SEACAR Continuous Water Quality Analysis: SE Region for Turbidity

Last compiled on 08 June, 2023

Contents

Important Notes

These scripts were created by J.E. Panzik (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_Continuous_ReportRender.R (https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/WQ_Continuous/SEACAR_WC_Continuous_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)
library(formatR)
options(scipen=999)
knitr::opts_chunk$set(
   warning=FALSE,
   message=FALSE,
   dpi=200
)
```

File Import

Imports file that is determined in the SEACAR_WC_Continuous_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 Continuous data is available at: https://usf.box.com/s/7ocbmdsm5bgfz6535t8btrnj3r73ysch

The file being used for the analysis is: **Combined_WQ_WC_NUT_cont_Turbidity_SE-2023-Jun-07.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="")
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_cont_Turbidity_SE-2023-Jun-07.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

The filtering that is performed by the script at this point removes rows that are missing values for `ResultValue` and `RelativeDepth`, and removes any activity type that has "Blank" in the description. Data passes the filtering the process if it is has an `Include` value of 1.

Creates a variable for each `MonitoringID` which is defined as a unique combination of `ManagedAreaName`, `ProgramID`, `ProgramAreaName`, and `ProgramLocationID`.

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 5 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 monitoring location has at least 2 consecutive years of observations with at least 2 repeating months takes the data, creates a list of the monitoring IDs and cycles through each monitoring ID. For each monitoring ID 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 location passes the criteria and is stored in a variable
7. The list of IDs 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 monitoring location. 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.

A variable with the monitoring IDs that pass all criteria is created and stored.

```r
# Converts Include to be a logical either TRUE or FALSE
data$Include <- as.logical(data$Include)
# Removes any data rows that do not have Include set to TRUE
data <- data[data$Include==TRUE,]
# Removes rows that have missing ResultValues
data <- data[!is.na(data$ResultValue),]
# Removes rows that have missing RelativeDepth
```

```r
data <- data[!is.na(data$RelativeDepth),]
# Rremoves rows that have an ActivityType with Blank
data <- data[!grep("Blank", data$ActivityType),]

# Removes any data below threshold value of 0, or 5 for Water Temperature
if(param_name=="Water_Temperature"){
   data <- data[data$ResultValue>=-5,]
} else{
   data <- data[data$ResultValue>=0,]
}

# Gets list of managed areas for the specific region being looked at
MA_All_Region <- MA_All[MA_All$Region==region,]

# Gets AreaID for data by merging data with the managed area list for the region
data <- merge.data.frame(MA_All_Region[,c("AreaID", "ManagedAreaName")],
                          data, by="ManagedAreaName", all=TRUE)
# Creates MonitoringID to more easily cycle through monitoring locations
data <- data %>%
    group_by(AreaID, ManagedAreaName, ProgramID, ProgramName,
             ProgramLocationID) %>%
    mutate(MonitoringID=cur_group_id())

# Creates function to checks monitoring location for at least 2 years of
# continuous consecutive data
ContinuousConsecutiveCheck <- function(con_data){
   # Gets MonitoringIDs
   IDs <- unique(con_data$MonitoringID[con_data$Include==TRUE &
                                          !is.na(con_data$Include)])
   # Loops through each MonitoringID
   for(i in 1:length(IDs)) {
      # Gets list of Years for MonitoringID
      Years <- unique(con_data$Year[con_data$MonitoringID==IDs[i] &
                                      con_data$Include==TRUE &
                                      !is.na(con_data$Include)])
      # Puts Years in order
      Years <- Years[order(Years)]
      # If there are fewer than 2 years, skip to next MonitoringID
      if(length(Years)<2) {
         next
      }
      # Starts loop to make sure there are at least 2 consecutive years with
      # consecutive months of data
      for(j in 2:length(Years)) {
         # If adjacent year entries are not 1 year apart, skip to the next set
         # of year entries
         if(Years[j]-Years[j-1]!=1) {
            next
         }
         # Gets the list of months from the first year
         Months1 <- unique(con_data$Month[con_data$MonitoringID==IDs[i] &
                                            con_data$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$Month[con_data$MonitoringID==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
         # MonitoringID passes the check and is stored
         if(length(intersect(Months1, Months2))>=2) {
            # Creates variable for stored MonitoringID if it doesn't exist
            if(exists("consecutive")==FALSE){
               consecutive <- IDs[i]
               break
            } else{
               # Adds to variable for storing MonitoringID if does exist
               consecutive <- append(consecutive, IDs[i])
               break
            }
         }
      }
   }
   # After going through all MonitoringID, return variable with list of all
   # that pass
   return(consecutive)
}

# Stores the MonitoringID that pass the consecutive year check
consMonthIDs <- ContinuousConsecutiveCheck(data)

# Creates data frame with summary for each monitoring location.
Mon_Summ <- data %>%
   group_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,
            ProgramLocationID) %>%
   summarize(ParameterName=parameter,
             RelativeDepth=unique(RelativeDepth),
             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]),
             LastSampleDate=max(SampleDate[Include==TRUE]),
             ConsecutiveMonths=ifelse(unique(MonitoringID) %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, na.rm=TRUE))
Mon_Summ$ConsecutiveMonths <- NULL

# Puts summary data in order based on MonitoringID
Mon_Summ <- as.data.table(Mon_Summ[order(Mon_Summ$MonitoringID), ])
```

```
# Creates column in data that determines how many years from the start for each
# Monitoring location
data <- data %>%
   group_by(MonitoringID) %>%
   mutate(YearFromStart=Year-min(Year))
# Adds SufficientData column to data table based on MonitoringID
data <- merge.data.frame(data, Mon_Summ[,c("MonitoringID", "SufficientData")],
                         by="MonitoringID")
# 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$SufficientData==TRUE,
                               TRUE, FALSE)
# Get list of and number of MonitoringID that are to be used in analysis
Mon_IDs <- unique(data$MonitoringID[data$Use_In_Analysis==TRUE])
Mon_IDs <- Mon_IDs[order(Mon_IDs)]
n <- length(Mon_IDs)
```

Monitoring Location Statistics

Gets summary statistics for each monitoring location. Excluded monitoring locations 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 `Use_In_Analysis` value of TRUE
2. Group data that have the same `ManagedAreaName`, `ProgramID`, `ProgramName`, `ProgramLocationID`, `Year`, and `Month`.
   - Second summary statistics consider the monitoring location grouping and `Year`.
   - Third summary statistics consider the monitoring location grouping and `Month`.
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, 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`, `ProgramID`, `ProgramName`, `ProgramLocationID`, `Year`, and `Month` in that order.
5. Write summary stats to a pipe-delimited .txt file in the output directory
   - WC Continuous Output Files in SEACAR GitHub (https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Continuous/output)

Because the continuous data is extensive and most measurements are taken every 15 minutes, a daily average is determined and used based on grouping `ManagedAreaName`, `ProgramID`, `ProgramName`, `ProgramLocationID`, and `SampleDate`. The new `ResultValue` is the mean of all values on that date from that specific monitoring location. Sets the `SampleDate` as a date object, and creates various scales of the date to be used by plotting functions.

```
# Create summary statistics for each monitoring location based on Year and Month
# intervals.
Mon_YM_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
   group_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,
            ProgramLocationID, Year, Month) %>%
   summarize(ParameterName=parameter,
             RelativeDepth=unique(RelativeDepth),
             EarliestSampleDate=min(SampleDate),
             LastSampleDate=max(SampleDate), N=length(ResultValue),
             Min=min(ResultValue), Max=max(ResultValue),
             Median=median(ResultValue), Mean=mean(ResultValue),
             StandardDeviation=sd(ResultValue))
# Puts the data in order based on ManagedAreaName, ProgramID, ProgramName,
# ProgramLocationID, Year, then Month
Mon_YM_Stats <- as.data.table(Mon_YM_Stats[order(Mon_YM_Stats$ManagedAreaName,
                                                 Mon_YM_Stats$ProgramID,
                                                 Mon_YM_Stats$ProgramName,
                                                 Mon_YM_Stats$ProgramLocationID,
                                                 Mon_YM_Stats$Year,
                                                 Mon_YM_Stats$Month), ])
# Writes summary statistics to file without MonitoringID
fwrite(select(Mon_YM_Stats, -MonitoringID),
       paste0(out_dir_param,"/WC_Continuous_", param_abrev, "_", region,
              "_MonLoc_MMYY_Stats.txt"), sep="|")
# Get year from start for each monitoring location
Mon_YM_Stats <- Mon_YM_Stats %>%
   group_by(MonitoringID) %>%
   mutate(YearFromStart=Year-min(Year))
# Create decimal value of year and month values
Mon_YM_Stats$YearMonthDec <- Mon_YM_Stats$Year + ((Mon_YM_Stats$Month-0.5) / 12)

# Create summary statistics for each monitoring location based on Year
# intervals.
Mon_Y_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
            Year) %>%
   summarize(ParameterName=parameter,
             RelativeDepth=unique(RelativeDepth),
             EarliestSampleDate=min(SampleDate),
             LastSampleDate=max(SampleDate), N=length(ResultValue),
             Min=min(ResultValue), Max=max(ResultValue),
             Median=median(ResultValue), Mean=mean(ResultValue),
             StandardDeviation=sd(ResultValue))
# Puts the data in order based on ManagedAreaName, ProgramID, ProgramName,
# ProgramLocationID, then Year
Mon_Y_Stats <- as.data.table(Mon_Y_Stats[order(Mon_Y_Stats$ManagedAreaName,
                                               Mon_Y_Stats$ProgramID,
                                               Mon_Y_Stats$ProgramName,
                                               Mon_Y_Stats$ProgramLocationID,
                                               Mon_Y_Stats$Year), ])
# Writes summary statistics to file
fwrite(Mon_Y_Stats, paste0(out_dir_param,"/WC_Continuous_", param_abrev, "_",
                           region, "_MonLoc_Yr_Stats.txt"), sep="|")

# Create summary statistics for each monitoring location based on Month
# intervals.
Mon_M_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
            Month) %>%
   summarize(ParameterName=parameter,
              RelativeDepth=unique(RelativeDepth),
              EarliestSampleDate=min(SampleDate),
              LastSampleDate=max(SampleDate), N=length(ResultValue),
              Min=min(ResultValue), Max=max(ResultValue),
              Median=median(ResultValue), Mean=mean(ResultValue),
              StandardDeviation=sd(ResultValue))
# Puts the data in order based on ManagedAreaName, ProgramID, ProgramName,
# ProgramLocationID, then Month
Mon_M_Stats <- as.data.table(Mon_M_Stats[order(Mon_M_Stats$ManagedAreaName,
                                               Mon_M_Stats$ProgramID,
                                               Mon_M_Stats$ProgramName,
                                               Mon_M_Stats$ProgramLocationID,
                                               Mon_M_Stats$Month), ])
# Writes summary statistics to file
fwrite(Mon_M_Stats, paste0(out_dir_param,"/WC_Continuous_", param_abrev, "_",
                           region, "_MonLoc_Mo_Stats.txt"), sep="|")
# Reduces size of data by getting a daily average
data <- data %>%
   group_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,
            ProgramLocationID, SampleDate) %>%
   summarise(Year=unique(Year), Month=unique(Month),
             RelativeDepth=unique(RelativeDepth),
             ResultValue=mean(ResultValue), Include=unique(Include),
             Use_In_Analysis=unique(Use_In_Analysis))
# Sets column formats to appropriate types
data$SampleDate <- as.Date(data$SampleDate)
data$YearMonth <- format(data$SampleDate, format = "%m-%Y")
data$YearMonthDec <- data$Year + ((data$Month-0.5) / 12)
data$DecDate <- decimal_date(data$SampleDate)
```

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 that performed at The Water Atlas: https://sarasota.wateratlas.usf.edu/water-quality-trends/#analysis-overview

The following steps are performed:

1. Define the trend function.
2. Take the `data` variable and only include rows that have a `Use_In_Analysis` value of `TRUE`
3. Group data that have the same `ManagedAreaName`, `ProgramID`, `ProgramName`, and `ProgramLocationID`.
4. 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,
5. For each group, a temporary variable is created to run 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.
- tau, Senn Slope (SennSlope), Senn Intercept (SennIntercept), and p are extracted from the model results.

6. The two stats tables are merged based on similar groups, and then Trend is determined from the user-defined function.

7. Write summary stats to a pipe-delimited .txt file in the output directory

   - WC Continuous Output Files in SEACAR GitHub (https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WQ_Continuous/output)

After the analysis is performed, a variable is created that stores the x & y coordinates of the SKT trend line to be used for plotting

```
# 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$p.value[1]
   trend <- trend_calculator(slope, stats.median, p)
   rm(ken)
   # Prints warnings if a fit does not exist and stores values as NA
   }, warning=function(w) {
      print(w)
   }, error=function(e) {
      print(e)
   }, finally={
      if (!exists("tau")) {
         tau <- NA
      }
      if (!exists("p")) {
         p <- NA
      }
      if (!exists("slope")) {
         slope <- NA
      }
      if (!exists("intercept")) {
         intercept <- NA
      }
      if (!exists("trend")) {
         trend <- NA
      }
   })
   # Puts variables in a vector for the monitoring location currently being
   # analyzed
   KT <-c(unique(dat$MonitoringID),
          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[8])) {
      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("MonitoringID", "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=nrow(Mon_Summ)))
   colnames(KT.Stats) <- c_names
   KT.Stats[, c("MonitoringID")] <- Mon_Summ[, c("MonitoringID")]
} else{
   # Starts cycling through Monitoring locations to determine seasonal
   # Kendall Tau
   for (i in 1:n) {
      # Gets the number of rows of data for the monitoring location
      x <- nrow(Mon_YM_Stats[Mon_YM_Stats$MonitoringID==Mon_IDs[i], ])
      # Perform analysis if there is more than 1 row
      if (x>0) {
         # Store the monitoring location summary statistics to be used in
         # trend analysis
         SKT.med <- Mon_Summ$Median[Mon_Summ$MonitoringID==Mon_IDs[i]]
         SKT.minYr <- Mon_Summ$EarliestYear[Mon_Summ$MonitoringID==Mon_IDs[i]]
         SKT.maxYr <- Mon_Summ$LatestYear[Mon_Summ$MonitoringID==Mon_IDs[i]]

         # Get seasonal Kendall Tau statistics by running data for monitoring
         # location through the functions
         KT.Stats <- runStats(Mon_YM_Stats[Mon_YM_Stats$MonitoringID==
                                              Mon_IDs[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 Mon_Summ
KT.Stats <-  merge.data.frame(Mon_Summ, KT.Stats,
                              by=c("MonitoringID"), all=TRUE)

KT.Stats <- as.data.table(KT.Stats[order(KT.Stats$MonitoringID), ])

# Writes combined statistics to file
fwrite(select(KT.Stats, -MonitoringID), paste0(out_dir_param,"/WC_Continuous_",
                                               param_abrev, "_", region,
                                               "_KendallTau_Stats.txt"),
       sep="|")

# Removes data rows with no ResultValue (created by merging with MA_All)
data <- data[!is.na(data$ResultValue),]

# Gets x and y values for starting point for trendline
KT.Plot <- KT.Stats %>%
   group_by(MonitoringID) %>%
   summarize(x=EarliestYear,
             y=SennIntercept)
# Gets x and y values for ending point for trendline
KT.Plot2 <- KT.Stats %>%
   group_by(MonitoringID) %>%
   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$MonitoringID), ])
KT.Plot <- KT.Plot[!is.na(KT.Plot$y),]
```

Appendix I: 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 `Use_In_Analysis` 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.

```
# 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))
# 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$Include==TRUE, ],
             aes(x=SampleDate, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Year",
        y=paste0("Values (", unit, ")")) +
   plot_theme
# Create plot object for y-axis scaled plot
p2 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=SampleDate, 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(0, y_scale) +
   plot_theme
# Create plot object for y-axis scaled plot for past 10 years
p3 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=Year, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",
        x="Year", y=paste0("Values (", unit, ")")) +
   ylim(0, y_scale) +
   scale_x_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+0.5),
                      breaks=seq(max(data$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
```

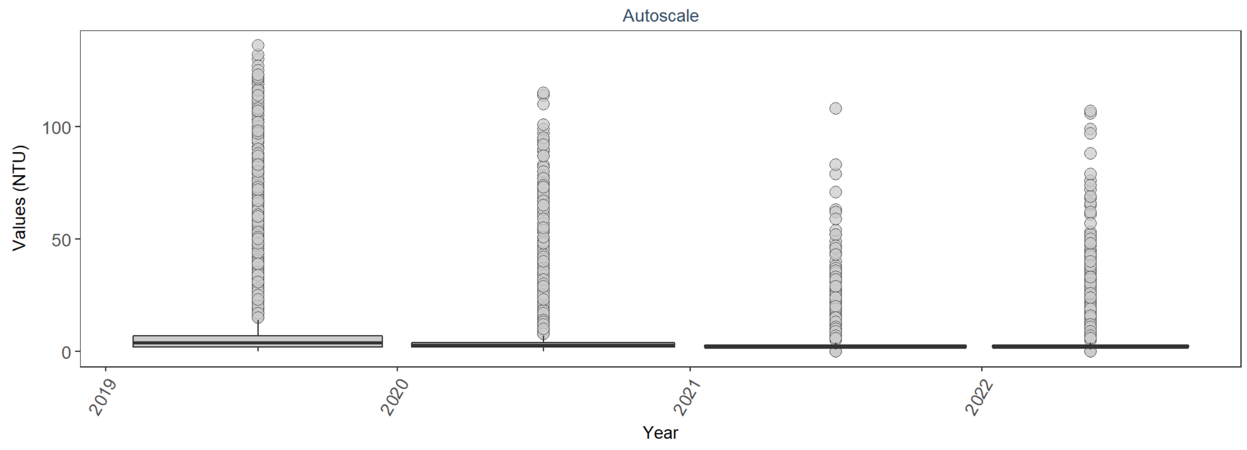
```r
p2 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Scaled to 5x Standard Deviation",
        x="Year", y=paste0("Values (", unit, ")")) +
   ylim(0, y_scale) +
   plot_theme +
   theme(legend.position="none")
# Create plot object for y-axis scaled plot for past 10 years
p3 <- ggplot(data=data[data$Include==TRUE, ],
             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 5x Standard Deviation, Last 10 Years",
        x="Year", y=paste0("Values (", unit, ")")) +
   ylim(0, y_scale) +
   scale_x_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+0.5),
                      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))
```

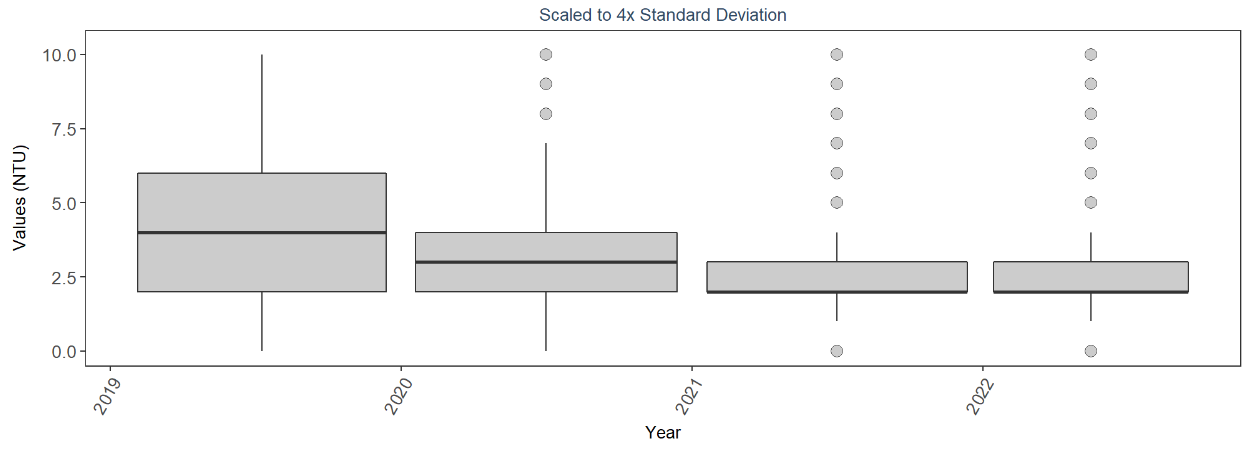
The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

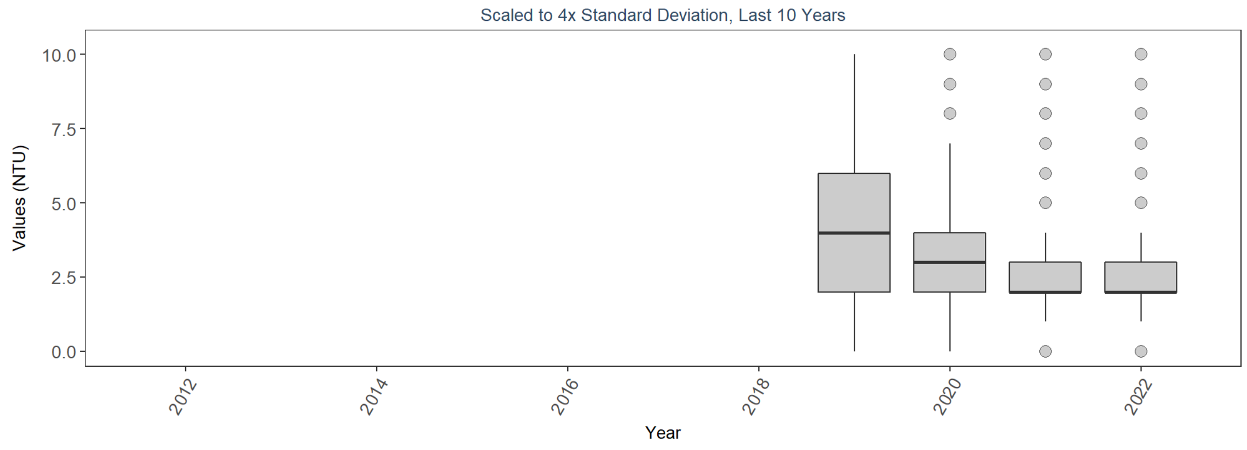
```r
# Create plot object for auto-scaled y-axis plot
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
   geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
                outlier.color="#333333", outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Month",
        y=paste0("Values (", unit, ")"), fill="Month") +
   scale_x_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal") +
   guides(fill=guide_legend(nrow=1))
# Create plot object for y-axis scaled plot
p2 <- ggplot(data=data[data$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 5x Standard Deviation",
        x="Month", y=paste0("Values (", unit, ")")) +
   ylim(0, 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 5x Standard Deviation, Last 10 Years",
        x="Month", y=paste0("Values (", unit, ")")) +
   ylim(0, 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))
```

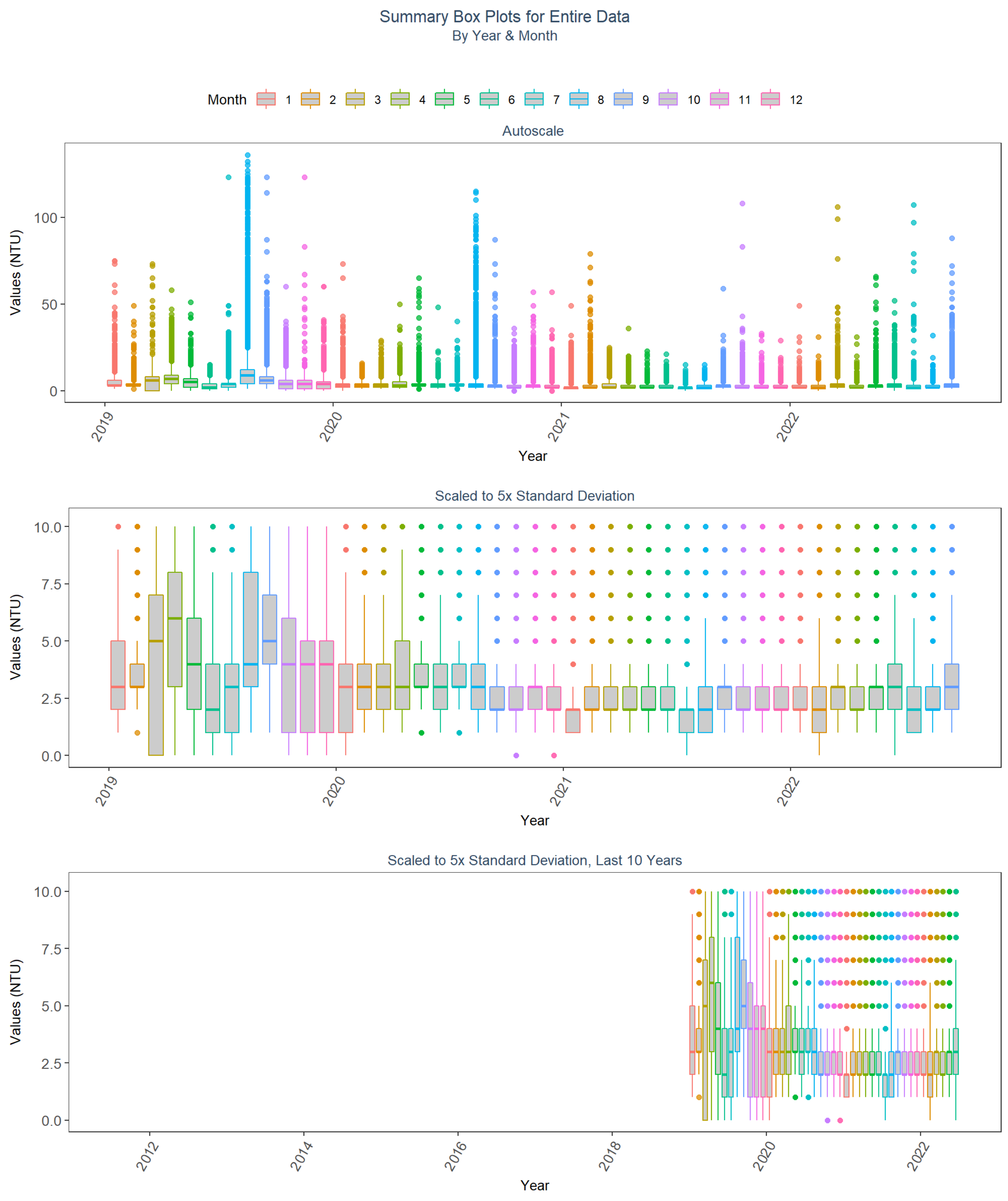
Summary Box Plots for Entire Data

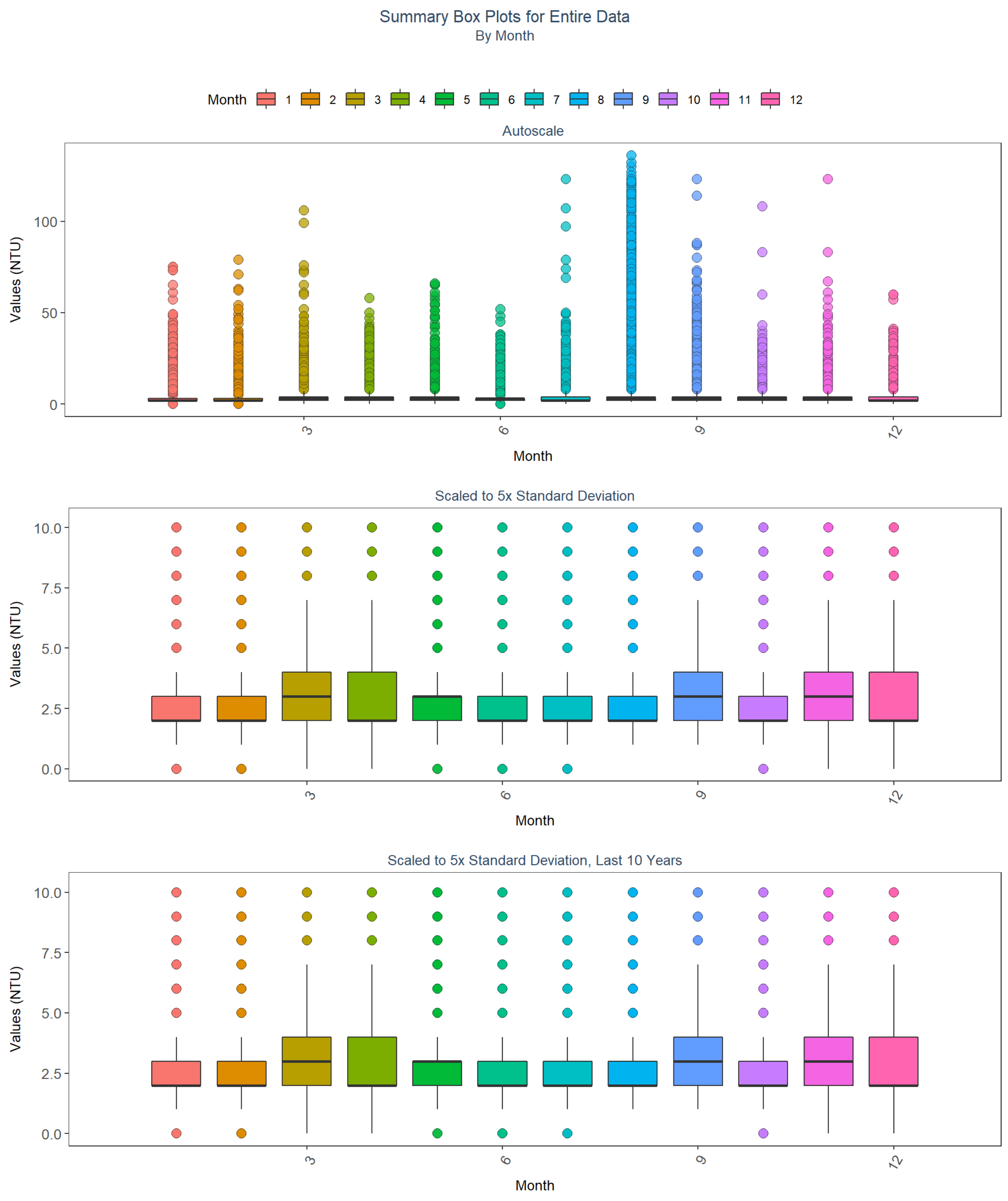
By Year








Summary Box Plots for Entire Data
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
0
50
100
2019
2020
2021
2022
Year
Scaled to 5x Standard Deviation
Values (NTU)
0.0
2.5
5.0
7.5
10.0
2019
2020
2021
2022
Year
Scaled to 5x Standard Deviation, Last 10 Years
Values (NTU)
0.0
2.5
5.0
7.5
10.0
2012
2014
2016
2018
2020
2022
Year



Appendix II: Monitoring Location Trendlines

The plots created in this section are designed to show the general trend of the data. Data is taken and grouped by `MonitoringID`. 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 averages that have been aggregated by year and month for the desired monitoring location
2. Determine the earliest and latest year of the data to create x-axis scale and intervals
3. Determine the x-axis scale
4. Set the plot type as a line and point plot with the specifics of each
5. Add the linear trend determined form the seasonal Kendall Tau slope and intercept
6. Create the title, x-axis, y-axis, and labels
7. Set the y and x limits
8. Apply the plot theme
9. Set the SKT analysis results as a table figure
10. Combine the plot and table to be displayed together

```
# Determines whether analyzed monitoring locations exist. If they do, begins
# looping through them
if(n==0){
   print("There are no monitoring locations that qualify.")
} else {
   # Begins looping through each monitoring location
   for (i in 1:n) {
      # Gets data to be used in plot for monitoring location
      plot_data <- Mon_YM_Stats[Mon_YM_Stats$MonitoringID==Mon_IDs[i],]
      # Gets trendline data for monitoring location
      KT.plot_data <- KT.Plot[KT.Plot$MonitoringID==Mon_IDs[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
      }
      # Get name of managed area
      MA_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID==Mon_IDs[i]]
      # Get program location name
      Mon_name <- paste0(KT.Stats$ProgramID[KT.Stats$MonitoringID==Mon_IDs[i]],
         "\n", KT.Stats$ProgramName[KT.Stats$MonitoringID==Mon_IDs[i]], "\n",
         KT.Stats$ProgramLocationID[KT.Stats$MonitoringID==Mon_IDs[i]])
      # 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_name, "\n", Mon_name),
              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$MonitoringID==Mon_IDs[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. Does not add space after last plot
      if(i!=n){
        cat("\n \n \n")
      }
      rm(plot_data)
      rm(KTset, leg)
      rm(plot_data)
      rm(KTset, leg)
   }
}
```

[1] "There are no monitoring locations that qualify."

Appendix III: Monitoring Location Summary Box Plots

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

1. Use the data set that only has `Use_In_Analysis` of `TRUE` for the desired monitoring location
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 `MonitoringID` with data grouped by `Year`, then `Year` and `Month`, then finally `Month` only. Each program 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

```r
# Determines whether analyzed monitoring locations exist. If they do, begins
# looping through them
if(n==0){
   print("There are no monitoring locations that qualify.")
} else {
   # Begin looping through monitoringg locations
   for (i in 1:n) {
      # Determine upper and lower bounds of time for x-axis
      year_lower <- min(data$Year[data$Use_In_Analysis==TRUE &
                                  data$MonitoringID==Mon_IDs[i]])
      year_upper <- max(data$Year[data$Use_In_Analysis==TRUE &
                                  data$MonitoringID==Mon_IDs[i]])
      # Determine upper and lower bounds of ResultValue for y-axis
      min_RV <- min(data$ResultValue[data$Use_In_Analysis==TRUE &
                                     data$MonitoringID==Mon_IDs[i]])
      mn_RV <- mean(data$ResultValue[data$Use_In_Analysis==TRUE &
                                     data$MonitoringID==Mon_IDs[i] &
                                     data$ResultValue <
                                     quantile(data$ResultValue, 0.98)])
      sd_RV <- sd(data$ResultValue[data$Use_In_Analysis==TRUE &
                                   data$MonitoringID==Mon_IDs[i] &
                                   data$ResultValue <
                                   quantile(data$ResultValue, 0.98)])
      # Sets x- and y-axis scale
      x_scale <- ifelse(year_upper - year_lower > 30, 10, 5)
      y_scale <- mn_RV + 4 * sd_RV
      # Gets managed area name for title
      MA_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID==Mon_IDs[i]]
      # Gets program location name for title
      Mon_name <- paste0(KT.Stats$ProgramID[KT.Stats$MonitoringID==Mon_IDs[i]],
         "\n", KT.Stats$ProgramName[KT.Stats$MonitoringID==Mon_IDs[i]], "\n",
         KT.Stats$ProgramLocationID[KT.Stats$MonitoringID==Mon_IDs[i]])

      ##Year plots
      # Create plot object for auto-scaled y-axis plot
      p1 <- ggplot(data=data[data$Use_In_Analysis==TRUE &
                             data$MonitoringID==Mon_IDs[i], ],
                   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=data[data$Use_In_Analysis==TRUE &
                                  data$MonitoringID==Mon_IDs[i], ],
                   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=data[data$Use_In_Analysis==TRUE &
                                  data$MonitoringID==Mon_IDs[i] &
                                  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_name, "\n", Mon_name),
                            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=data[data$Use_In_Analysis==TRUE &
                                  data$MonitoringID==Mon_IDs[i], ],
                   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 +
```

```r
    theme(legend.position="none")
  # Create plot object for y-axis scaled plot
  p5 <- ggplot(data=data[data$Use_In_Analysis==TRUE &
                              data$MonitoringID==Mon_IDs[i], ],
               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=data[data$Use_In_Analysis==TRUE &
                              data$MonitoringID==Mon_IDs[i], ],
               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_name, "\n", Mon_name),
                         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=data[data$Use_In_Analysis==TRUE &
                              data$MonitoringID==Mon_IDs[i], ],
               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=data[data$Use_In_Analysis==TRUE &
                            data$MonitoringID==Mon_IDs[i], ],
                   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=data[data$Use_In_Analysis==TRUE &
                            data$MonitoringID==Mon_IDs[i] &
                            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_name, "\n", Mon_name),
                              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(0.1, 1)))
      print(ggarrange(p00, YMset, ncol=1, heights=c(0.1, 1)))
      print(ggarrange(p000, Mset, ncol=1, heights=c(0.1, 1)))

      rm(plot_data)
      rm(p1, p2, p3, p4, p5, p6, p7, p8, p9, p0, p00, p000, leg1, leg2,
         Yset, YMset, Mset)
   }
}
```

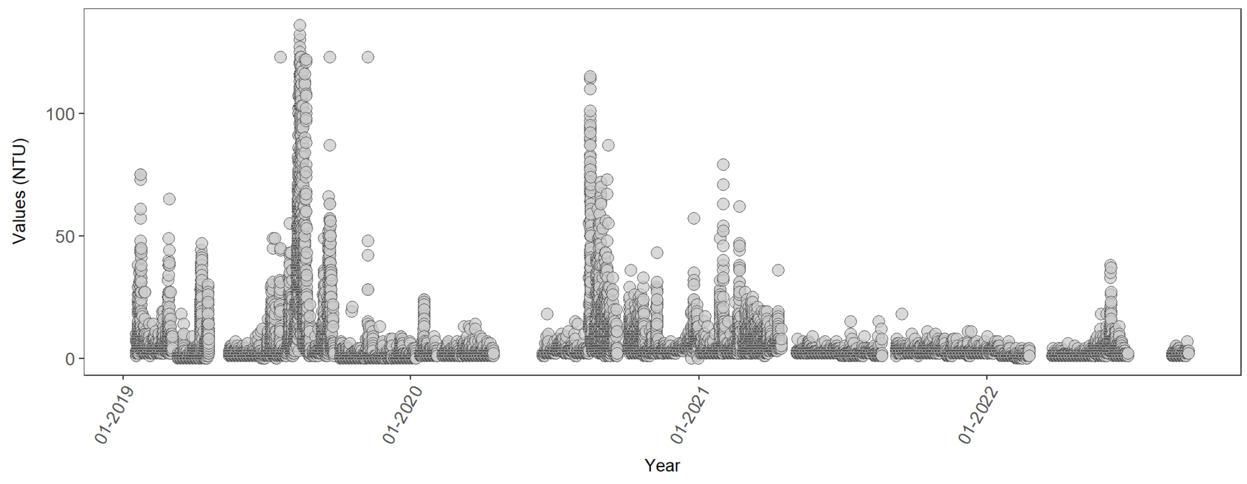
```
## [1] "There are no monitoring locations that qualify."
```

Appendix IV: Excluded Monitoring Locations

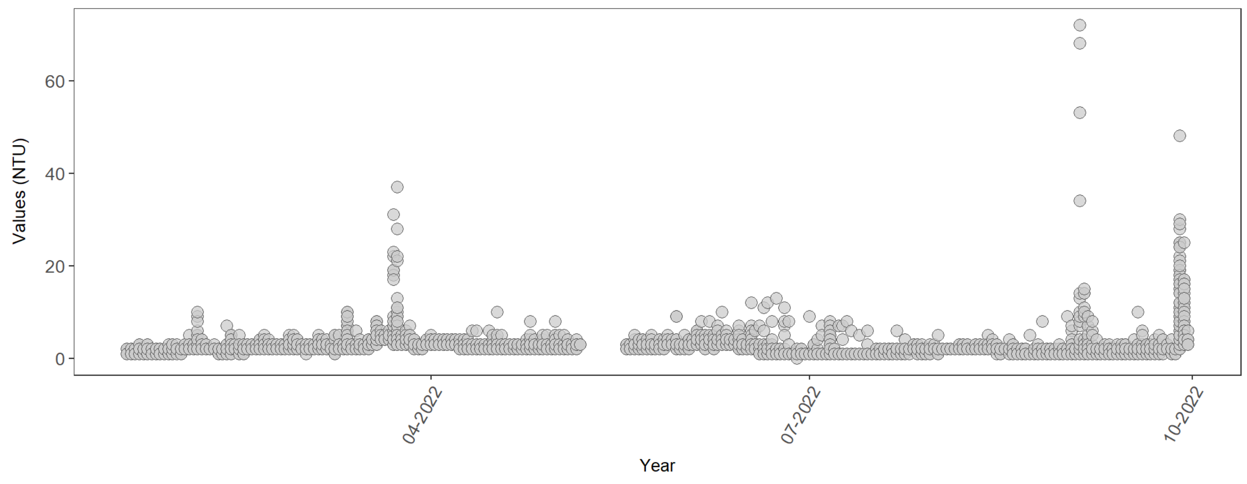
Scatter plots of data values are created for monitoring locations that have fewer than 5 separate years of data entries.

```r
# Get list of monitoring locations that have data, but without sufficient data
Mon_Exclude <- Mon_Summ[Mon_Summ$SufficientData==FALSE & N_Years>0,]
Mon_Exclude <- Mon_Exclude[order(Mon_Exclude$MonitoringID),]
z=nrow(Mon_Exclude)
# Determines whether excluded monitoring locations exist. If they do, begins
# looping through them
if(z==0){
   print("There are no monitoring locations that qualify.")
} else {
   for(i in 1:z){
      # Get managed area name for title
      MA_name <- unique(data$ManagedAreaName[
         data$MonitoringID==Mon_Exclude$MonitoringID[i]])
      # Get program name for title
      Mon_name <- paste0(unique(data$ProgramID[
         data$MonitoringID==Mon_Exclude$MonitoringID[i]]), "\n",
         unique(data$ProgramName[
            data$MonitoringID==Mon_Exclude$MonitoringID[i]]), "\n",
         unique(data$ProgramLocationID[
            data$MonitoringID==Mon_Exclude$MonitoringID[i]]))
      # Create scatter plot with data
      p1<-ggplot(data=data[data$MonitoringID==Mon_Exclude$MonitoringID[i]&
                              data$Include==TRUE, ],
                 aes(x=SampleDate, y=ResultValue)) +
         geom_point(shape=21, size=3, color="#333333", fill="#cccccc",
                    alpha=0.75) +
         labs(title=paste0(MA_name, "\n",
                           Mon_name, " (", Mon_Exclude$N_Years[i],
                        " Unique Years)"),
              subtitle="Autoscale", x="Year",
              y=paste0("Values (", unit, ")")) +
         plot_theme +
         scale_x_date(labels=date_format("%m-%Y"))
      print(p1)
   }
}
```

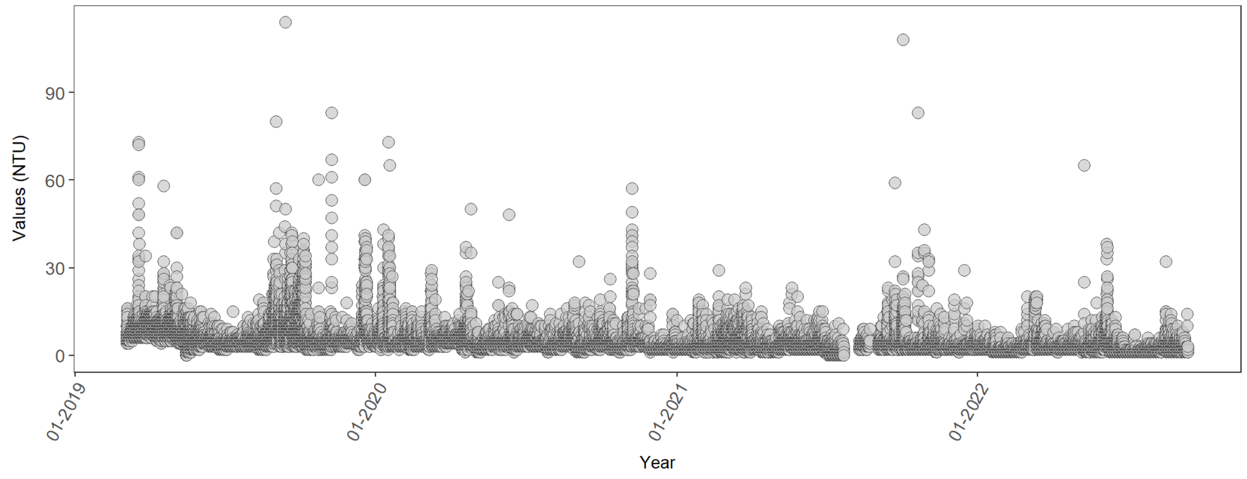
Biscayne Bay Aquatic Preserve
5077
Biscayne Bay Aquatic Preserves Continuous Water Quality Monitoring
BBBB14 (4 Unique Years)
Autoscale



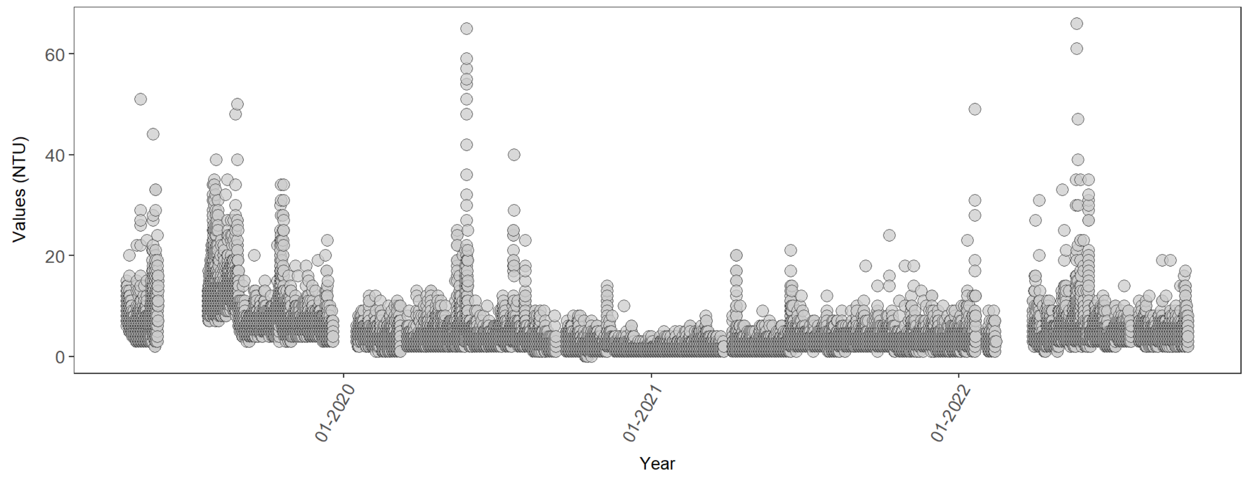
Biscayne Bay Aquatic Preserve
5077
Biscayne Bay Aquatic Preserves Continuous Water Quality Monitoring
BBCWA4 (1 Unique Years)
Autoscale



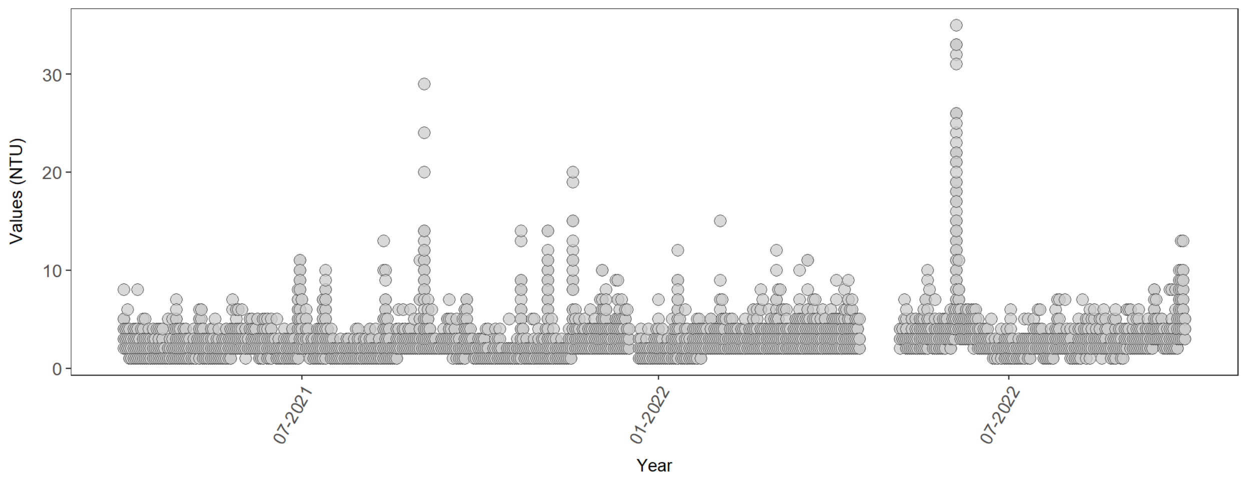
Biscayne Bay Aquatic Preserve
5077
Biscayne Bay Aquatic Preserves Continuous Water Quality Monitoring
BBJT71 (4 Unique Years)
Autoscale



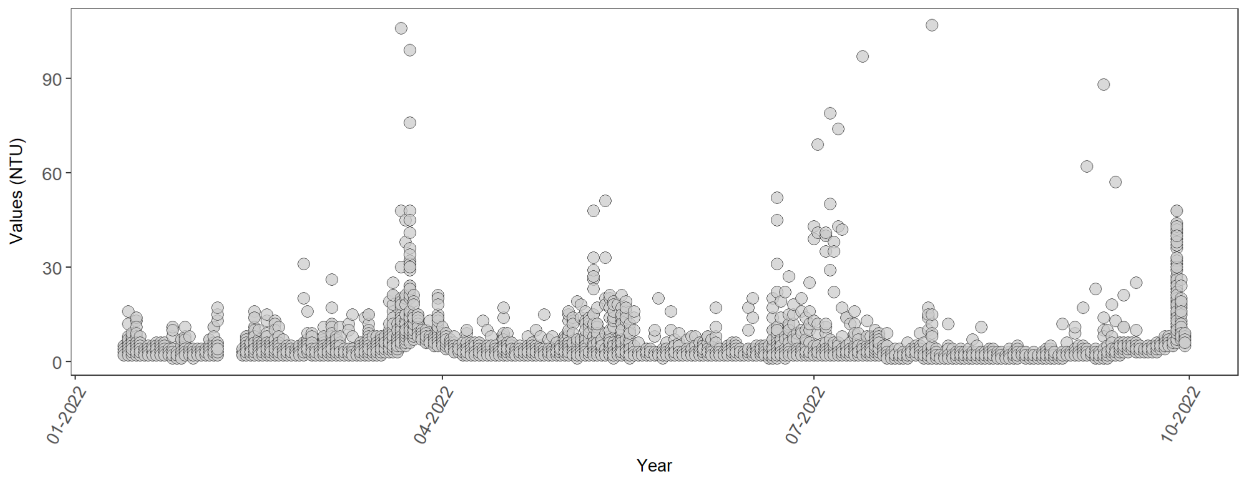
Biscayne Bay Aquatic Preserve
5077
Biscayne Bay Aquatic Preserves Continuous Water Quality Monitoring
BBLR03 (4 Unique Years)
Autoscale



Biscayne Bay Aquatic Preserve
5077
Biscayne Bay Aquatic Preserves Continuous Water Quality Monitoring
BBMRDW (2 Unique Years)
Autoscale



Biscayne Bay Aquatic Preserve
5077
Biscayne Bay Aquatic Preserves Continuous Water Quality Monitoring
BBMRRB (1 Unique Years)
Autoscale


Values (NTU)
Year