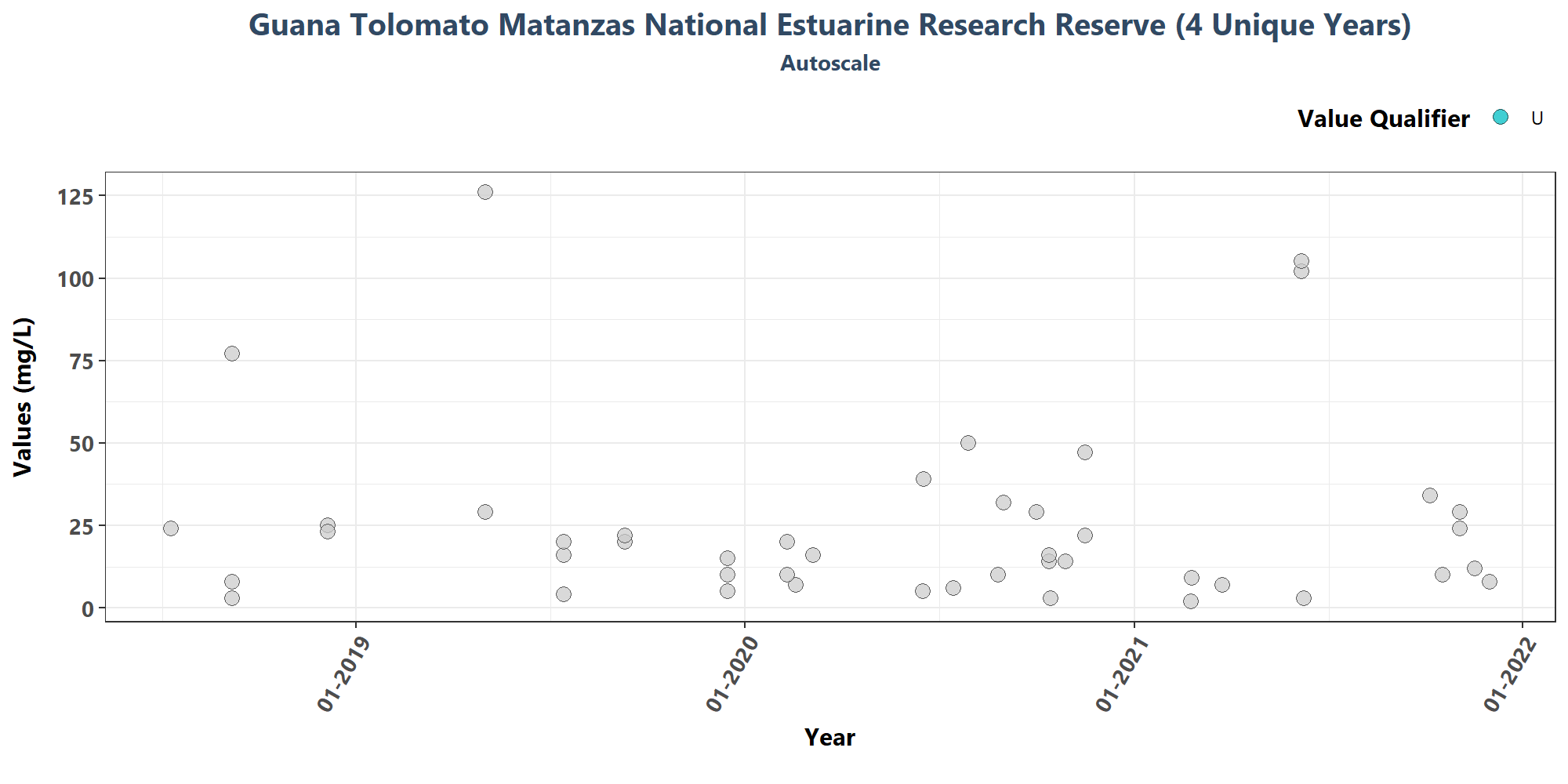
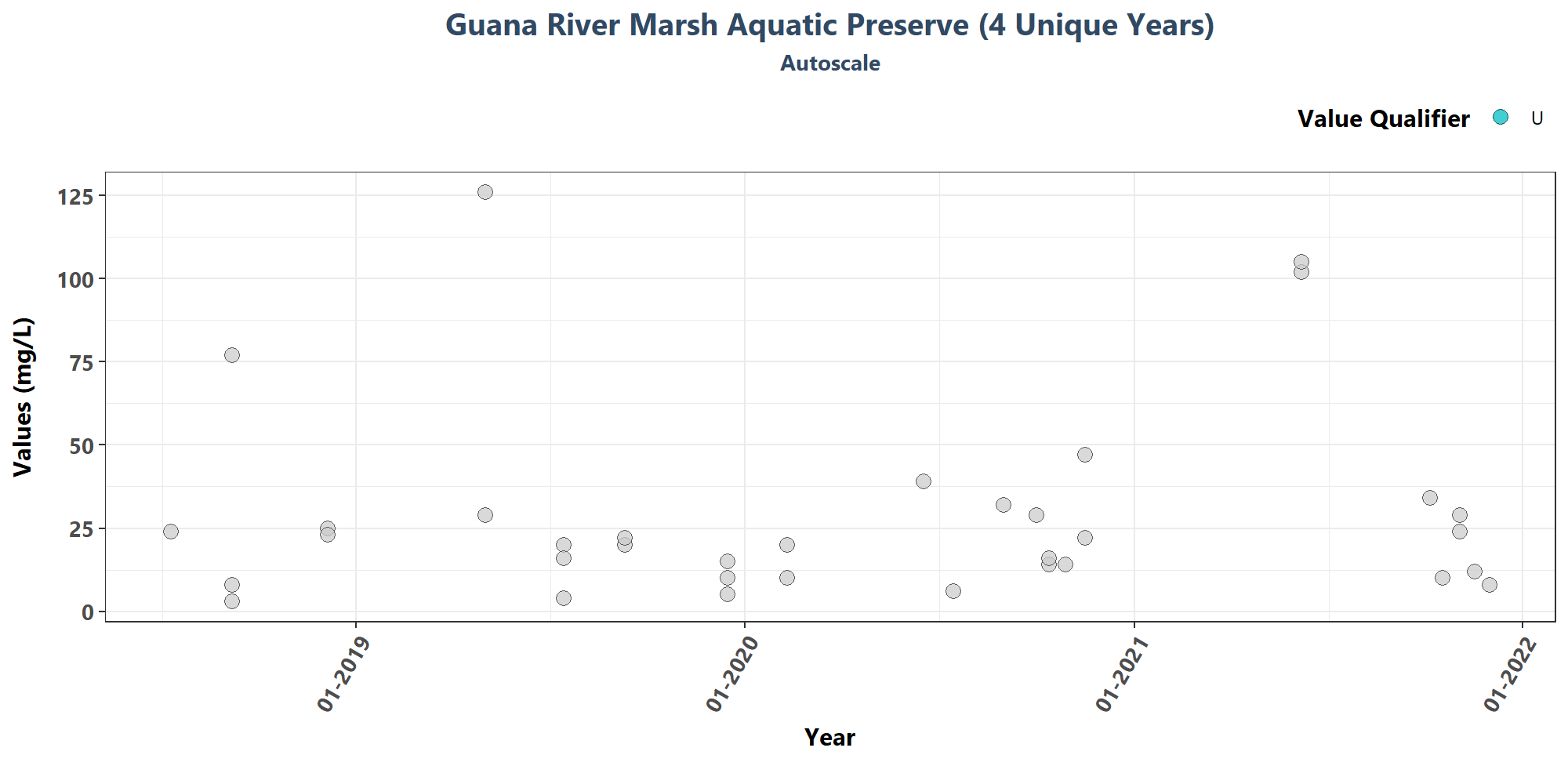
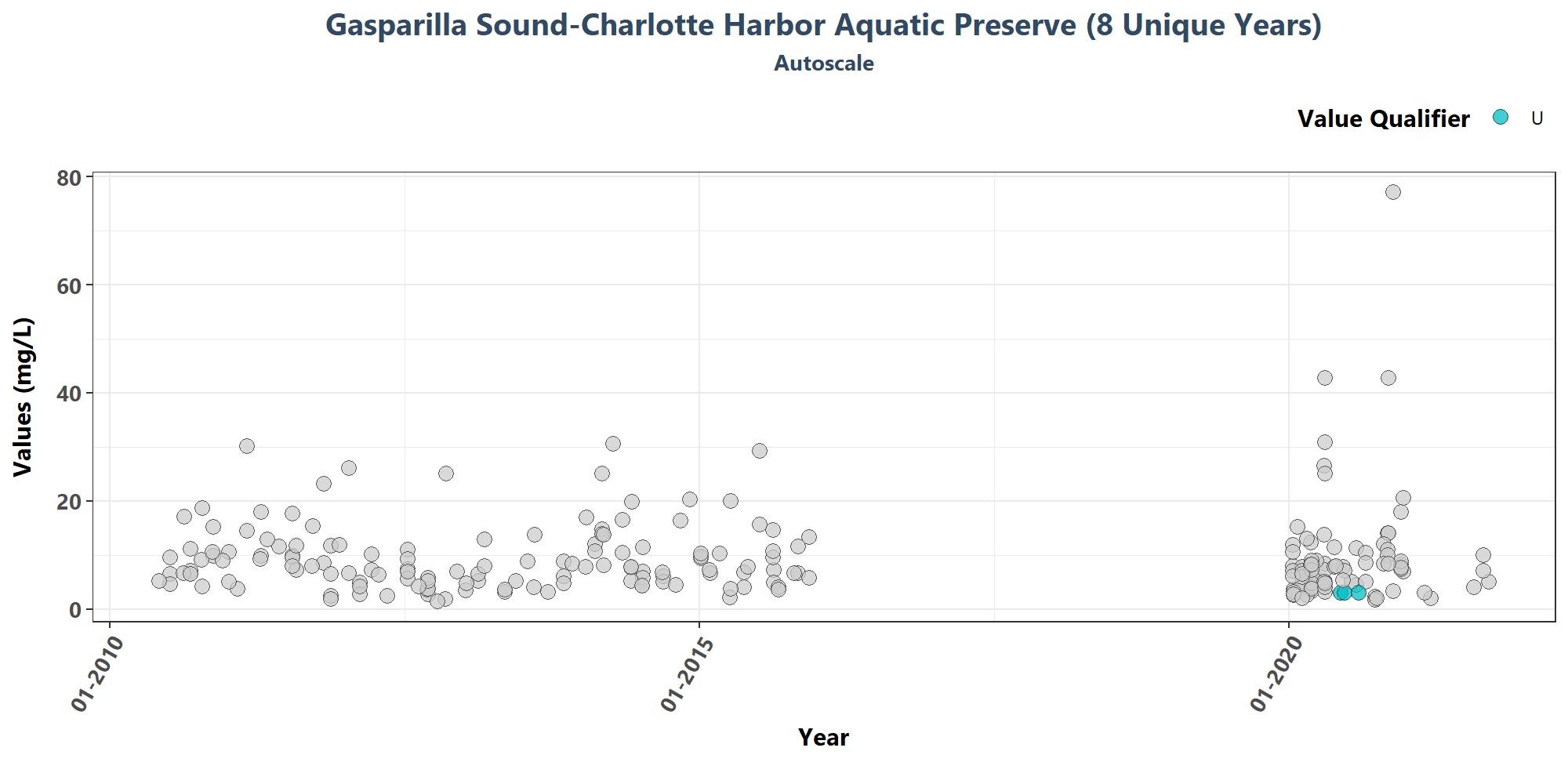
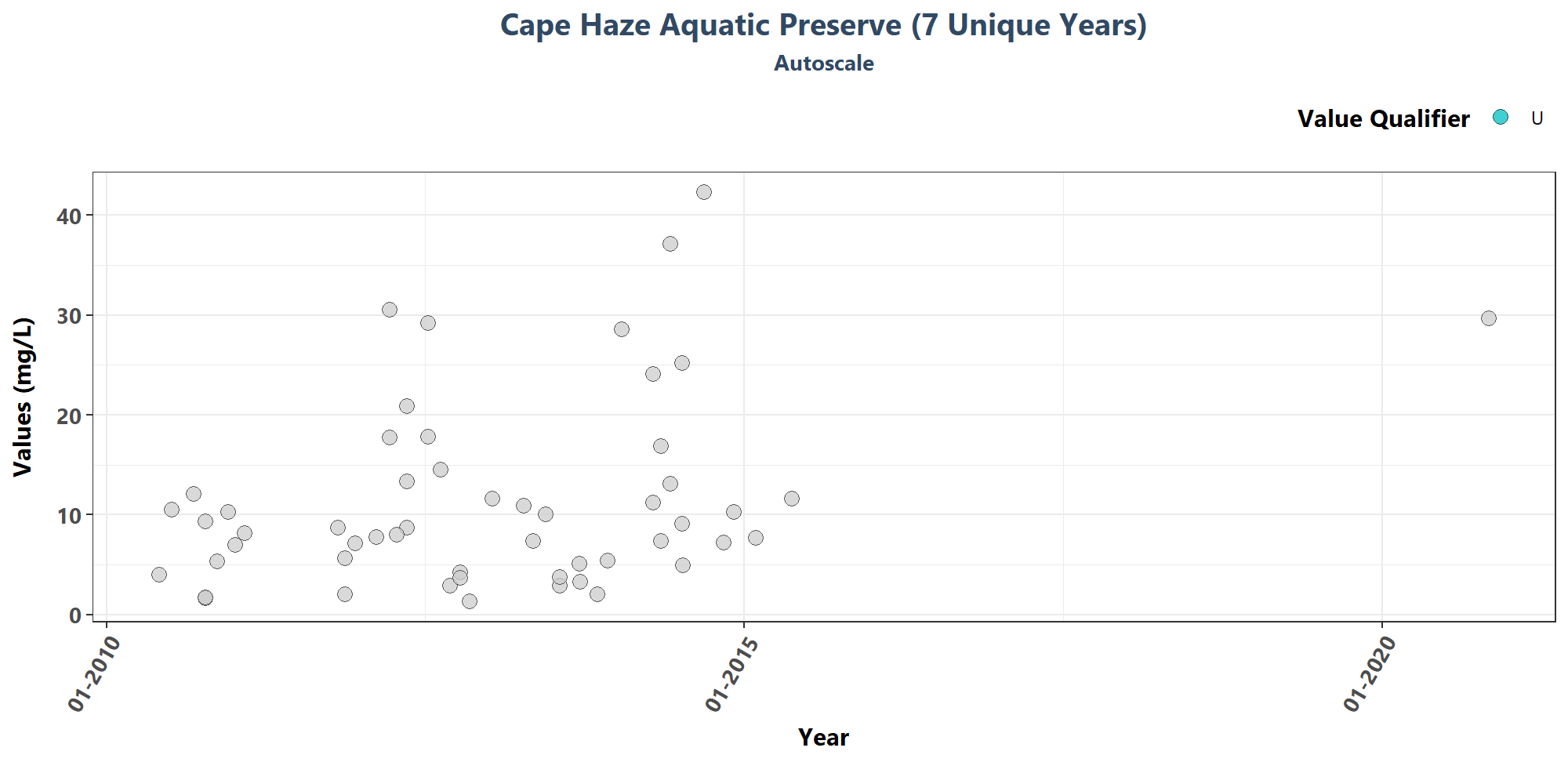
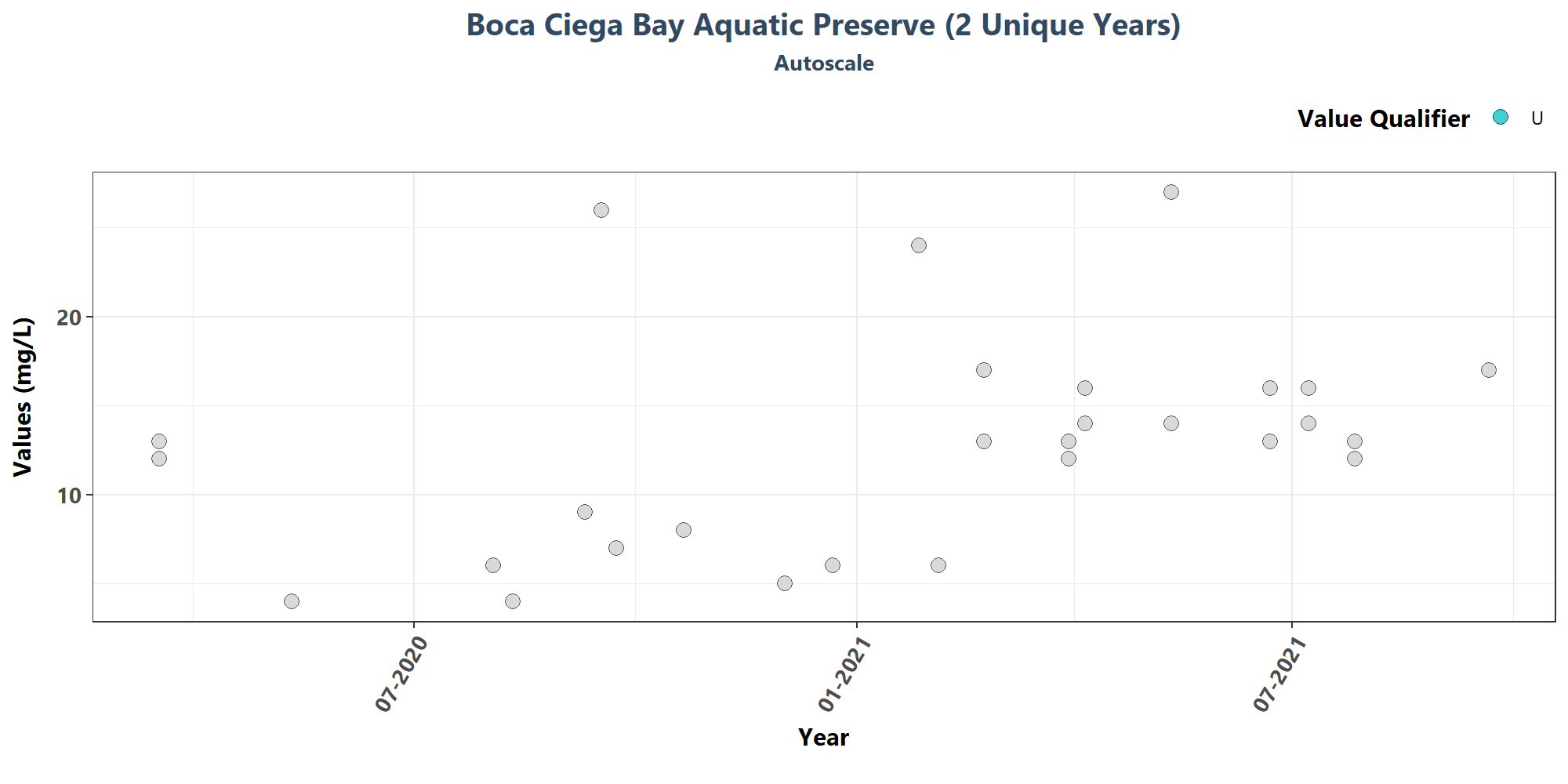
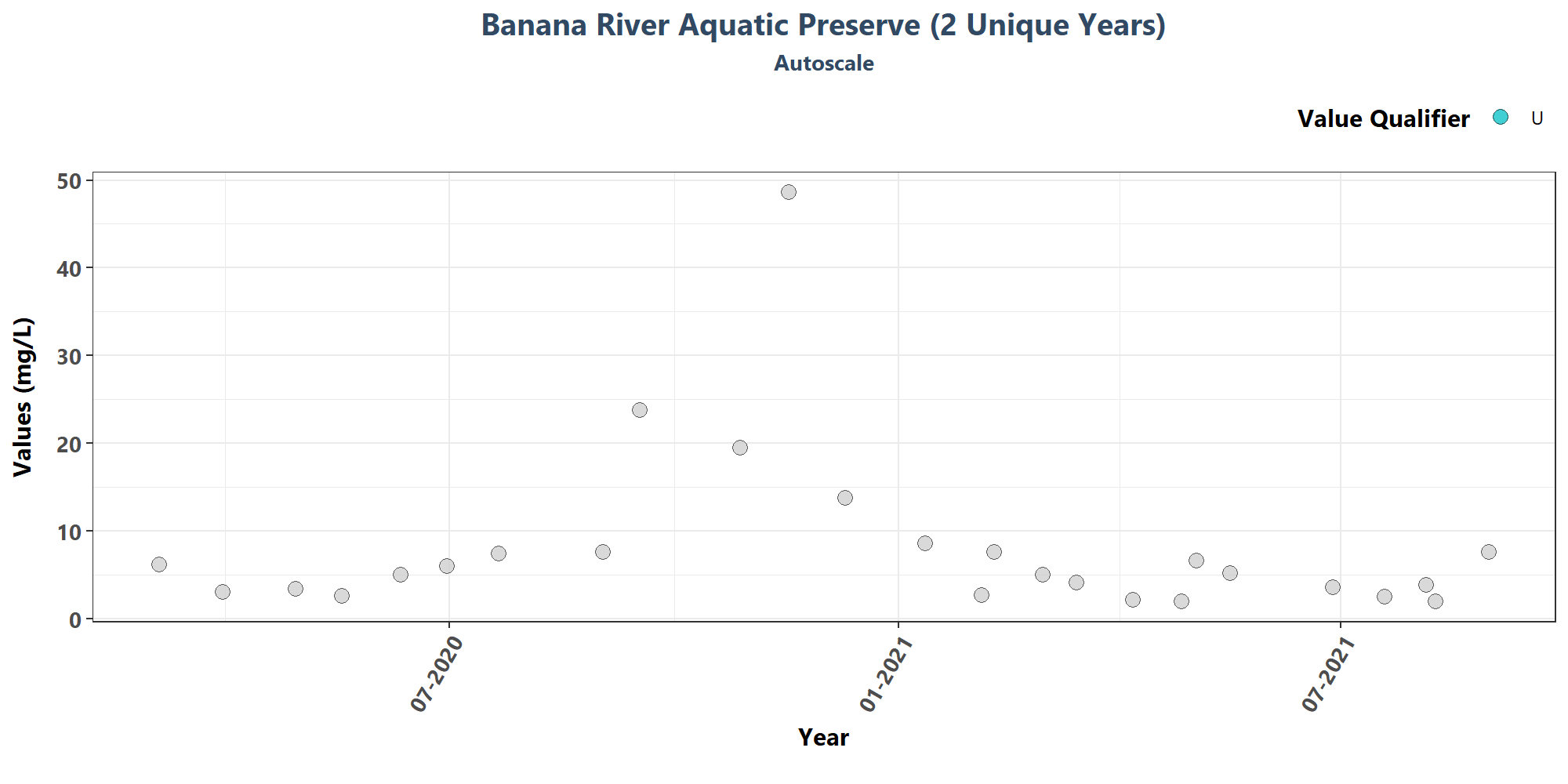
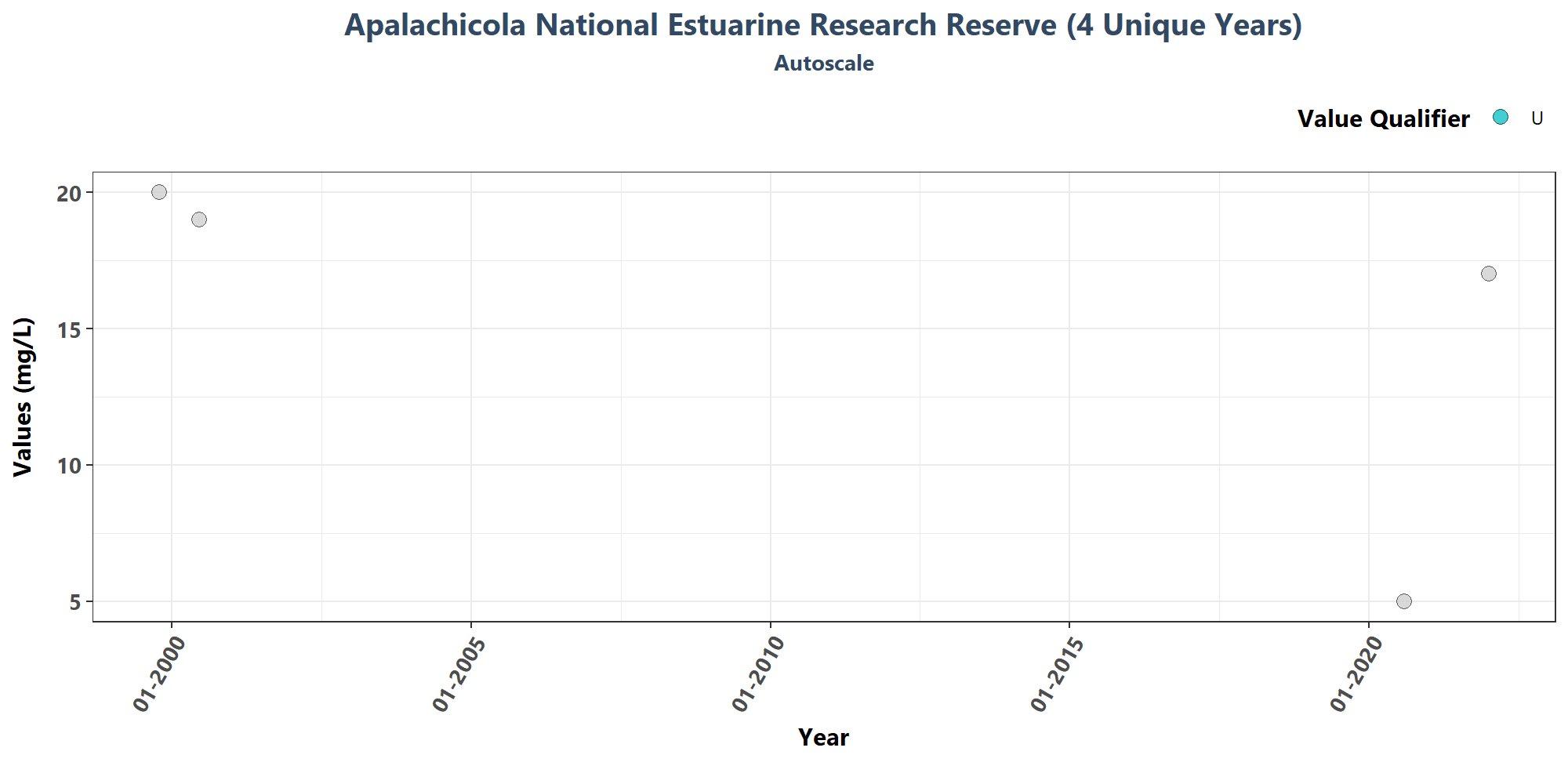
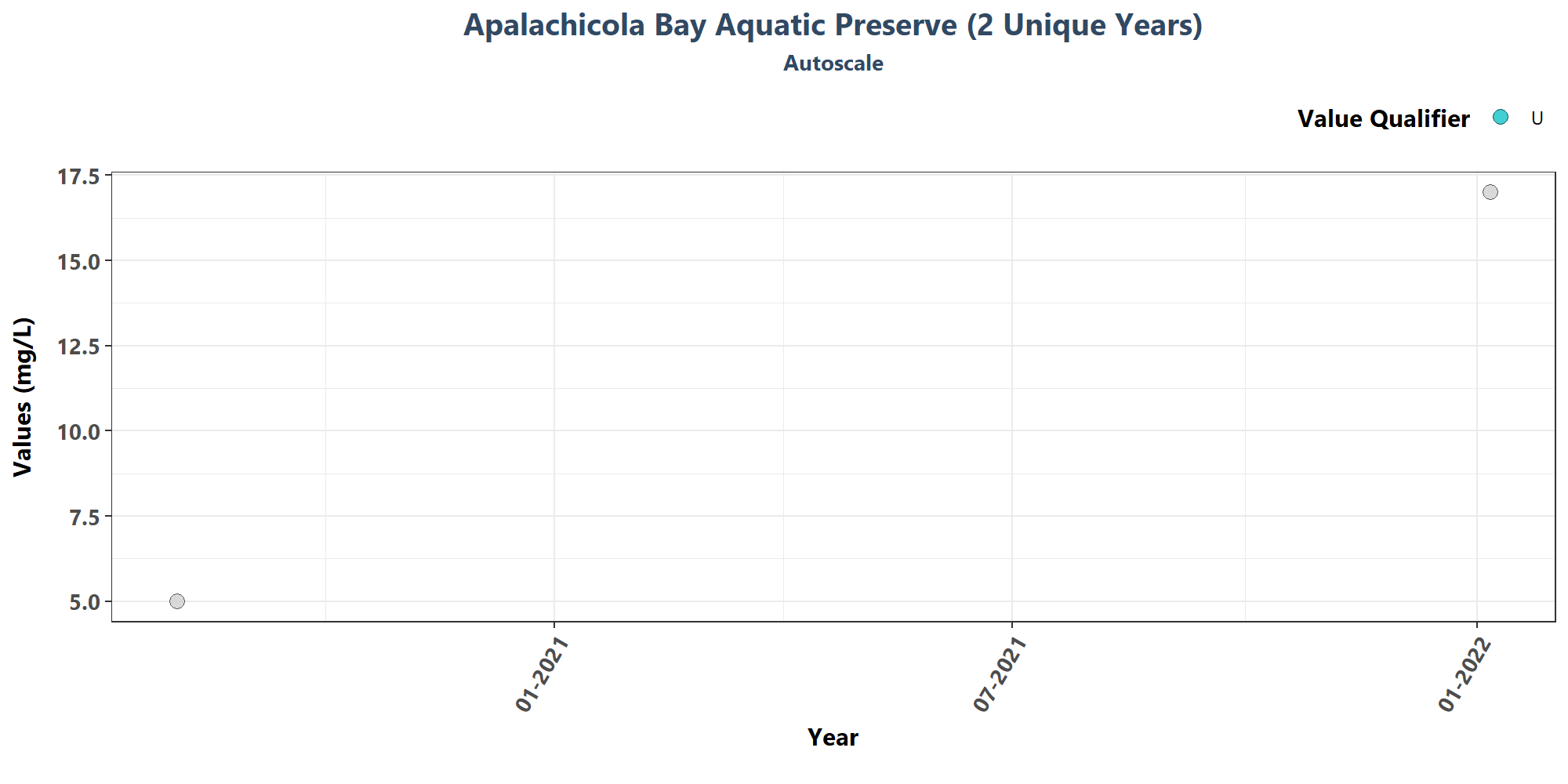
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Month",  
 y=paste0("Values (", 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))  
  
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")  
  
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")  
leg <- get\_legend(p1)  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
  
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())  
  
Mset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))



# Appendix III: 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.

if(z==0){  
 print("There are no managed areas that qualify.")  
} else {  
 for(i in 1:z){  
 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)  
 }  
}

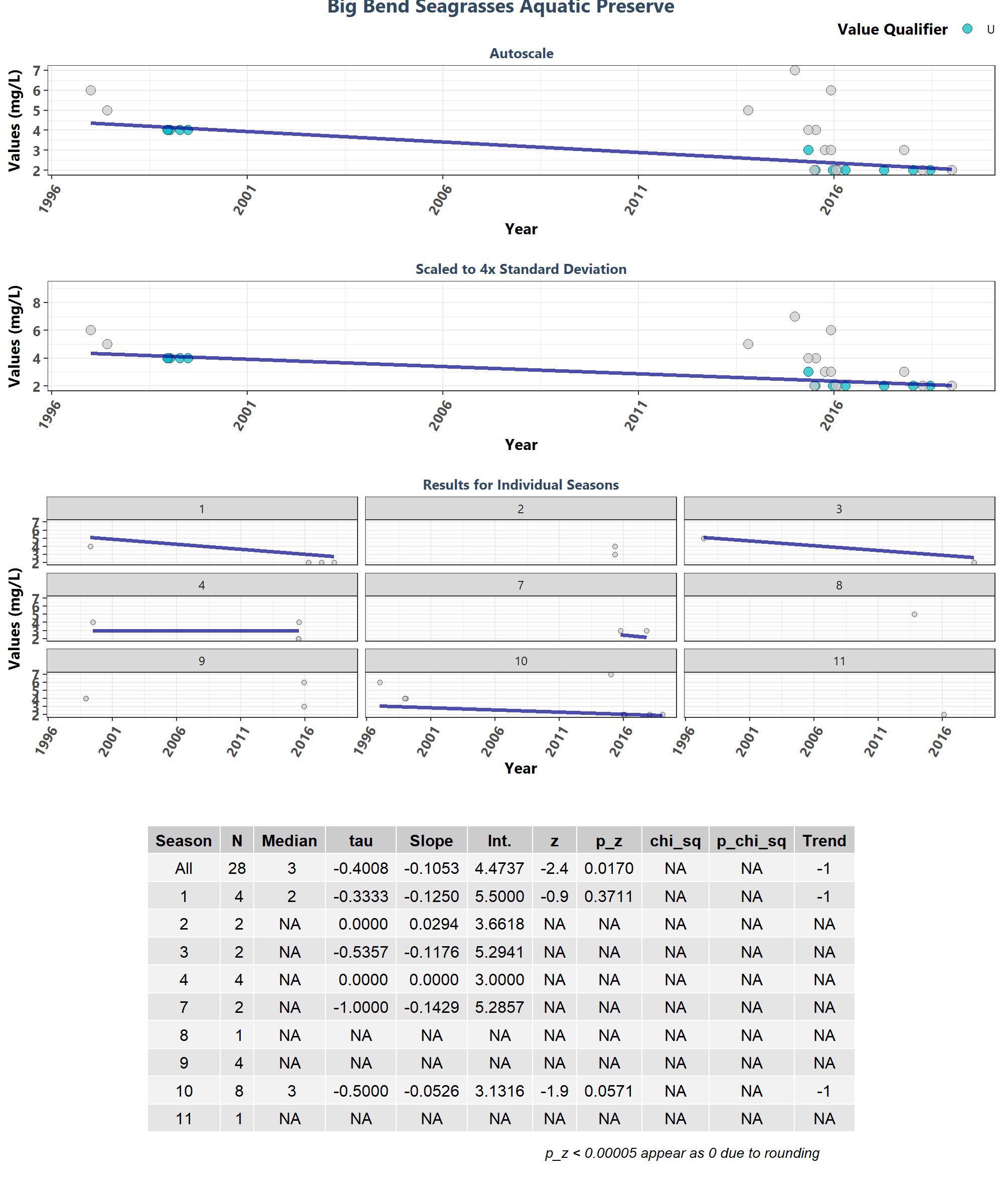


# Appendix IV: Managed Area Trendlines

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

if(n==0){  
 print("There are no managed areas that qualify.")  
} else {  
 for (i in 1:n) {  
 plot\_data <- data[data$SufficientData==TRUE &  
 data$ManagedAreaName==MA\_Include[i],]  
 plot\_data$Season <- factor(plot\_data$Month, levels=c("All", seq(1, 12)),  
 ordered=TRUE)  
 year\_lower <- min(plot\_data$relyear)  
 year\_upper <- max(plot\_data$relyear)  
 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)])  
 x\_scale <- ifelse(year\_upper - year\_lower > 30, 10, 5)  
 y\_scale <- mn\_RV + 4 \* sd\_RV  
   
 tau <- KT.Stats$tau[KT.Stats$ManagedAreaName==MA\_Include[i]]  
 s\_slope <- KT.Stats$SennSlope[KT.Stats$ManagedAreaName==MA\_Include[i]]  
 s\_int <- KT.Stats$SennIntercept[KT.Stats$ManagedAreaName==MA\_Include[i]]  
 trend <- KT.Stats$Trend[KT.Stats$ManagedAreaName==MA\_Include[i]]  
 z <- KT.Stats$z[KT.Stats$ManagedAreaName==MA\_Include[i]]  
 p\_z <- KT.Stats$p\_z[KT.Stats$ManagedAreaName==MA\_Include[i]]  
 chi\_sq <- KT.Stats$chi\_sq[KT.Stats$ManagedAreaName==MA\_Include[i]]  
 p\_chi\_sq <- KT.Stats$p\_chi\_sq[KT.Stats$ManagedAreaName==MA\_Include[i]]  
   
 # model <- lm(ResultValue ~ relyear\_dd,  
 # data=plot\_data)  
 # m\_int <- coef(model)[[1]]  
 # m\_slope <- coef(model)[[2]]  
 # rm(model)  
   
 xbrks <- seq(round\_any(min(plot\_data$relyear\_dd), 5, floor), round\_any(max(plot\_data$relyear\_dd), 5, ceiling),   
 by=(round\_any(max(plot\_data$relyear\_dd), 5, ceiling) - round\_any(min(plot\_data$relyear\_dd), 5, floor))/5)  
   
 xlabs <- seq(max(plot\_data$Year) - round\_any(max(plot\_data$relyear\_dd), 5,  
 ceiling),  
 max(plot\_data$Year),   
 by=(max(plot\_data$Year) - (max(plot\_data$Year) - round\_any(max(plot\_data$relyear\_dd), 5, ceiling)))/5)  
   
 KT.Stats[, season := Season]  
 KT.Stats[ManagedAreaName==MA\_Include[i] & season != "All", `:=`  
 (N\_Data=nrow(plot\_data[Season==season &  
 ManagedAreaName==MA\_Include[i], ]),  
 relyear\_dd\_lower=min(plot\_data[Season==season &  
 ManagedAreaName==  
 MA\_Include[i],  
 relyear\_dd]),  
 relyear\_dd\_upper=max(plot\_data[Season==season &  
 ManagedAreaName==  
 MA\_Include[i],  
 relyear\_dd])), by="season"]  
 KT.Stats[ManagedAreaName==MA\_Include[i] & season=="All", `:=`  
 (relyear\_dd\_lower=min(plot\_data[ManagedAreaName==MA\_Include[i],  
 relyear\_dd]),  
 relyear\_dd\_upper=max(plot\_data[  
 ManagedAreaName==MA\_Include[i], relyear\_dd])), by="season"]  
 KT.Stats[, season := NULL]  
   
 # plot\_data[is.na(VQ\_Plot), VQ\_Plot := "None"]  
 p1 <- ggplot(data=plot\_data,  
 aes(x=relyear\_dd, y=ResultValue, fill=VQ\_Plot)) +  
 geom\_point(shape=21, size=3, color="#333333", alpha=0.75) +  
 # geom\_abline(aes(slope=s\_slope, intercept=s\_int),  
 # color="#000099", size=1.2, alpha=0.7) +  
 geom\_segment(data=KT.Stats[ManagedAreaName==MA\_Include[i] &  
 Season=="All", ],  
 aes(x=relyear\_dd\_lower, y=relyear\_dd\_lower \* SennSlope +  
 SennIntercept,  
 xend=relyear\_dd\_upper,  
 yend=relyear\_dd\_upper \* SennSlope + SennIntercept),  
 color="#000099", size=1.2, alpha=0.7, inherit.aes=FALSE) +  
 labs(subtitle="Autoscale",  
 x="Year", y=paste0("Values (", unit, ")"),  
 fill="Value Qualifier") +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal",  
 legend.justification="right") +  
 {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")  
 }} +  
 scale\_x\_continuous(breaks=xbrks,  
 labels=xlabs)  
   
 p2 <- ggplot(data=plot\_data,  
 aes(x=relyear\_dd, y=ResultValue, fill=VQ\_Plot)) +  
 geom\_point(shape=21, size=3, color="#333333", alpha=0.75) +  
 # geom\_abline(aes(slope=s\_slope, intercept=s\_int),  
 # color="#000099", size=1.2, alpha=0.7) +  
 geom\_segment(data=KT.Stats[ManagedAreaName==MA\_Include[i] &  
 Season=="All", ],  
 aes(x=relyear\_dd\_lower, y=relyear\_dd\_lower \* SennSlope +  
 SennIntercept, xend=relyear\_dd\_upper,  
 yend=relyear\_dd\_upper \* SennSlope + SennIntercept),  
 color="#000099", size=1.2, alpha=0.7, inherit.aes=FALSE) +  
 ylim(min\_RV, y\_scale) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 plot\_theme +  
 theme(legend.position="none") +  
 {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")  
 }} +  
 scale\_x\_continuous(breaks=xbrks,  
 labels=xlabs)  
   
 splot <- ggplot(plot\_data, aes(x=relyear\_dd, y=ResultValue)) +  
 geom\_point(shape=21, size=1.5, color="#333333", fill="#cccccc",  
 alpha=0.75) +  
 geom\_segment(data=KT.Stats[ManagedAreaName==MA\_Include[i] &  
 Season != "All", ],  
 aes(x=relyear\_dd\_lower, y=relyear\_dd\_lower \* SennSlope +  
 SennIntercept, xend=relyear\_dd\_upper,  
 yend=relyear\_dd\_upper \* SennSlope + SennIntercept),  
 color="#000099", size=1.2, alpha=0.7) +  
 #ylim(min\_RV-0.1\*y\_scale, y\_scale) +  
 scale\_x\_continuous(breaks=xbrks,  
 labels=xlabs) +  
 labs(y=paste0("Values (", unit, ")"), x="Year",  
 subtitle="Results for Individual Seasons") +  
 facet\_wrap(~Season, ncol=3) +  
 plot\_theme  
   
 leg <- get\_legend(p1)  
 KTset <- ggarrange(leg, p1 + theme(legend.position="none"), p2,  
 splot, ncol=1, heights=c(0.1, 1, 1, 1.5))  
   
 p0 <- ggplot() + labs(title=paste0(MA\_Include[i])) +  
 plot\_theme + theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(),  
 axis.line=element\_blank())  
   
 KT.Stats[ManagedAreaName==MA\_Include[i], `:=` (N=N\_Data,  
 Median=round(Median, 2),  
 Slope=round(SennSlope, 4),  
 Int.=round(SennIntercept, 4),  
 z=round(z, 1),  
 chi\_sq=round(chi\_sq, 1))]  
   
 #print(ggarrange(p0, KTset, ncol=1, heights=c(0.1, 1.25)))  
 ResultTable <- KT.Stats[KT.Stats$ManagedAreaName==MA\_Include[i], ] %>%  
 select(Season, N, Median, tau, Slope, Int., z, p\_z, chi\_sq, p\_chi\_sq, Trend)  
   
 t1 <- ggtexttable(ResultTable, rows=NULL, theme=ttheme(base\_size=11.5)) %>%  
 tab\_add\_footnote(text="p\_z < 0.00005 appear as 0 due to rounding",  
 size=10, face="italic")  
   
 print(ggarrange(p0, KTset, t1, ncol=1, heights=c(0.015, 0.645, 0.34)))  
 cat('\n \n \n')  
 rm(plot\_data)  
 rm(KTset, leg)  
 }  
}



# Appendix V: Managed Area Summary Box Plots

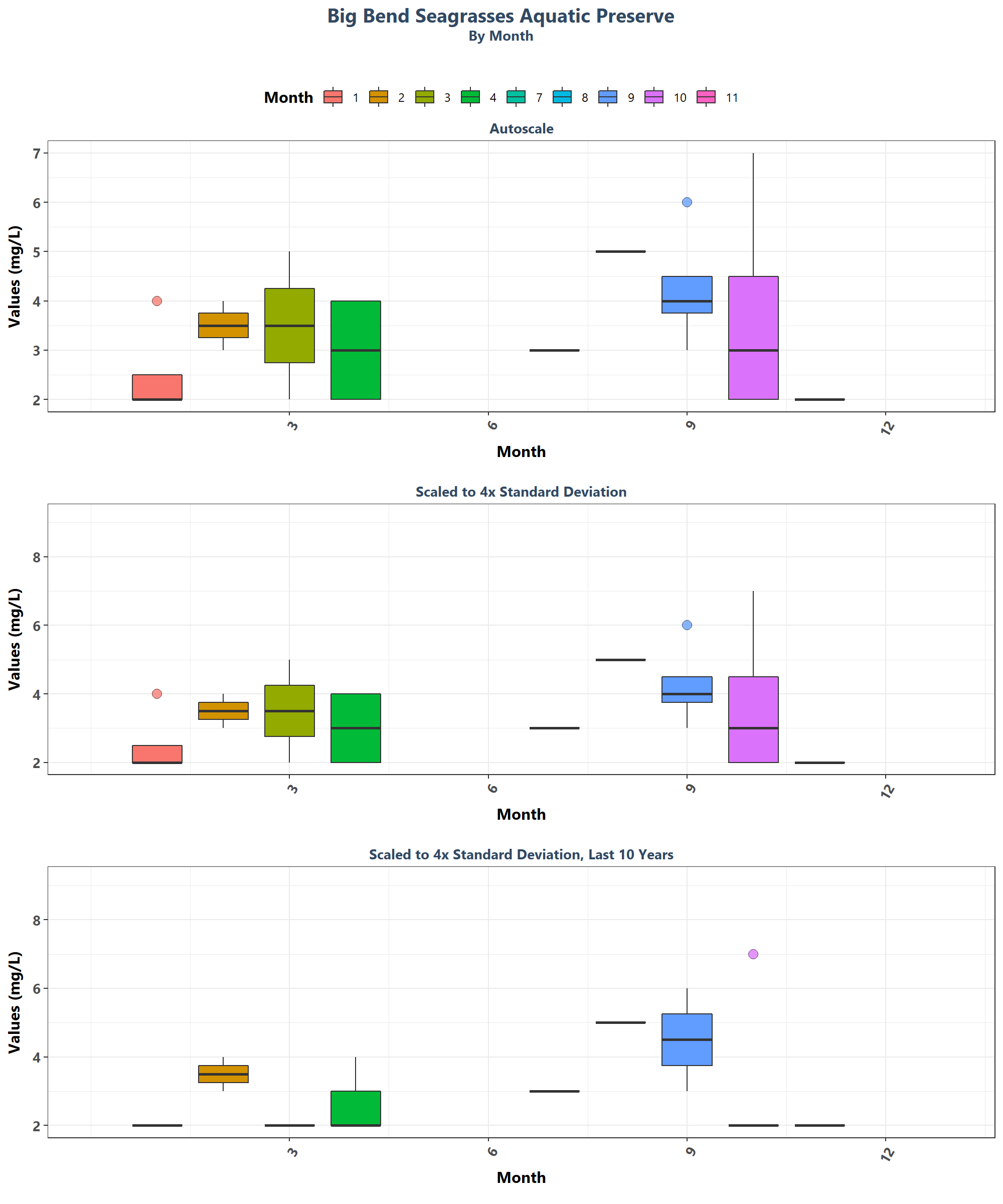
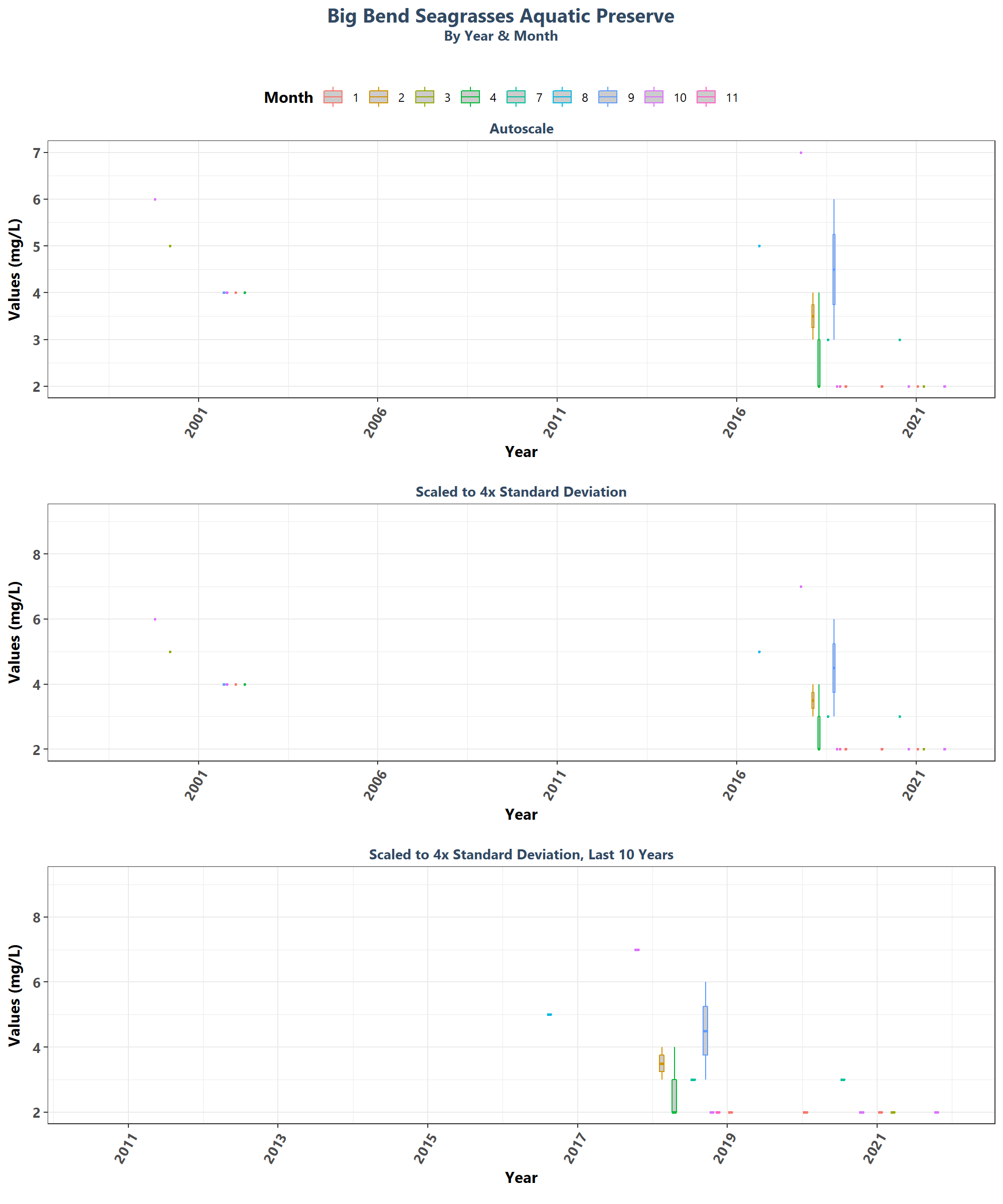
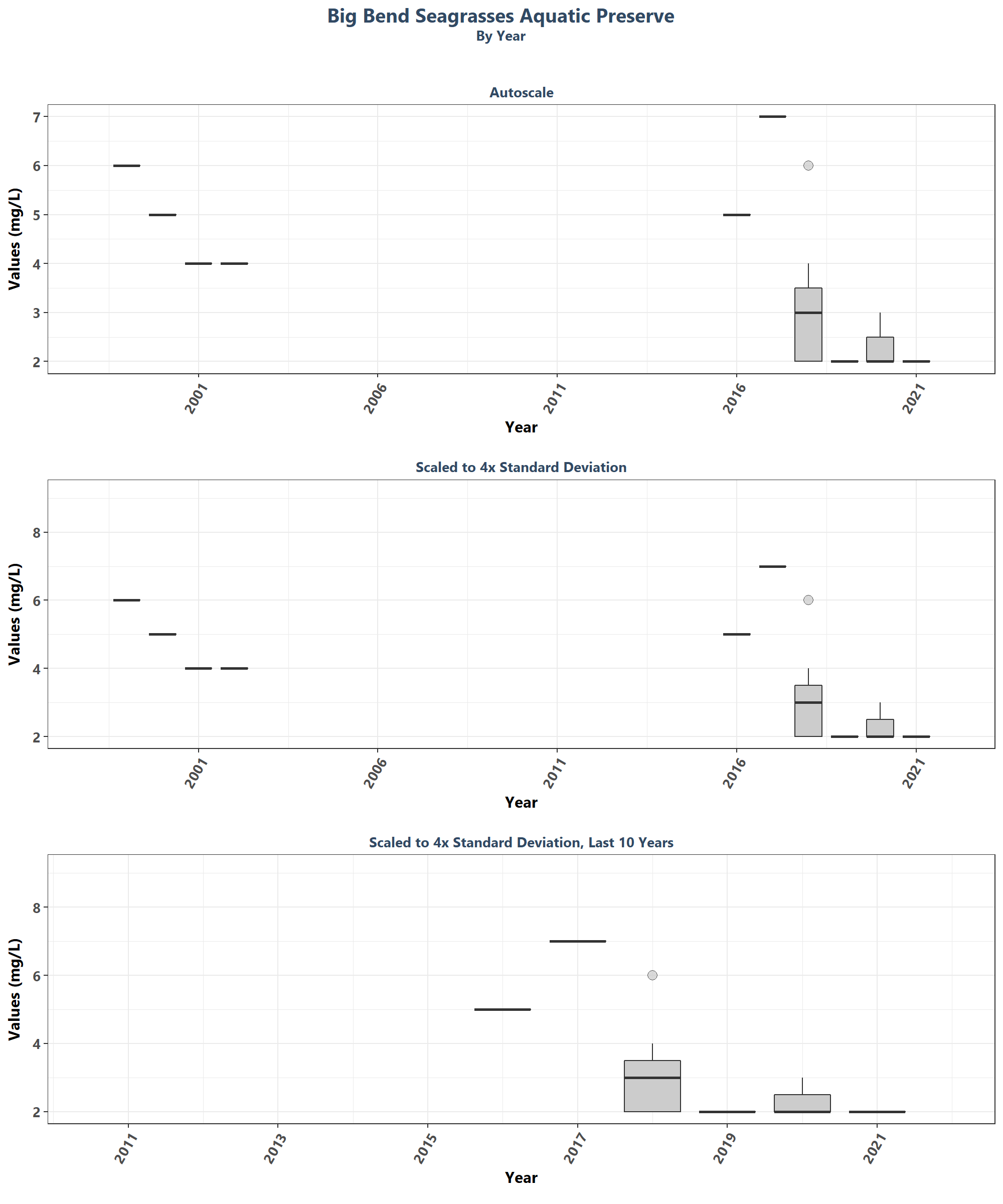
Data is taken and grouped by ManagedAreaName. The scripts that create plots follow this format

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

if(n==0){  
 print("There are no managed areas that qualify.")  
} else {  
 for (i in 1:n) {  
 plot\_data <- data[data$SufficientData==TRUE &  
 data$ManagedAreaName==MA\_Include[i],]  
 year\_lower <- min(plot\_data$Year)  
 year\_upper <- max(plot\_data$Year)  
 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)])  
 x\_scale <- ifelse(year\_upper - year\_lower > 30, 10, 5)  
 y\_scale <- mn\_RV + 4 \* sd\_RV  
  
 ##Year plots  
 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  
  
 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  
  
 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  
  
 Yset <- ggarrange(p1, p2, p3, ncol=1)  
  
 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  
 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")  
  
 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))  
  
 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")  
  
 leg1 <- get\_legend(p5)  
 YMset <- ggarrange(leg1, p4, p5 + theme(legend.position="none"), p6,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
  
 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  
 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")  
  
 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))  
  
 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")  
  
 leg2 <- get\_legend(p8)  
 Mset <- ggarrange(leg2, p7, p8 + theme(legend.position="none"), p9,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
  
 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())  
  
 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)  
 }  
}



rm(list=setdiff(ls(), c("all\_params", "all\_depths", "all\_activity",  
 "param\_name", "depth", "activity", "KT.Stats\_all",  
 "MA\_All", "APP\_Plots", "out\_dir", "file\_in", "file\_out")))