

SEACAR Discrete Water Quality Analysis: Bottom Total Nitrogen

Last compiled on 18 May, 2022

Contents

Purpose	1
Adjustable Inputs	2
Libraries	2
File Import	3
Data Filtering and Data Impacted by Specific Value Qualifiers	4
Managed Area Statistics	6
Monitoring Location Statistics	7
Seasonal Kendall Tau Analysis	8
Appendix I: Scatter Plot of Entire Dataset	11
Appendix II: Dataset Summary Box Plots	14
Appendix III: Excluded Managed Areas	20
Appendix IV: Managed Area Trendlines	26
Appendix V: Managed Area Summary Box Plots	39

Purpose

The purpose of this script is to analyze the discrete bottom total nitrogen data that is created from the SEACAR database, apply filtering criteria, create summary plots, and perform seasonal Kendall Tau analysis for each program location and summary statistics for values measured at the desired depth.

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.

Adjustable Inputs

This is placed early so that it is easier to edit parameters that users may want to adjust.

The first variable is whether you want to create the summary plots in the appendices. If you want to see all appendix plots, set `APP_Plots` to `TRUE`. If you would like to only perform the analysis and export the data files with minimal plots, set `APP_Plots` to `FALSE`. This option is available because generating the plots in the appendices increases the processing time significantly.

Since the file names all have similar structure with only the parameter name being varied, the code below sets variables to include standard string information that is the same across all data files.

This includes: the raw data directory (`in_dir`), output file directory (`out_dir`), file prefix (`file_pref`), date the files were created from the database (`file_date`), the name of the parameter of interest (`param_name`), and the relative depth of interest (`depth`). The complete file name is created by pasting all of the strings together with the specific parameter name without spaces (`paste0` command).

```
APP_Plots <- TRUE
in_dir <- "data/"
out_dir <- "output/"
file_pref <- "Combined_WQ_WC_NUT_"
file_date <- "2022-Apr-13"
param_name <- "Total_Nitrogen"
depth <- "Bottom"
activity <- "Sample"
```

Libraries

Loads libraries used in the script. The inclusion of `scipen` option limits how frequently R defaults to scientific notation.

```
library(knitr)
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.2.0
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.0
```

```
library(lubridate)
library(ggplot2)
library(ggpubr)
library(scales)
library(EnvStats)
```

```
## Warning: package 'EnvStats' was built under R version 4.2.0
```

```
library(tidyr)
options(scipen = 999)
```

File Import

Creates file name from inputs above and read in the file from txt format with pipe delimiters.

The code creates output directories for the output files if they don't exist in the directory.

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 units of the parameter, sets the `SampleDate` as a date object, and creates various scales of the date to be used by plotting functions.

```
if(!file.exists("output")){
  dir.create("output")}

file_in <- paste0(in_dir, file_pref, param_name, "-", file_date, ".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 = "")

MA_All <- fread("../../ManagedArea.csv", sep = ",", header = TRUE, stringsAsFactors = FALSE,
  na.strings = "")
data <- merge.data.frame(MA_All[,c("AreaID", "ManagedAreaName")],
  data, by = "ManagedAreaName")

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 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

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.) 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.

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, 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.

```
if(depth=="Bottom"){
  data$RelativeDepth[grep("12Q", data$SEACAR_QAQCFlagCode[
    data$RelativeDepth == "Surface"])] <- "Bottom"
}

data$Include <- as.logical(data$Include)
data <- data[!is.na(data$ResultValue),]
data <- data[!is.na(data$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$Include[grep("H", data$ValueQualifier[data$ProgramID==476])] <- TRUE

MA_Years <- data[data$Include == TRUE, ] %>%
  group_by(ManagedAreaName) %>%
  summarize(N = length(unique(Year)))
MA_Years <- as.data.table(MA_Years[order(MA_Years$ManagedAreaName), ])
MA_Years$Enough_Time <- ifelse(MA_Years$N < 10, FALSE, TRUE)
data <- merge.data.frame(data, MA_Years[,c("ManagedAreaName", "Enough_Time")],
  by = "ManagedAreaName")
```

```

data$Use_In_Analysis <- ifelse(data$Include == TRUE &
                              data$Enough_Time == TRUE,
                              TRUE, FALSE)

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_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$VQ_Plot[data$ProgramID!=476] <- gsub("[^U]+", "",
                                           data$VQ_Plot[data$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: 10982, Number Passed Filter: 10966
## I Codes: 49 (0.446185%)
## Q Codes: 7 (0.063741%)
## U Codes: 13 (0.118376%)

```

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 `Use_In_Analysis` 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

```

MA_names <- unique(data$ManagedAreaName[data$Use_In_Analysis == TRUE])
MA_names <- MA_names[order(MA_names)]
n <- length(MA_names)

MA_YM_Stats <- data[data$Use_In_Analysis == TRUE, ] %>%
  group_by(ManagedAreaName, Year, Month) %>%
  summarize(N = length(ResultValue),
            Min = min(ResultValue),
            Max = max(ResultValue),
            Median = median(ResultValue),
            Mean = mean(ResultValue),
            StandardDeviation = sd(ResultValue),
            ProgramIDs = paste(sort(unique(ProgramID), decreasing = FALSE),
                               collapse = ', '))
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                              MA_YM_Stats$Year,
                                              MA_YM_Stats$Month), ])
fwrite(MA_YM_Stats, paste0(out_dir, "/", param_name, "_", file_date, "_",

```

```

        activity, "_", depth,
        "_ManagedArea_YearMonth_Stats.txt"), sep = "|")

MA_Y_Stats <- data[data$Use_In_Analysis == TRUE, ] %>%
  group_by(ManagedAreaName, Year) %>%
  summarize(N = length(ResultValue),
            Min = min(ResultValue),
            Max = max(ResultValue),
            Median = median(ResultValue),
            Mean = mean(ResultValue),
            StandardDeviation = sd(ResultValue),
            ProgramIDs = paste(sort(unique(ProgramID), decreasing = FALSE),
                               collapse = ', '))
MA_Y_Stats <- as.data.table(MA_Y_Stats[order(MA_Y_Stats$ManagedAreaName,
                                             MA_Y_Stats$Year), ])
fwrite(MA_Y_Stats, paste0(out_dir, "/", param_name, "_", file_date, "_",
                          activity, "_", depth,
                          "_ManagedArea_Year_Stats.txt"), sep = "|")

MA_M_Stats <- data[data$Use_In_Analysis == TRUE, ] %>%
  group_by(ManagedAreaName, Month) %>%
  summarize(N = length(ResultValue),
            Min = min(ResultValue),
            Max = max(ResultValue),
            Median = median(ResultValue),
            Mean = mean(ResultValue),
            StandardDeviation = sd(ResultValue),
            ProgramIDs = paste(sort(unique(ProgramID), decreasing = FALSE),
                               collapse = ', '))
MA_M_Stats <- as.data.table(MA_M_Stats[order(MA_M_Stats$ManagedAreaName,
                                             MA_M_Stats$Month), ])
fwrite(MA_M_Stats, paste0(out_dir, "/", param_name, "_", file_date, "_",
                          activity, "_", depth,
                          "_ManagedArea_Month_Stats.txt"), sep = "|")

```

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 `Use_In_Analysis` 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

```
Mon_Stats <- data[data$Use_In_Analysis == TRUE, ] %>%
  group_by(ManagedAreaName, ProgramID, ProgramName, ProgramLocationID) %>%
  summarize(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, "_", file_date, "_",
                          activity, "_", depth,
                          "_MonitoringLoc_Stats.txt"), sep = "|")
```

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 `Use_In_Analysis` 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


```

tauSeasonal <- function(data, independent, stats.median, stats.minYear,
                        stats.maxYear) {
  tau <- NULL
  tryCatch({
    ken <-
      kendallSeasonalTrendTest(
        y = data$ResultValue,
        season = data$Month,
        year = data$Year,
        independent.obs = independent
      )
    tau <- ken$estimate[1]
    p <- ken$p.value[2]
    slope <- ken$estimate[2]
    intercept <- ken$estimate[3]
    trend <- trend_calculator(slope, stats.median, p)
  }, warning = function(w) {
    print(w)
  }, error = function(e) {
    print(e)
  }, finally = {
    if (!exists("tau")) {
      tau <- NULL
    }
    if (!exists("p")) {
      p <- NULL
    }
    if (!exists("slope")) {
      slope <- NULL
    }
    if (!exists("intercept")) {
      intercept <- NULL
    }
    if (!exists("trend")) {
      trend <- NULL
    }
  })
  KT <-c(unique(data$AreaID),
        unique(data$ManagedAreaName),
        parameter,
        depth,
        activity,
        independent,
        stats.median,
        nrow(data),
        stats.minYear,
        stats.maxYear,
        tau,
        p,
        slope,
        intercept,
        trend)
  return(KT)
}

```

```

}
runStats <- function(data) {
  data$Index <- as.Date(data$SampleDate) # , "%Y-%m-%d")
  data$ResultValue <- as.numeric(data$ResultValue)
  # Calculate basic stats
  stats.median <- median(data$ResultValue, na.rm = TRUE)
  stats.minYear <- min(data$Year, na.rm = TRUE)
  stats.maxYear <- max(data$Year, na.rm = TRUE)
  # Calculate Kendall Tau and Slope stats, then update appropriate columns and table
  KT <- tauSeasonal(data, TRUE, stats.median,
                    stats.minYear, stats.maxYear)
  if (is.null(KT[11])) {
    KT <- tauSeasonal(data, FALSE, stats.median,
                      stats.minYear, stats.maxYear)
  }
  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", "ParameterName", "RelativeDepth",
             "ActivityType", "Independent", "Median", "N", "EarliestYear",
             "LatestYear", "tau", "p", "SennSlope", "SennIntercept", "Trend")
if(n==0){
  KT.Stats <- data.frame(matrix(ncol=15, nrow=0))
  colnames(KT.Stats) <- c_names
  fwrite(KT.Stats, paste0(out_dir,"/", param_name, "_", file_date, "_",

```

```

        activity, "-", depth,
        "_KendallTau_Stats.txt"), sep = "|")
} else{
  for (i in 1:n) {
    values <- data[data$Use_In_Analysis == TRUE &
                    data$ManagedAreaName == MA_names[i], ]
    if (nrow(values) > 0) {
      KT.Stats <- runStats(values)
    }
  }
  KT.Stats <- as.data.frame(KT.Stats)
  c_names <- c("AreaID", "ManagedAreaName", "ParameterName", "RelativeDepth",
               "ActivityType", "Independent", "Median", "N", "EarliestYear",
               "LatestYear", "tau", "p", "SennSlope", "SennIntercept", "Trend")
  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$Median <- as.numeric(KT.Stats$Median)
  KT.Stats$N <- as.integer(KT.Stats$N)
  KT.Stats$EarliestYear <- as.integer(KT.Stats$EarliestYear)
  KT.Stats$LatestYear <- as.integer(KT.Stats$LatestYear)
  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$Trend <- as.integer(KT.Stats$Trend)
  fwrite(KT.Stats, paste0(out_dir, "/", param_name, "_", file_date, "_",
                           activity, "_", depth,
                           "_KendallTau_Stats.txt"), sep = "|")
}

```

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.

```

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,
                 color=VQ_Plot)) +
  geom_point(size = 1.5) +
  labs(subtitle = "Autoscale",
       x = "Year", y = paste0("Values (", unit, ")"),
       color="Value Qualifier") +
  theme(legend.position = "top", legend.box = "horizontal",
        legend.justification = "right",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face="bold")) +
  scale_x_date(labels = date_format("%Y")) +
  {if(inc_H==TRUE){
    scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
                                   "HU" = "#7CAE00"), na.value="black")
  } else {
    scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
  }}

p2 <- ggplot(data = data[data$Include==TRUE,],
             aes(x = SampleDate, y = ResultValue,
                 color=VQ_Plot)) +
  geom_point(size = 1.5) +
  ylim(min_RV, y_scale) +
  labs(subtitle = "Scaled to 4x Standard Deviation",
       x = "Year", y = paste0("Values (", unit, ")")) +
  theme(legend.position = "none",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold")) +
  scale_x_date(labels = date_format("%Y")) +
  {if(inc_H==TRUE){
    scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
                                   "HU" = "#7CAE00"), na.value="black")
  } else {
    scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
  }}

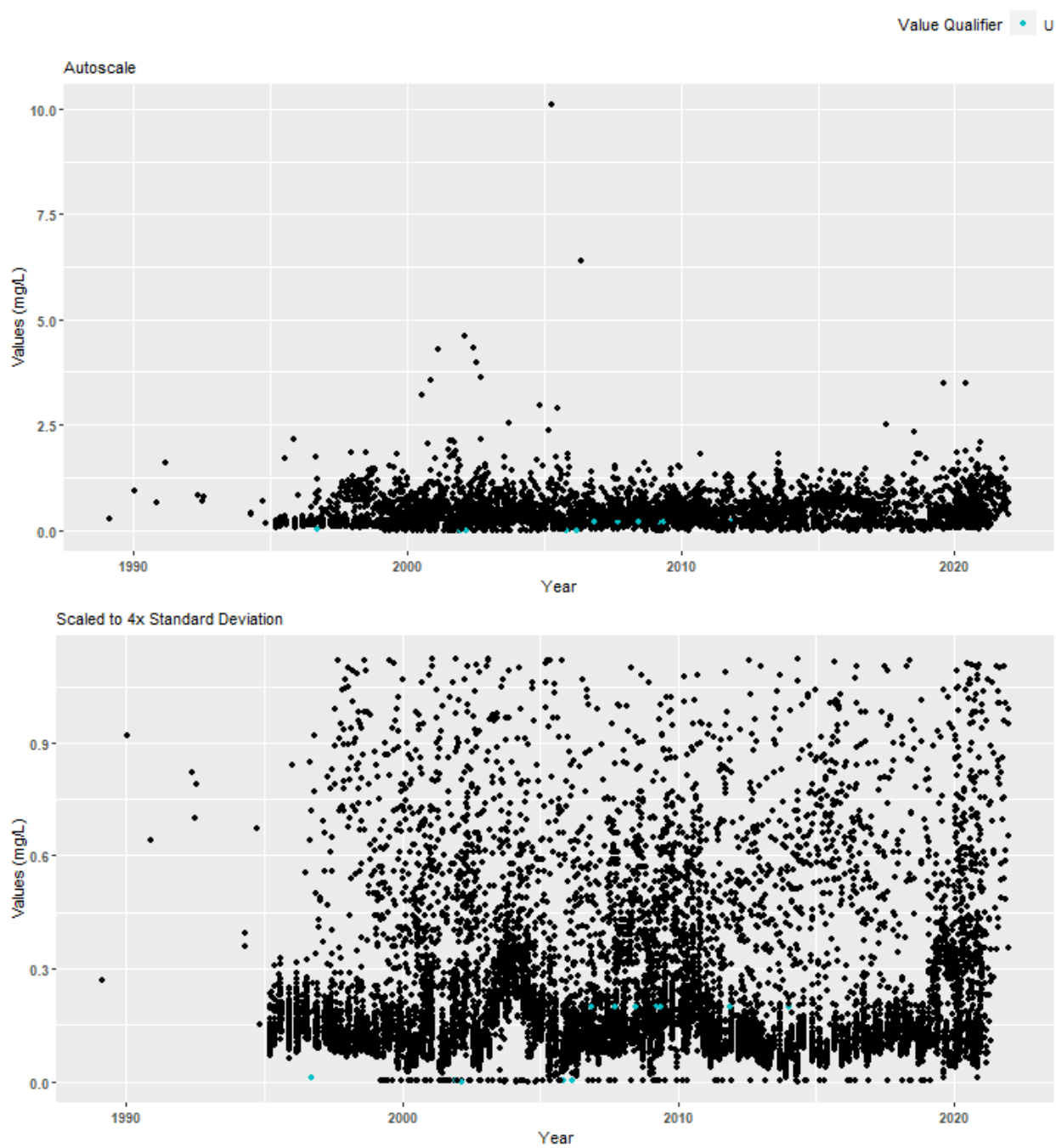
leg <- get_legend(p1)
pset <- ggarrange(leg, p1 + theme(legend.position = "none"), p2,
                  ncol = 1, heights = c(0.1, 1, 1))

p0 <- ggplot() + labs(title = "Scatter Plot for Entire Dataset") +
  theme_bw() + theme(plot.title = element_text(face="bold"),
                     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))

```

Scatter Plot for Entire Dataset



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 `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.

```
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(outlier.size = 0.5) +
  labs(subtitle = "Autoscale", x = "Year",
       y = paste0("Values (", unit, ")")) +
  theme(axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

p2 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = Year, y = ResultValue, group = Year)) +
  geom_boxplot(outlier.size = 0.5) +
  labs(subtitle = "Scaled to 4x Standard Deviation", x = "Year",
       y = paste0("Values (", unit, ")")) +
  ylim(min_RV, y_scale) +
  theme(axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

p3 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = as.integer(Year), y = ResultValue, group = Year)) +
  geom_boxplot(outlier.size = 0.5) +
  labs(subtitle = "Scaled to 4x Standard Deviation, Last 10 Years",
       x = "Year", y = paste0("Values (", unit, ")")) +
  ylim(min_RV, y_scale) +
  scale_x_continuous(limits = c(max(data$Year) - 10.5, max(data$Year)+1),
                    breaks = seq(max(data$Year) - 10, max(data$Year), 2)) +
  theme(axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))
```

```

set <- ggarrange(p1, p2, p3, ncol = 1)

p0 <- ggplot() + labs(title = "Summary Box Plots for Entire Data",
                     subtitle = "By Year") + theme_bw() +
  theme(plot.title = element_text(face="bold"),
        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(outlier.size = 0.5) +
  labs(subtitle = "Autoscale", x = "Year",
       y = paste0("Values (", unit, ")"), color="Month") +
  theme(legend.position = "top", legend.box = "horizontal",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold")) +
  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(outlier.size = 0.5) +
  labs(subtitle = "Scaled to 4x Standard Deviation",
       x = "Year", y = paste0("Values (", unit, ")")) +
  ylim(min_RV, y_scale) +
  theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

p3 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = YearMonthDec, y = ResultValue,
                 group = YearMonth, color = as.factor(Month))) +
  geom_boxplot(outlier.size = 0.5) +
  labs(subtitle = "Scaled to 4x Standard Deviation, Last 10 Years",
       x = "Year", y = paste0("Values (", unit, ")")) +
  ylim(min_RV, y_scale) +
  scale_x_continuous(limits = c(max(data$Year) - 10.5, max(data$Year)+1),
                    breaks = seq(max(data$Year) - 10, max(data$Year), 2)) +
  theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

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") + theme_bw() +
  theme(plot.title = element_text(face="bold"),
        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(outlier.size = 0.5) +
  labs(subtitle = "Autoscale", x = "Month",
       y = paste0("Values (", unit, ")"), fill="Month") +
  scale_x_continuous(limits = c(0, 13), breaks = seq(3, 12, 3)) +
  theme(legend.position = "top", legend.box = "horizontal",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold")) +
  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(outlier.size = 0.5) +
  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)) +
  theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

p3 <- ggplot(data = data[data$Include == TRUE &
                         data$Year >= max(data$Year) - 10, ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
  geom_boxplot(outlier.size = 0.5) +
  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)) +
  theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

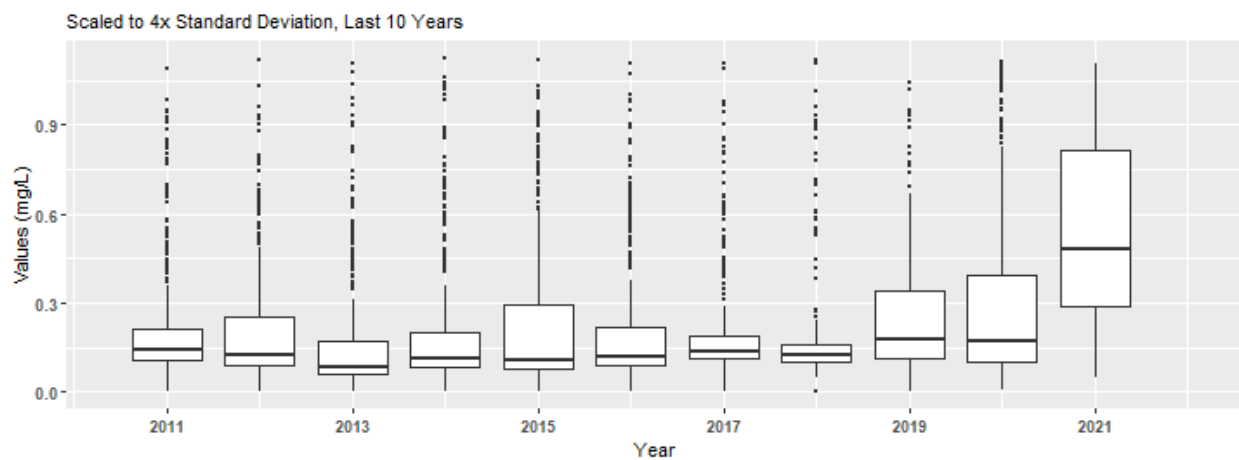
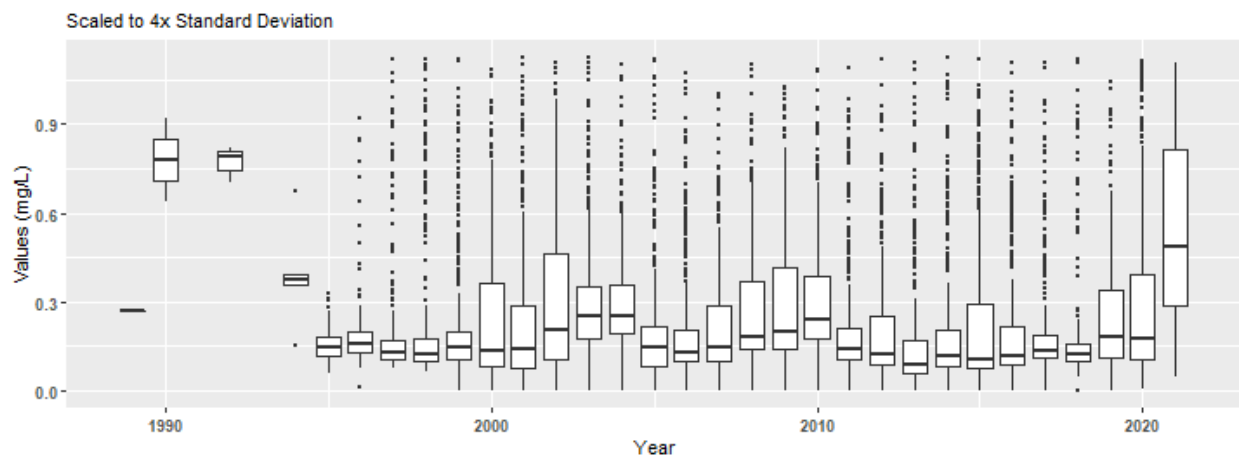
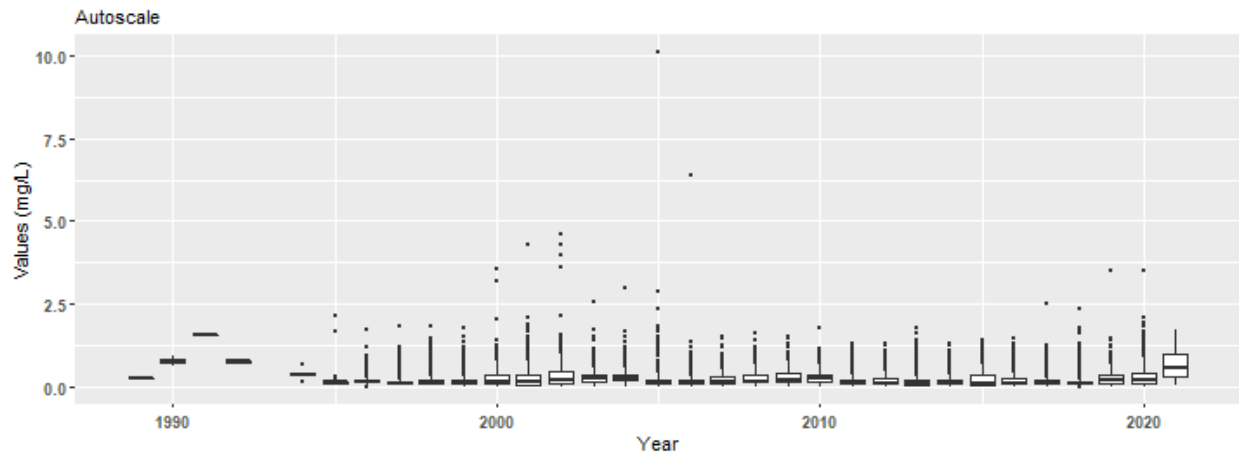
leg <- get_legend(p1)
set <- ggarrange(leg, p1 + theme(legend.position = "none"), p2, p3, ncol = 1,
                 heights = c(0.1, 1, 1, 1))

p0 <- ggplot() + labs(title = "Summary Box Plots for Entire Data",
                    subtitle = "By Month") + theme_bw() +
  theme(plot.title = element_text(face="bold"),
        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))
```

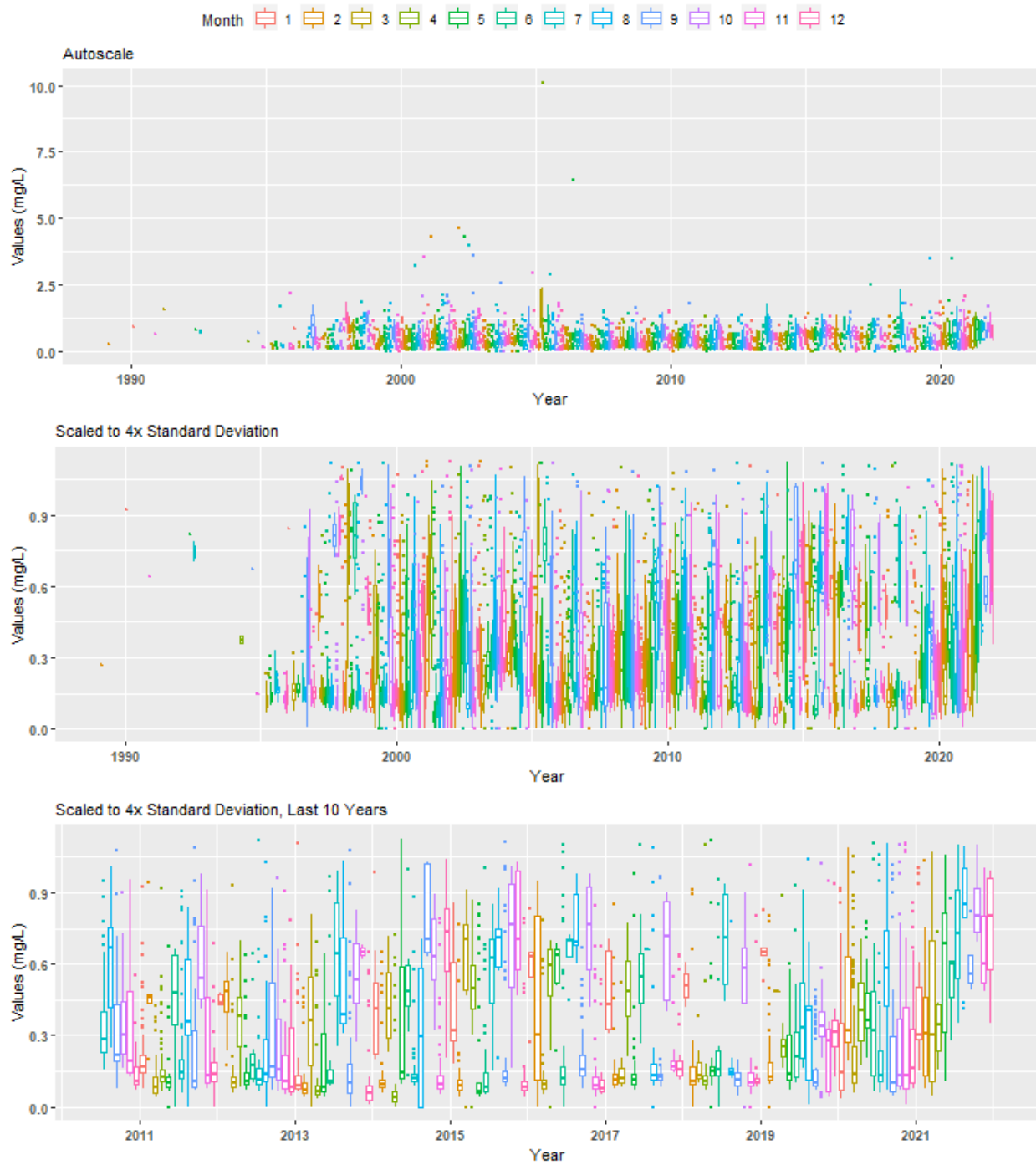

Summary Box Plots for Entire Data

By Year



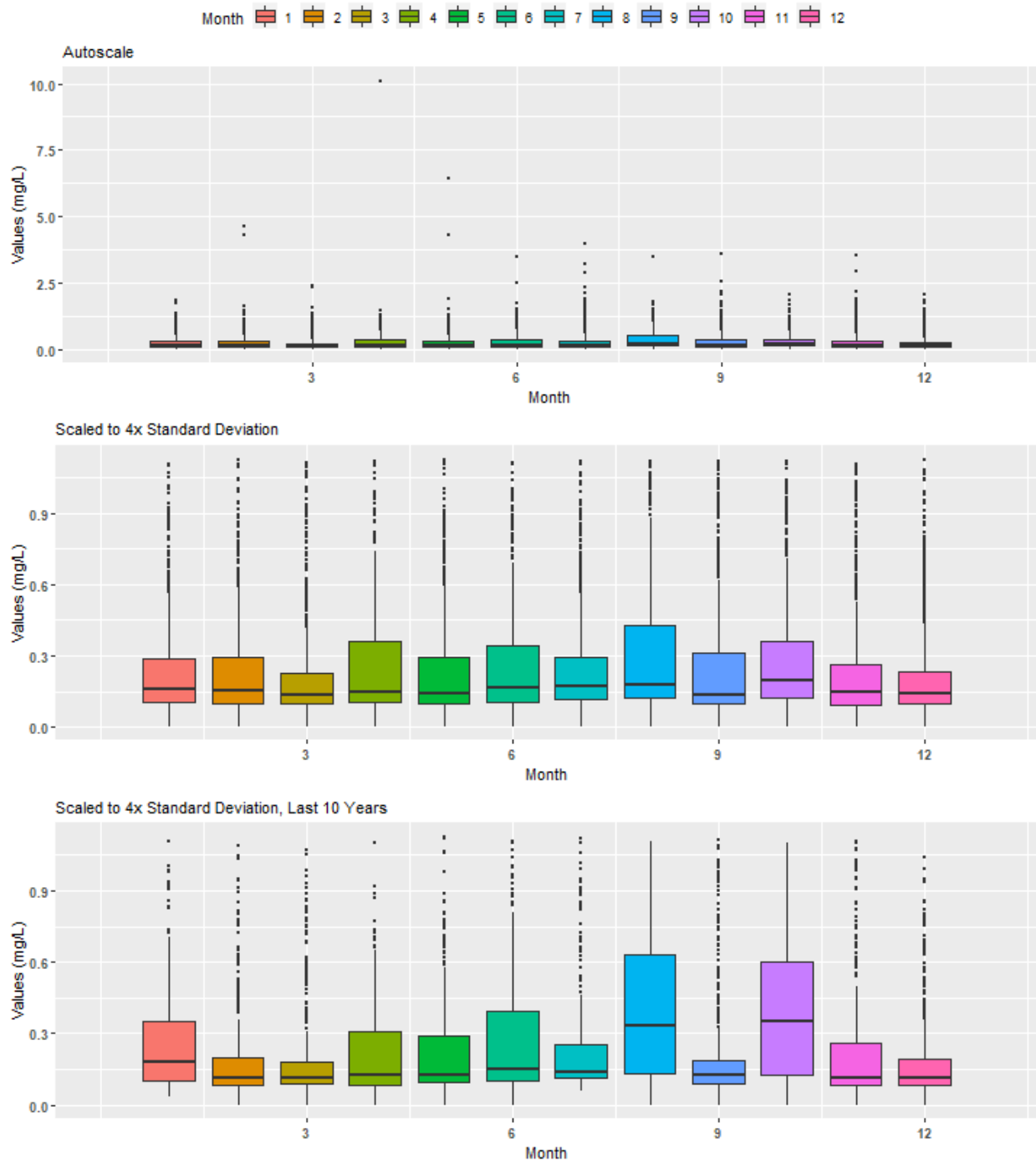
Summary Box Plots for Entire Data

By Year & Month



Summary Box Plots for Entire Data

By Month



Appendix III: Excluded Managed Areas

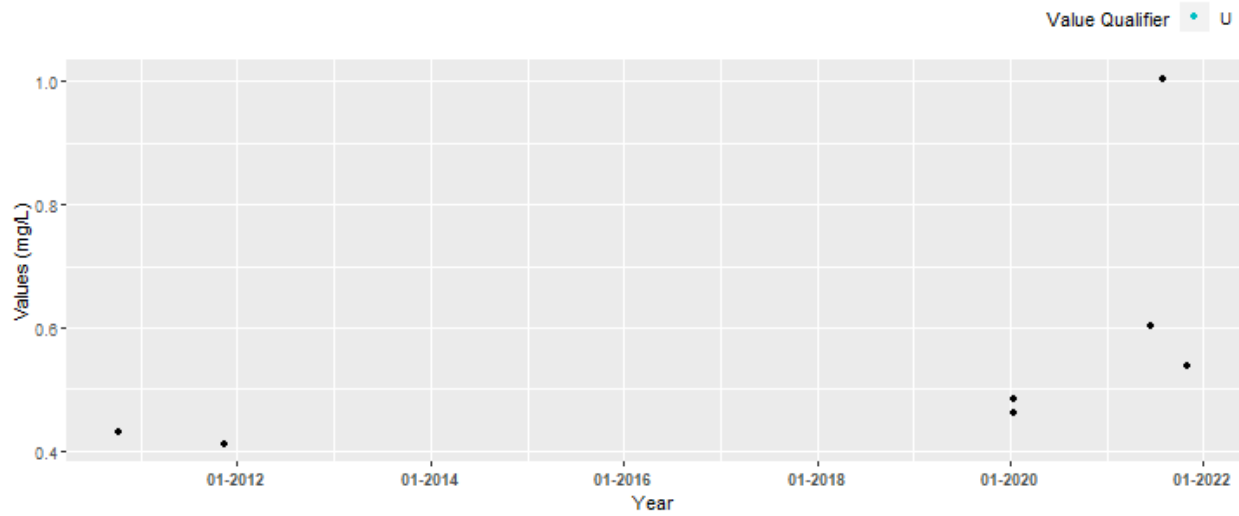
Scatter plots of data values are created for managed areas that have fewer than 10 separate years of data entries. Data points are colored based on specific value qualifiers of interest.

```
MA_Exclude <- MA_Years[MA_Years$Enough_Time==FALSE,]
MA_Exclude <- MA_Exclude[order(MA_Exclude$ManagedAreaName),]
z=length(MA_Exclude$ManagedAreaName)

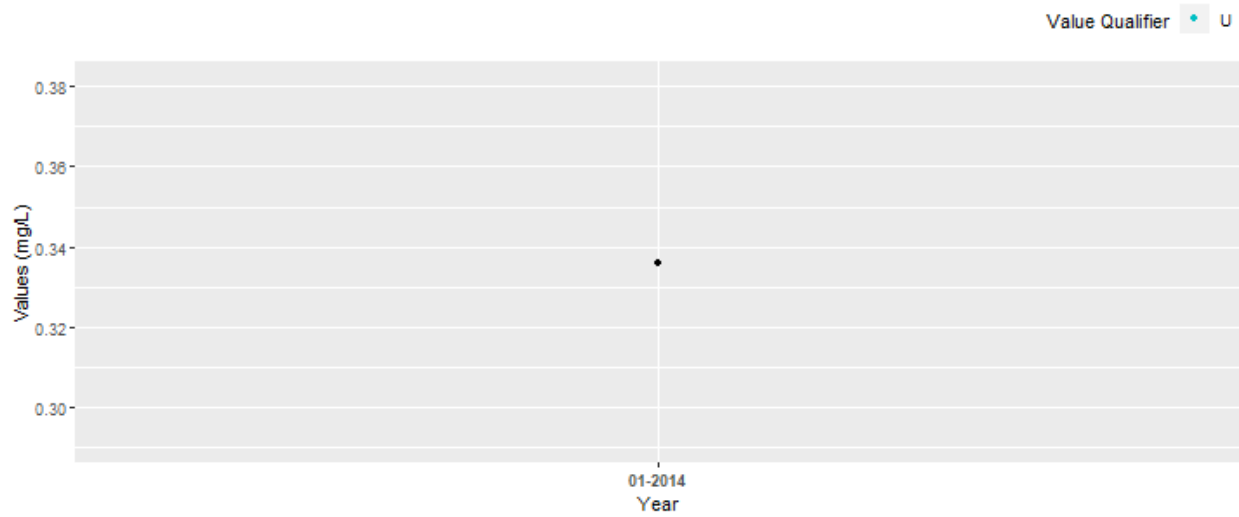
if(z==0){
  print("There are no managed areas that qualify.")
} else {
  for(i in 1:z){

    p1<-ggplot(data=data[data$ManagedAreaName==MA_Exclude$ManagedAreaName[i]&
      data$Include == TRUE, ],
      aes(x = SampleDate, y = ResultValue, color=VQ_Plot)) +
    geom_point() +
    labs(title = paste0("Scatter Plot of Excluded Managed Area\n",
      MA_Exclude$ManagedAreaName[i], " (",
      MA_Exclude$N[i], " Unique Years)"),
      subtitle="Autoscale", x = "Year",
      y = paste0("Values (", unit, ")"), color="Value Qualifier") +
    theme(legend.position = "top", legend.box = "horizontal",
      legend.justification = "right",
      axis.text.x = element_text(face = "bold")) +
    scale_x_date(labels = date_format("%m-%Y")) +
    {if(inc_H==TRUE){
      scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
        "HU" = "#7CAE00"), na.value="black")
    } else {
      scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
    }}
    print(p1)
  }
}
```

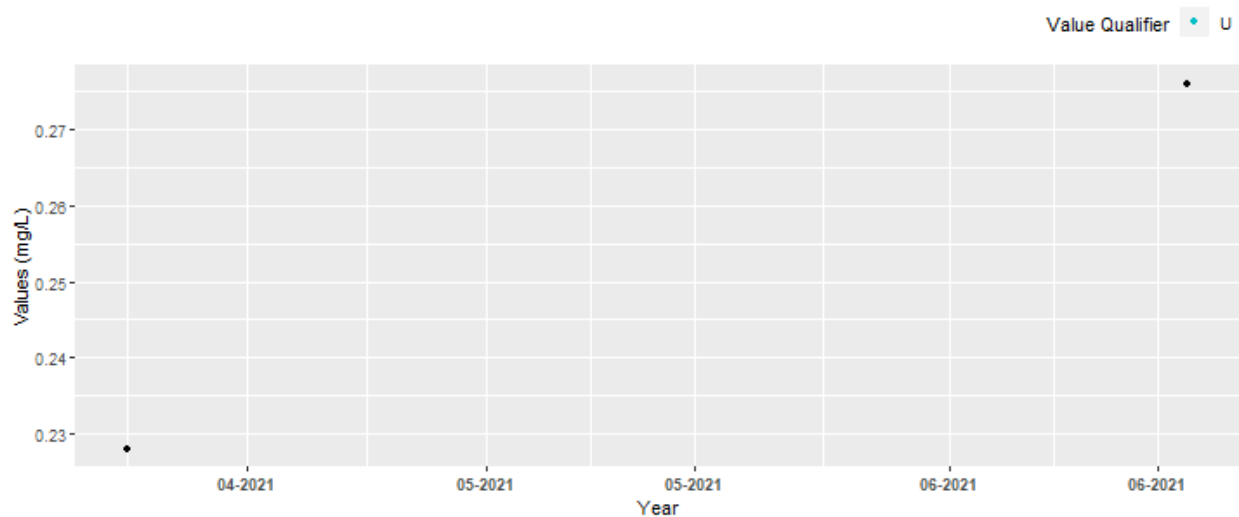
Scatter Plot of Excluded Managed Area
Cape Haze Aquatic Preserve (4 Unique Years)
Autoscale



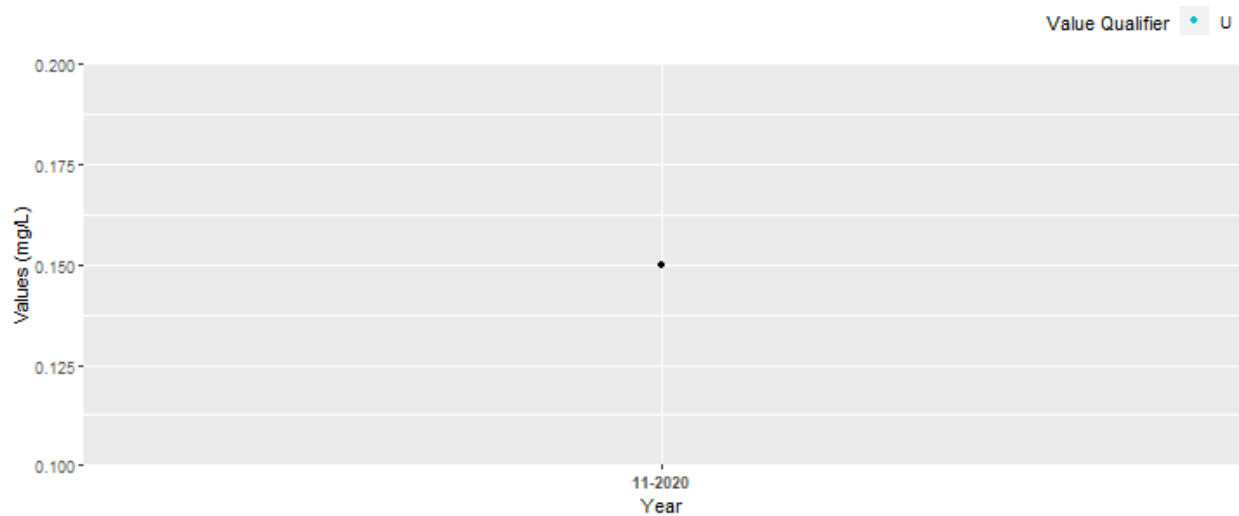
Scatter Plot of Excluded Managed Area
Fort Clinch State Park Aquatic Preserve (1 Unique Years)
Autoscale



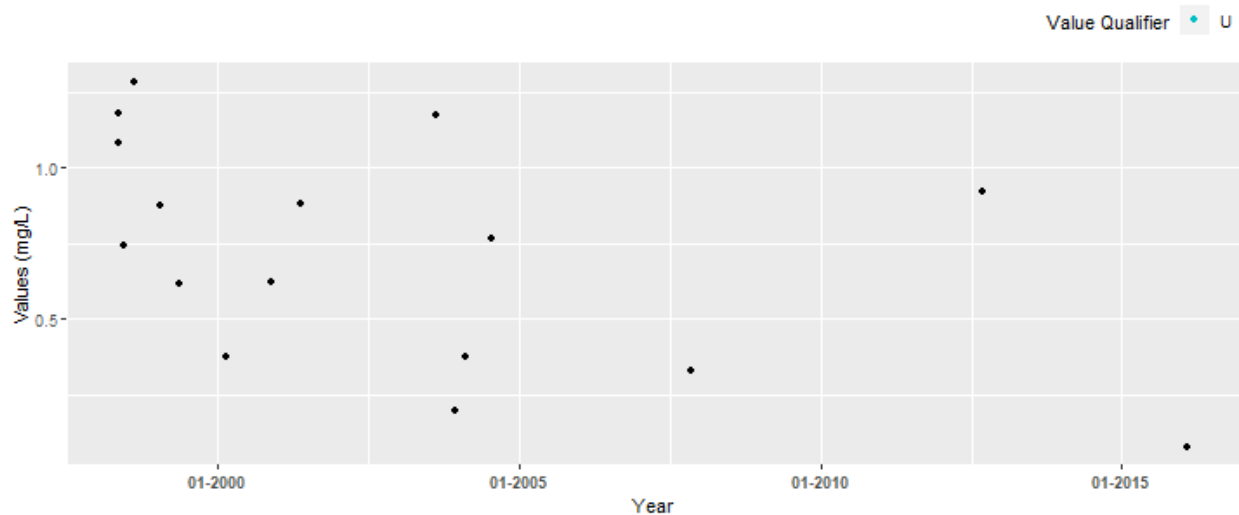
Scatter Plot of Excluded Managed Area
Jensen Beach to Jupiter Inlet Aquatic Preserve (1 Unique Years)
Autoscale



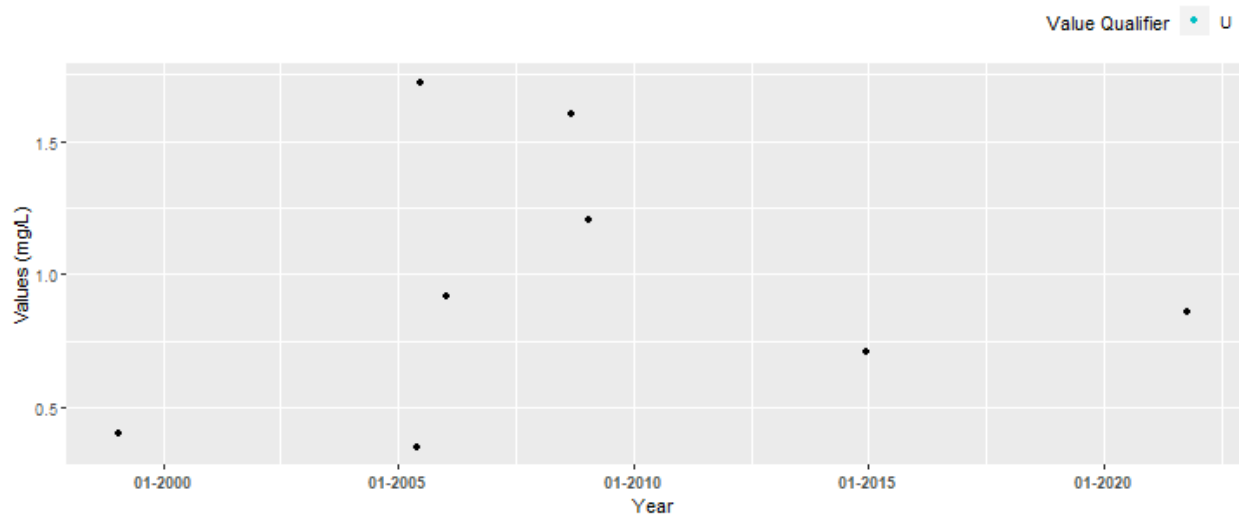
Scatter Plot of Excluded Managed Area
Lignumvitae Key Aquatic Preserve (1 Unique Years)
Autoscale



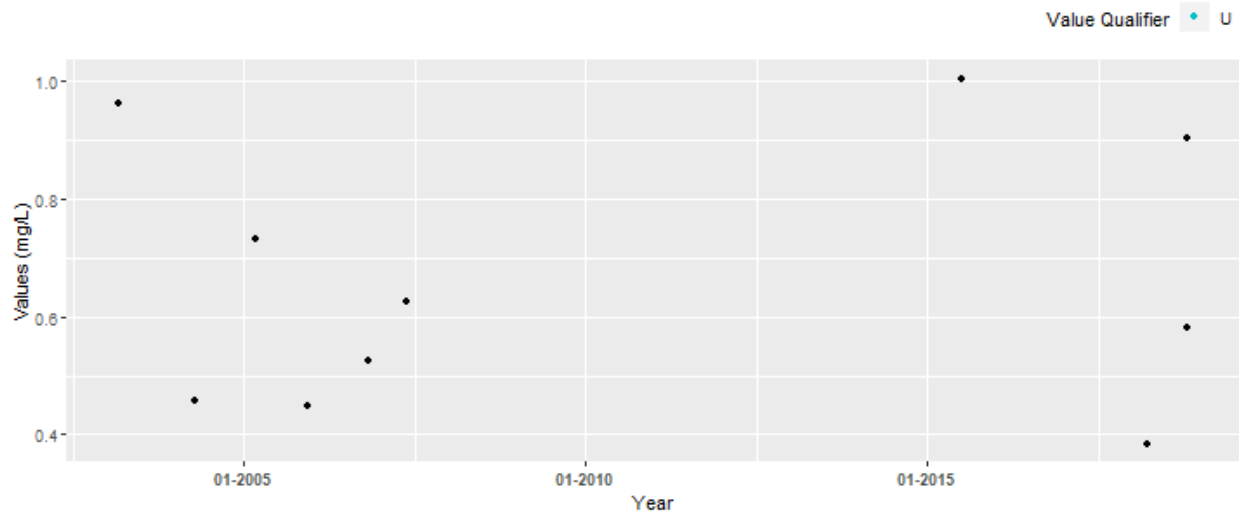
Scatter Plot of Excluded Managed Area
Mosquito Lagoon Aquatic Preserve (9 Unique Years)
Autoscale



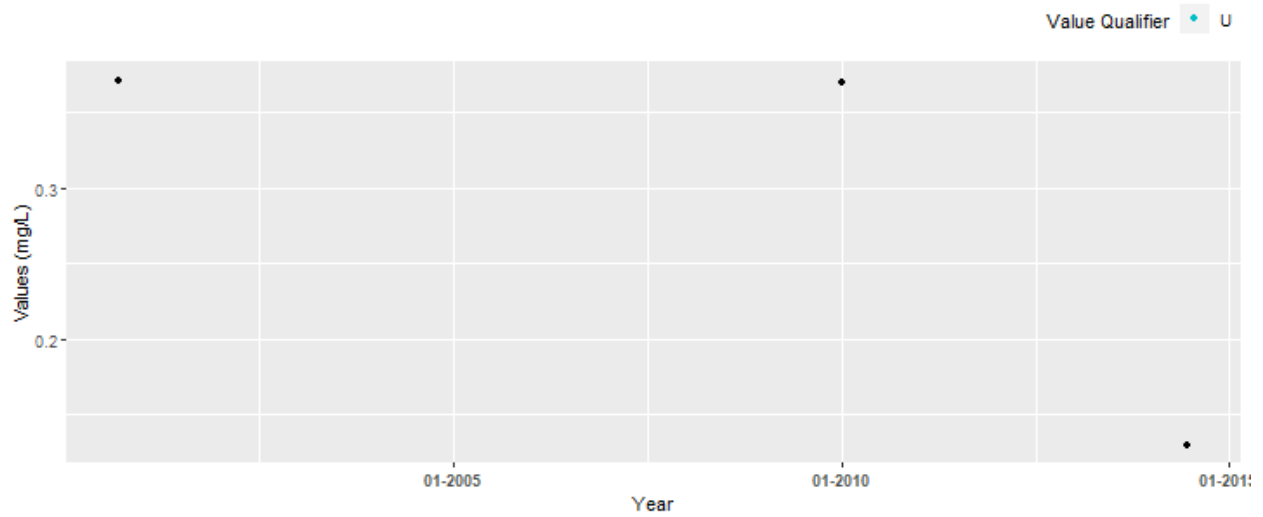
Scatter Plot of Excluded Managed Area
North Fork St. Lucie Aquatic Preserve (7 Unique Years)
Autoscale

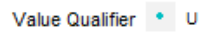
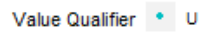


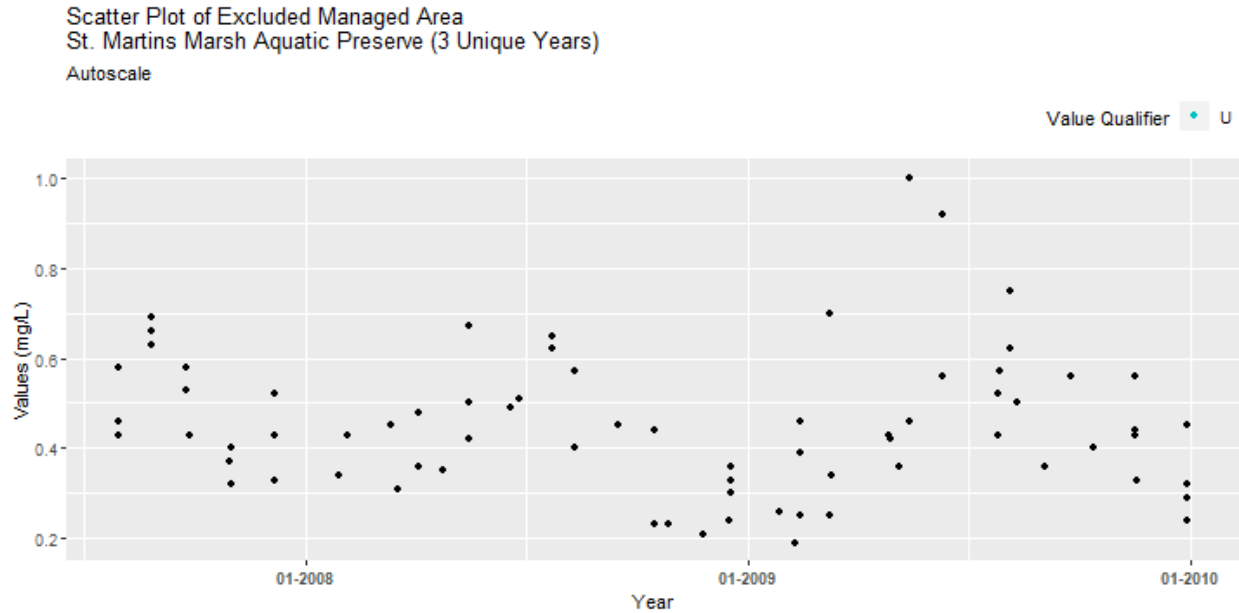
Scatter Plot of Excluded Managed Area
Pellicer Creek Aquatic Preserve (7 Unique Years)
Autoscale



Scatter Plot of Excluded Managed Area
Pine Island Sound Aquatic Preserve (3 Unique Years)
Autoscale



Value Qualifier UValue Qualifier U



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 **Use_In_Analysis** 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$Use_In_Analysis == TRUE &
                      data$ManagedAreaName == MA_names[i],]

    year_lower <- min(plot_data$Year)
    year_upper <- max(plot_data$Year)
    min_RV <- min(plot_data$ResultValue)
```

```

mn_RV <- mean(plot_data$ResultValue[data$ResultValue <
                                     quantile(data$ResultValue, 0.98)])
sd_RV <- sd(plot_data$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

tau <- KT.Stats$tau[KT.Stats$ManagedAreaName==MA_names[i]]
s_slope <- KT.Stats$SennSlope[KT.Stats$ManagedAreaName==MA_names[i]]
s_int <- KT.Stats$SennIntercept[KT.Stats$ManagedAreaName==MA_names[i]]
trend <- KT.Stats$Trend[KT.Stats$ManagedAreaName==MA_names[i]]
p <- KT.Stats$p[KT.Stats$ManagedAreaName==MA_names[i]]

model <- lm(ResultValue ~ DecDate,
             data = plot_data)
m_int <- coef(model)[[1]]
m_slope <- coef(model)[[2]]
p1 <- ggplot(data = plot_data,
             aes(x = DecDate, y = ResultValue,
                 color=VQ_Plot)) +
  geom_point(size = 1.5) +
  geom_abline(aes(slope=s_slope, intercept=s_int),
              color="red", size=1.5) +
  labs(subtitle = "Autoscale",
       x = "Year", y = paste0("Values (", unit, ")"),
       color="Value Qualifier") +
  theme(legend.position = "top", legend.box = "horizontal",
        legend.justification = "right",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face="bold")) +
  {if(inc_H==TRUE){
    scale_color_manual(values = c("H" = "#F8766D", "U" = "#00BFC4",
                                  "HU" = "#7CAE00"), na.value="black")
  } else {
    scale_color_manual(values = c("U" = "#00BFC4"), na.value="black")
  }}

p2 <- ggplot(data = plot_data,
             aes(x = DecDate, y = ResultValue,
                 color=VQ_Plot)) +
  geom_point(size = 1.5) +
  geom_abline(aes(slope=s_slope, intercept=s_int),
              color="red", size=1.5) +
  ylim(min_RV, y_scale) +
  labs(subtitle = "Scaled to 4x Standard Deviation",
       x = "Year", y = paste0("Values (", unit, ")")) +
  theme(legend.position = "none",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face="bold")) +
  {if(inc_H==TRUE){
    scale_color_manual(values = c("H" = "#F8766D", "U" = "#00BFC4",
                                  "HU" = "#7CAE00"), na.value="black")
  } else {

```

```

    scale_color_manual(values = c("U" = "#00BFC4"), na.value="black")
  }}
leg <- get_legend(p1)
KTset <- ggarrange(leg, p1 + theme(legend.position = "none"), p2,
  ncol = 1, heights = c(0.1, 1, 1))

p0 <- ggplot() + labs(title = paste0("Data Points with Trendlines for ",
  MA_names[i]),
  subtitle = paste0("Senn Slope = ", s_slope,
    ", Senn Intercept = ", s_int,
    "\nTrend = ", trend,
    ", tau = ", tau,
    ", p = ", p,
    "\nLinear Trendline: ",
    "y = ", m_slope,"x + ",m_int)) +
  theme_bw() + theme(plot.title = element_text(face="bold"),
    panel.border = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_blank())

print(ggarrange(p0, KTset, ncol = 1, heights = c(0.15, 1)))
}
}

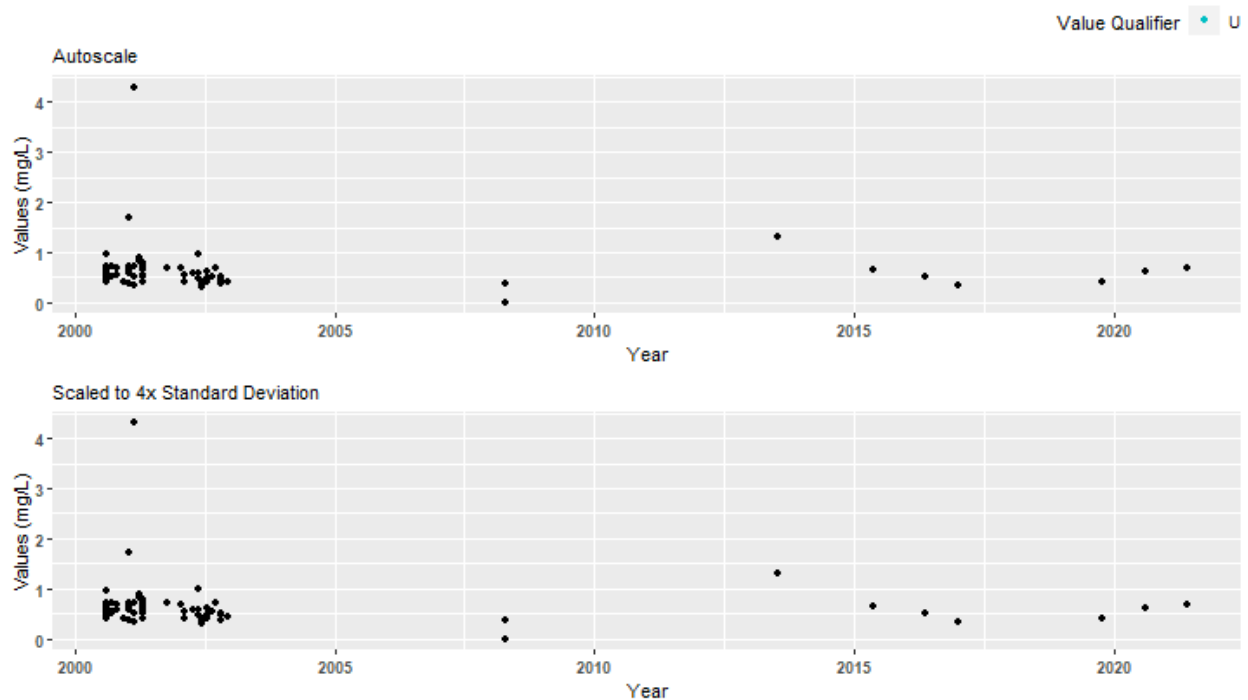
```

Data Points with Trendlines for Apalachicola Bay Aquatic Preserve

Senn Slope = -0.0178, Senn Intercept = -3.826

Trend = 0, tau = -0.0753, p = 0.1112

Linear Trendline: $y = -0.00729985544989324x + 15.2705972886423$

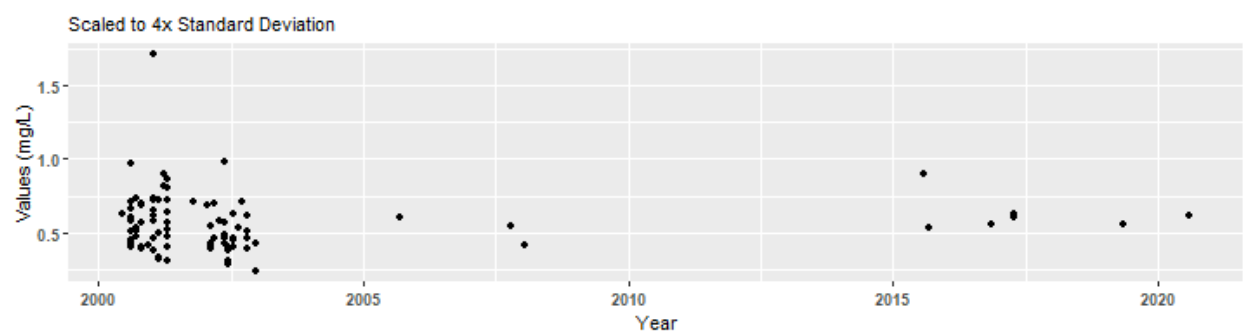
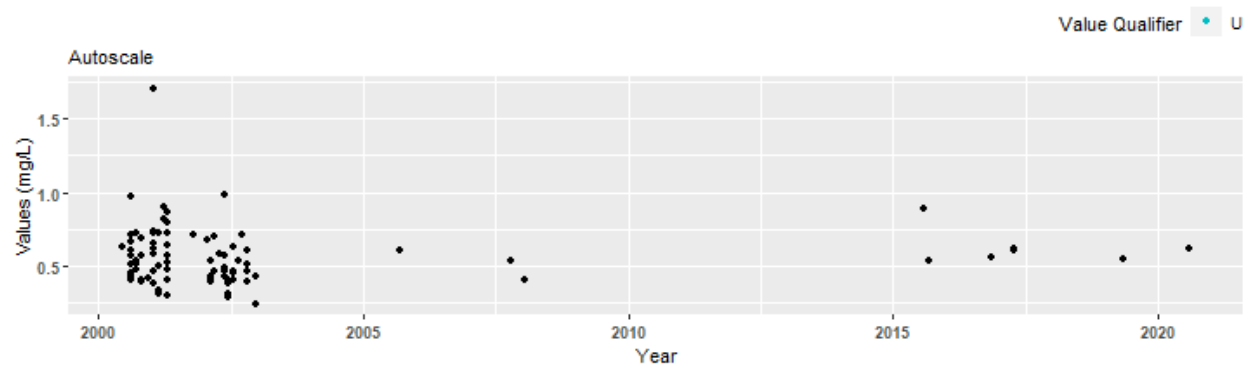


Data Points with Trendlines for Apalachicola National Estuarine Research Reserve

Senn Slope = 0.0013666666666667, Senn Intercept = 6.30816666666667

Trend = 0, tau = -0.0173, p = 0.8334

Linear Trendline: $y = 0.00127920298309486x + -1.99585171762746$

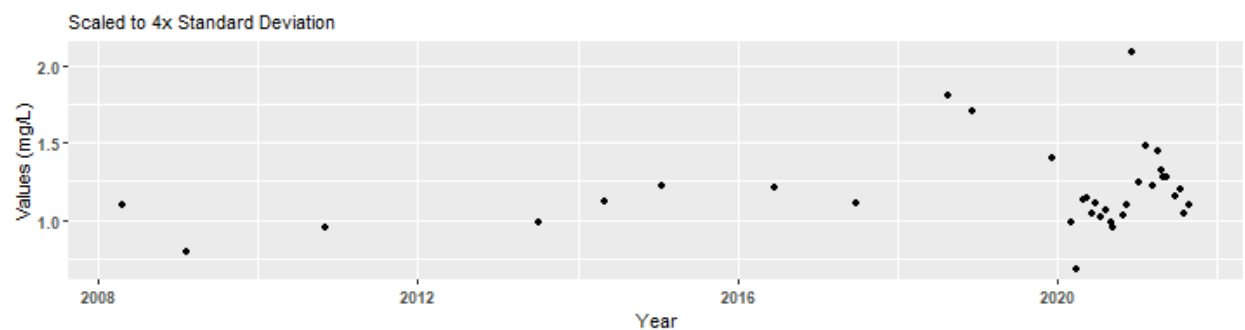
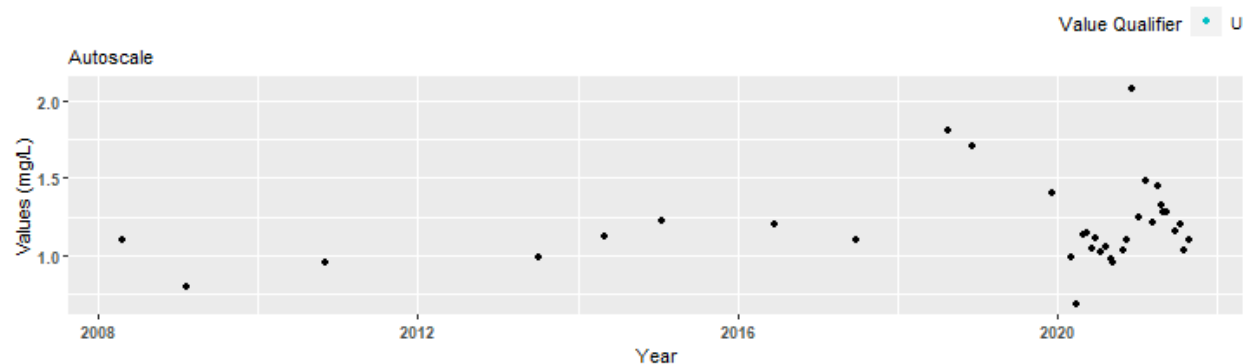


Data Points with Trendlines for Banana River Aquatic Preserve

Senn Slope = 0.0169769230769231, Senn Intercept = -47.6185964285715

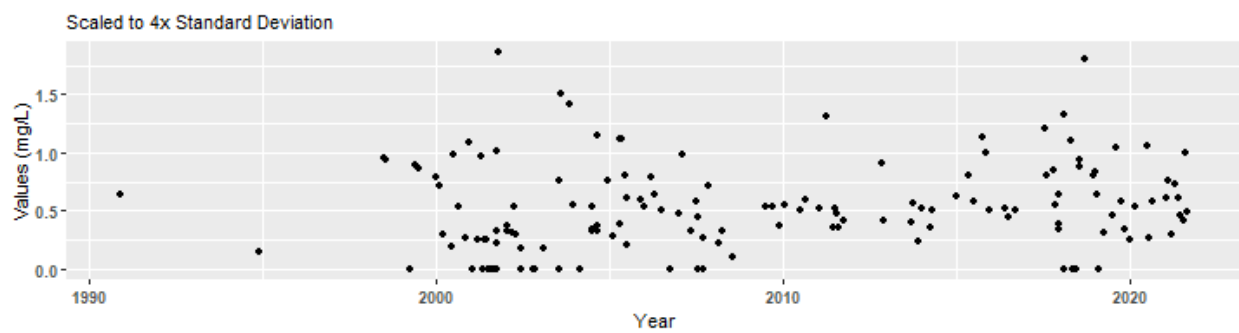
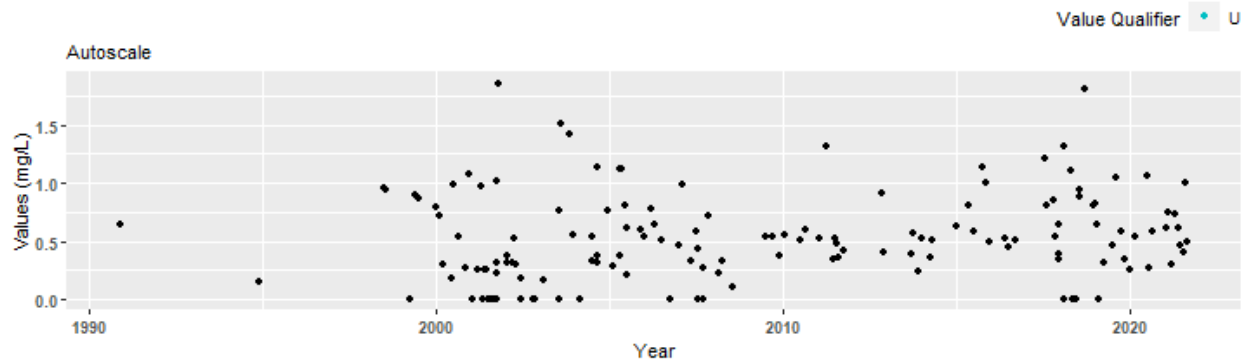
Trend = 0, tau = 0.5052, p = 0.0515

Linear Trendline: $y = 0.0180993213980443x + -35.3576051083604$



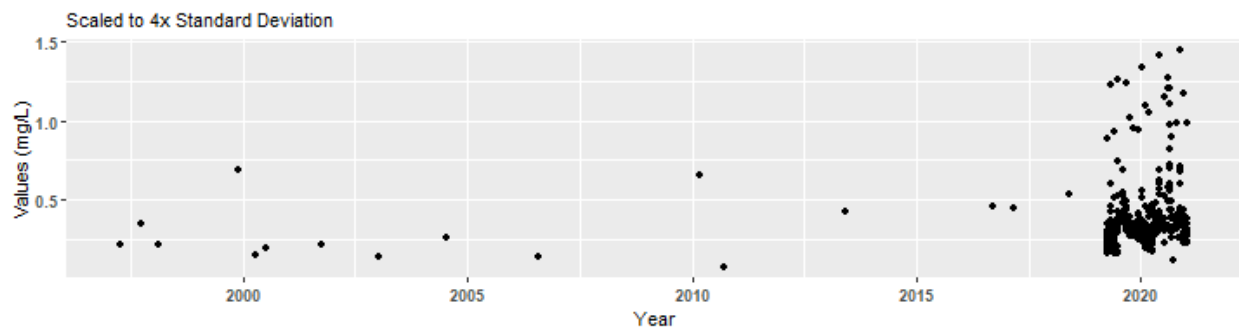
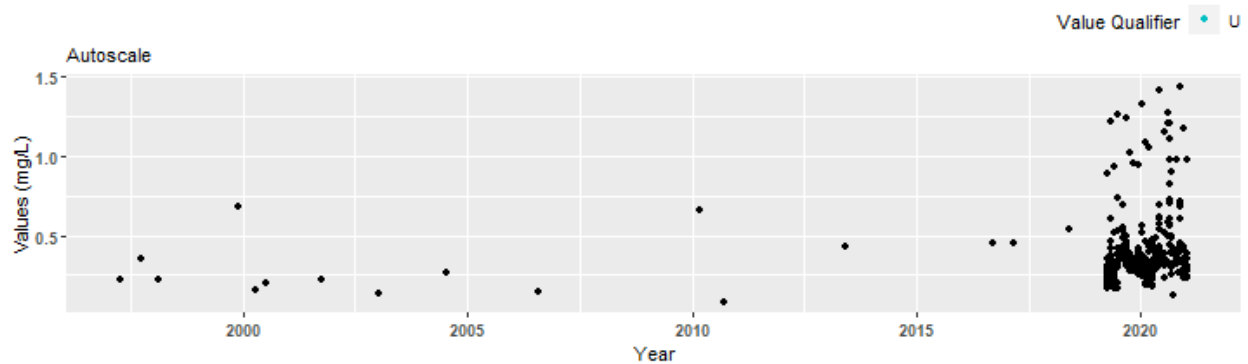
Data Points with Trendlines for Big Bend Seagrasses Aquatic Preserve

Senn Slope = 0.006999999999999999, Senn Intercept = -19.3246295454545
Trend = 0, tau = 0.1184, p = 0.1069
Linear Trendline: $y = 0.00849527568383328x - 16.5427160958967$



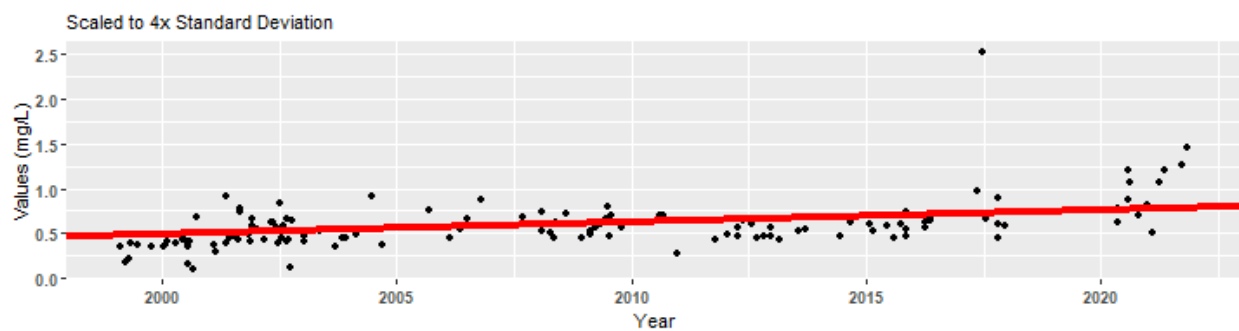
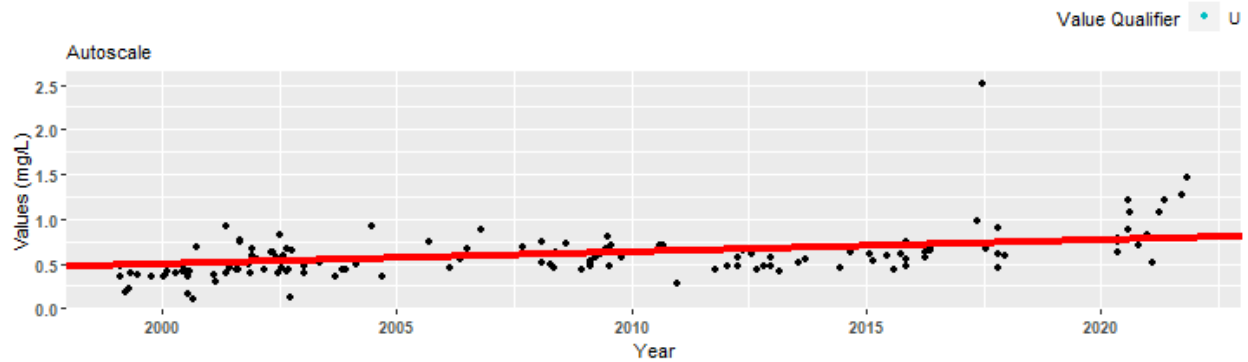
Data Points with Trendlines for Biscayne Bay Aquatic Preserve

Senn Slope = 0.019, Senn Intercept = -16.8255842105263
Trend = 1, tau = 0.1301, p = 0
Linear Trendline: $y = 0.00809814238033476x - 15.9510241849654$



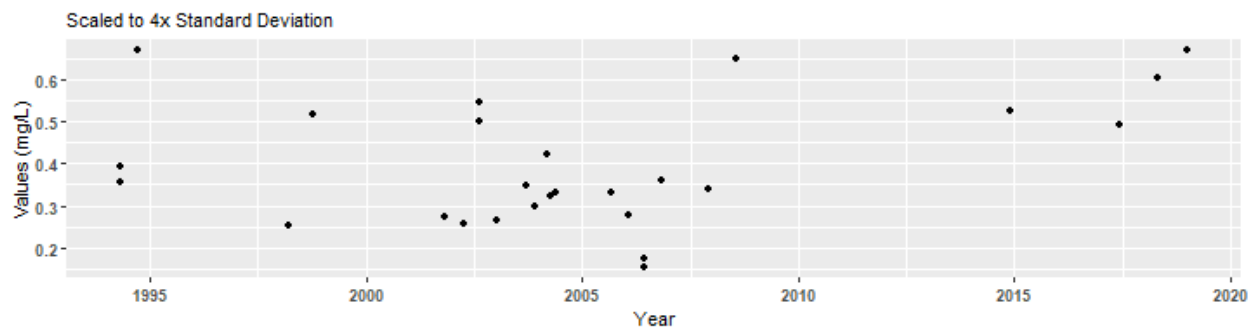
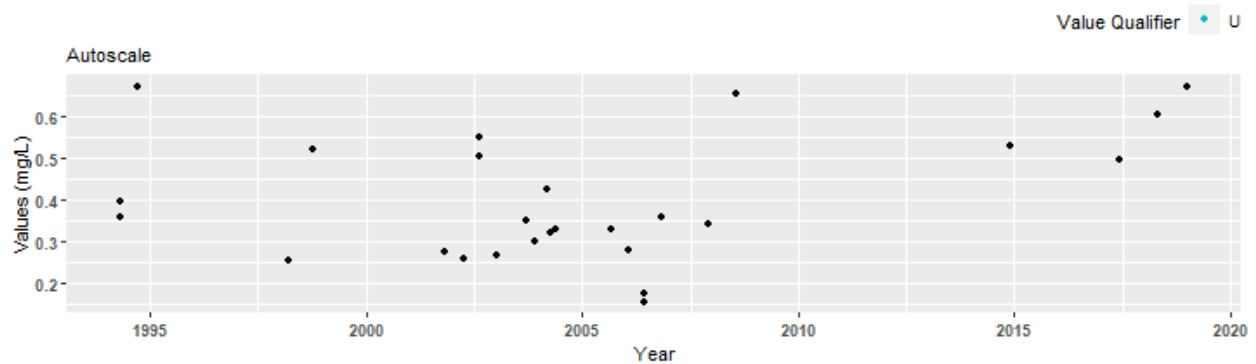
Data Points with Trendlines for Boca Ciega Bay Aquatic Preserve

Senn Slope = 0.0133333333333333, Senn Intercept = -26.1592857142857
Trend = 1, tau = 0.4543, p = 0
Linear Trendline: $y = 0.0202861411216817x + -40.1600969529729$



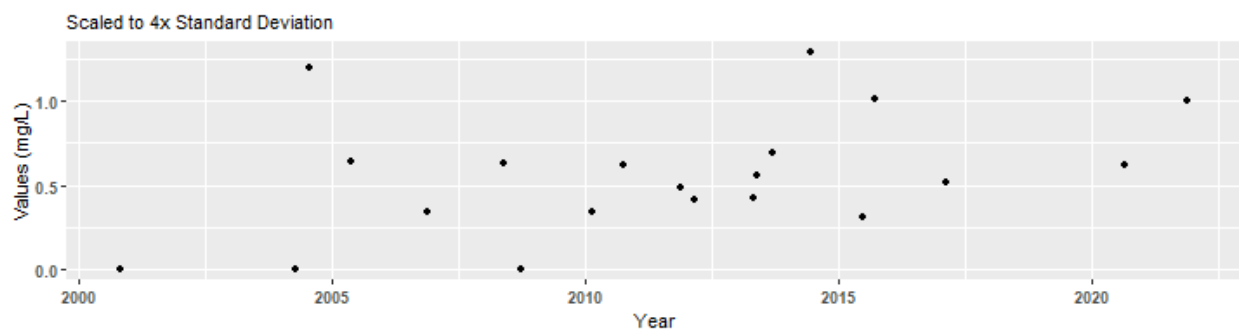
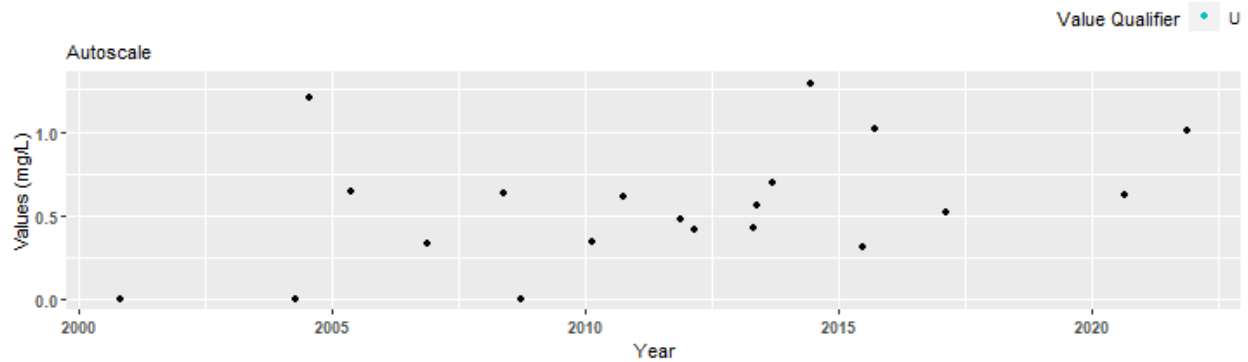
Data Points with Trendlines for Cape Romano-Ten Thousand Islands Aquatic Preserve

Senn Slope = 0.010375, Senn Intercept = -40.0934850649351
Trend = 0, tau = 0.2273, p = 1
Linear Trendline: $y = 0.00668510827089997x + -13.0059080585713$



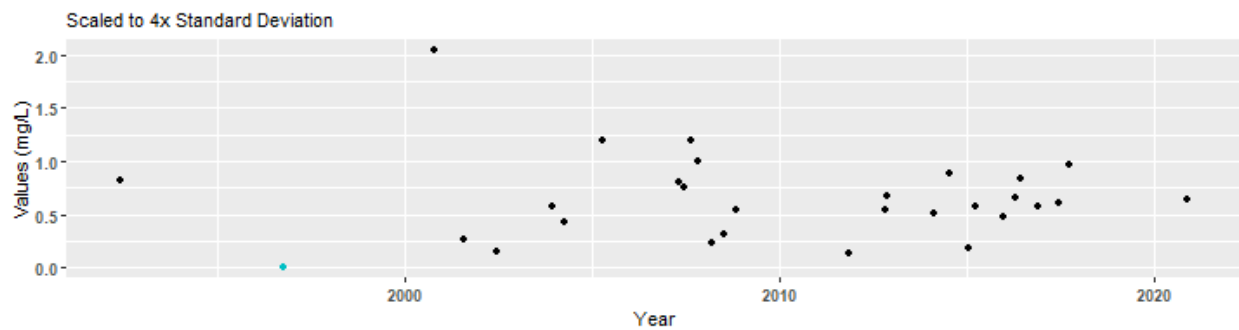
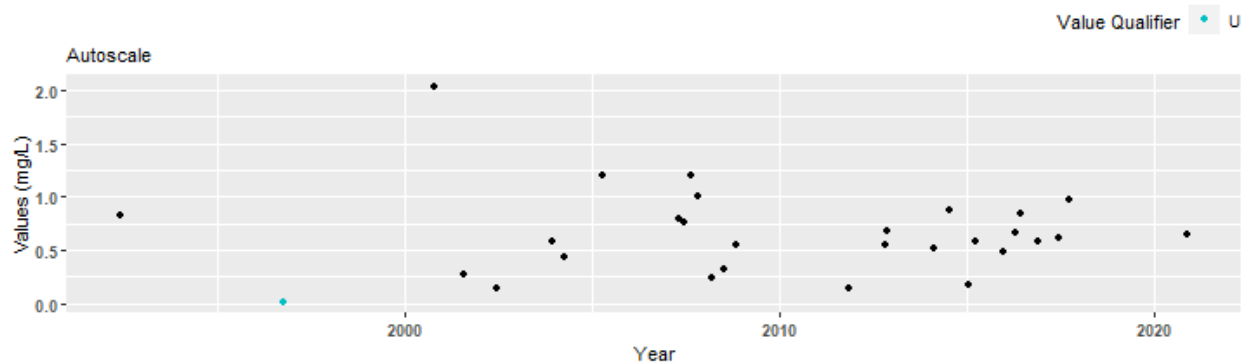
Data Points with Trendlines for Cockroach Bay Aquatic Preserve

Senn Slope = 0.036, Senn Intercept = -69.1166571428572
Trend = 0, tau = 0.4118, p = 0.0857
Linear Trendline: $y = 0.0273659757490388x + -54.4912893356316$



Data Points with Trendlines for Estero Bay Aquatic Preserve

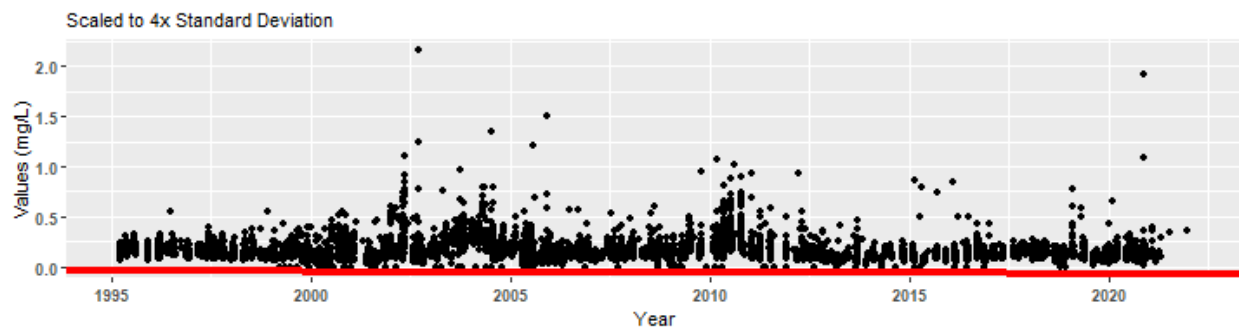
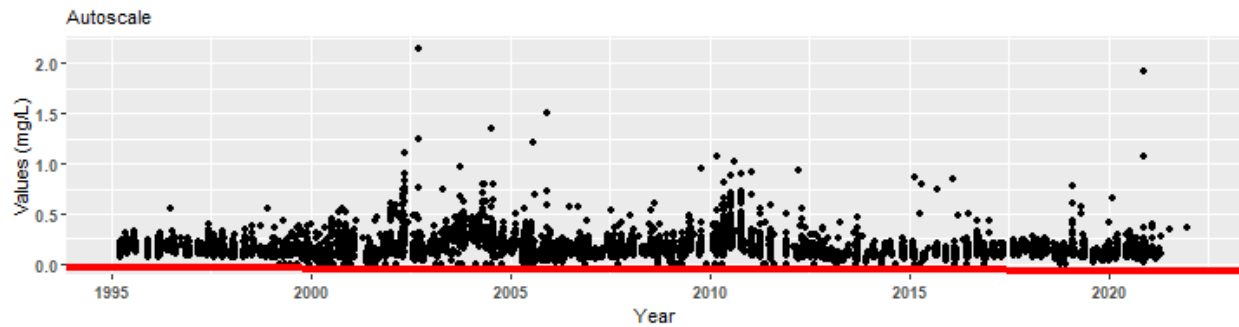
Senn Slope = 0, Senn Intercept = -33.6022943181818
Trend = 0, tau = 0.0133, p = 1
Linear Trendline: $y = -0.00317085685569322x + 7.01611786428982$



Data Points with Trendlines for Florida Keys National Marine Sanctuary

Senn Slope = -0.000923836587301587, Senn Intercept = 1.8081684785
Trend = -1, tau = -0.0733, p = 0
Linear Trendline: $y = -0.00149092998909285x + 3.1467672532726$

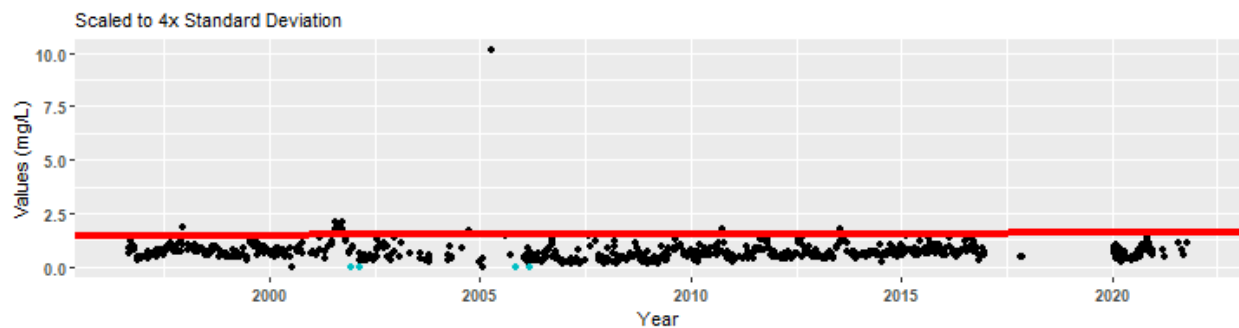
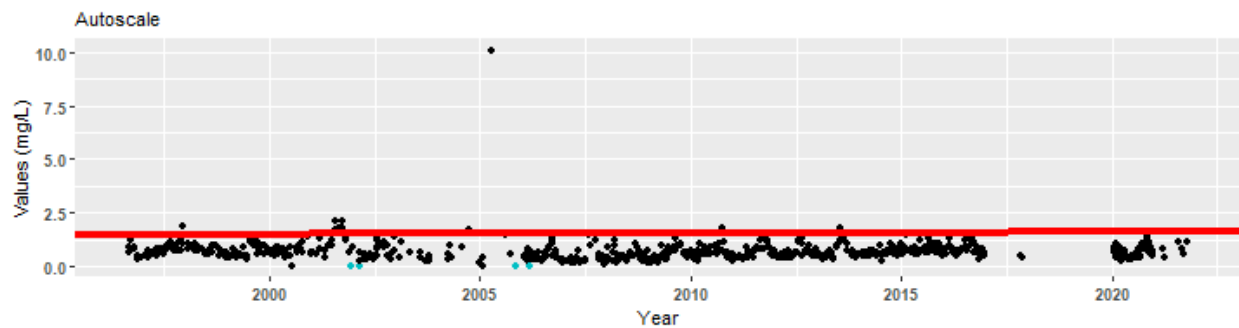
Value Qualifier • U



Data Points with Trendlines for Gasparilla Sound-Charlotte Harbor Aquatic Preserve

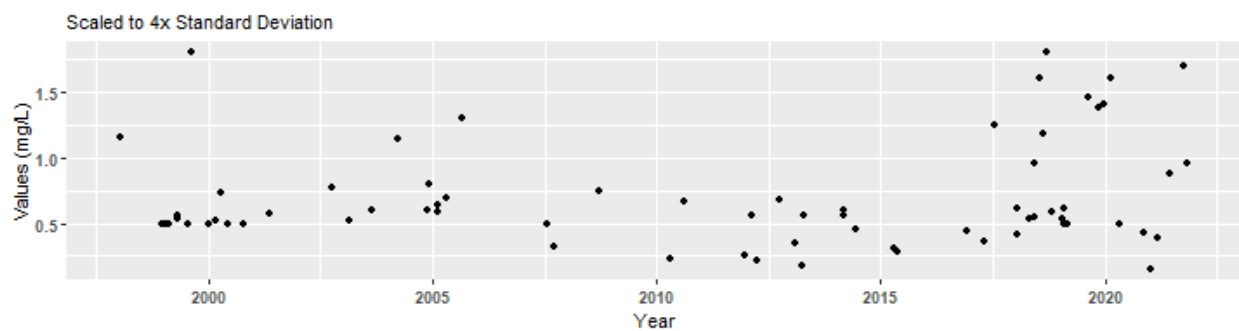
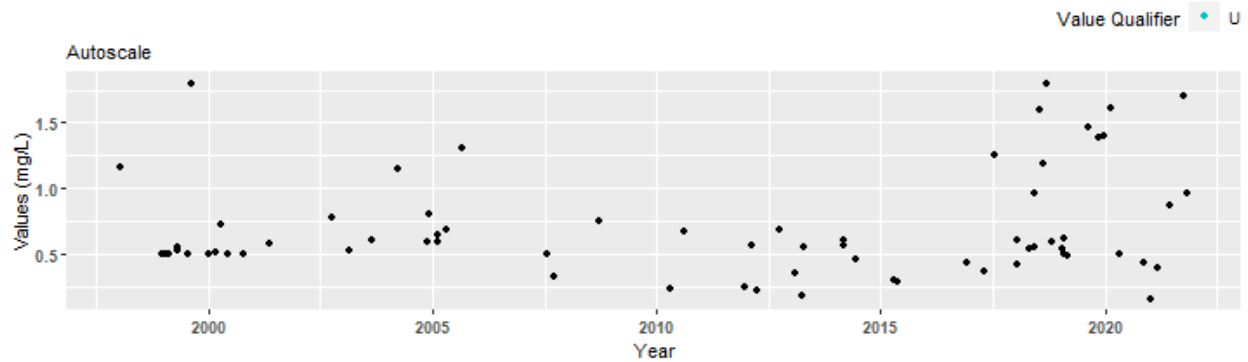
Senn Slope = 0.004749999999999998, Senn Intercept = -7.96653312969926
Trend = 1, tau = 0.0775, p = 0.004
Linear Trendline: $y = -0.000779337972028821x + 2.27155584376205$

Value Qualifier • U



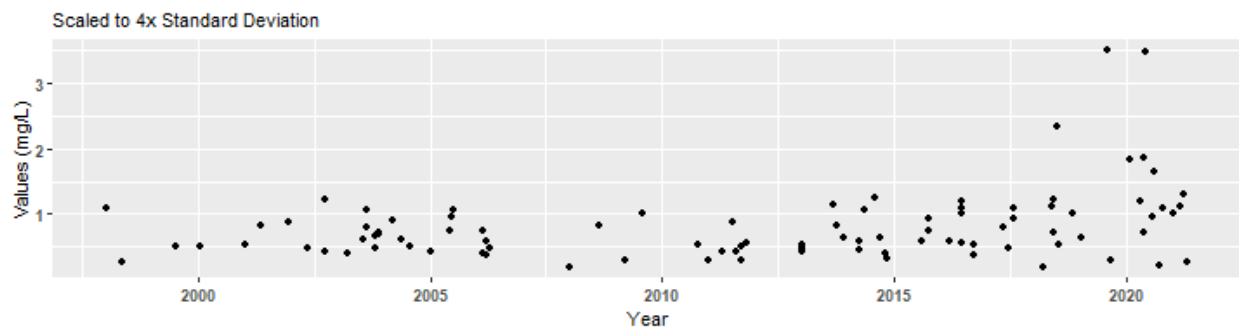
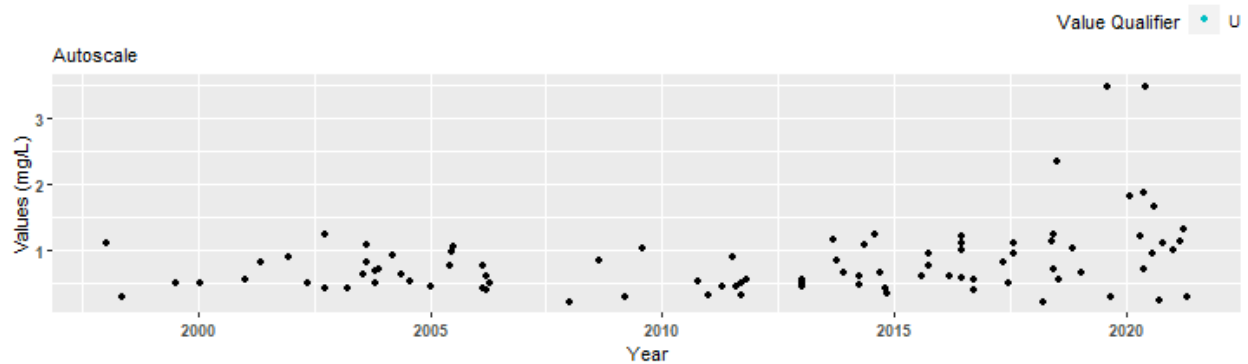
Data Points with Trendlines for Guana River Marsh Aquatic Preserve

Senn Slope = 0.002383333333333334, Senn Intercept = -13.2301704634979
Trend = 0, tau = 0.122, p = 0.4169
Linear Trendline: $y = 0.0077104188696597x + -14.8051006163082$



Data Points with Trendlines for Guana Tolomato Matanzas National Estuarine Research Reserve

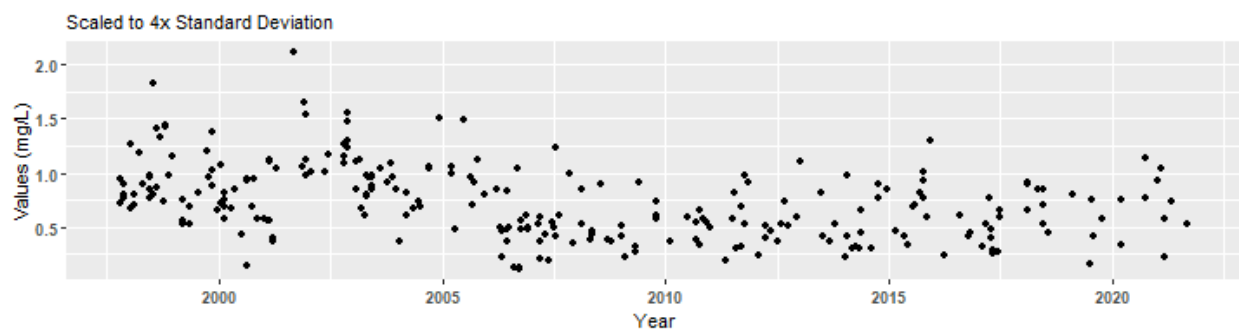
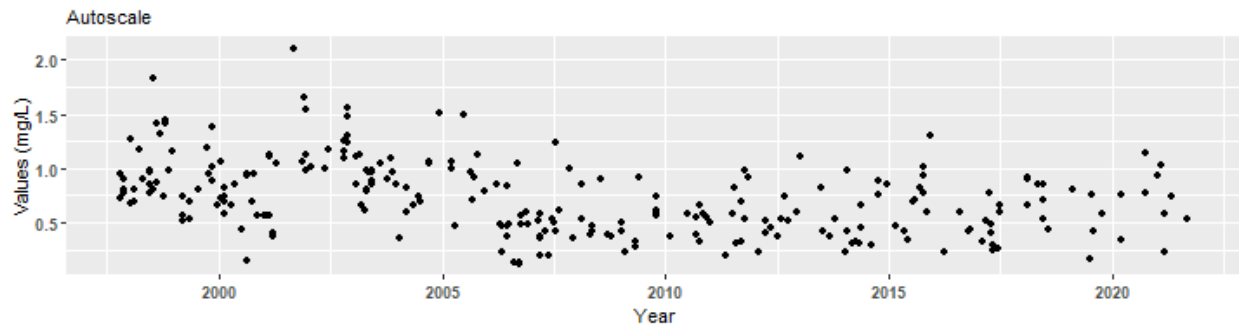
Senn Slope = 0.0193588235294118, Senn Intercept = -46.6856090909091
Trend = 1, tau = 0.196, p = 0.0103
Linear Trendline: $y = 0.0267234322872083x + -52.9655946833325$



Data Points with Trendlines for Indian River-Malabar to Vero Beach Aquatic Preserve

Senn Slope = -0.0177774725274725, Senn Intercept = 35.44453875
 Trend = -1, tau = -0.2612, p = 0
 Linear Trendline: $y = -0.0198303069173957x + 40.5425048754526$

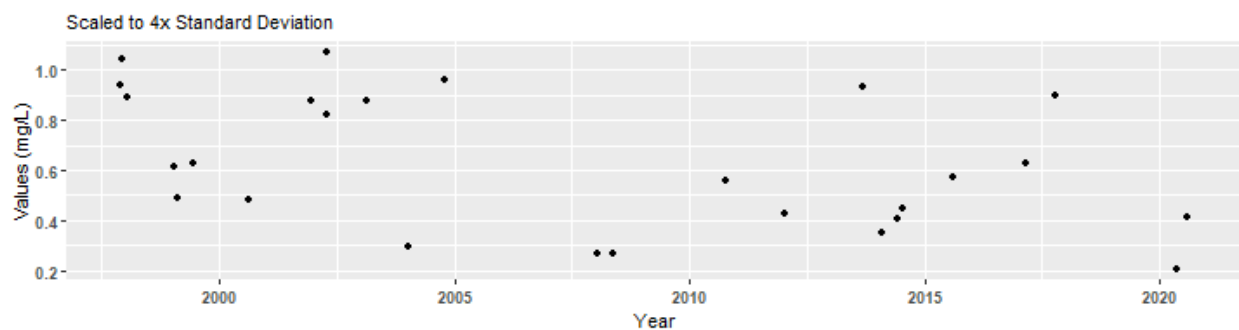
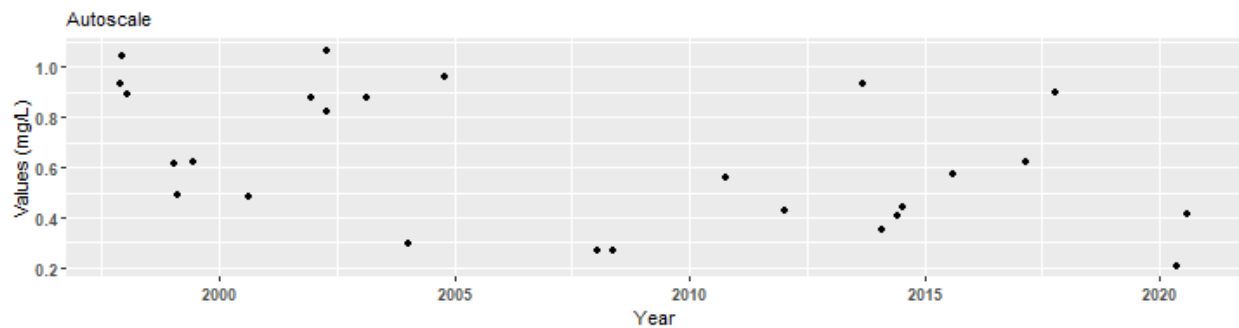
Value Qualifier • U



Data Points with Trendlines for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

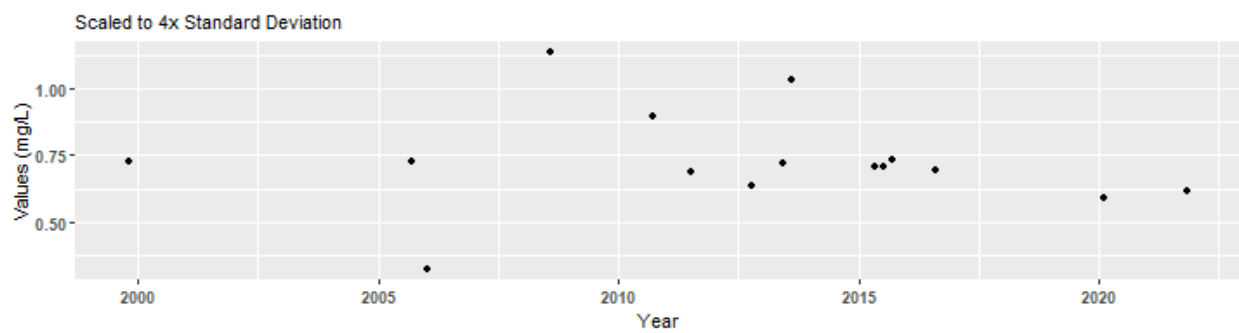
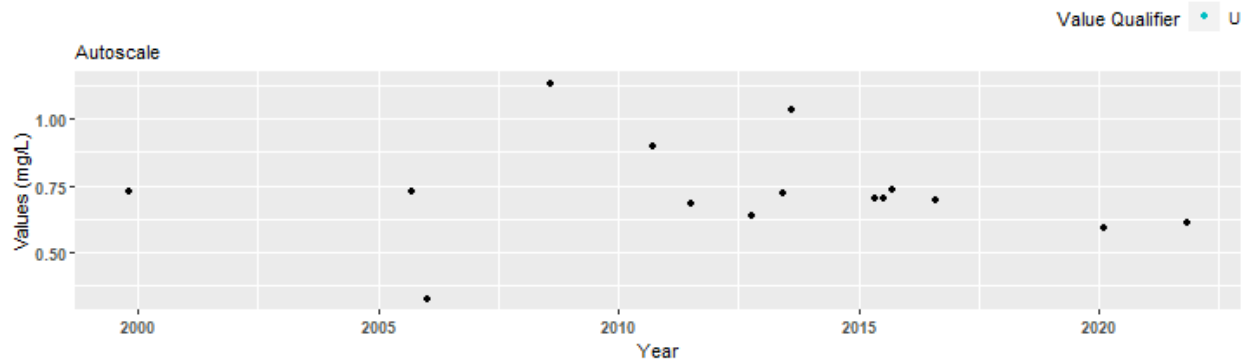
Senn Slope = -0.0117311282051282, Senn Intercept = 10.0011461538462
 Trend = 0, tau = -0.4, p = 0.1408
 Linear Trendline: $y = -0.0157201513018425x + 32.1887697949138$

Value Qualifier • U



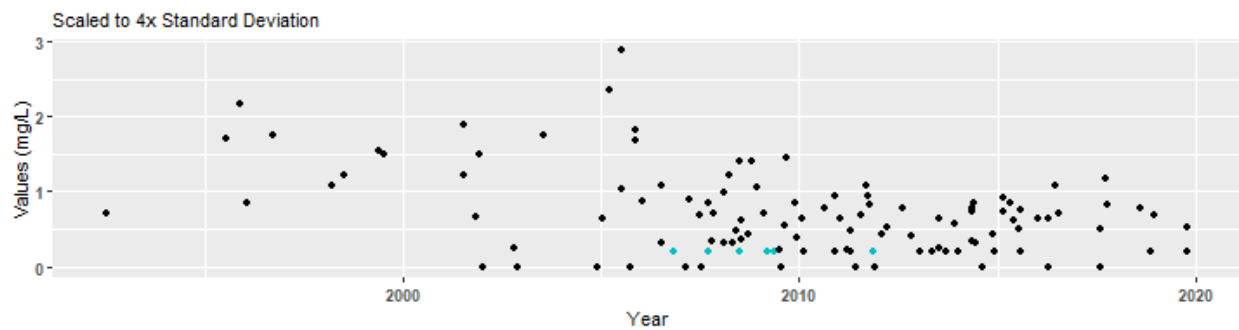
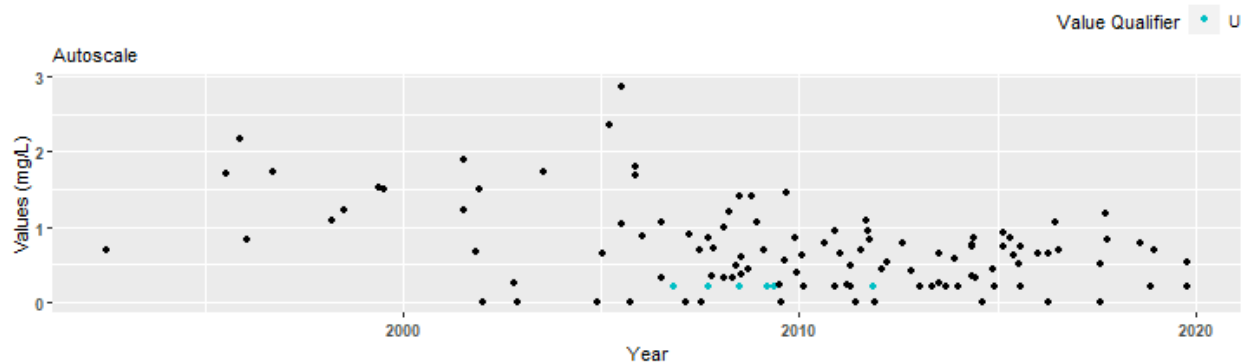
Data Points with Trendlines for Lemon Bay Aquatic Preserve

Senn Slope = -0.0134615384615385, Senn Intercept = 6.64461538461538
Trend = 0, tau = -0.2, p = 0.7434
Linear Trendline: $y = -0.00267778158958581x + 6.11725389866826$



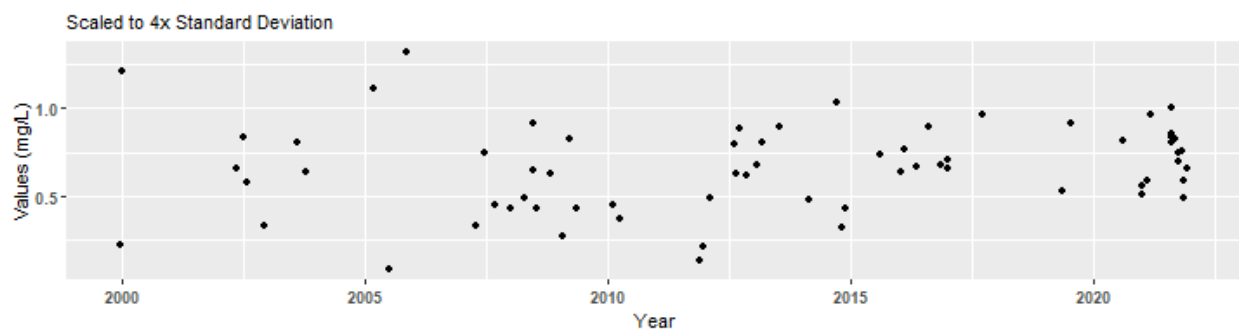
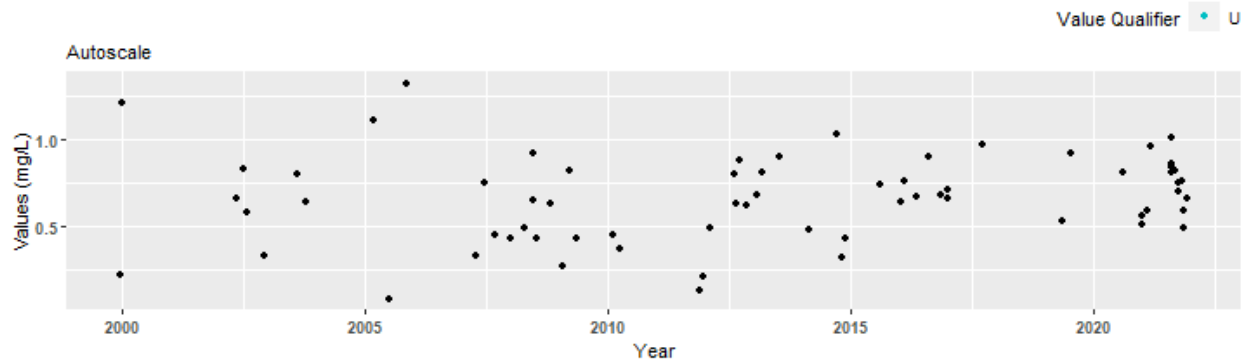
Data Points with Trendlines for Loxahatchee River-Lake Worth Creek Aquatic Preserve

Senn Slope = -0.0373529411764706, Senn Intercept = 10.57876825
Trend = -1, tau = -0.1239, p = 0.0127
Linear Trendline: $y = -0.0418473050604718x + 84.7989130088188$



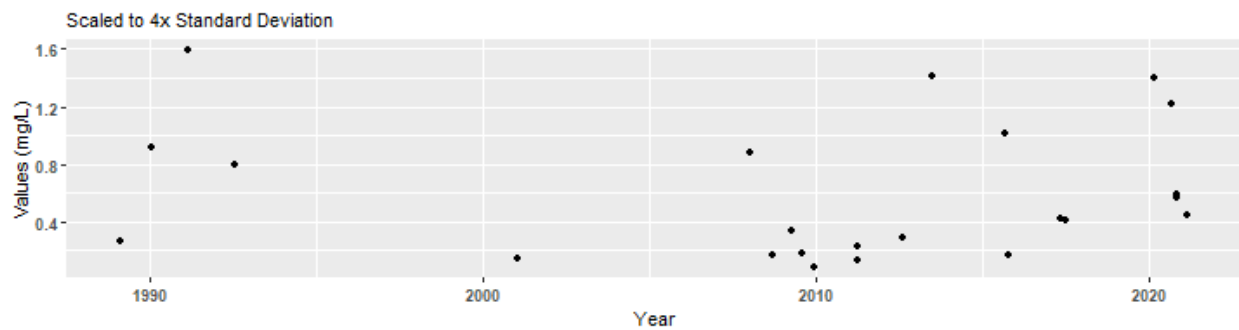
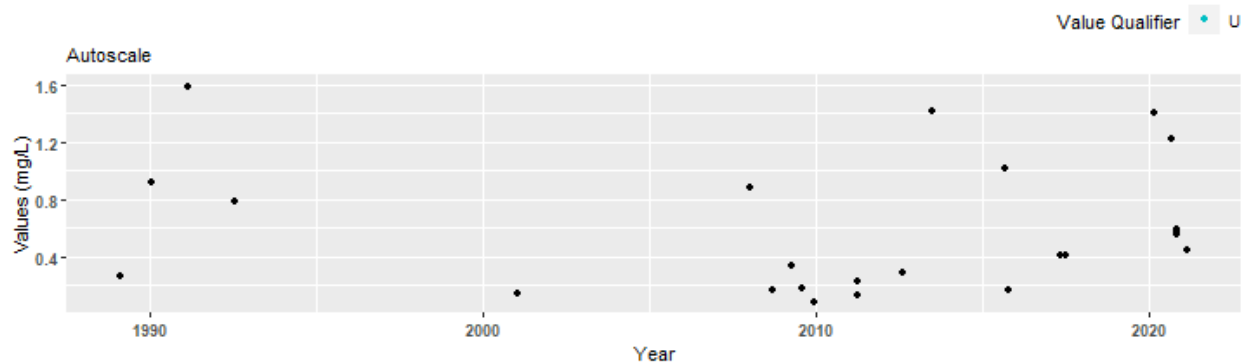
Data Points with Trendlines for Matlacha Pass Aquatic Preserve

Senn Slope = 0.00855714285714286, Senn Intercept = -26.0038771929825
Trend = 1, tau = 0.2164, p = 0.0372
Linear Trendline: $y = 0.00562464703326074x - 10.6701759560136$



Data Points with Trendlines for Nassau River-St. Johns River Marshes Aquatic Preserve

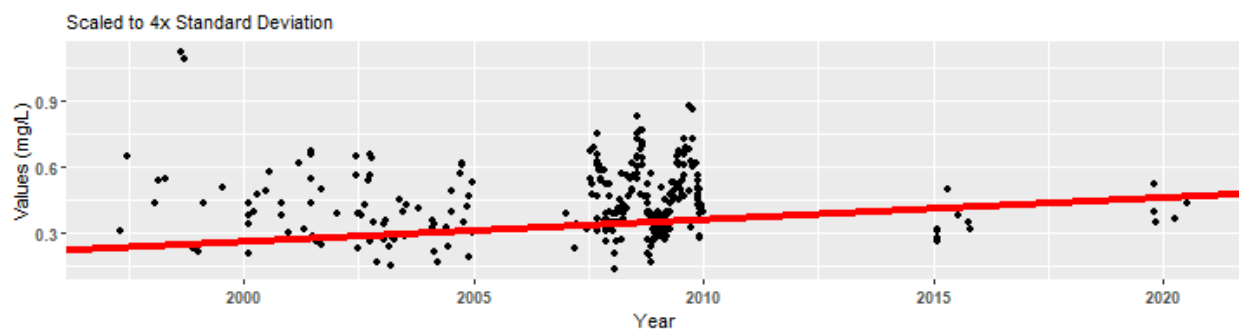
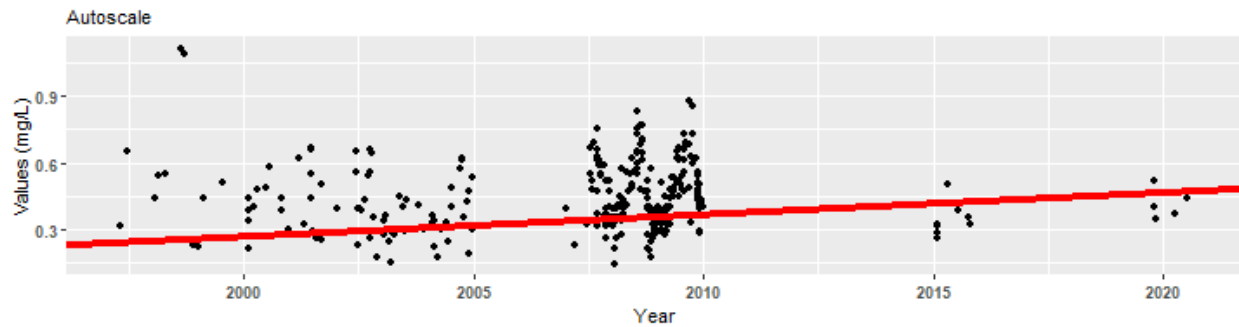
Senn Slope = 0.0330768049155146, Senn Intercept = -11.03975
Trend = 0, tau = 0.0667, p = 0.5403
Linear Trendline: $y = -0.00263244292387478x + 5.88488161967016$



Data Points with Trendlines for Nature Coast Aquatic Preserve

Senn Slope = 0.01, Senn Intercept = -19.735
Trend = 1, tau = 0.1413, p = 0.0001
Linear Trendline: $y = 0.00066264116606371x + -0.880976778511838$

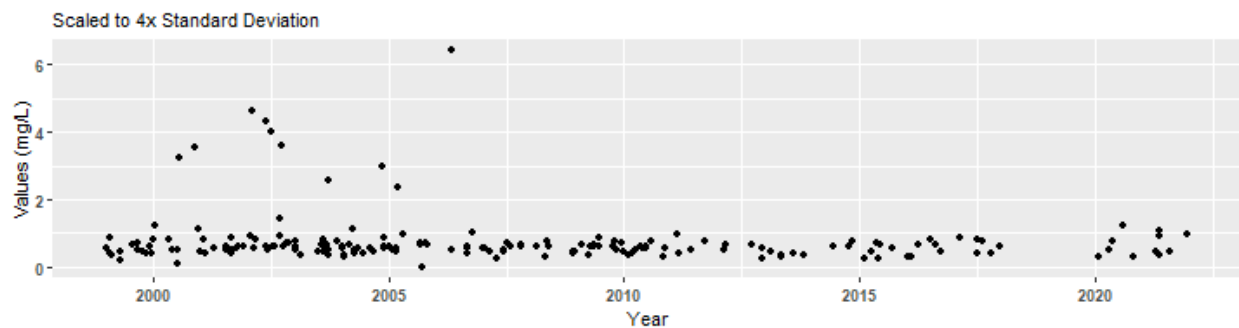
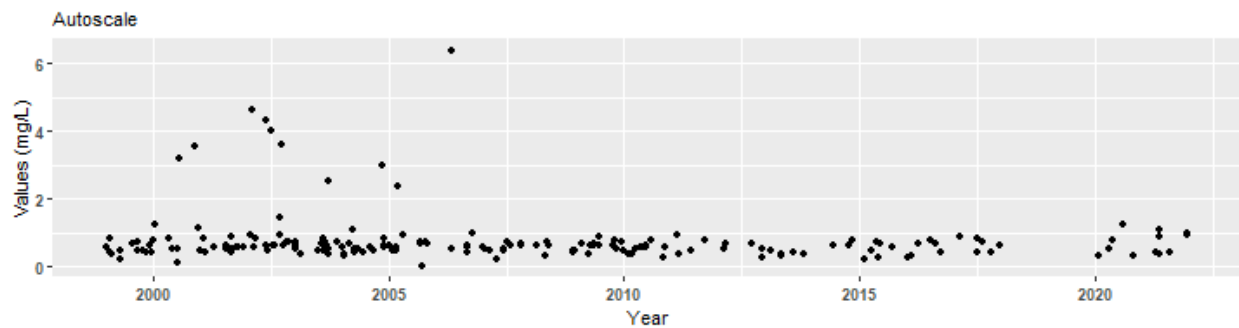
Value Qualifier • U



Data Points with Trendlines for Pinellas County Aquatic Preserve

Senn Slope = -0.0033333333333334, Senn Intercept = 0.478035714285714
Trend = 0, tau = -0.0429, p = 0.3357
Linear Trendline: $y = -0.0198058161583817x + 40.5136143453975$

Value Qualifier • U



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 `Use_In_Analysis` 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 + 5 time the standard deviation
3. Y-axis set to be mean + 5 time 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) {
    year_lower <- min(data$Year[data$Use_In_Analysis == TRUE &
                               data$ManagedAreaName == MA_names[i]])
    year_upper <- max(data$Year[data$Use_In_Analysis == TRUE &
                               data$ManagedAreaName == MA_names[i]])
    min_RV <- min(data$ResultValue[data$Use_In_Analysis == TRUE &
                                     data$ManagedAreaName == MA_names[i]])
    mn_RV <- mean(data$ResultValue[data$Use_In_Analysis == TRUE &
                                     data$ManagedAreaName == MA_names[i] &
                                     data$ResultValue <
                                     quantile(data$ResultValue, 0.98)])
    sd_RV <- sd(data$ResultValue[data$Use_In_Analysis == TRUE &
                                   data$ManagedAreaName == MA_names[i] &
                                   data$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 = data[data$Use_In_Analysis == TRUE &
                              data$ManagedAreaName == MA_names[i], ],
                 aes(x = Year, y = ResultValue, group = Year)) +
      geom_boxplot(outlier.size = 0.5) +
      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))) +
theme(axis.text.x = element_text(face = "bold"),
      axis.text.y = element_text(face = "bold"))

p2 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i], ],
            aes(x = Year, y = ResultValue, group = Year)) +
geom_boxplot(outlier.size = 0.5) +
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))) +
theme(axis.text.x = element_text(face = "bold"),
      axis.text.y = element_text(face = "bold"))

p3 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i] &
                        data$Year >= year_upper - 10, ],
            aes(x = Year, y = ResultValue, group = Year)) +
geom_boxplot(outlier.size = 0.5) +
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))) +
theme(axis.text.x = element_text(face = "bold"),
      axis.text.y = element_text(face = "bold"))

Yset <- ggarrange(p1, p2, p3, ncol = 1)

p0 <- ggplot() + labs(title = paste0("Summary Box Plots for ",
                                    MA_names[i]), subtitle = "By Year") +
theme_bw() + theme(plot.title = element_text(face="bold"),
                  panel.border = element_blank(),
                  panel.grid.major = element_blank(),
                  panel.grid.minor = element_blank(), axis.line = element_blank())

## Year & Month Plots
p4 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i], ],
            aes(x = YearMonthDec, y = ResultValue,
                group = YearMonth, color = as.factor(Month))) +
geom_boxplot(outlier.size = 0.5) +
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))) +

```



```

theme(legend.position = "none",
      axis.text.x = element_text(face = "bold"),
      axis.text.y = element_text(face = "bold"))

p5 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i], ],
            aes(x = YearMonthDec, y = ResultValue,
                group = YearMonth, color = as.factor(Month))) +
  geom_boxplot(outlier.size = 0.5) +
  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))) +
  theme(legend.position = "top", legend.box = "horizontal",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold")) +
  guides(color = guide_legend(nrow = 1))

p6 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i], ],
            aes(x = YearMonthDec, y = ResultValue,
                group = YearMonth, color = as.factor(Month)
            )) +
  geom_boxplot(outlier.size = 0.5) +
  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))) +
  theme(legend.position = "none",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

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("Summary Box Plots for ",
                                     MA_names[i]),
                     subtitle = "By Year & Month") + theme_bw() +
  theme(plot.title = element_text(face="bold"),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), axis.line = element_blank())

## Month Plots
p7 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i], ],
            aes(x = Month, y = ResultValue,
                group = Month, fill = as.factor(Month))) +
  geom_boxplot(outlier.size = 0.5) +

```

```

    labs(subtitle = "Autoscale",
          x = "Month", y = paste0("Values (", unit, ")"), fill = "Month") +
    scale_x_continuous(limits = c(0, 13), breaks = seq(3, 12, 3)) +
    theme(legend.position = "none",
          axis.text.x = element_text(face = "bold"),
          axis.text.y = element_text(face = "bold"))

p8 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i], ],
            aes(x = Month, y = ResultValue,
                group = Month, fill = as.factor(Month))) +
  geom_boxplot(outlier.size = 0.5) +
  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)) +
  theme(legend.position = "top", legend.box = "horizontal",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold")) +
  guides(fill = guide_legend(nrow = 1))

p9 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i] &
                        data$Year >= year_upper - 10, ],
            aes(x = Month, y = ResultValue,
                group = Month, fill = as.factor(Month))) +
  geom_boxplot(outlier.size = 0.5) +
  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)) +
  theme(legend.position = "none",
        axis.text.x = element_text(face = "bold"),
        axis.text.y = element_text(face = "bold"))

leg2 <- get_legend(p8)
Mset <- ggarrange(leg2, p7, p8 + theme(legend.position = "none"), p9,
                  ncol = 1, heights = c(0.1, 1, 1, 1))

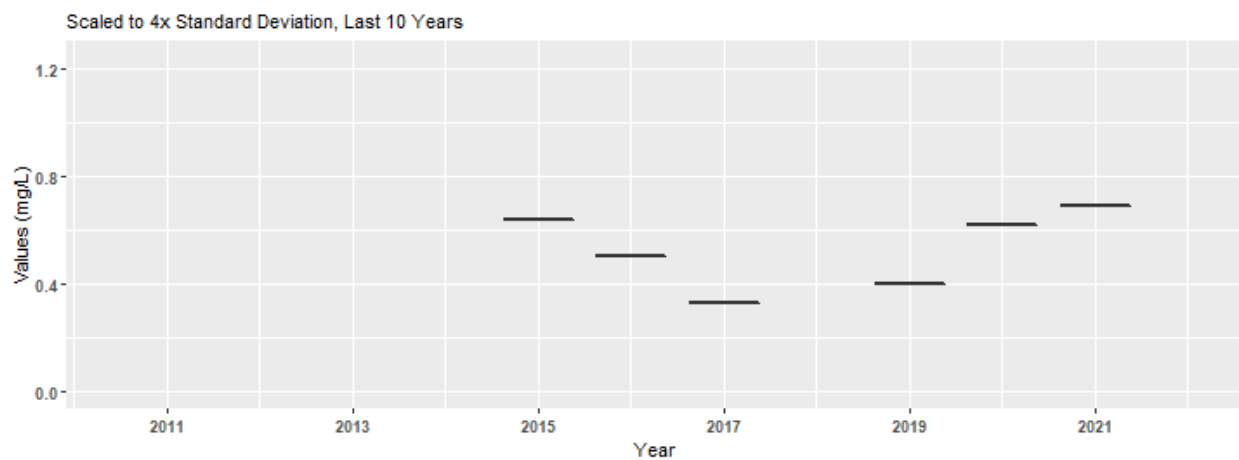
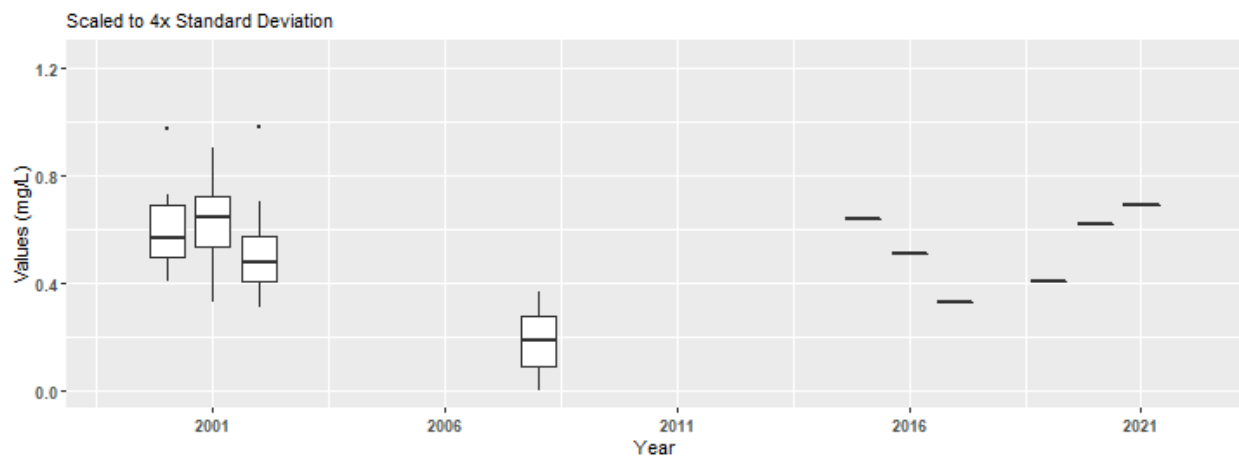
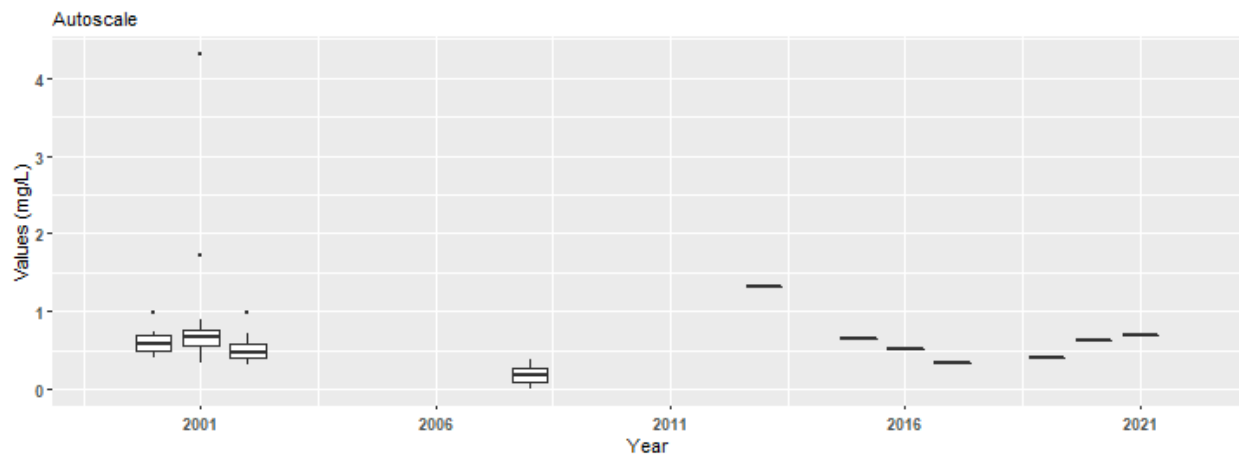
p000 <- ggplot() + labs(title = paste0("Summary Box Plots for ",
                                       MA_names[i]),
                      subtitle = "By Month") + theme_bw() +
  theme(plot.title = element_text(face="bold"),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        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)))
}
}

```

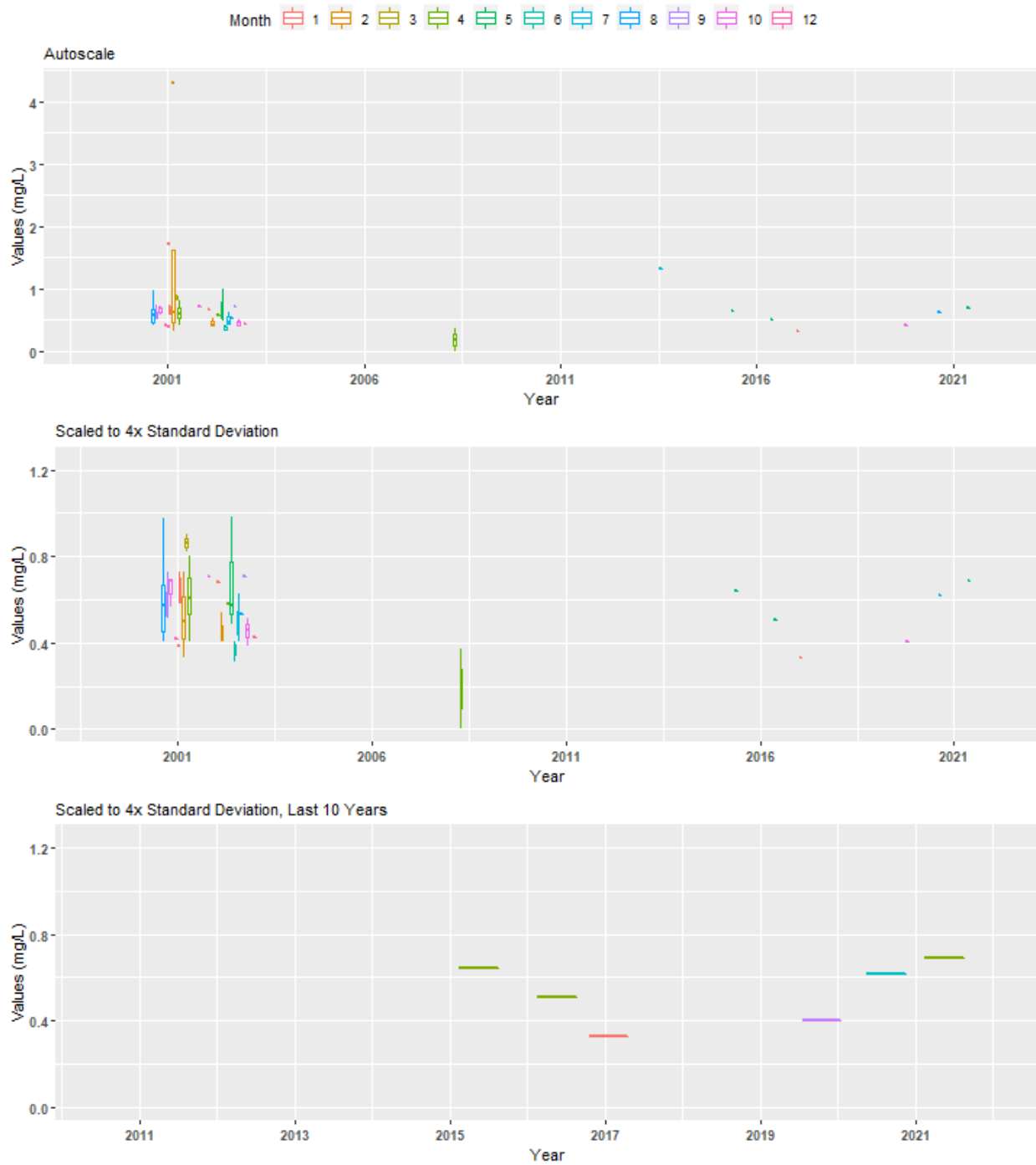
Summary Box Plots for Apalachicola Bay Aquatic Preserve

By Year



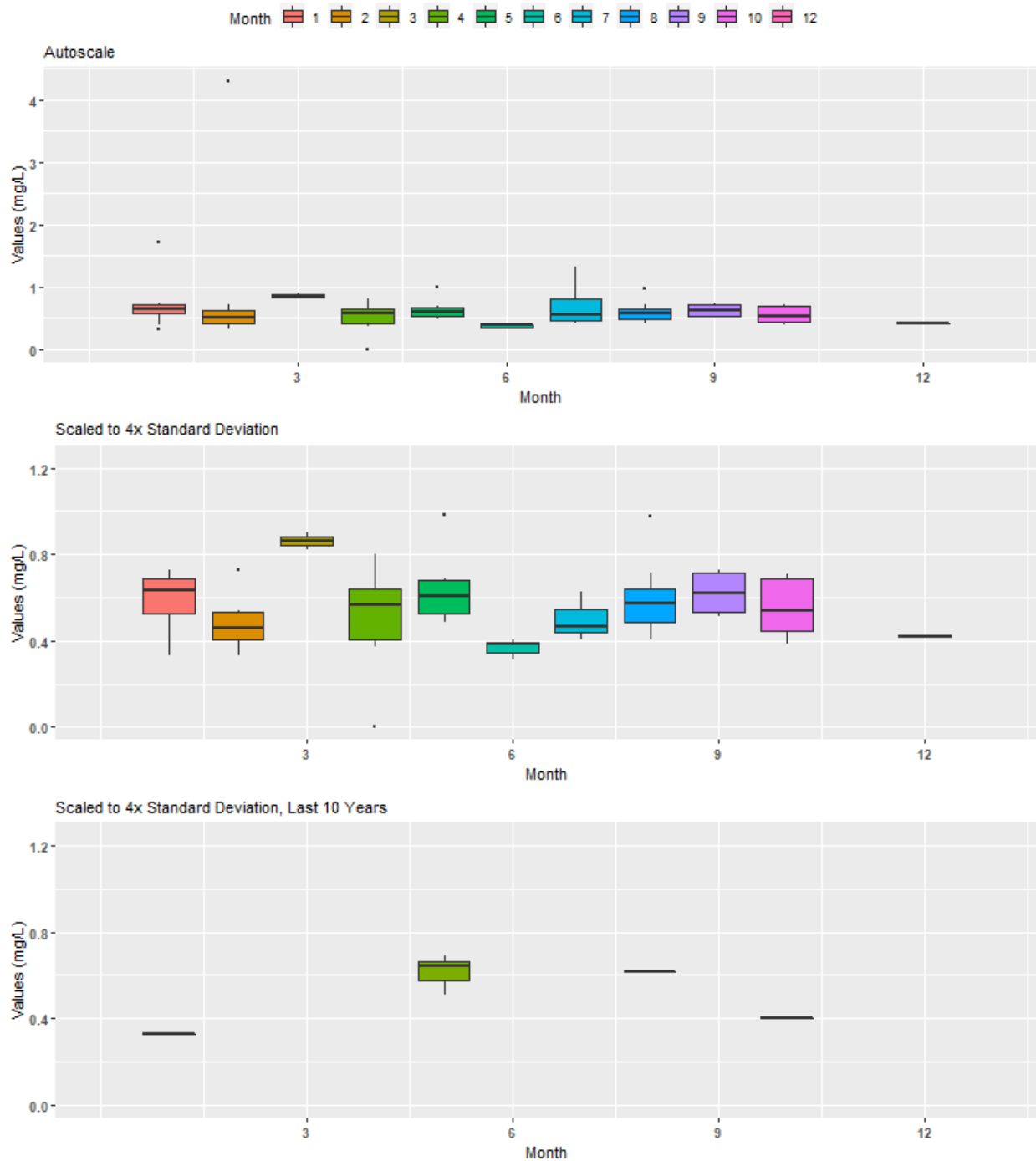
Summary Box Plots for Apalachicola Bay Aquatic Preserve

By Year & Month



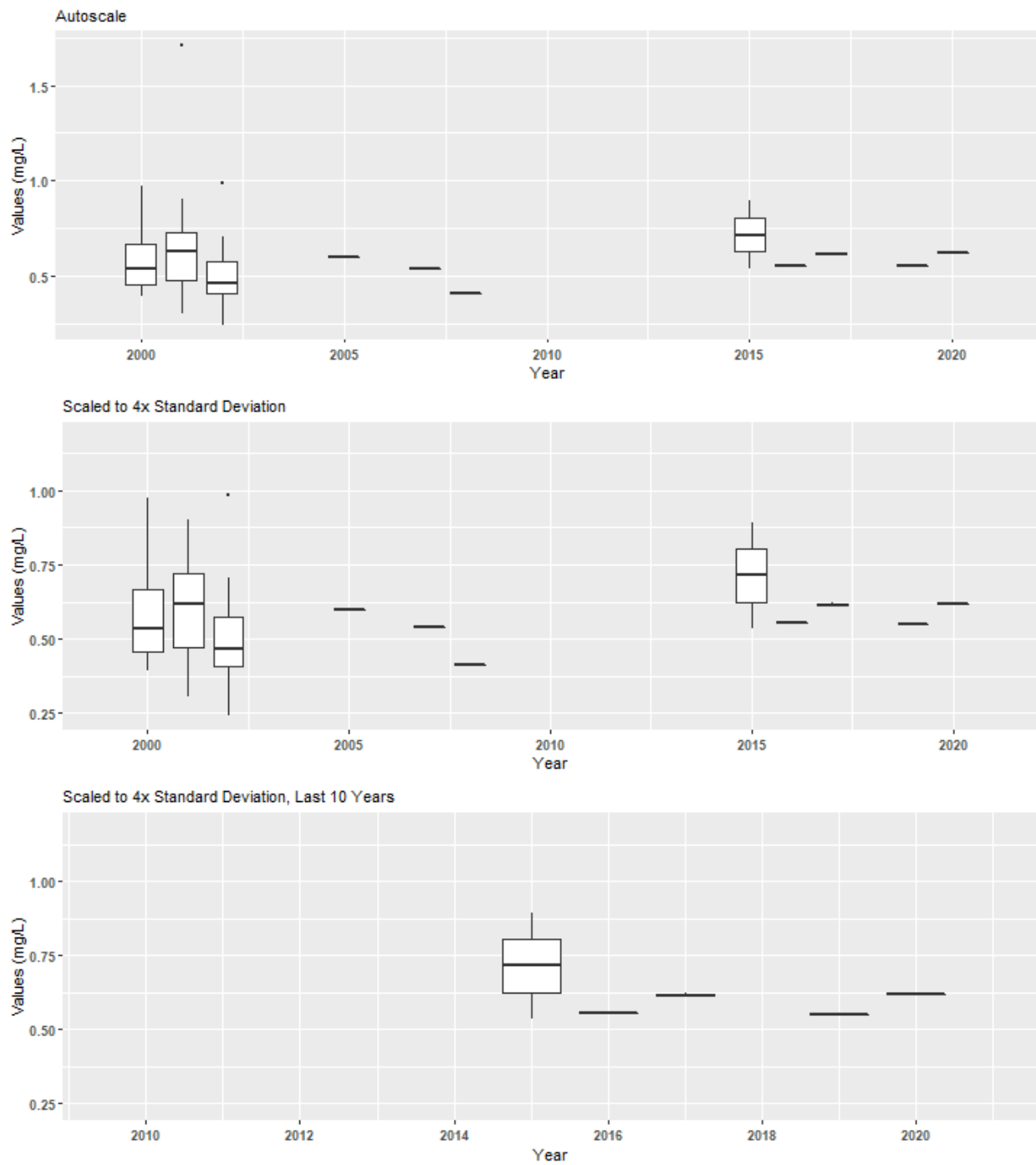
Summary Box Plots for Apalachicola Bay Aquatic Preserve

By Month



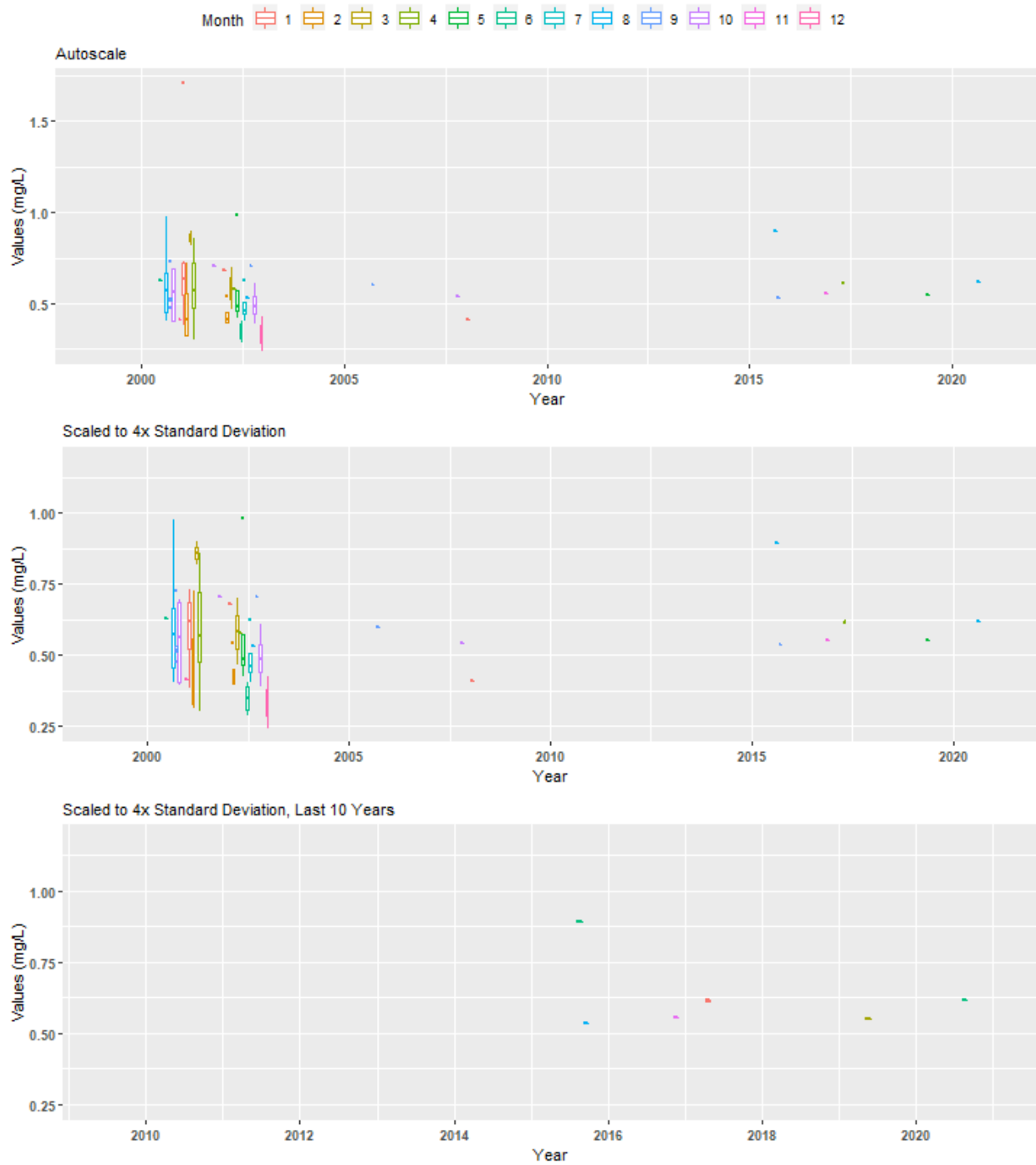
Summary Box Plots for Apalachicola National Estuarine Research Reserve

By Year



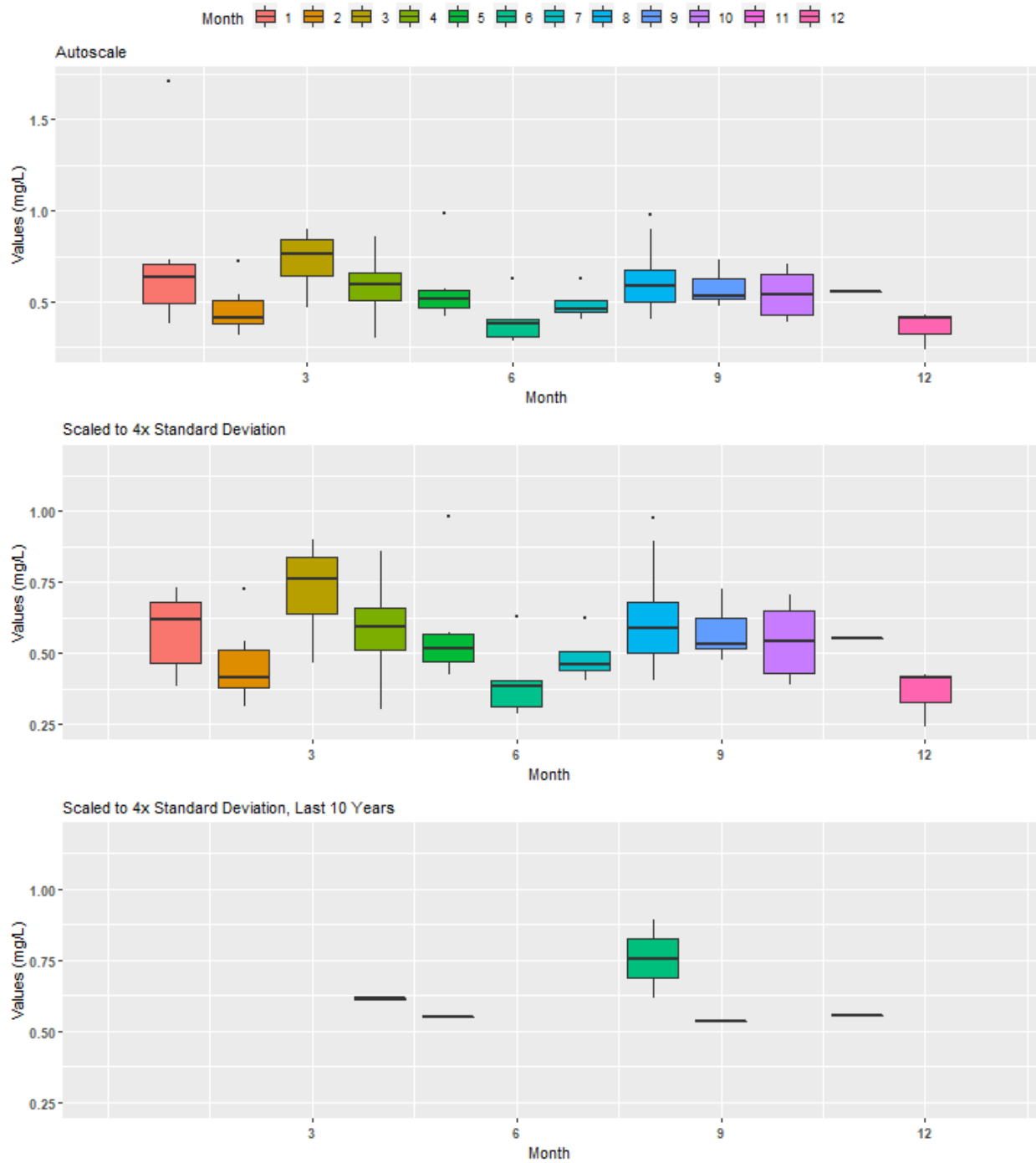
Summary Box Plots for Apalachicola National Estuarine Research Reserve

By Year & Month



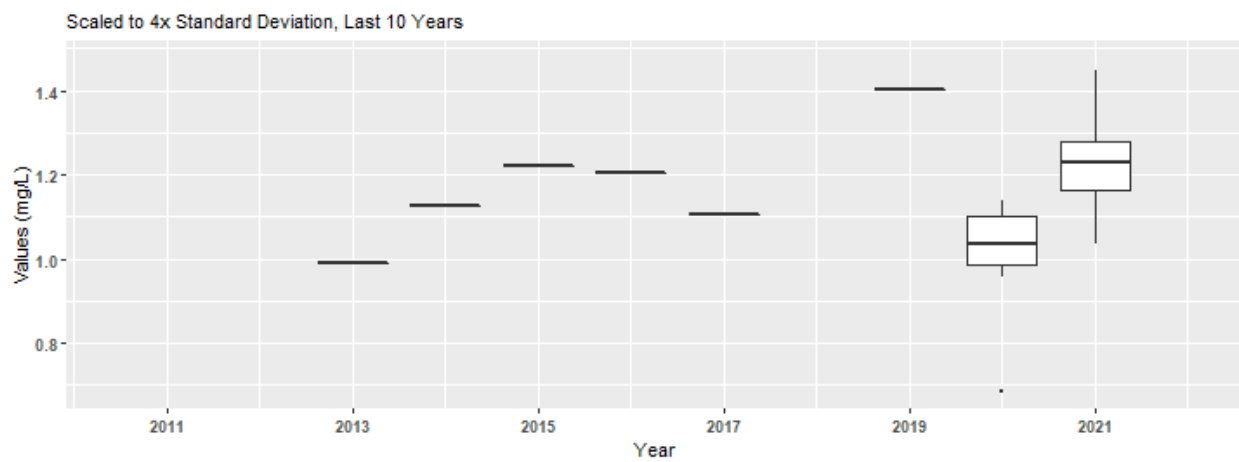
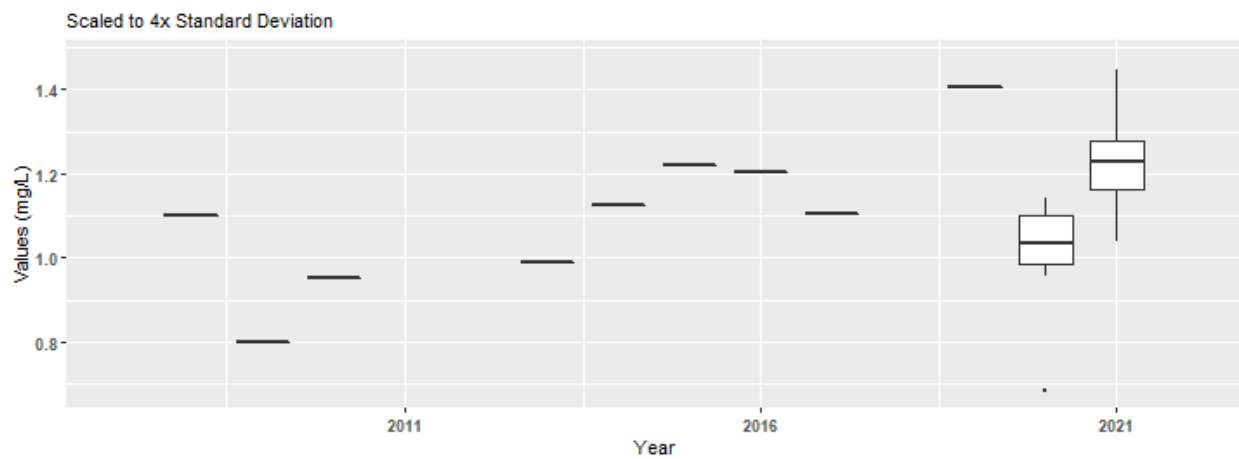
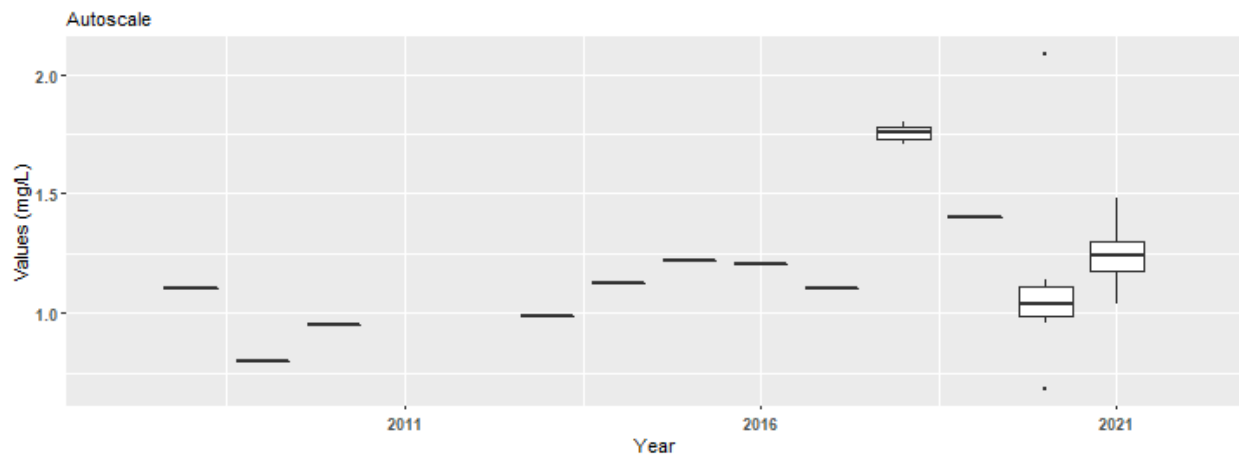
Summary Box Plots for Apalachicola National Estuarine Research Reserve

By Month



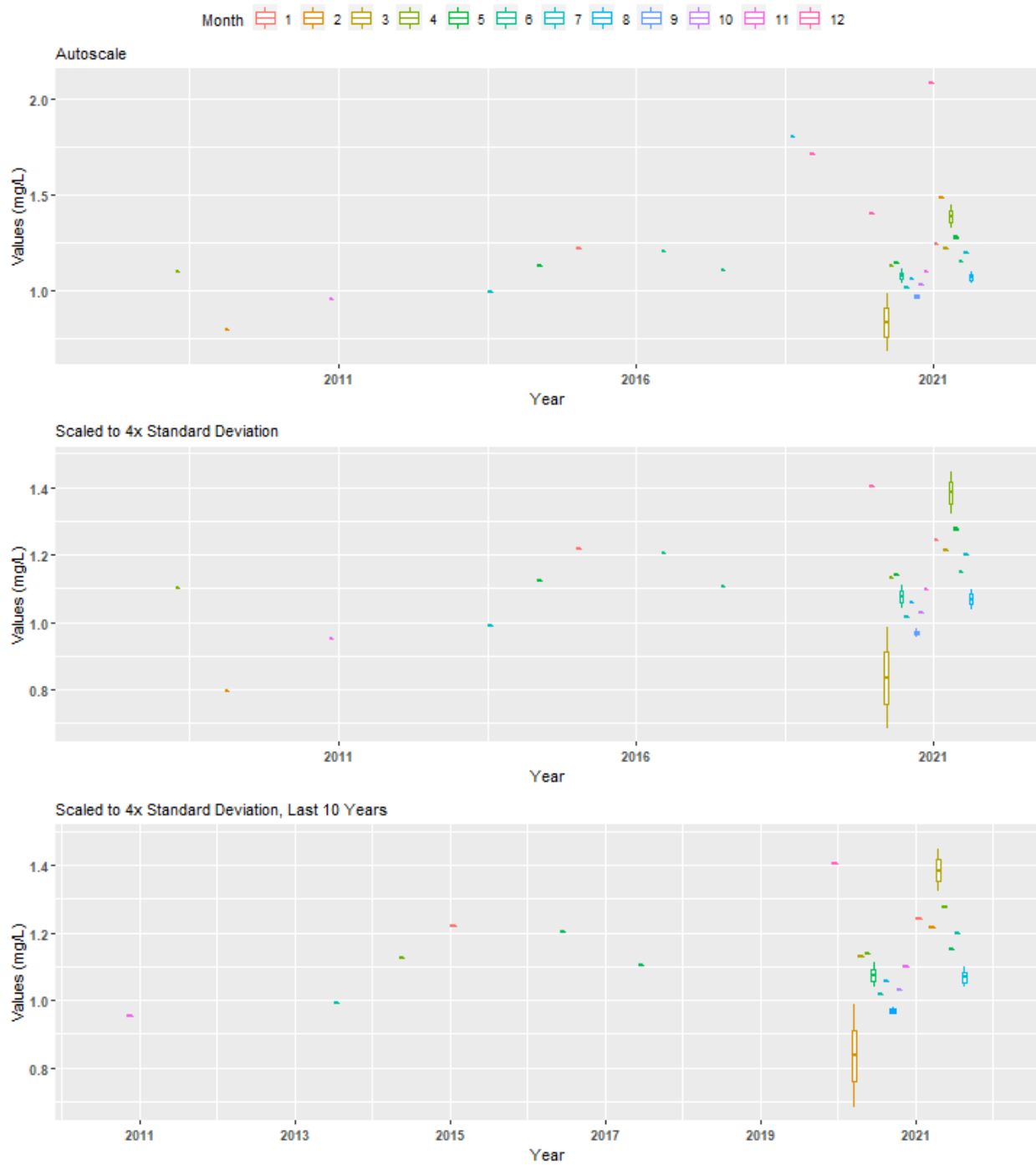
Summary Box Plots for Banana River Aquatic Preserve

By Year



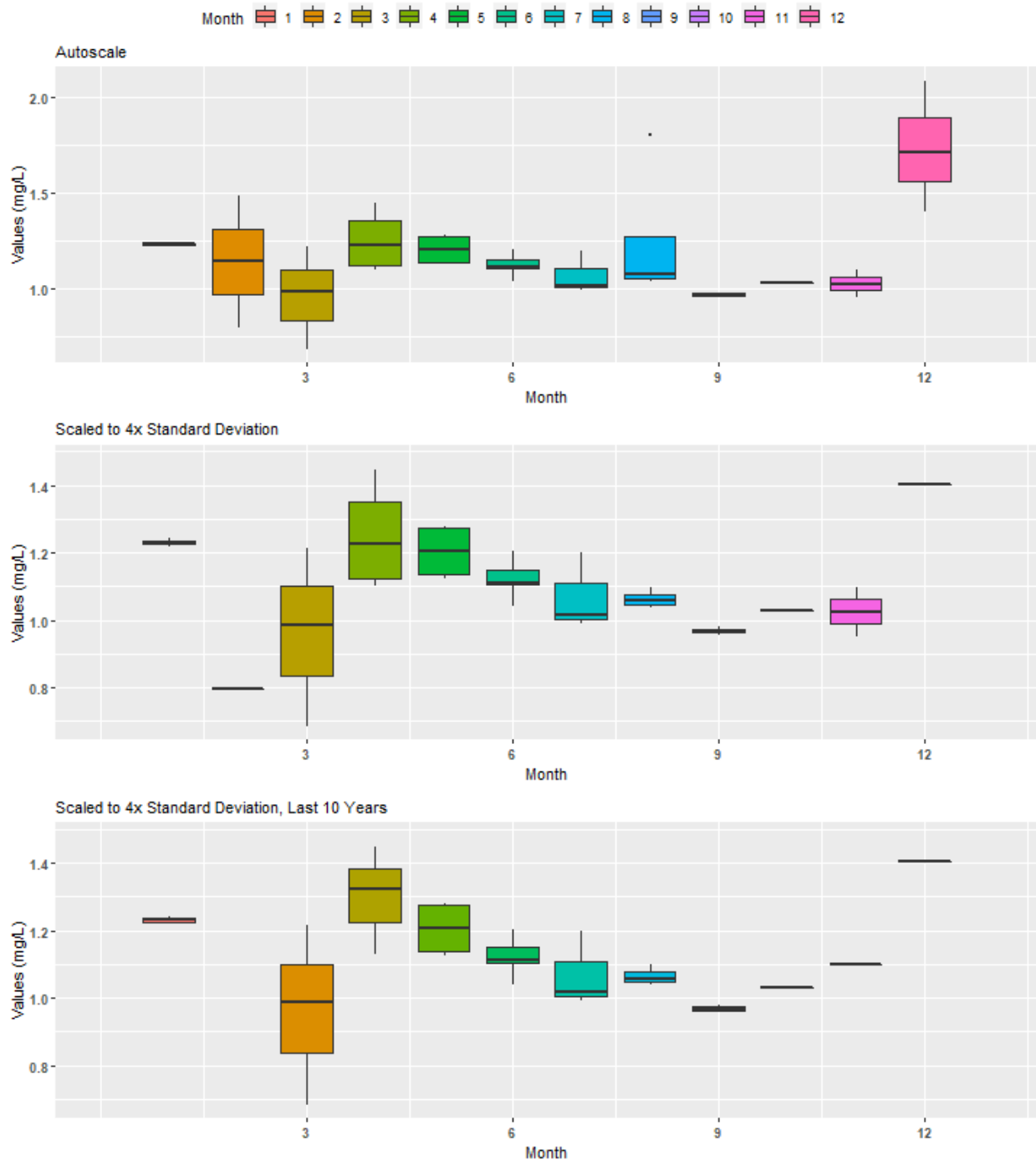
Summary Box Plots for Banana River Aquatic Preserve

By Year & Month



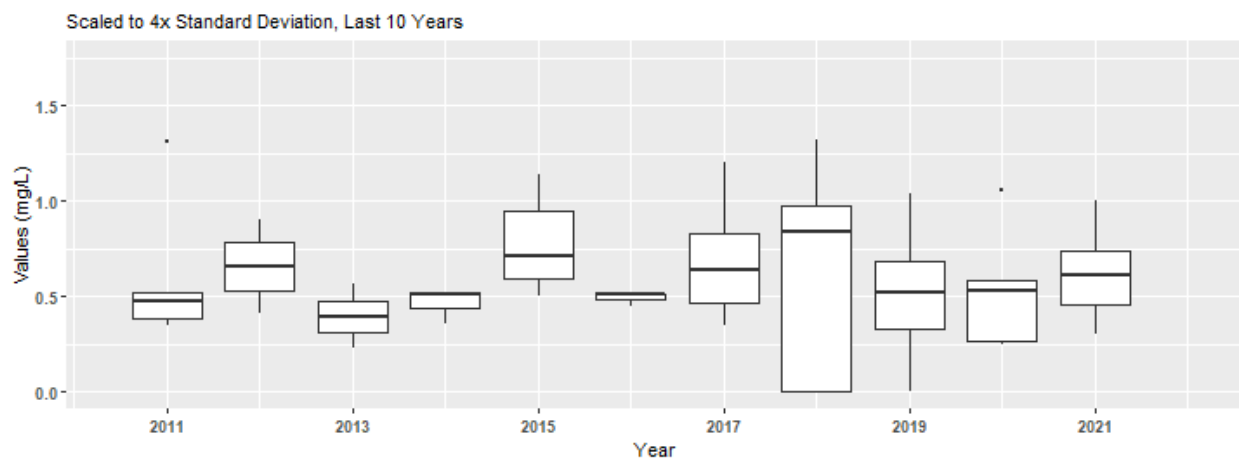
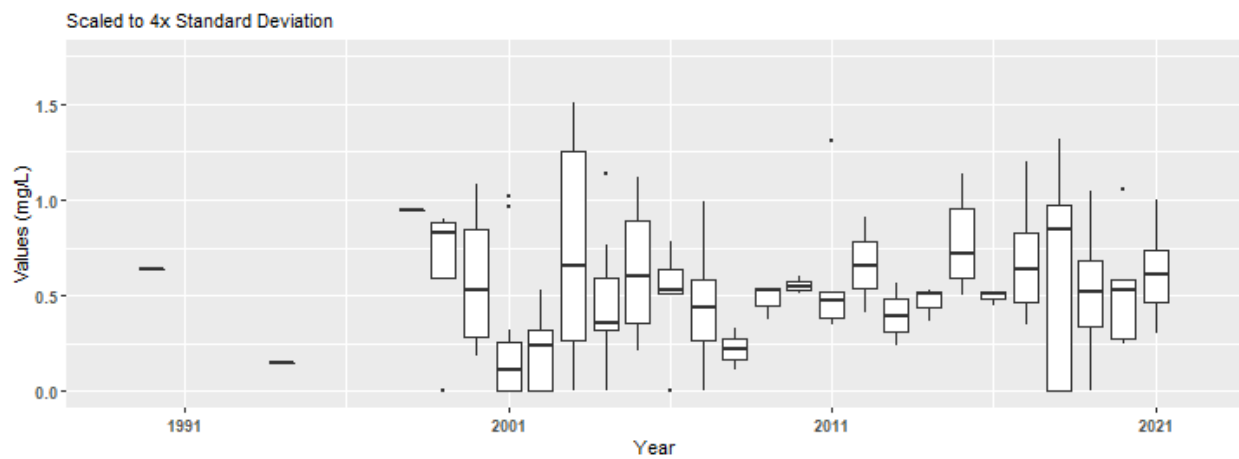
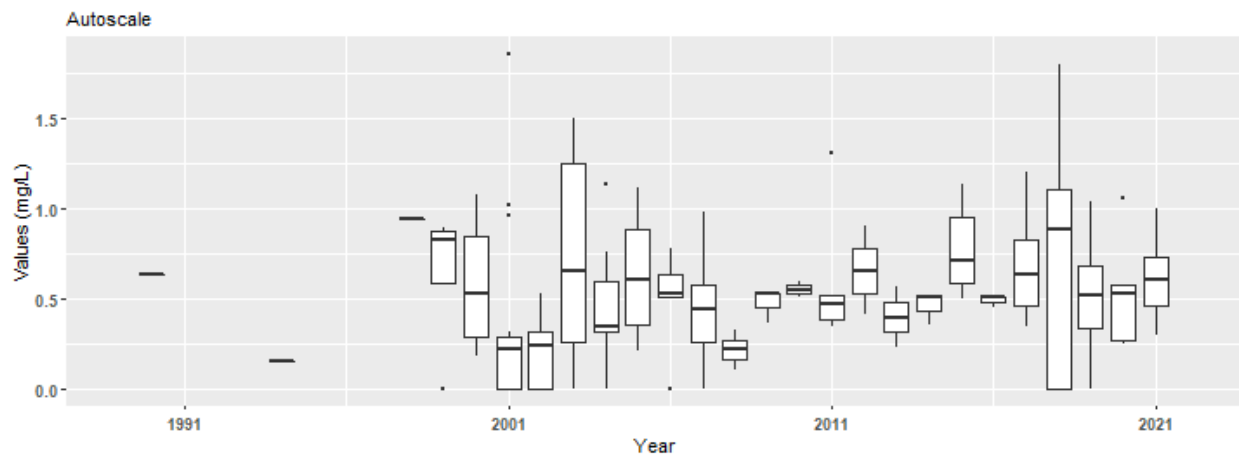
Summary Box Plots for Banana River Aquatic Preserve

By Month



Summary Box Plots for Big Bend Seagrasses Aquatic Preserve

By Year



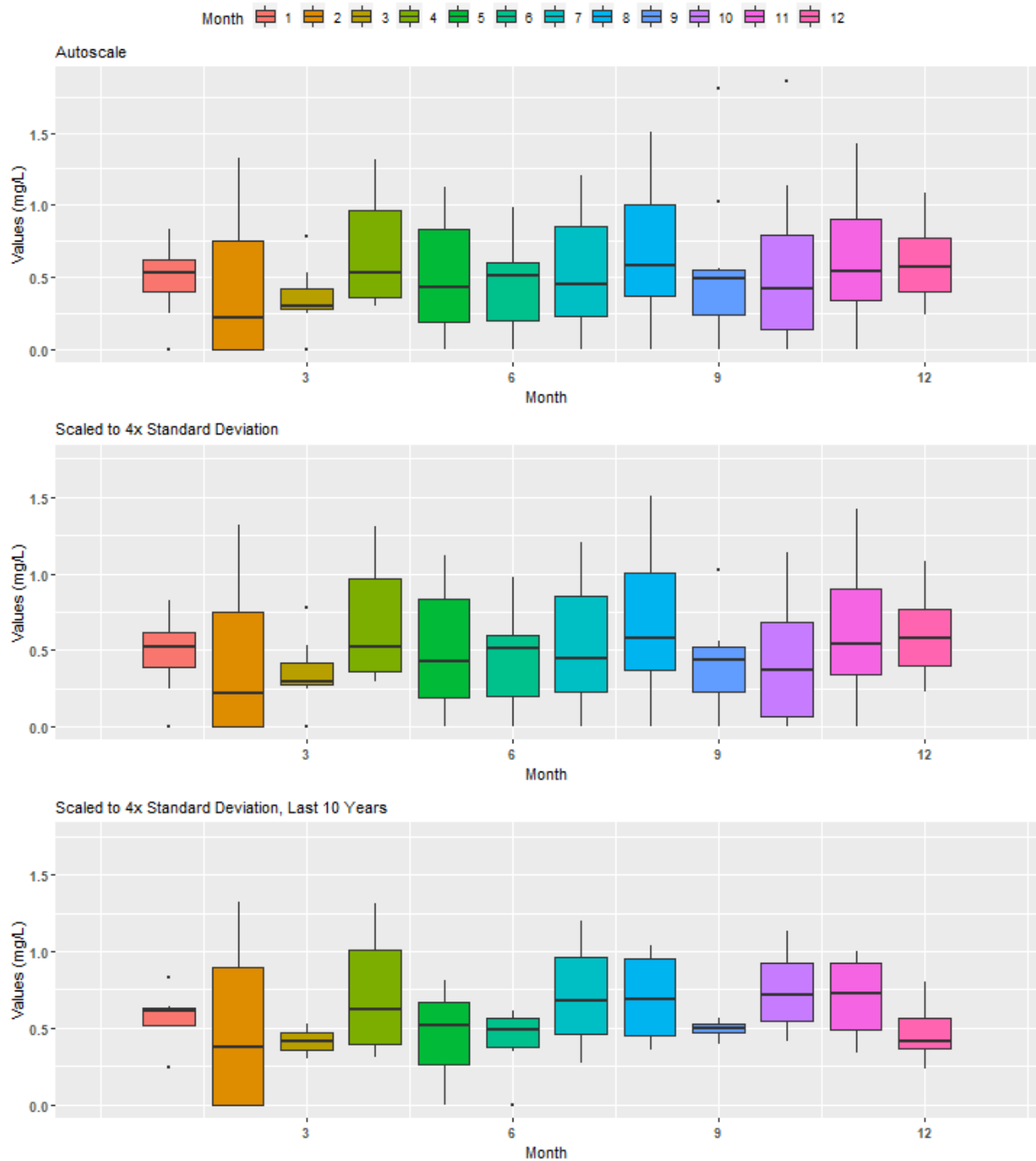
Summary Box Plots for Big Bend Seagrasses Aquatic Preserve

By Year & Month



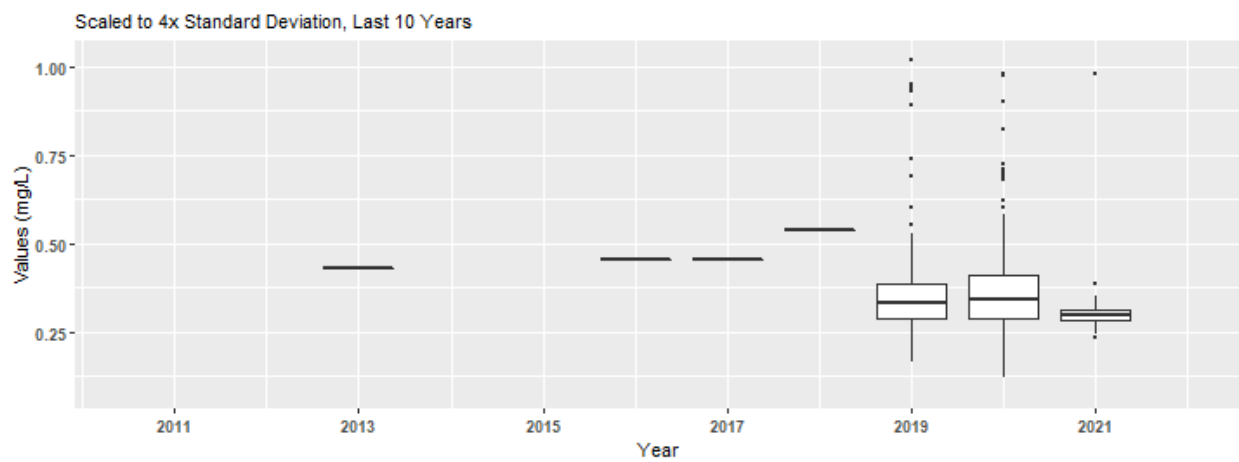
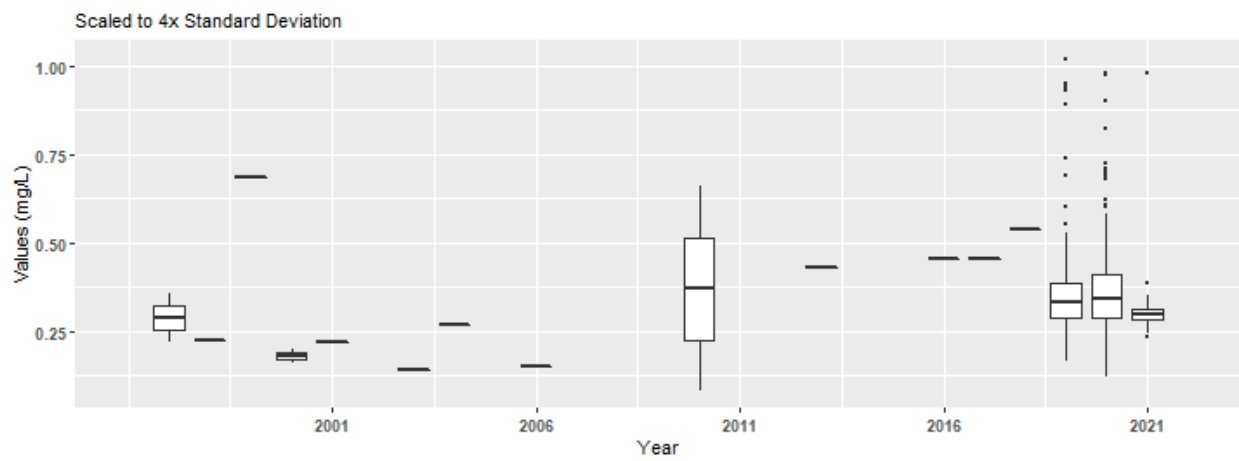
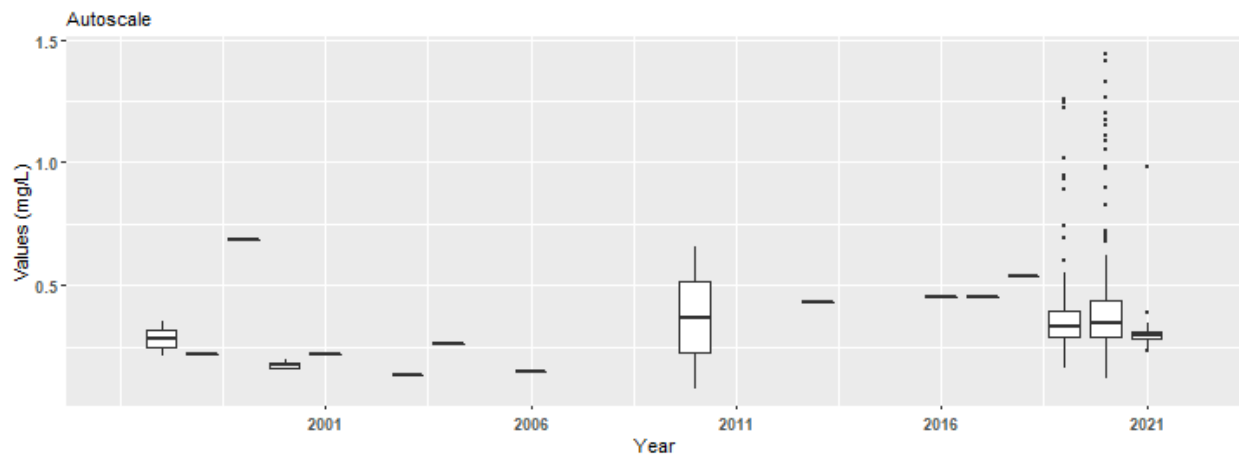
Summary Box Plots for Big Bend Seagrasses Aquatic Preserve

By Month



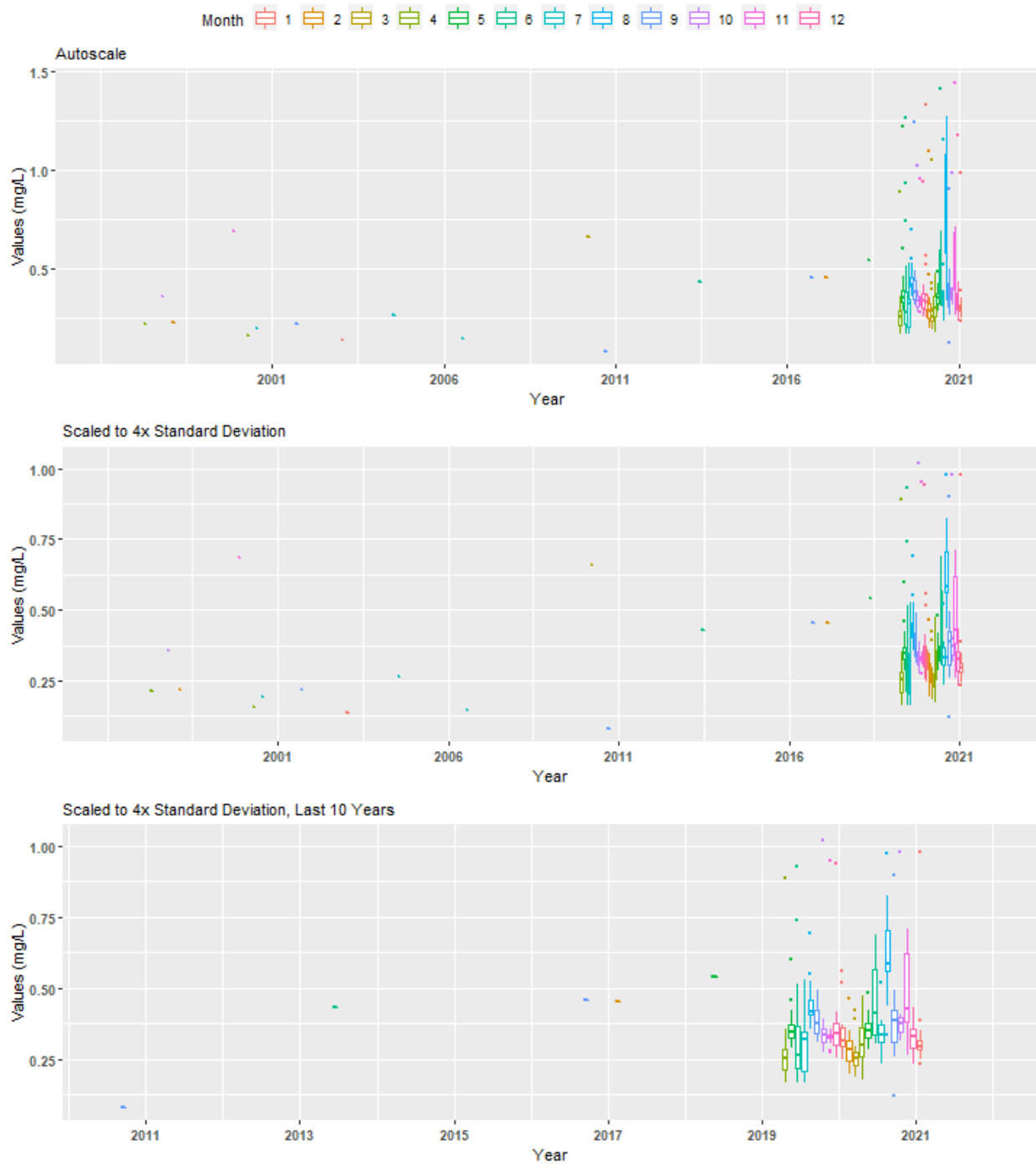
Summary Box Plots for Biscayne Bay Aquatic Preserve

By Year



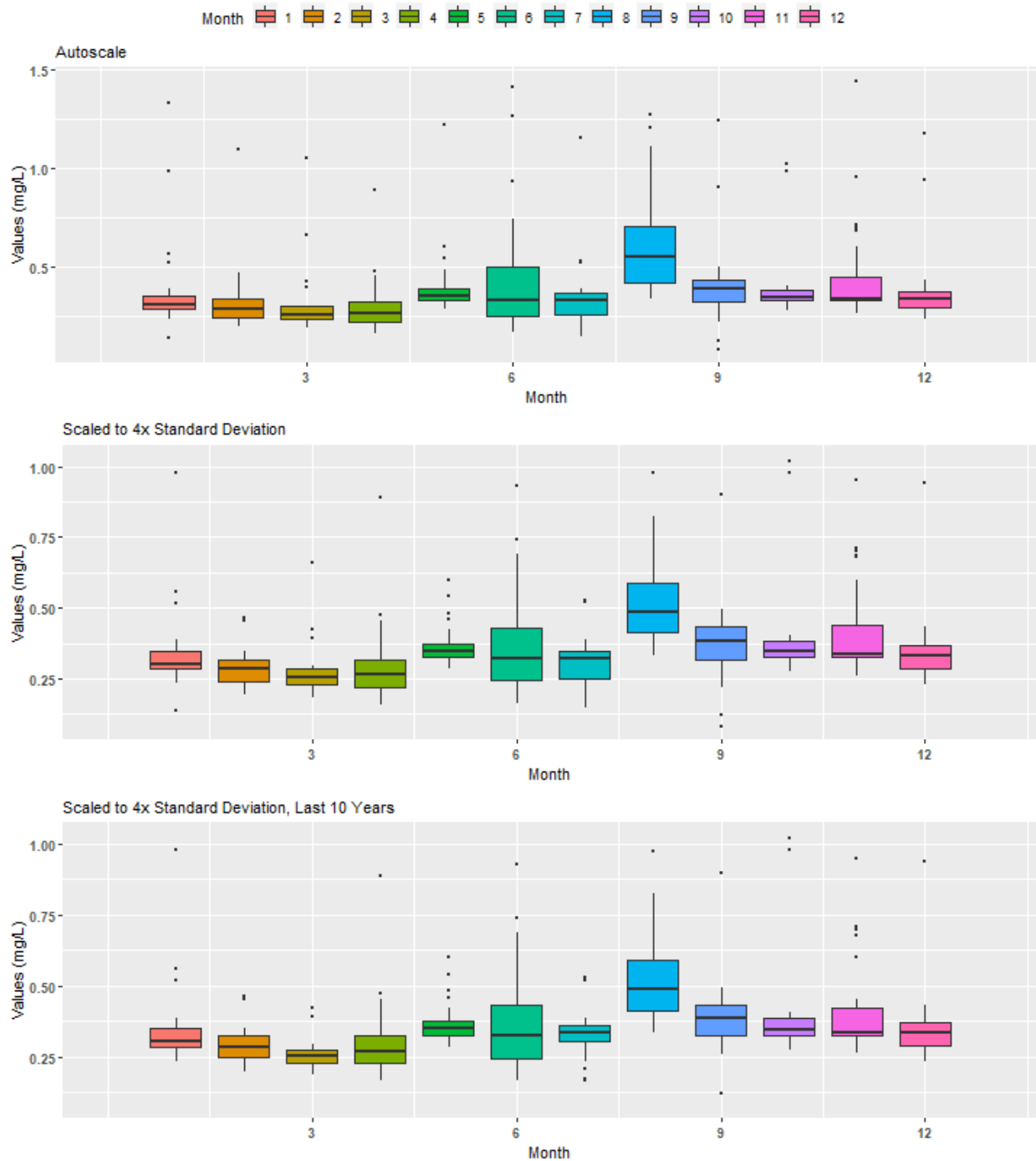
Summary Box Plots for Biscayne Bay Aquatic Preserve

By Year & Month



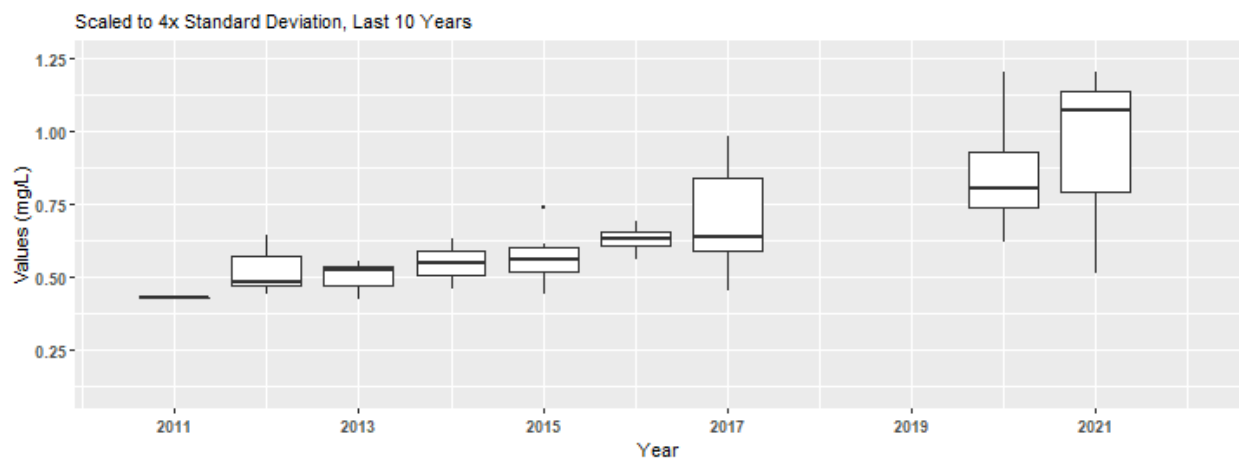
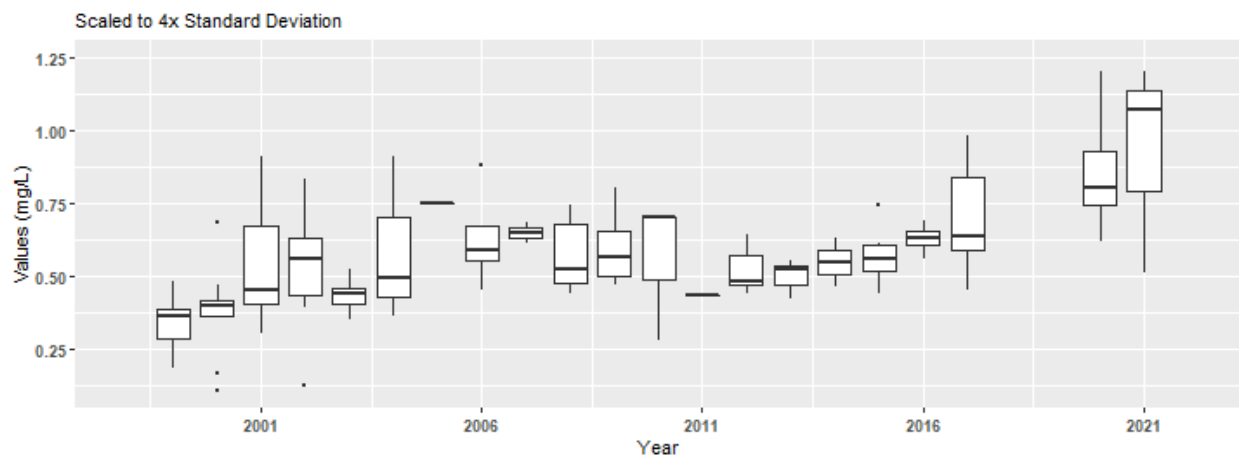
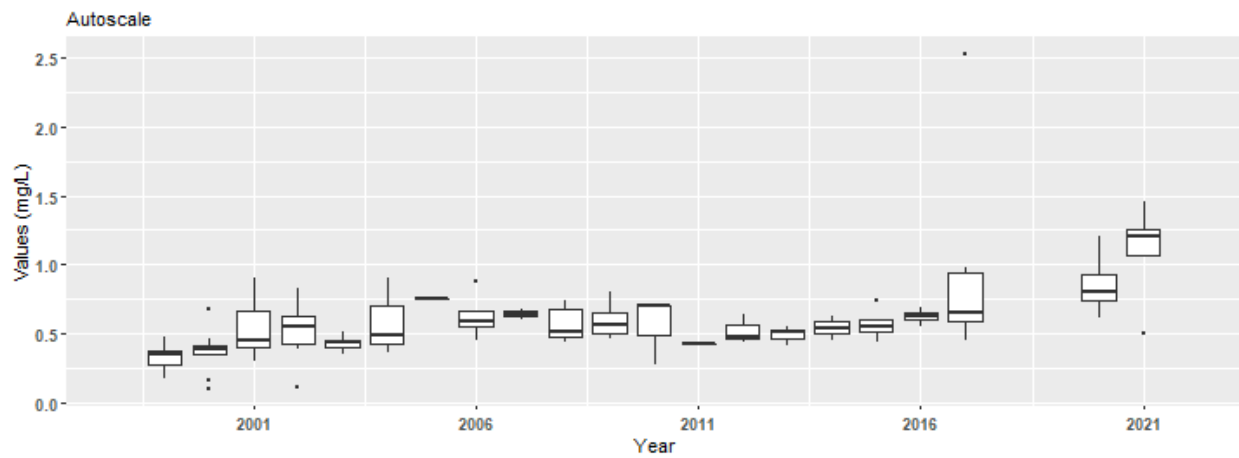
Summary Box Plots for Biscayne Bay Aquatic Preserve

By Month



Summary Box Plots for Boca Ciega Bay Aquatic Preserve

By Year



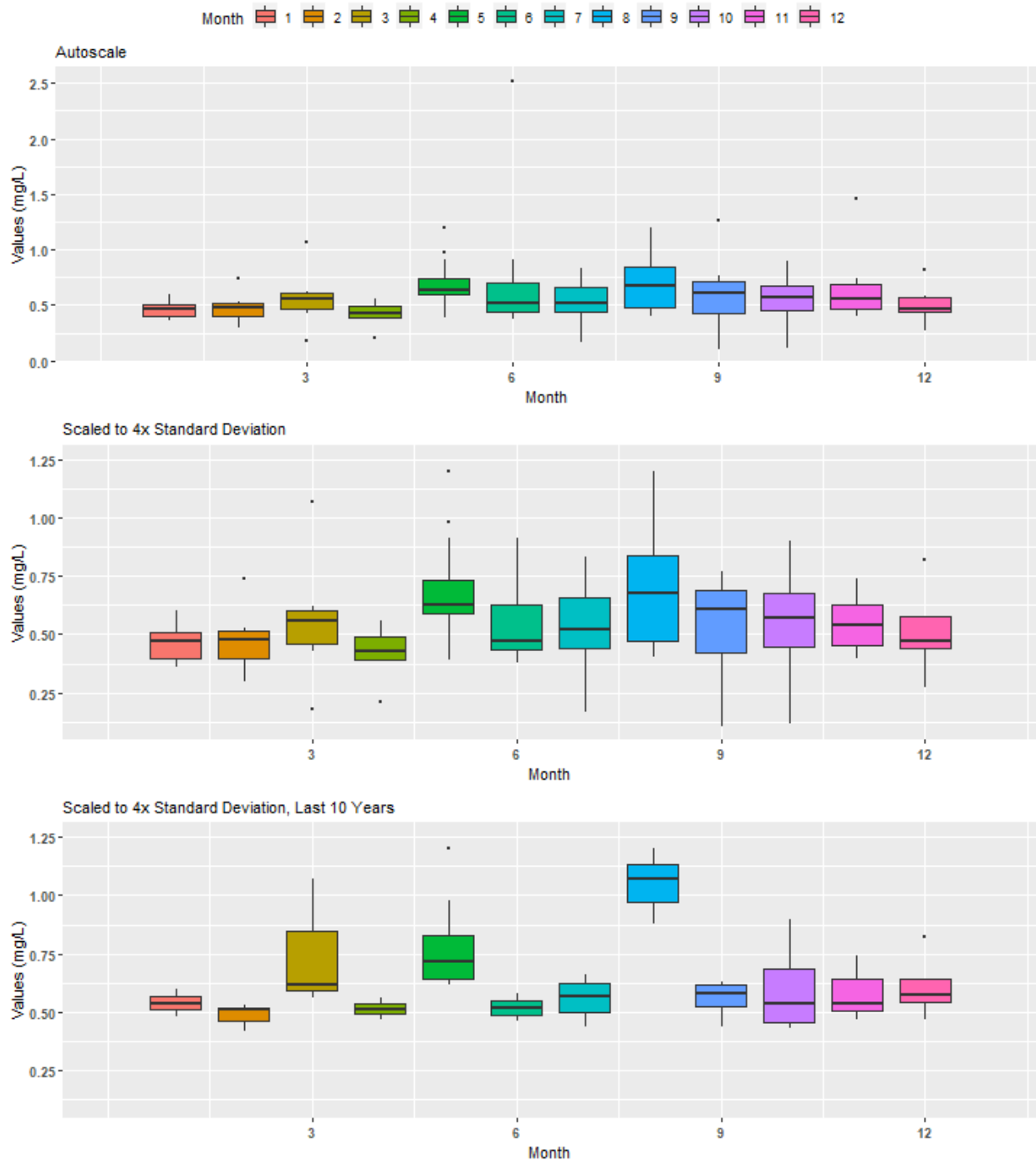
Summary Box Plots for Boca Ciega Bay Aquatic Preserve

By Year & Month

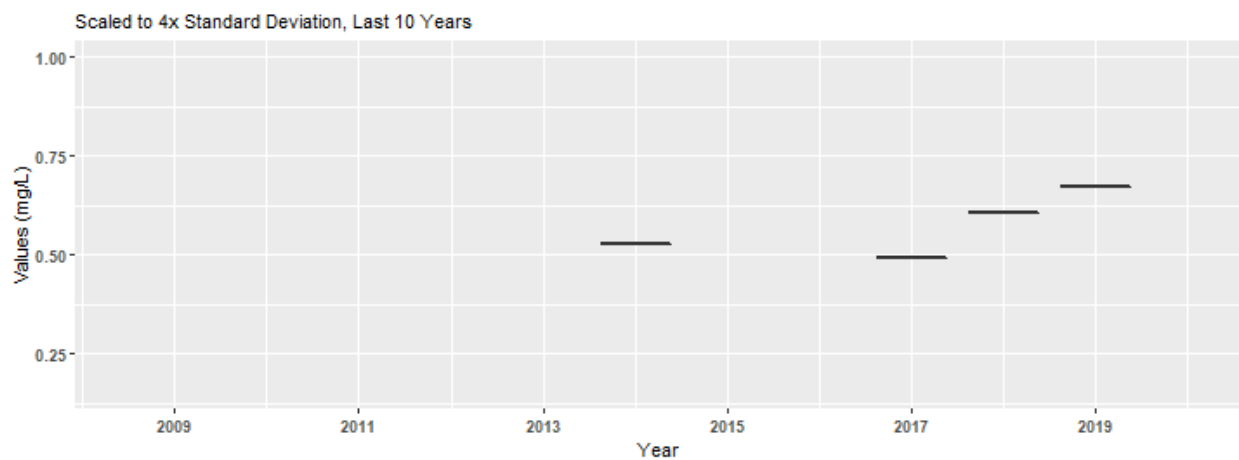
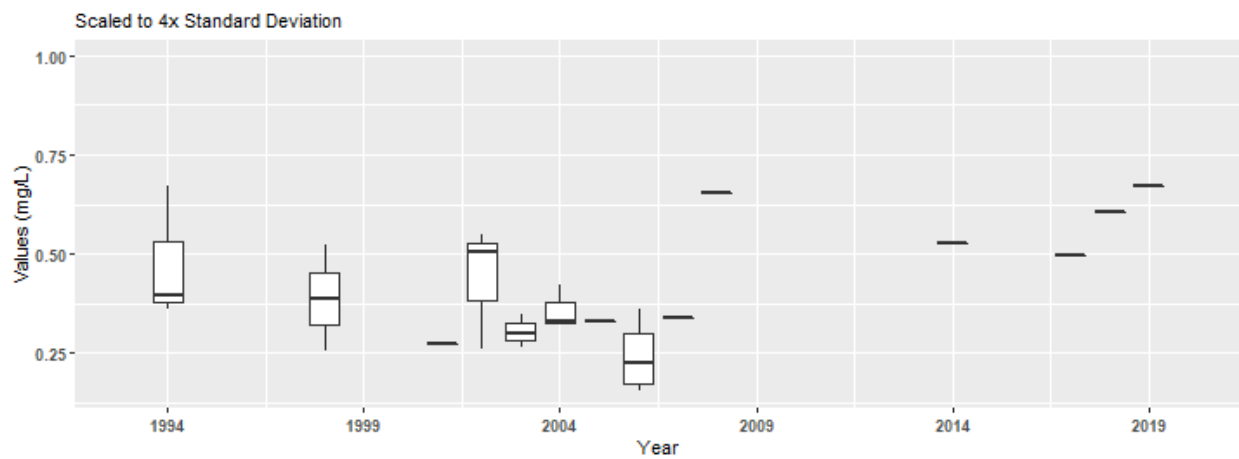
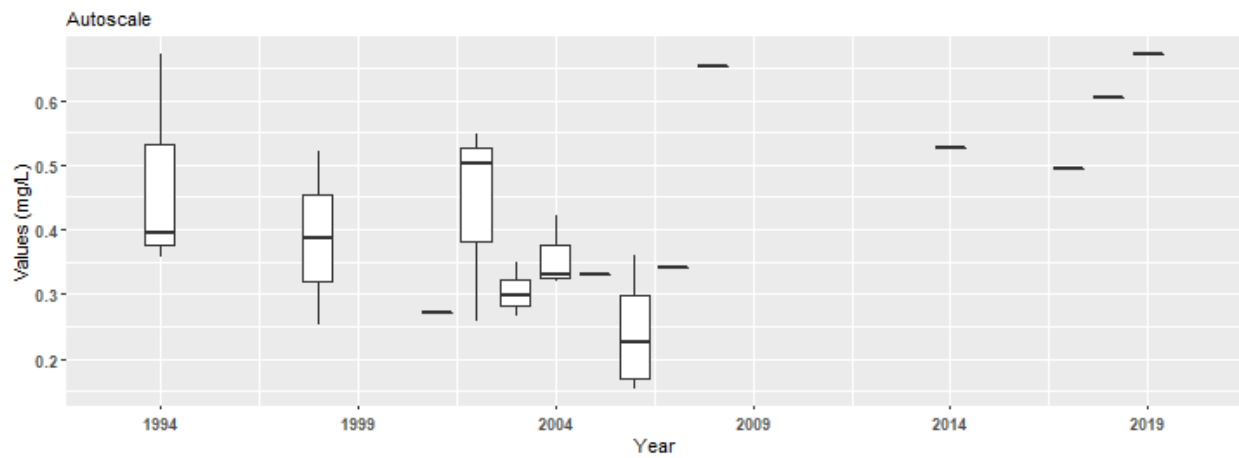


Summary Box Plots for Boca Ciega Bay Aquatic Preserve

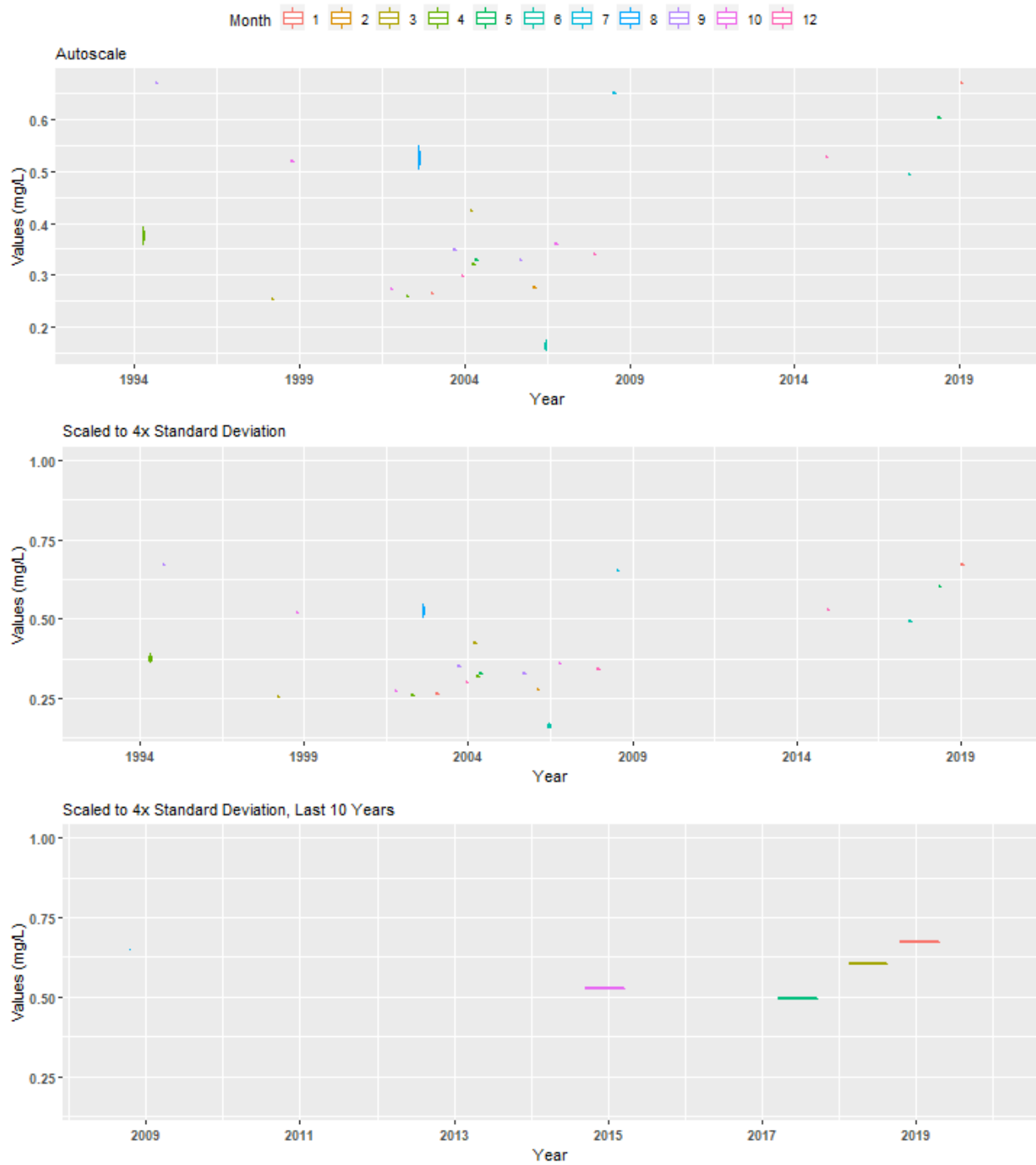
By Month



Summary Box Plots for Cape Romano-Ten Thousand Islands Aquatic Preserve By Year

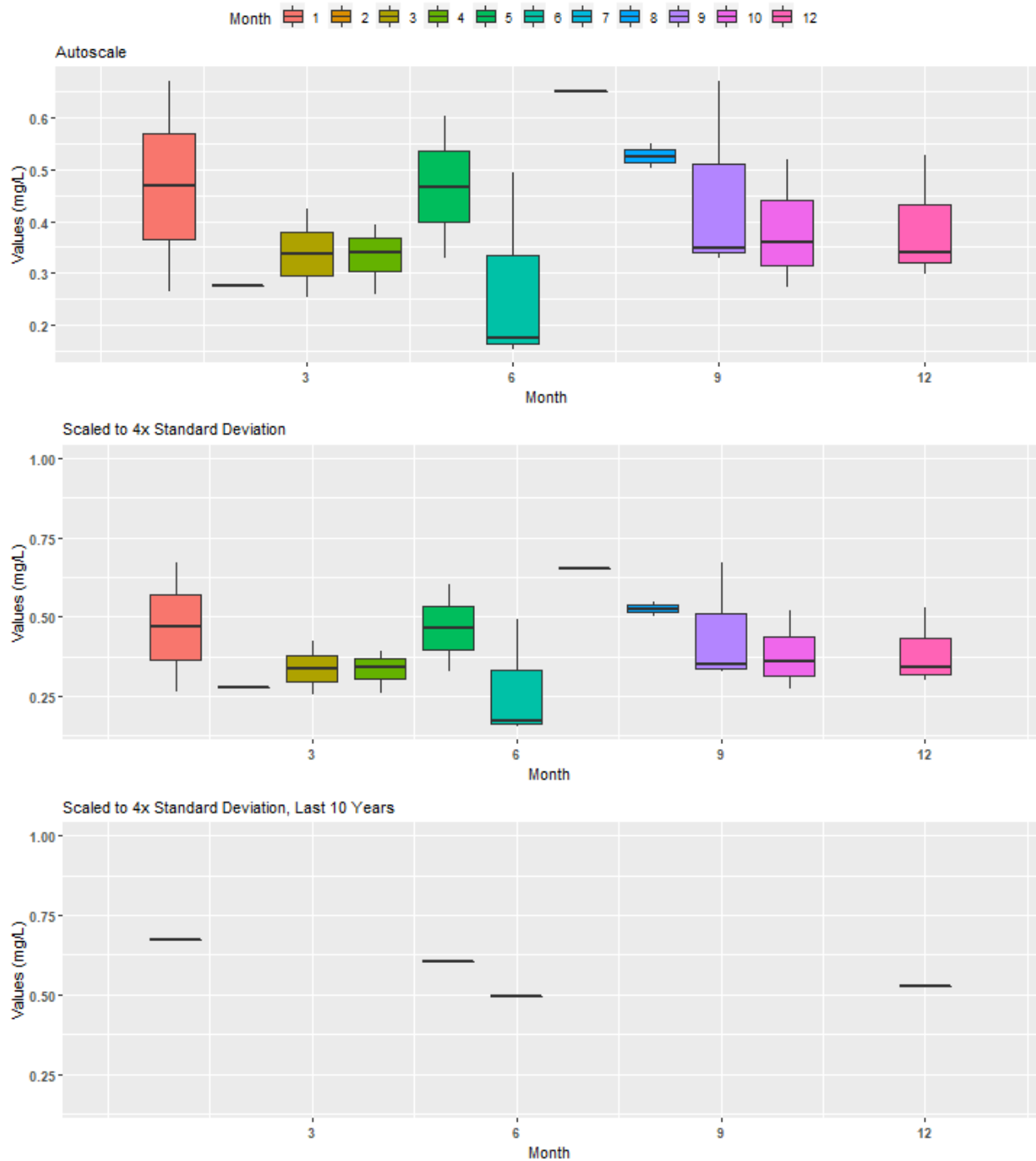


Summary Box Plots for Cape Romano-Ten Thousand Islands Aquatic Preserve By Year & Month



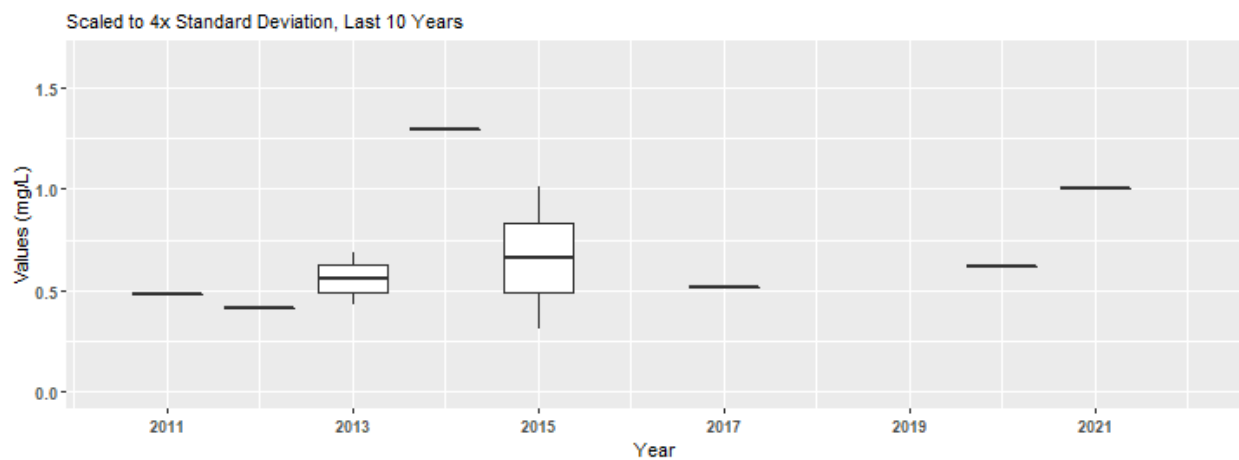
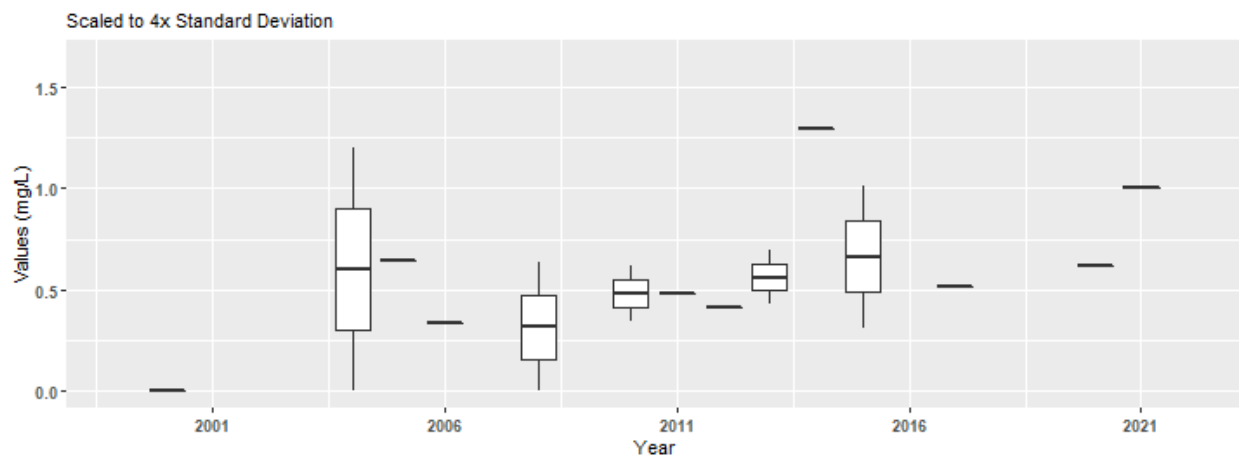
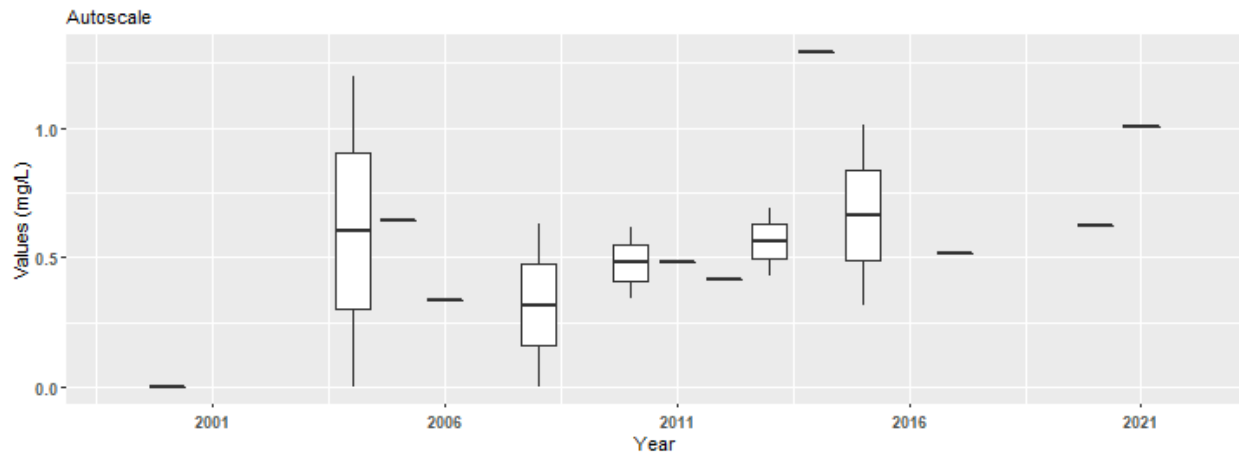
Summary Box Plots for Cape Romano-Ten Thousand Islands Aquatic Preserve

By Month



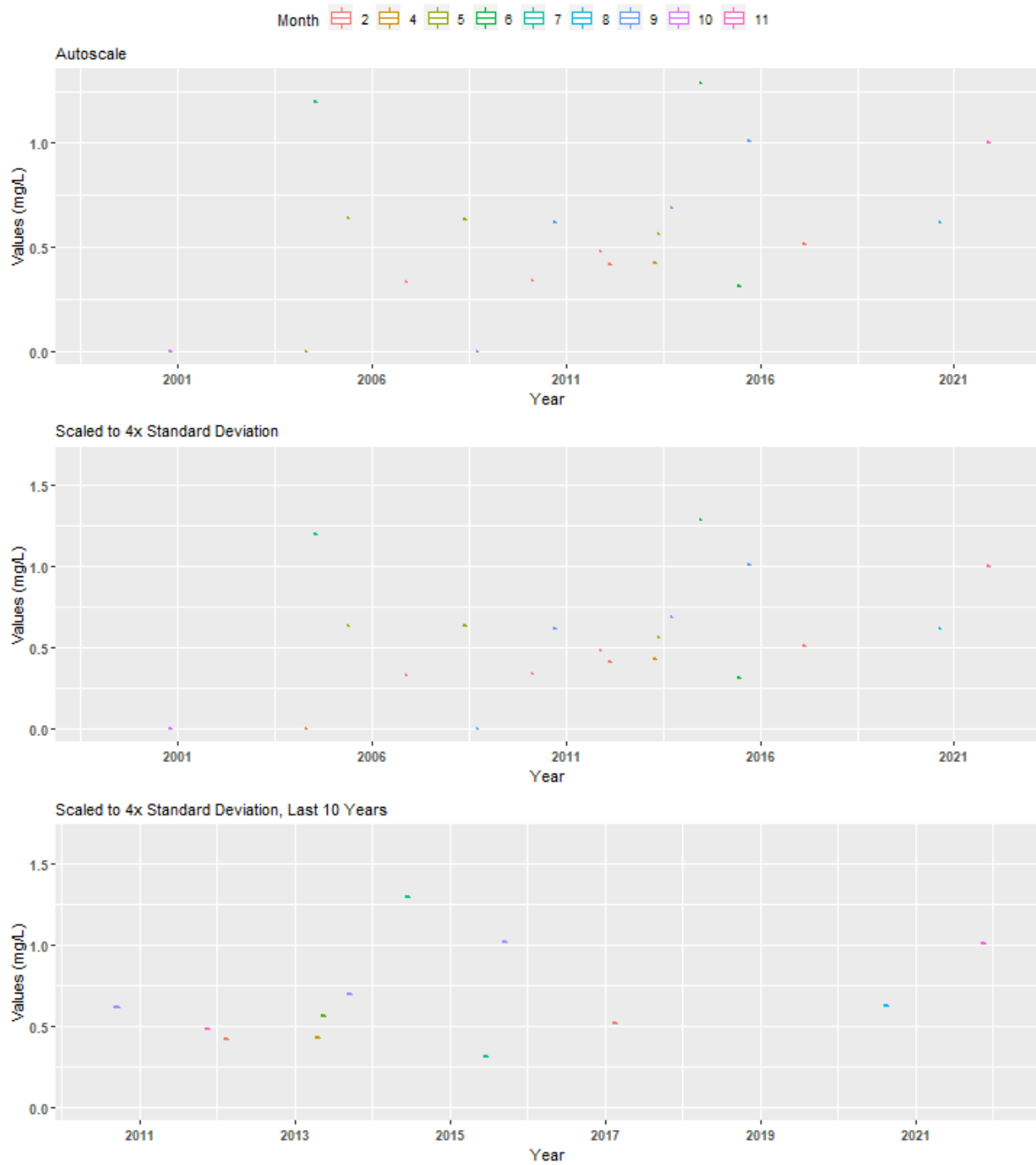
Summary Box Plots for Cockroach Bay Aquatic Preserve

By Year



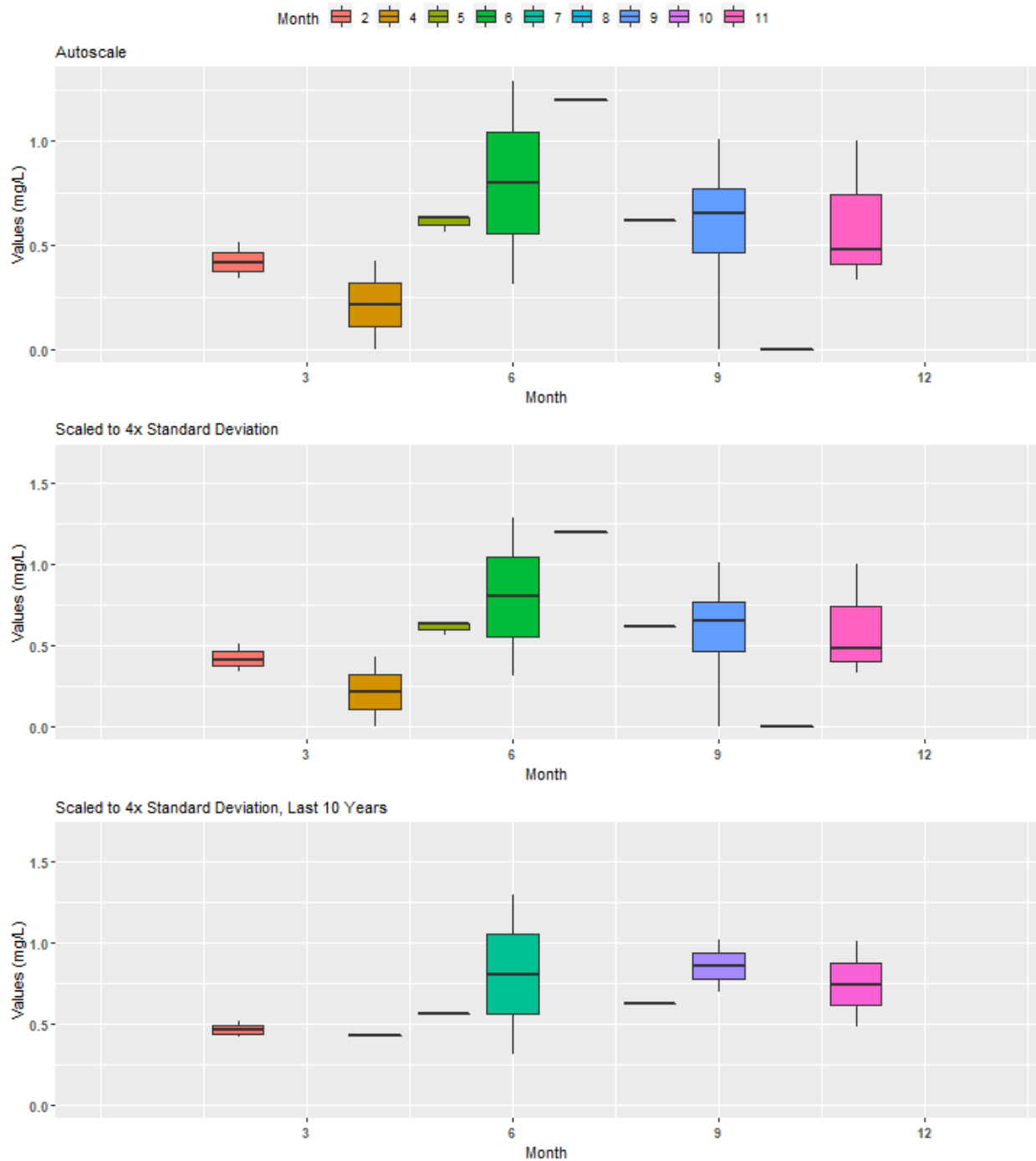
Summary Box Plots for Cockroach Bay Aquatic Preserve

By Year & Month



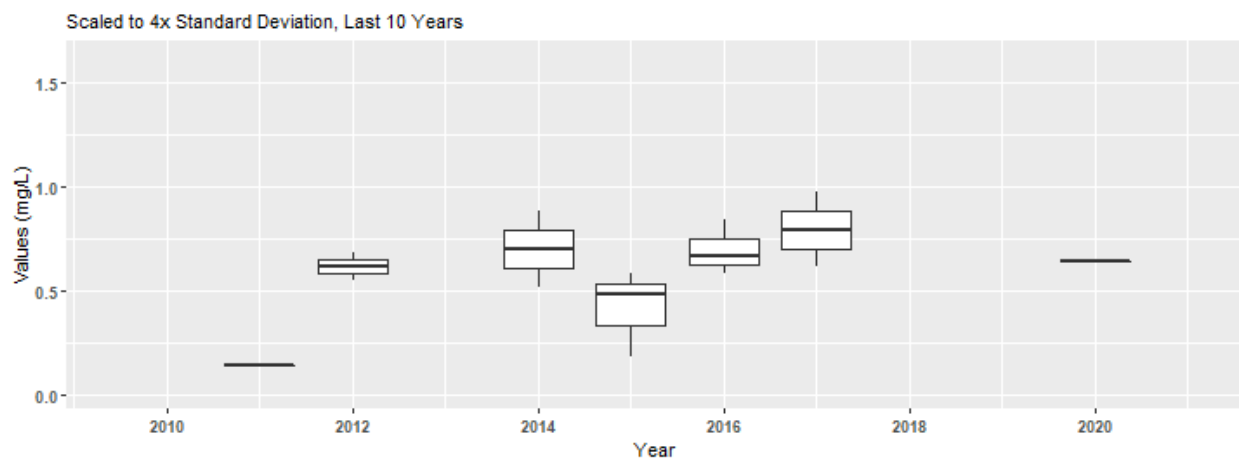
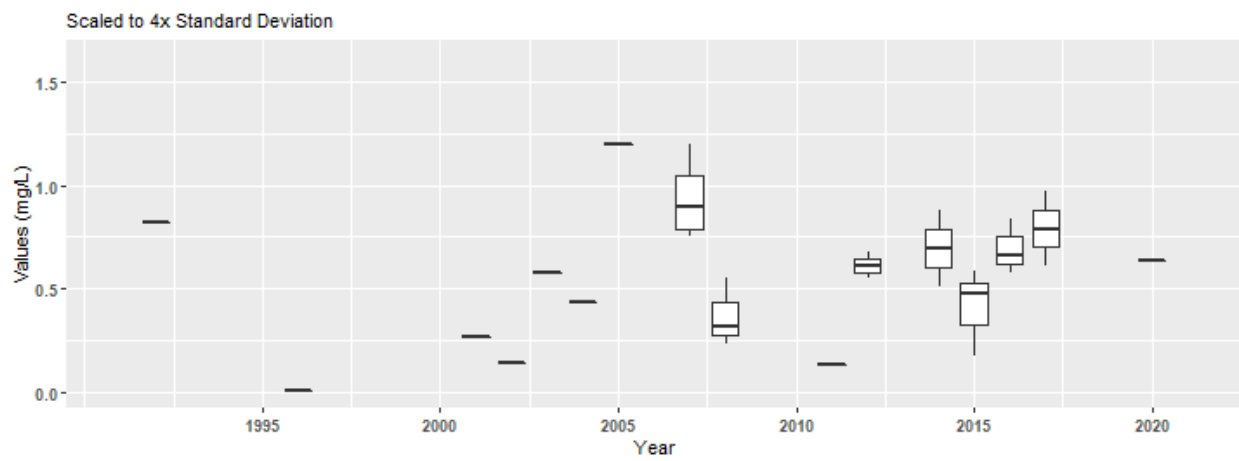
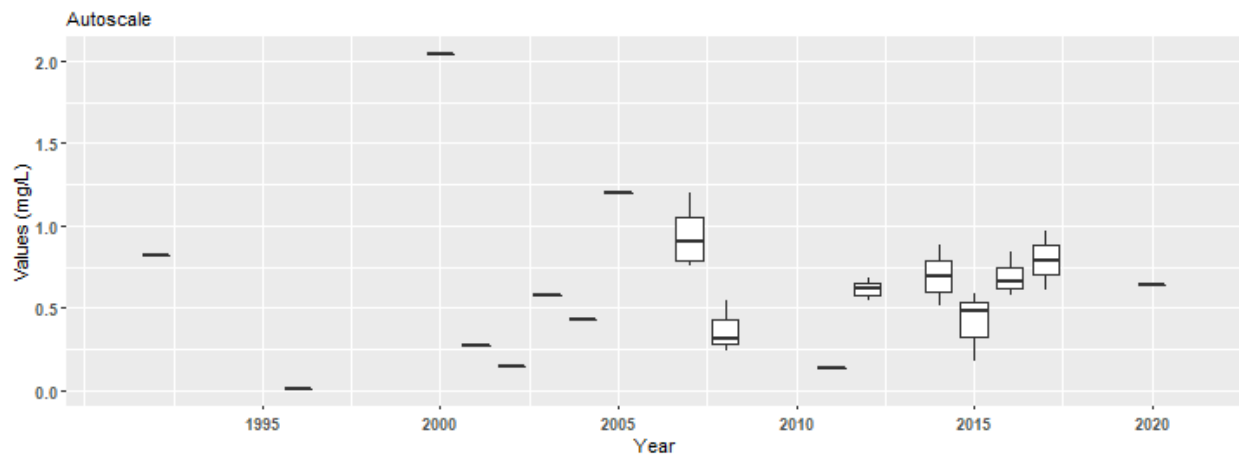
Summary Box Plots for Cockroach Bay Aquatic Preserve

By Month



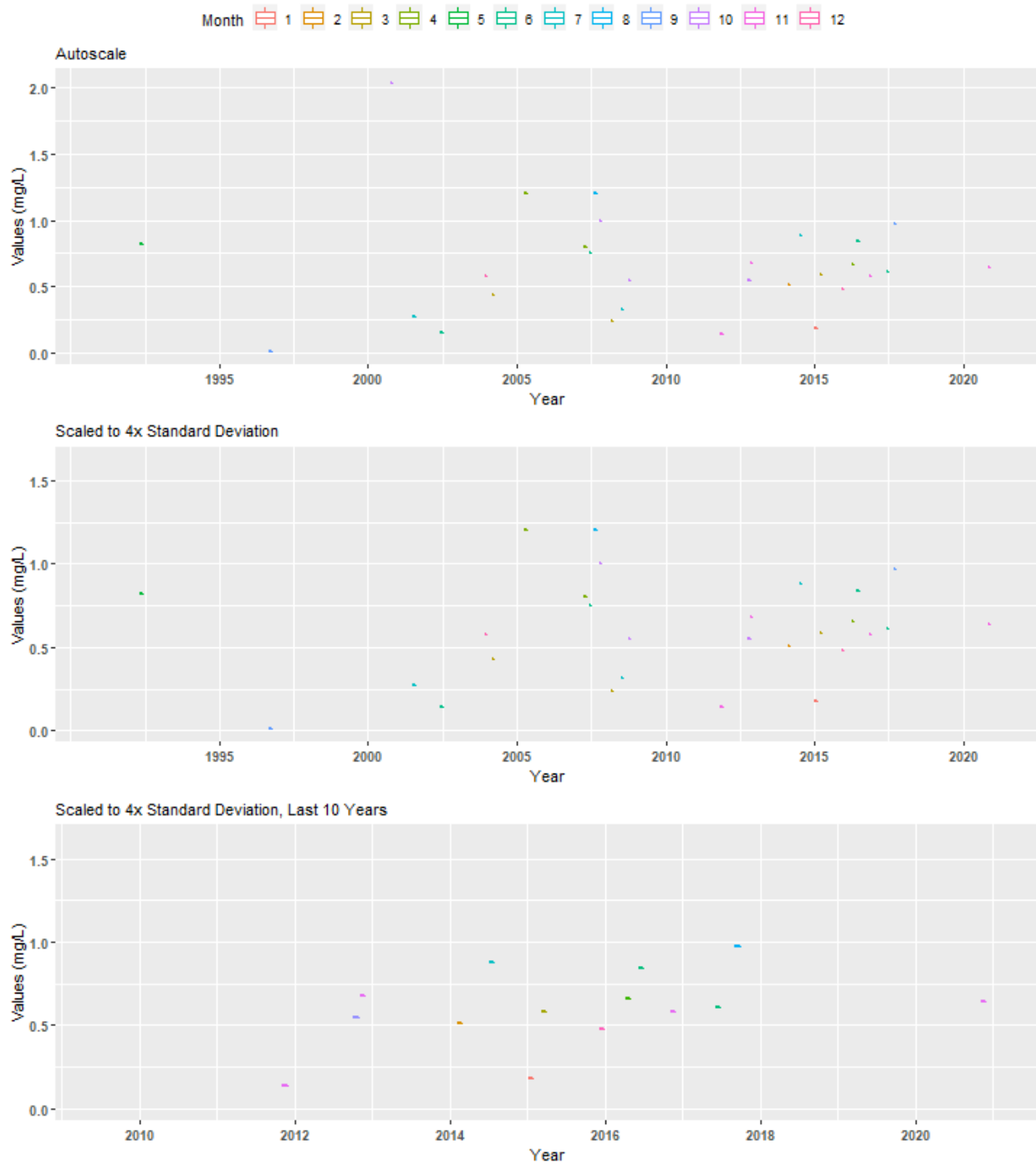
Summary Box Plots for Estero Bay Aquatic Preserve

By Year



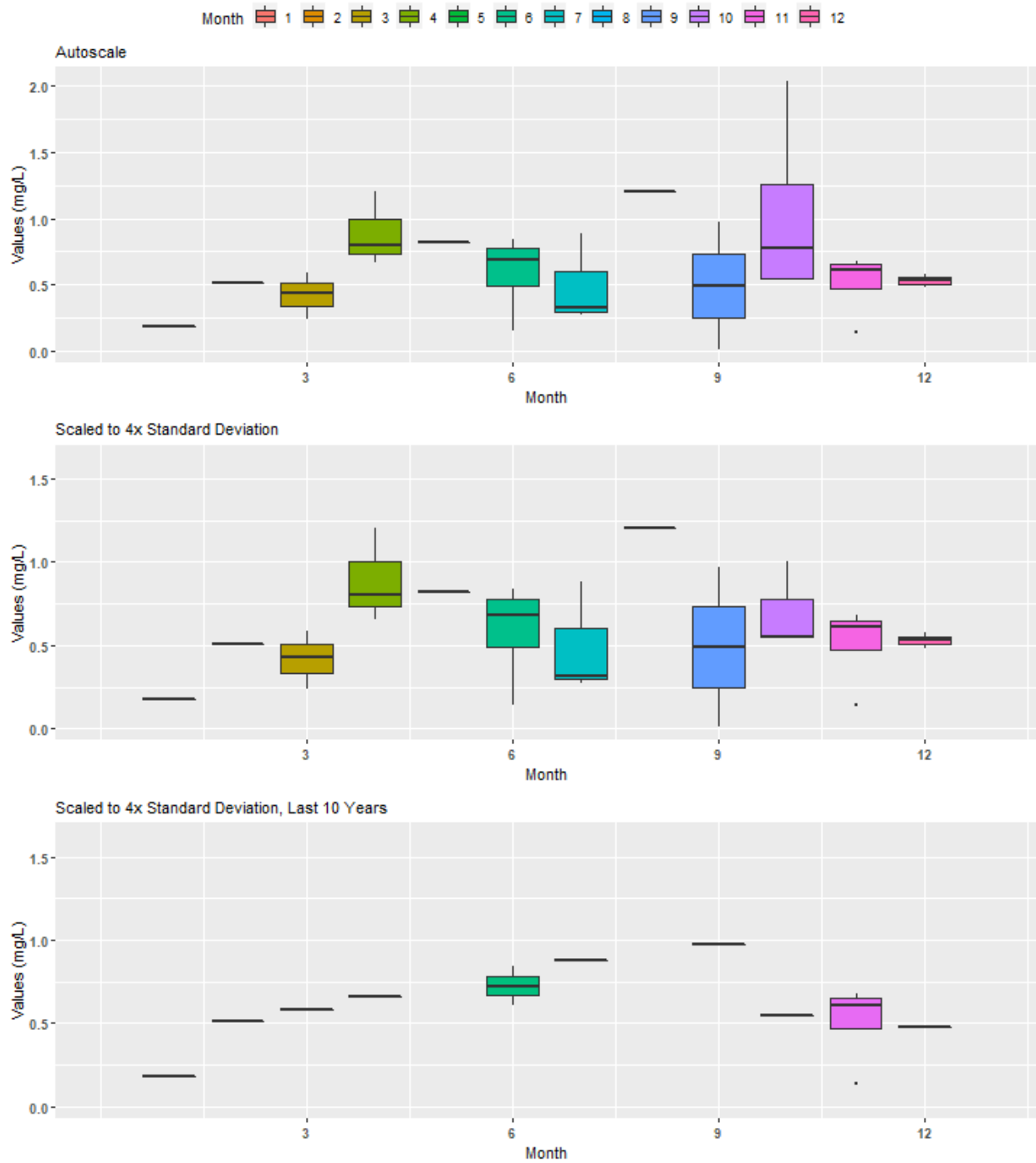
Summary Box Plots for Estero Bay Aquatic Preserve

By Year & Month



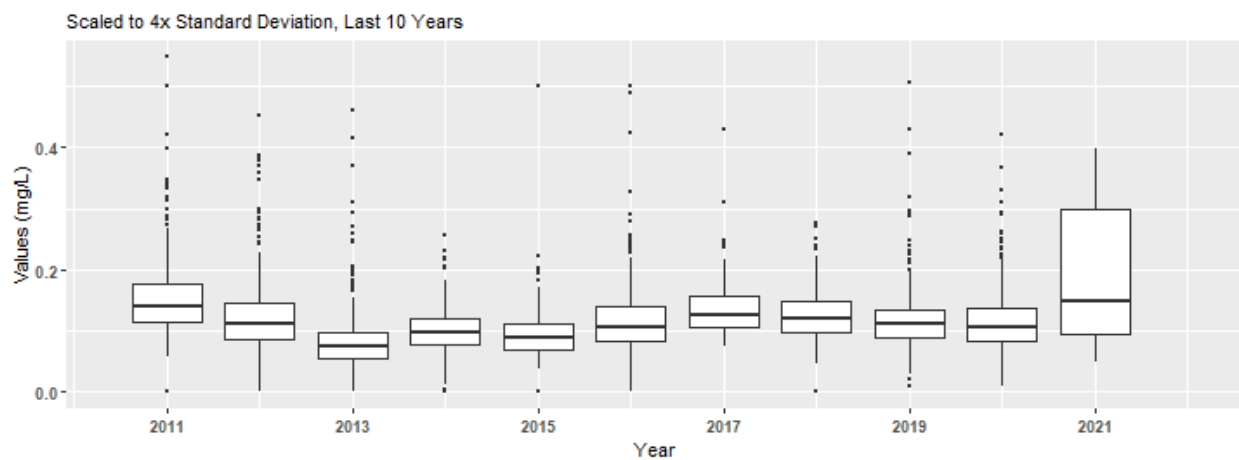
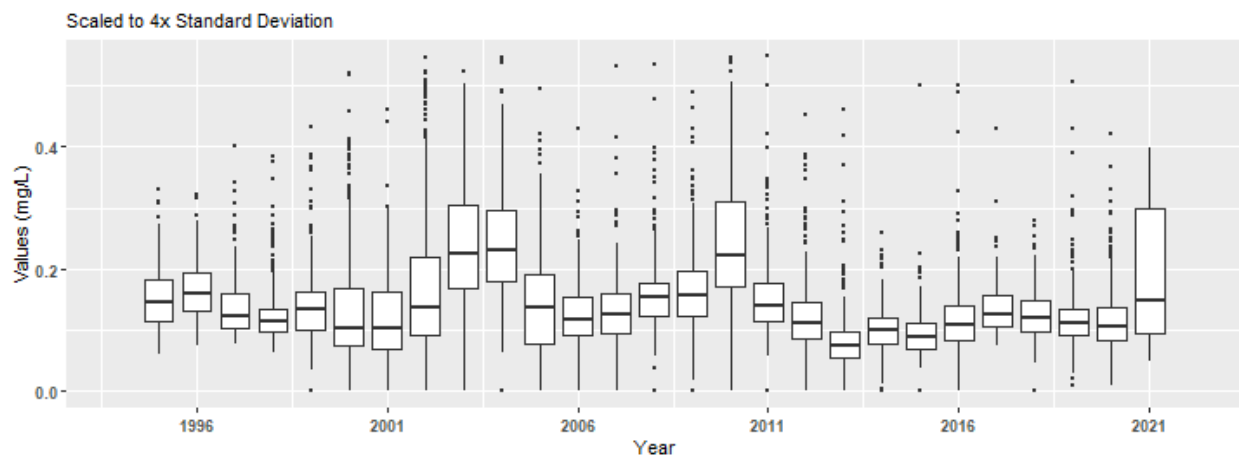
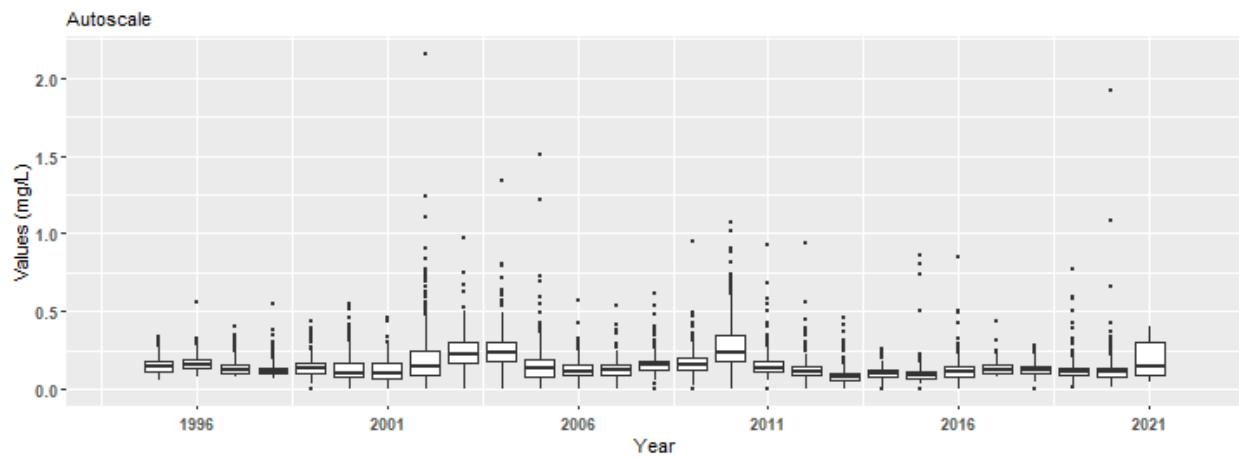
Summary Box Plots for Estero Bay Aquatic Preserve

By Month



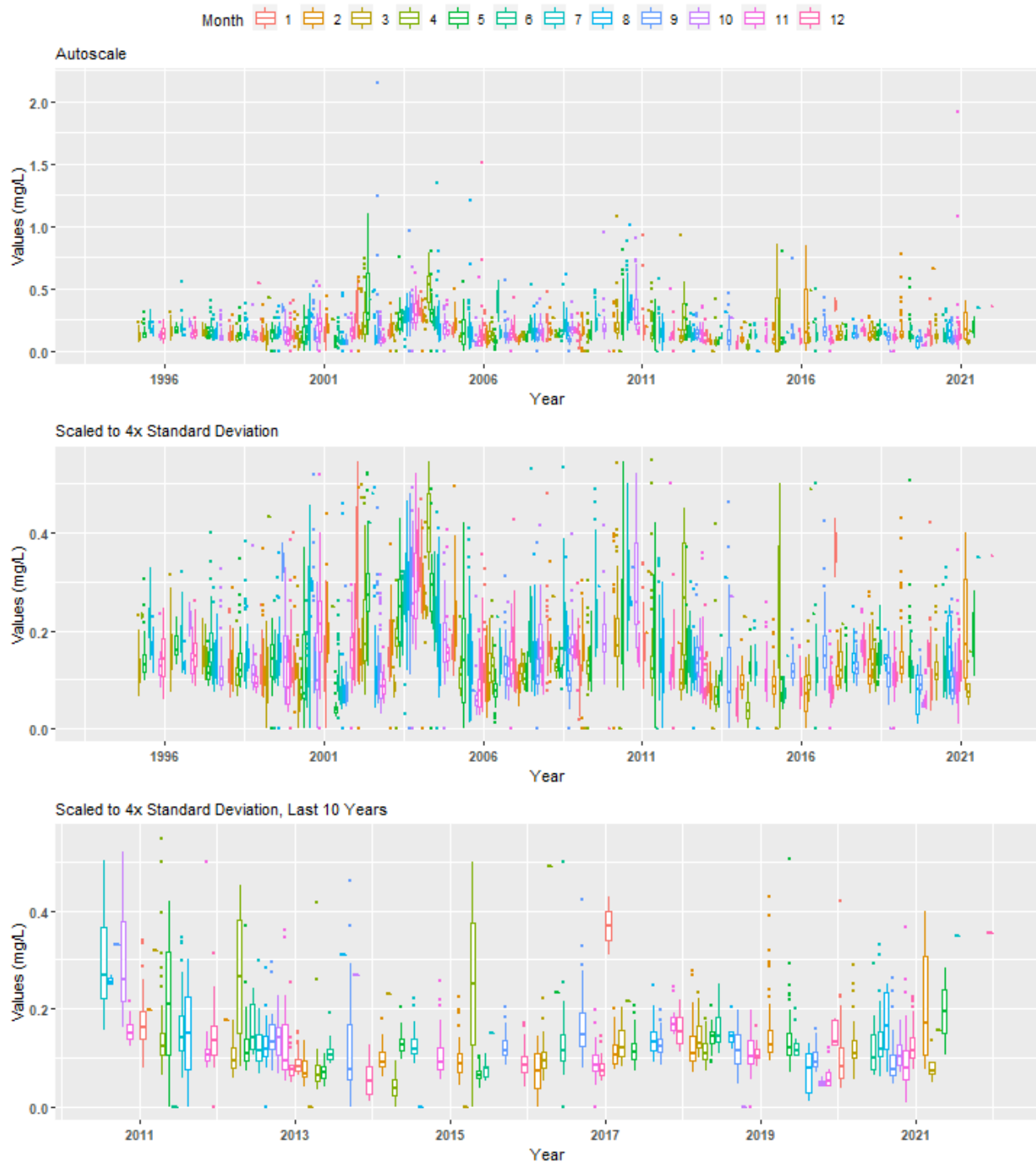
Summary Box Plots for Florida Keys National Marine Sanctuary

By Year



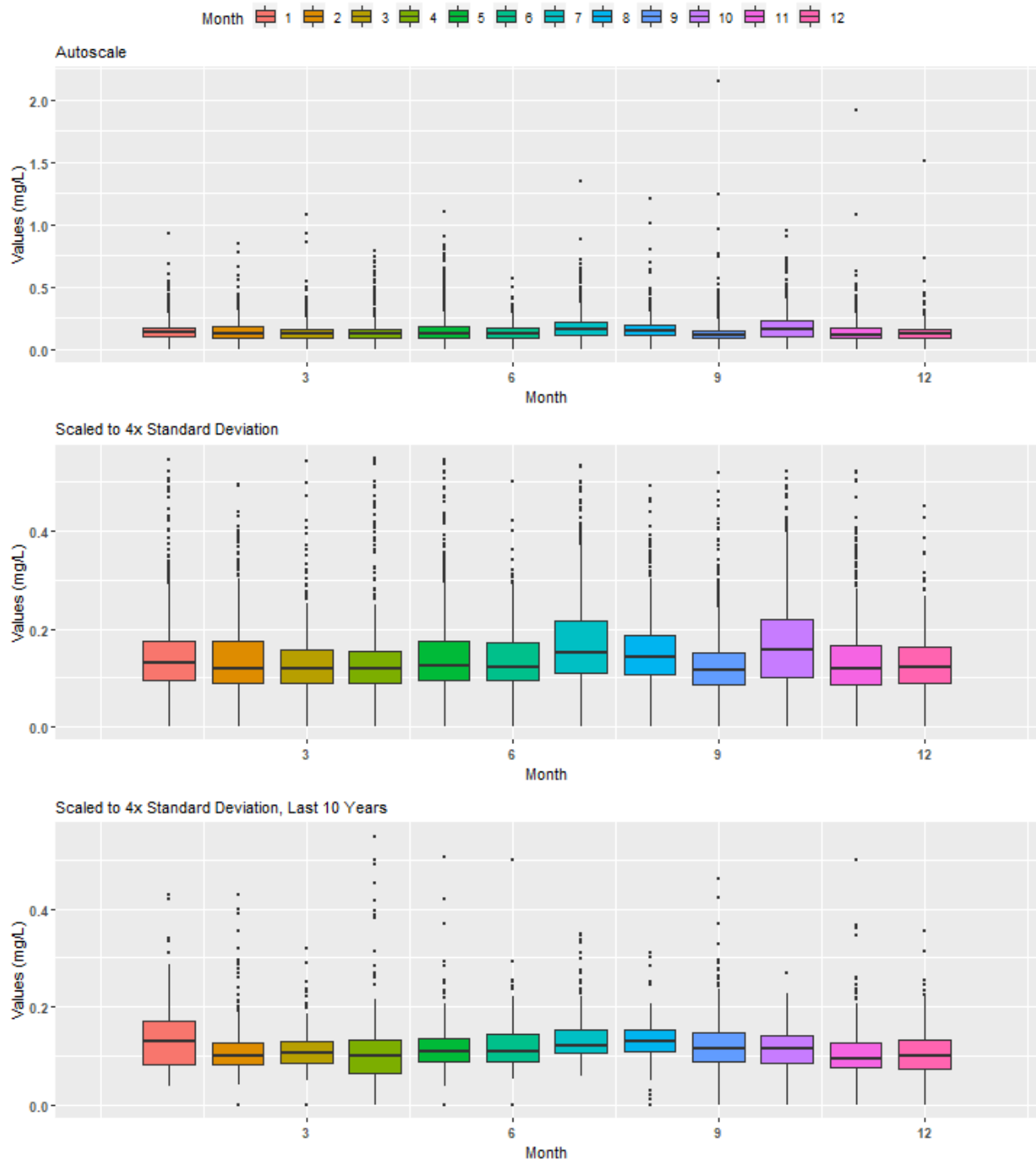
Summary Box Plots for Florida Keys National Marine Sanctuary

By Year & Month



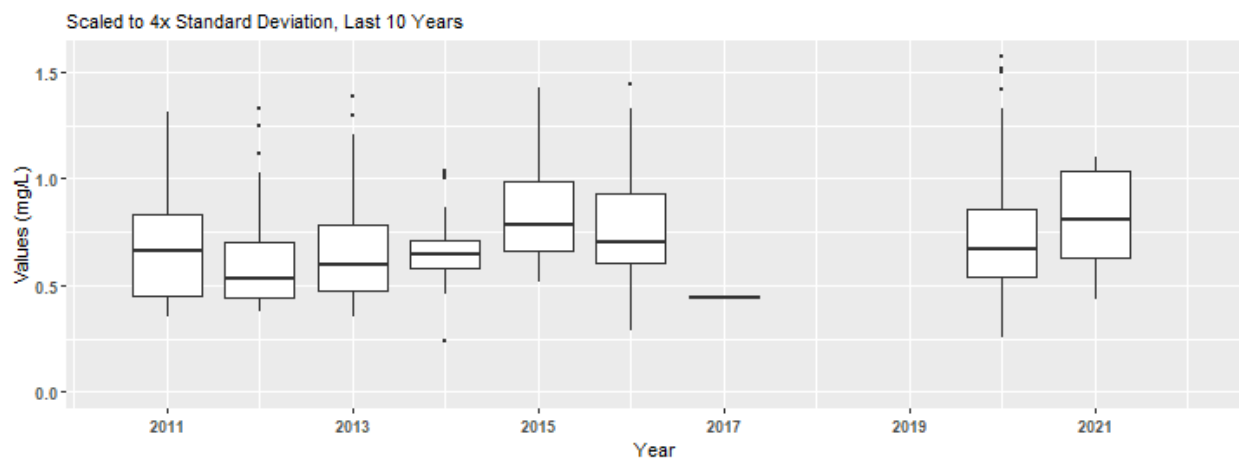
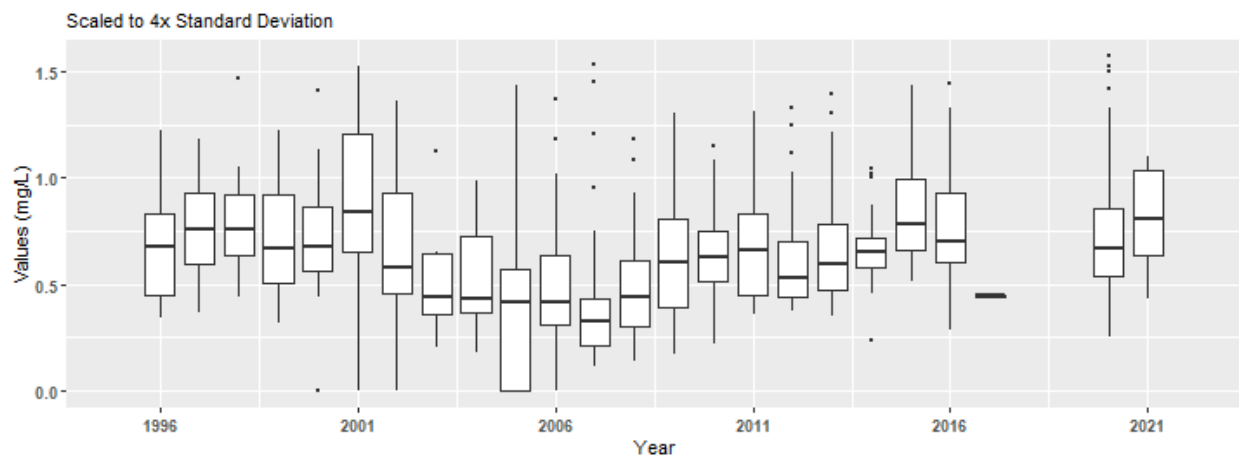
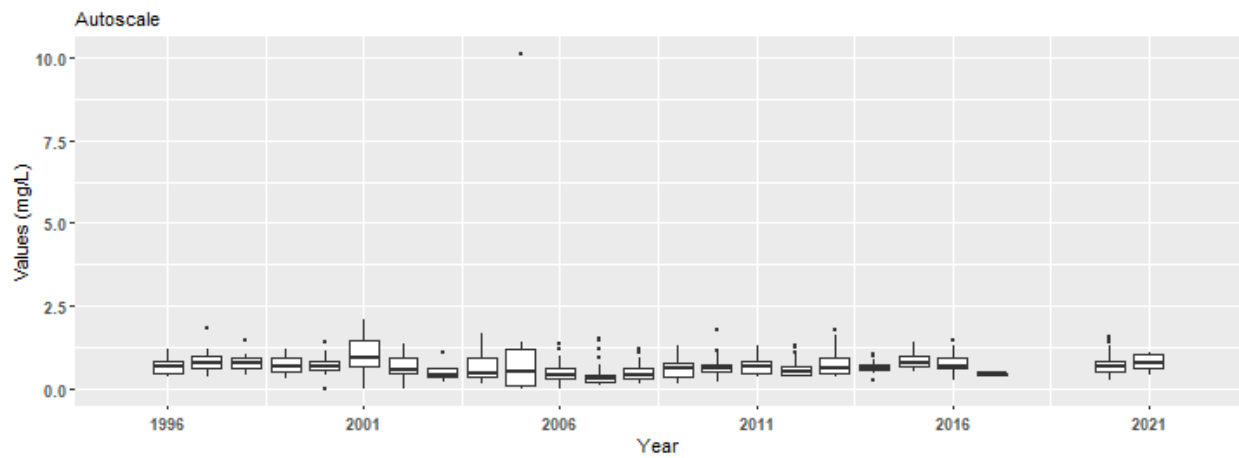
Summary Box Plots for Florida Keys National Marine Sanctuary

By Month



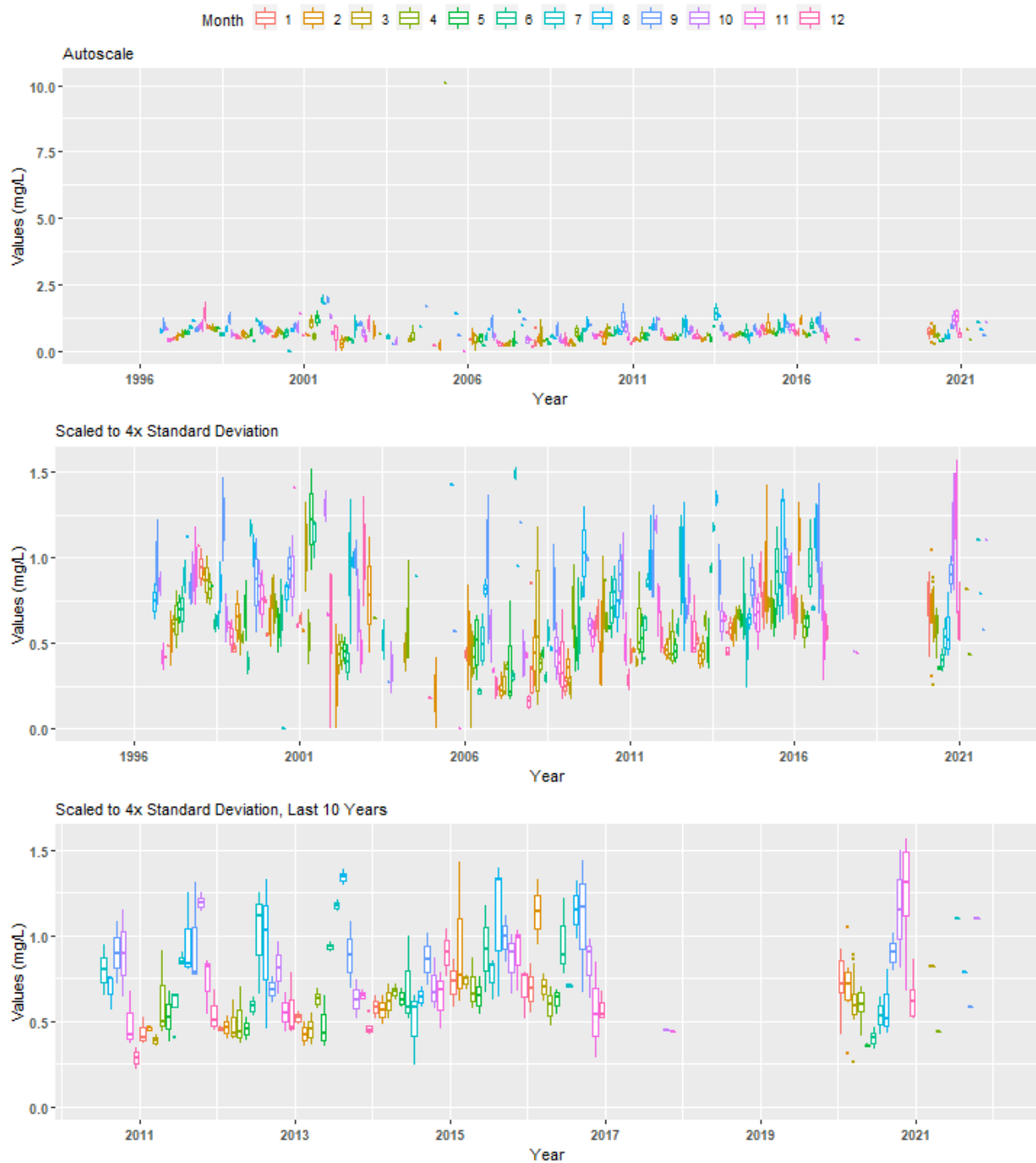
Summary Box Plots for Gasparilla Sound-Charlotte Harbor Aquatic Preserve

By Year



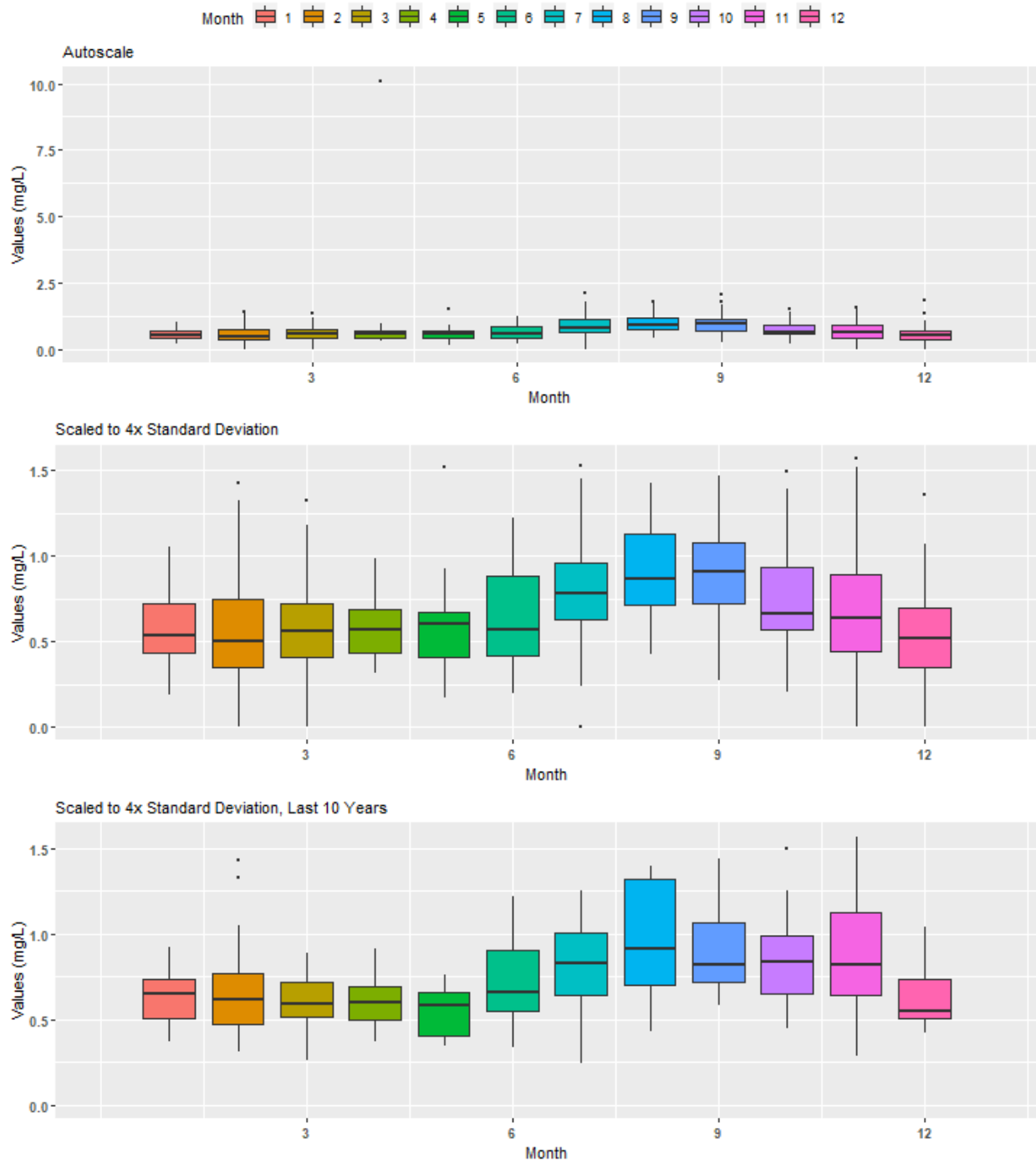
Summary Box Plots for Gasparilla Sound-Charlotte Harbor Aquatic Preserve

By Year & Month



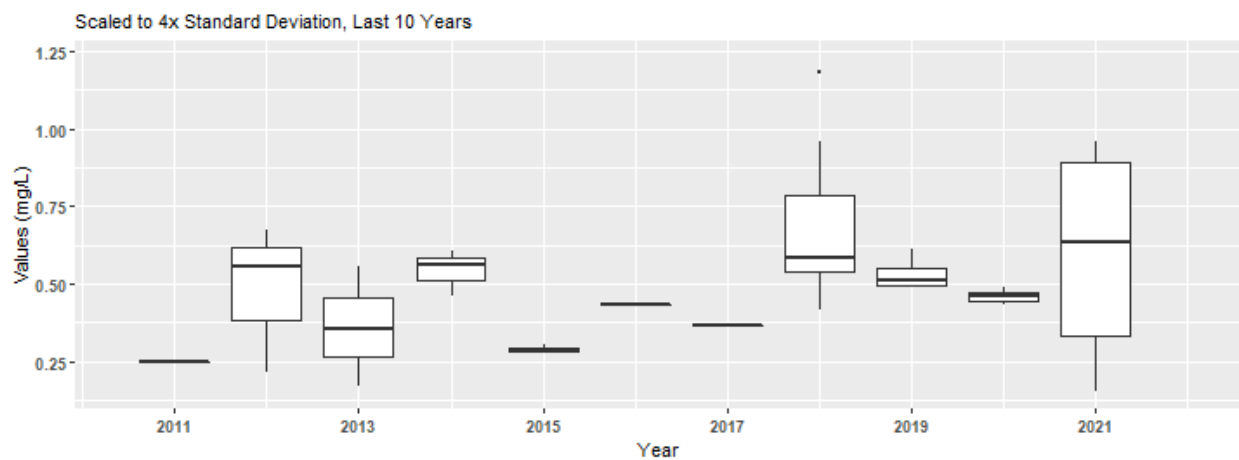
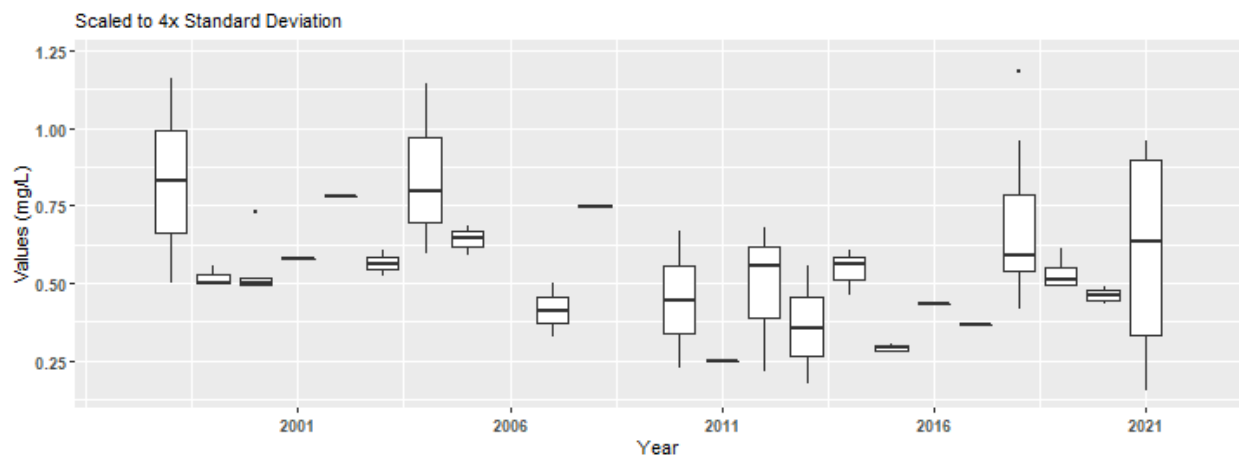
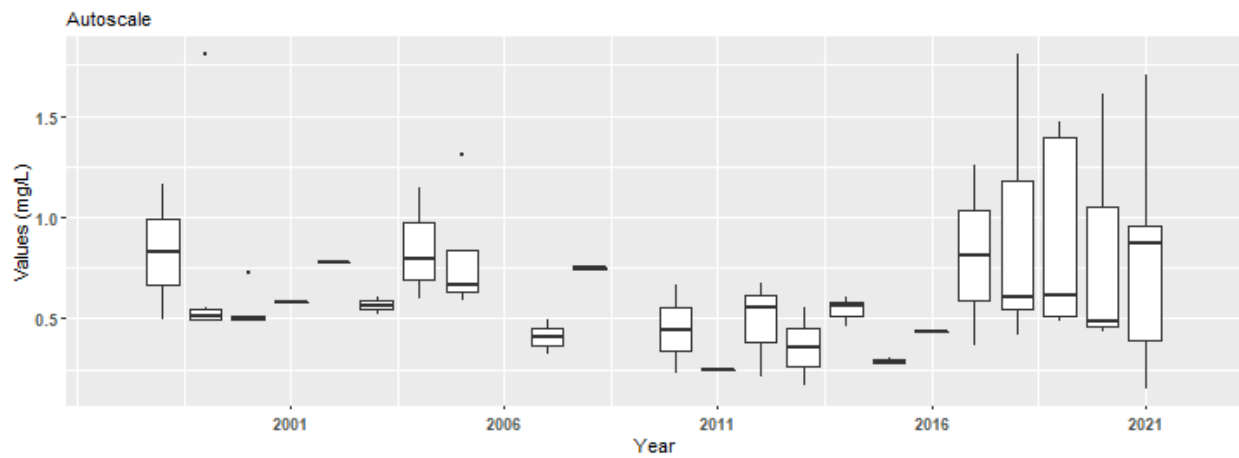
Summary Box Plots for Gasparilla Sound-Charlotte Harbor Aquatic Preserve

By Month



Summary Box Plots for Guana River Marsh Aquatic Preserve

By Year



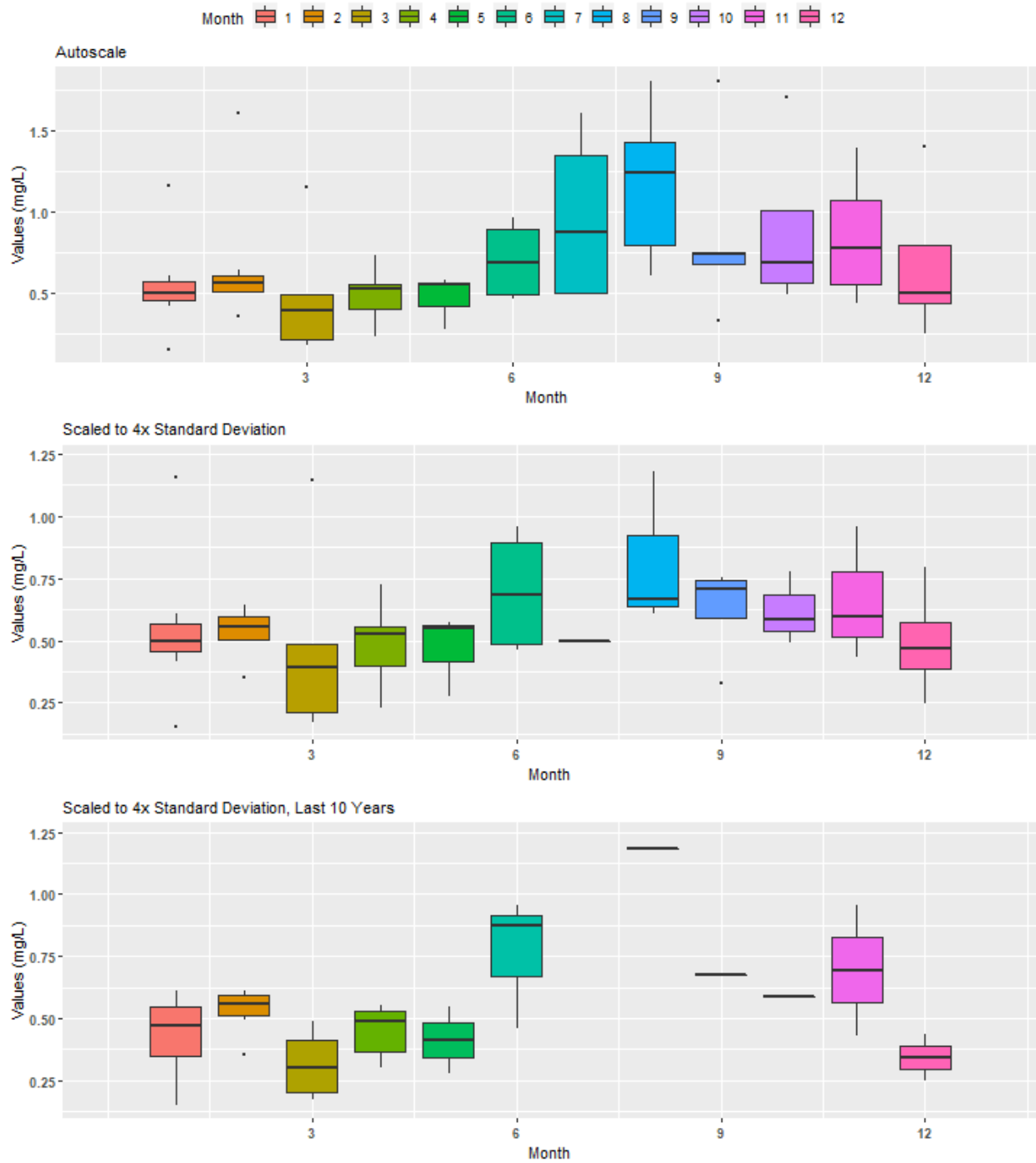
Summary Box Plots for Guana River Marsh Aquatic Preserve

By Year & Month



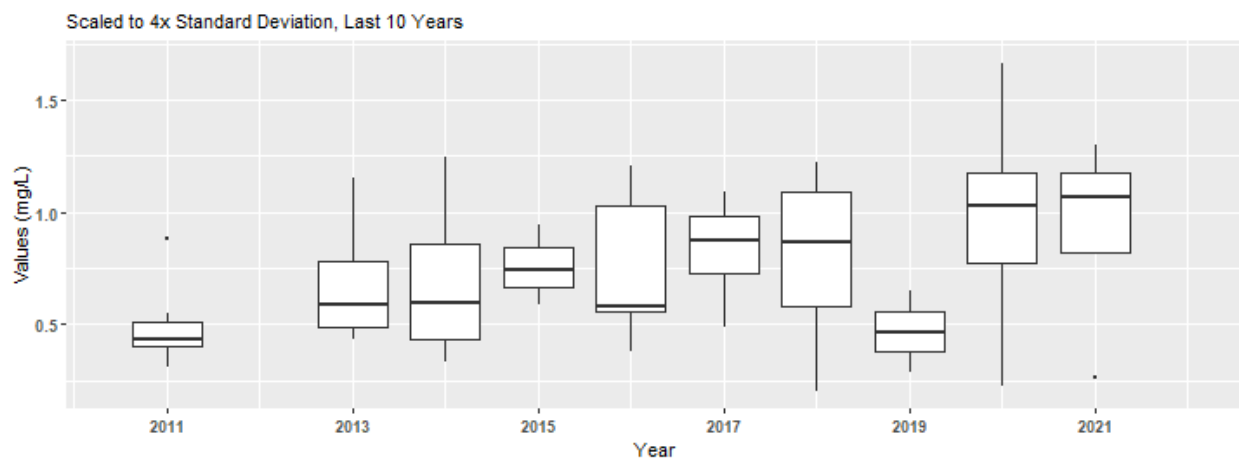
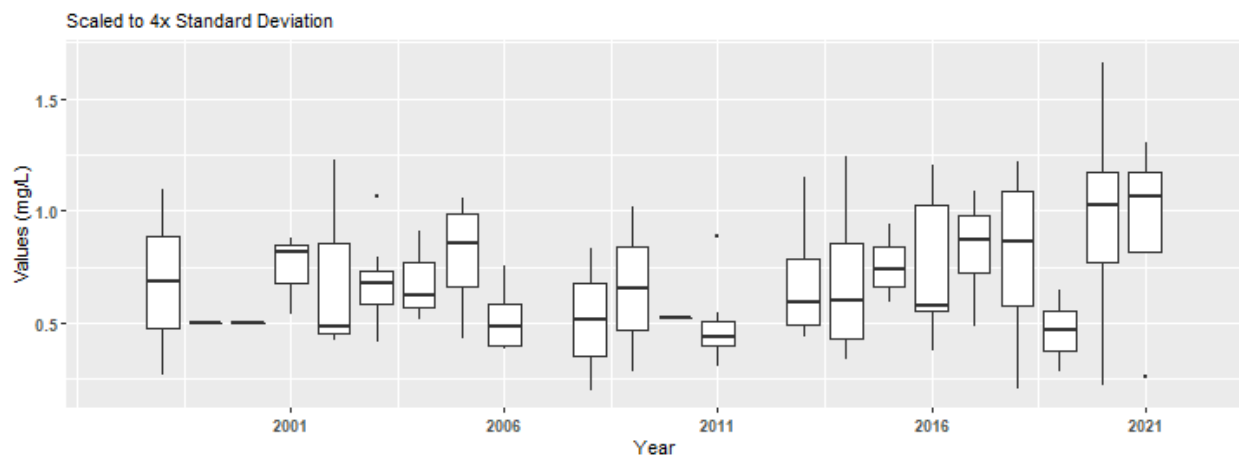
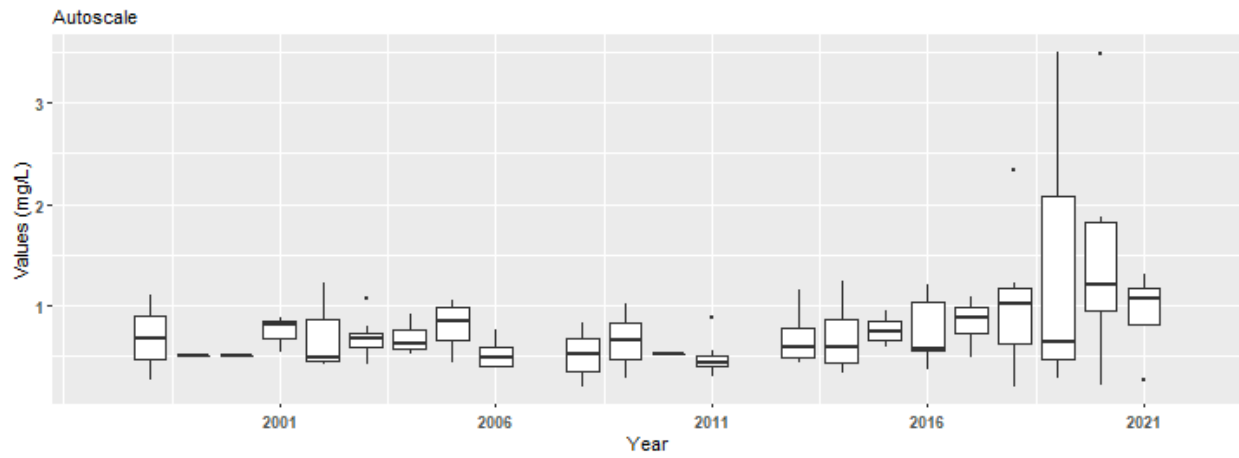
Summary Box Plots for Guana River Marsh Aquatic Preserve

By Month



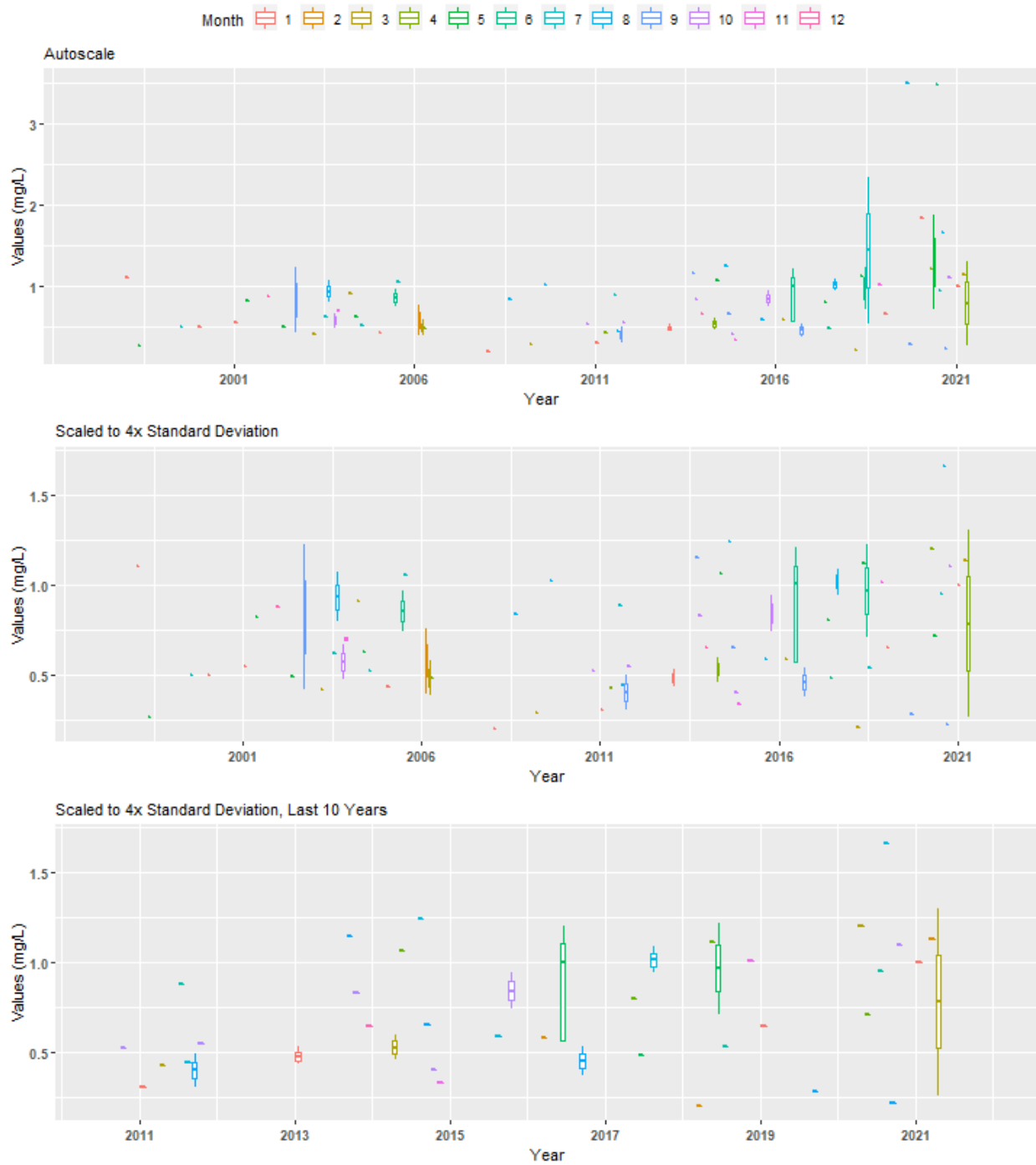
Summary Box Plots for Guana Tolomato Matanzas National Estuarine Research Reserve

By Year



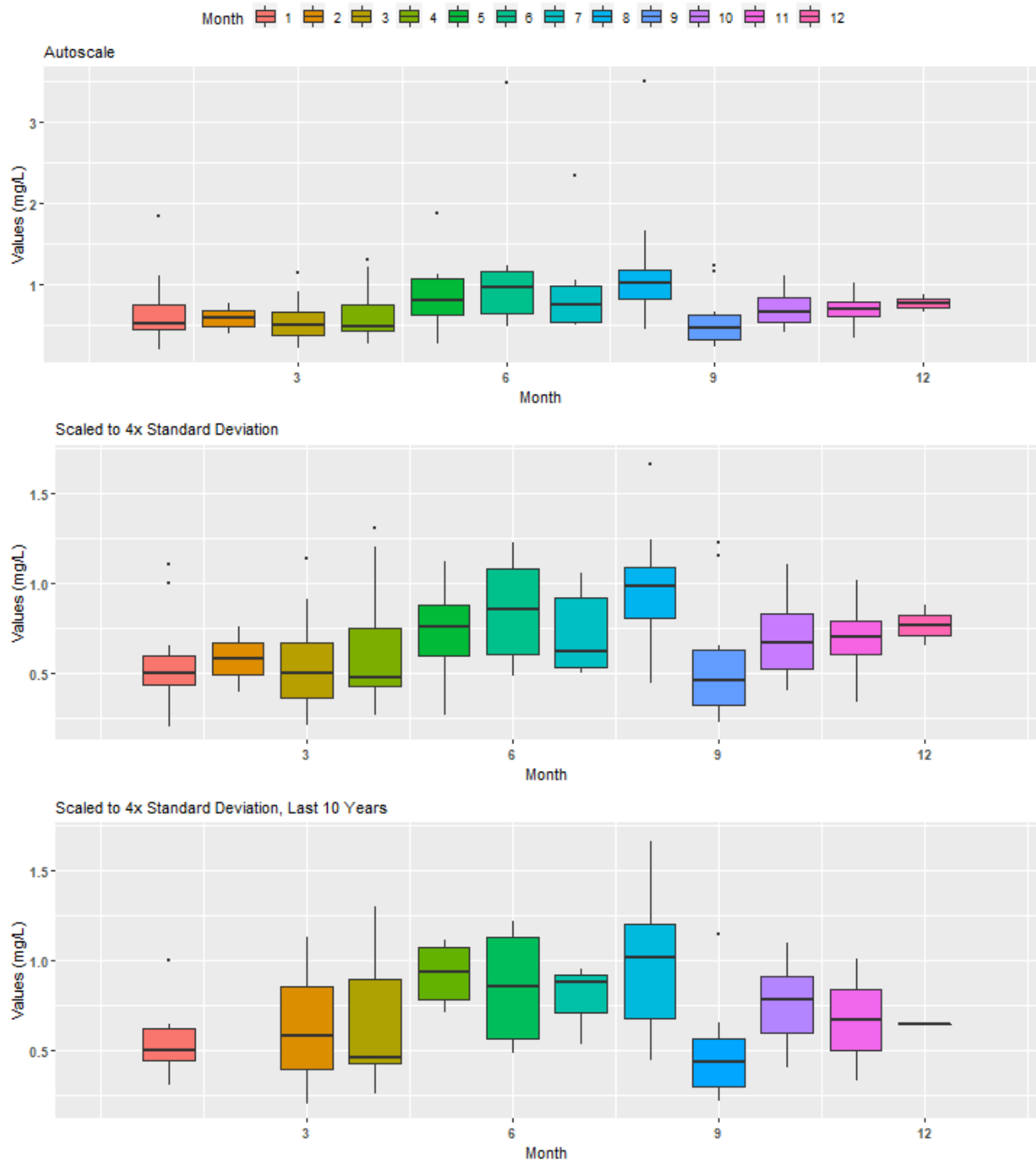
Summary Box Plots for Guana Tolomato Matanzas National Estuarine Research Reserve

By Year & Month



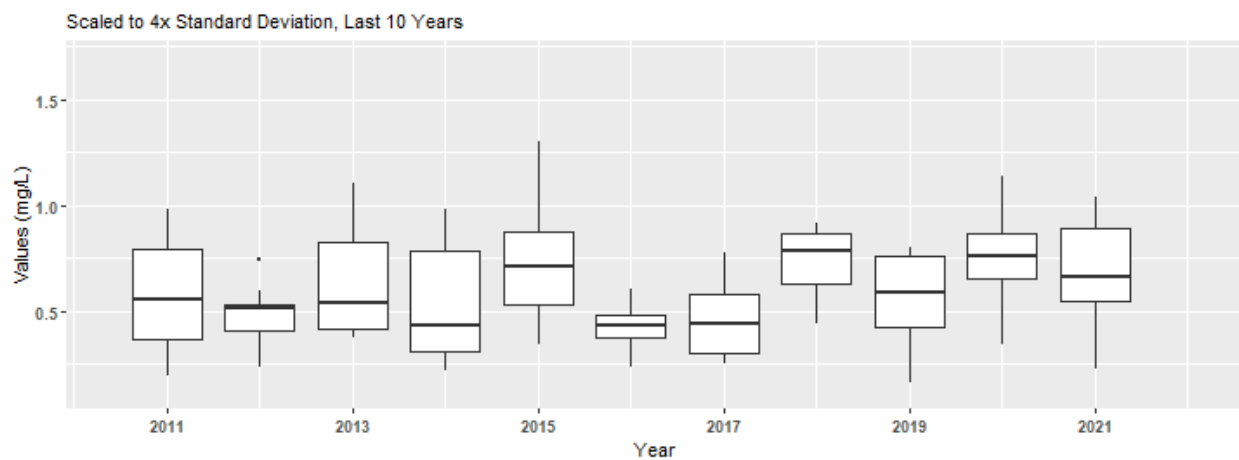
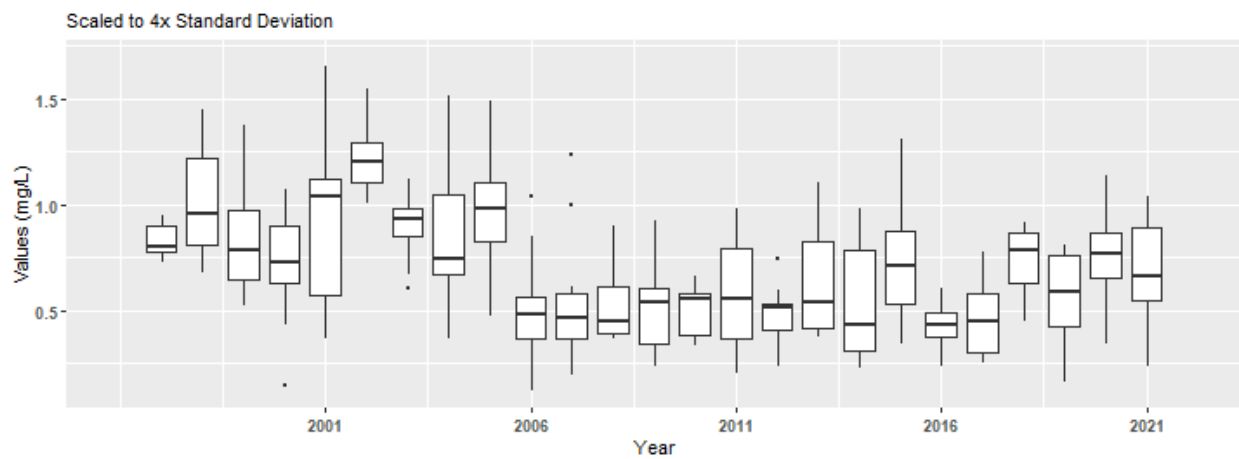
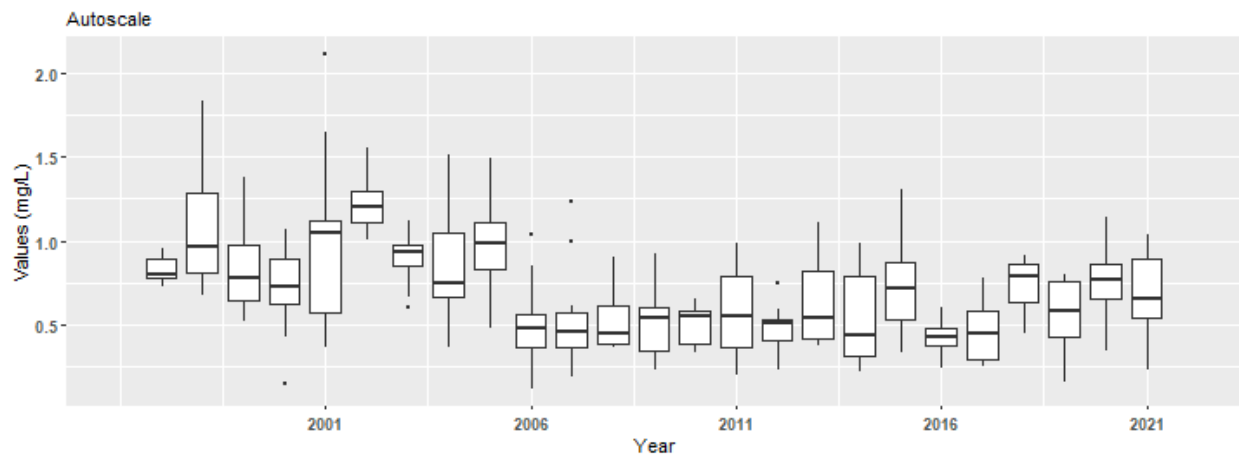
Summary Box Plots for Guana Tolomato Matanzas National Estuarine Research Reserve

By Month



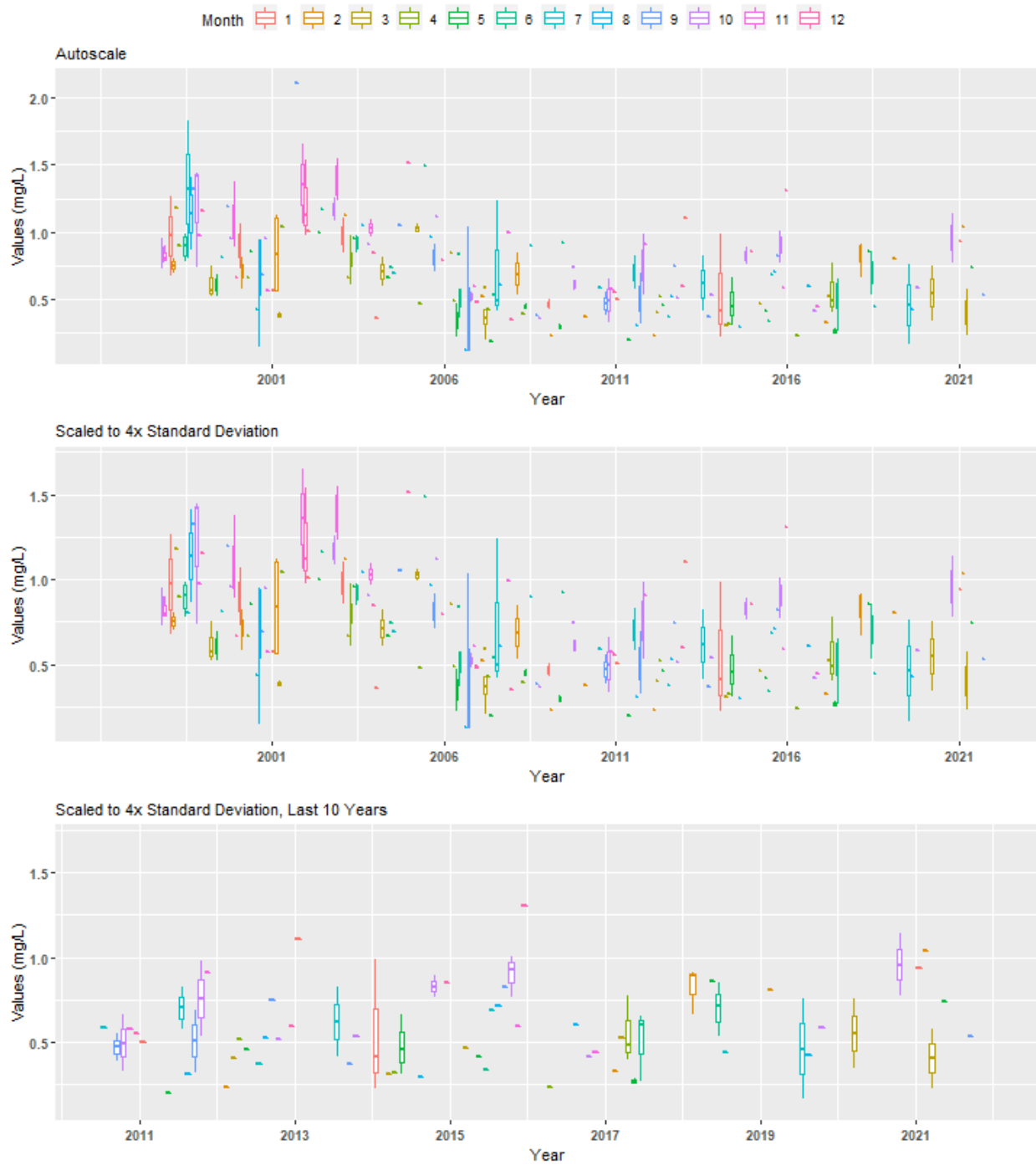
Summary Box Plots for Indian River-Malabar to Vero Beach Aquatic Preserve

By Year



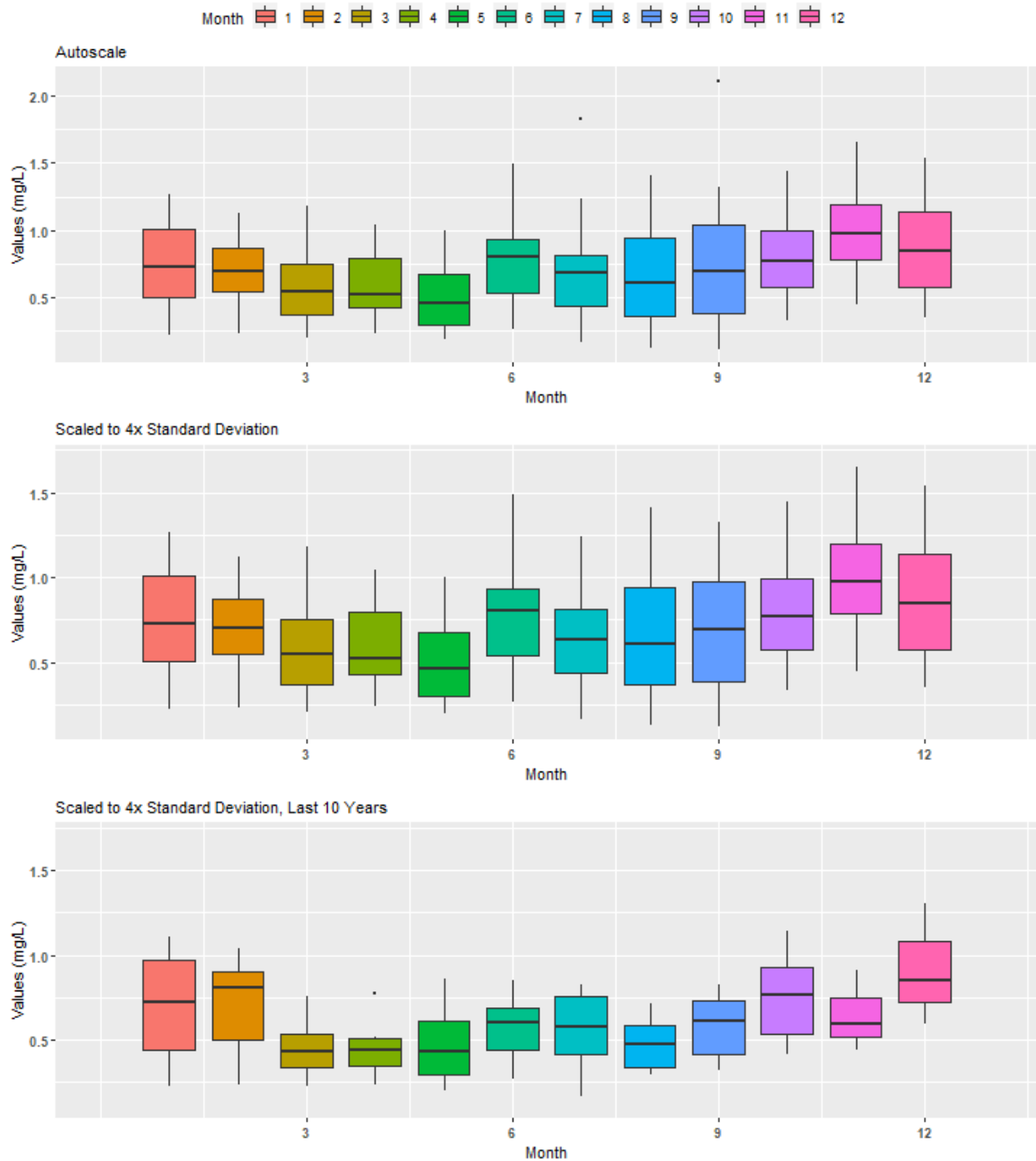
Summary Box Plots for Indian River-Malabar to Vero Beach Aquatic Preserve

By Year & Month



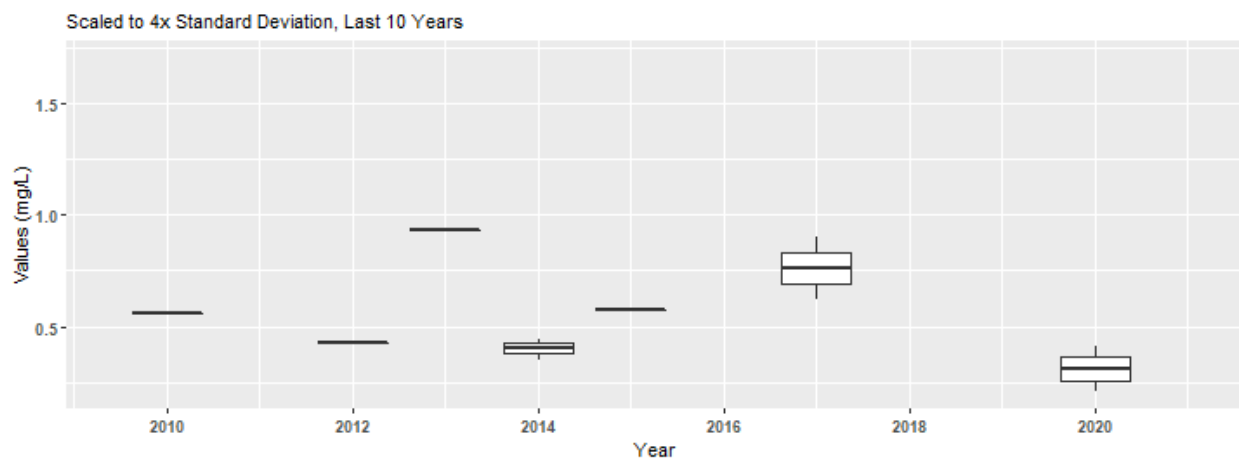
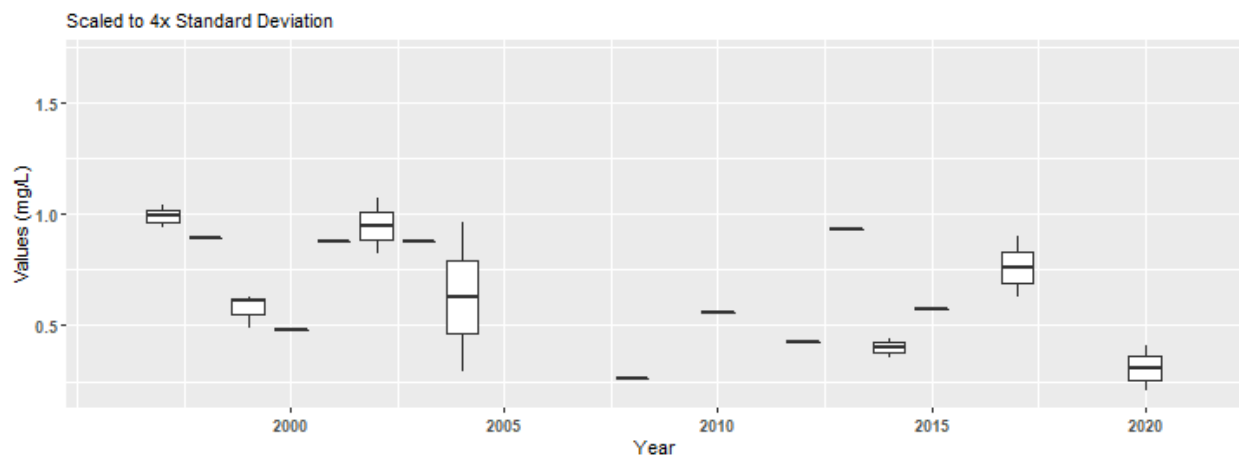
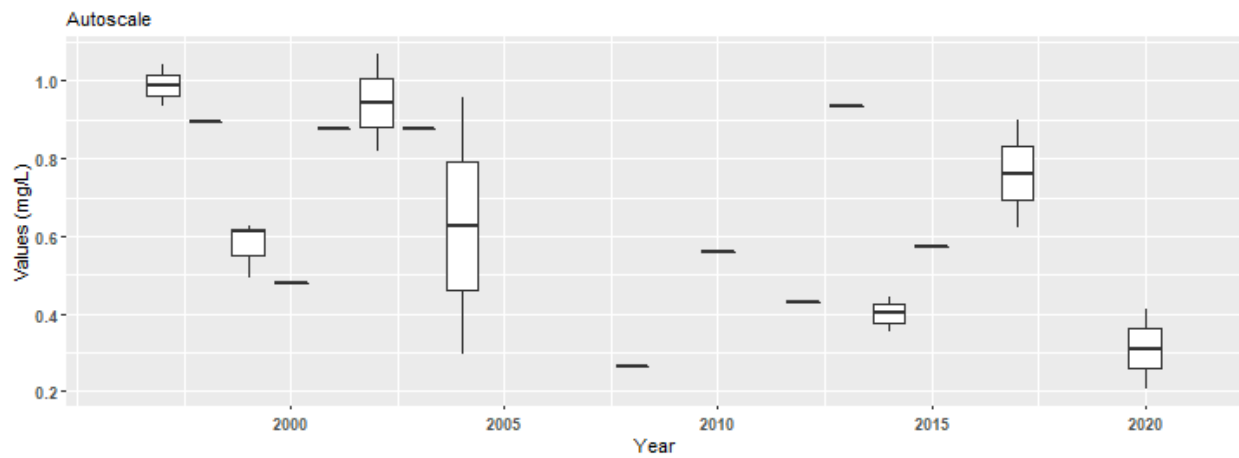
Summary Box Plots for Indian River-Malabar to Vero Beach Aquatic Preserve

By Month



Summary Box Plots for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

By Year



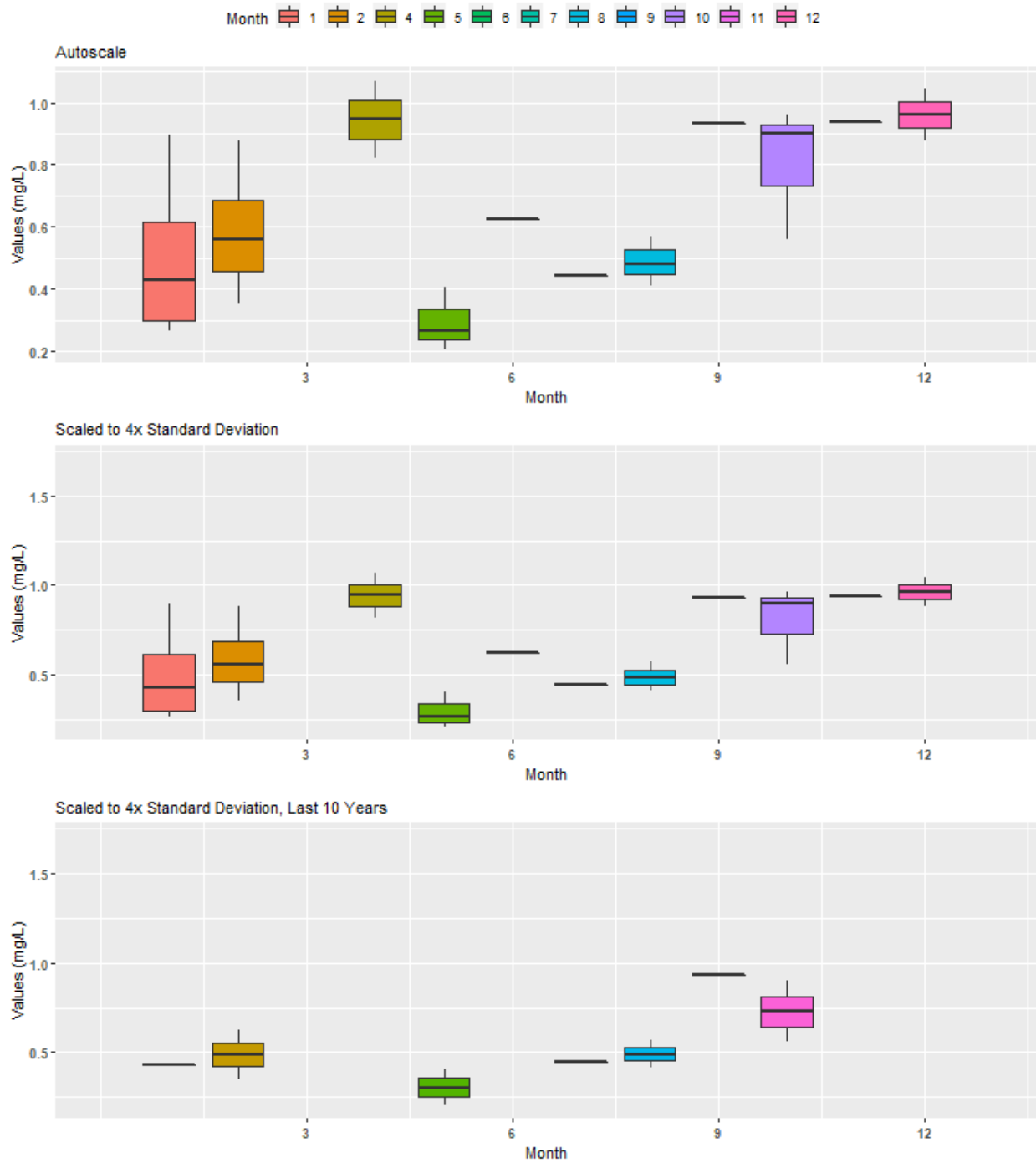
Summary Box Plots for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

By Year & Month

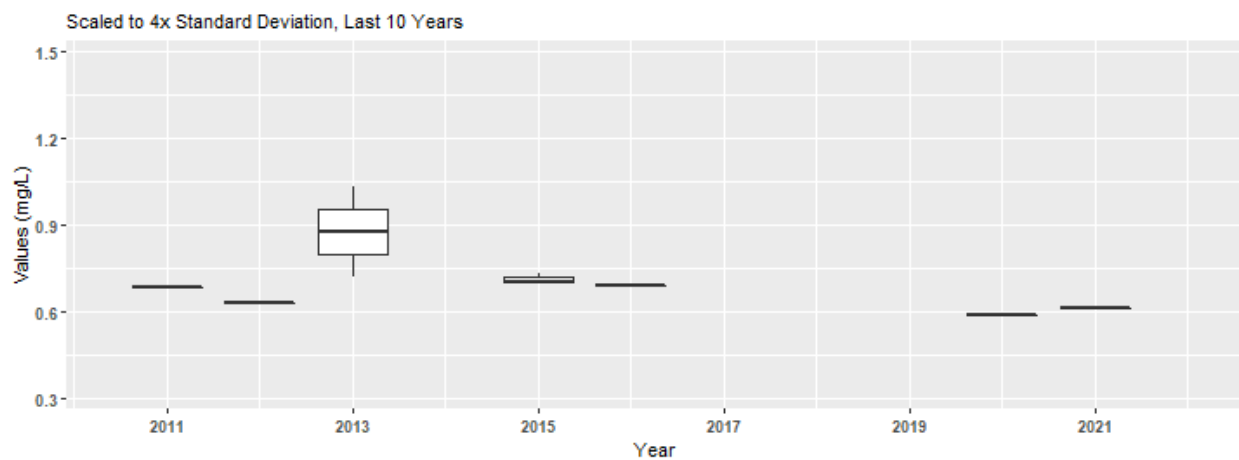
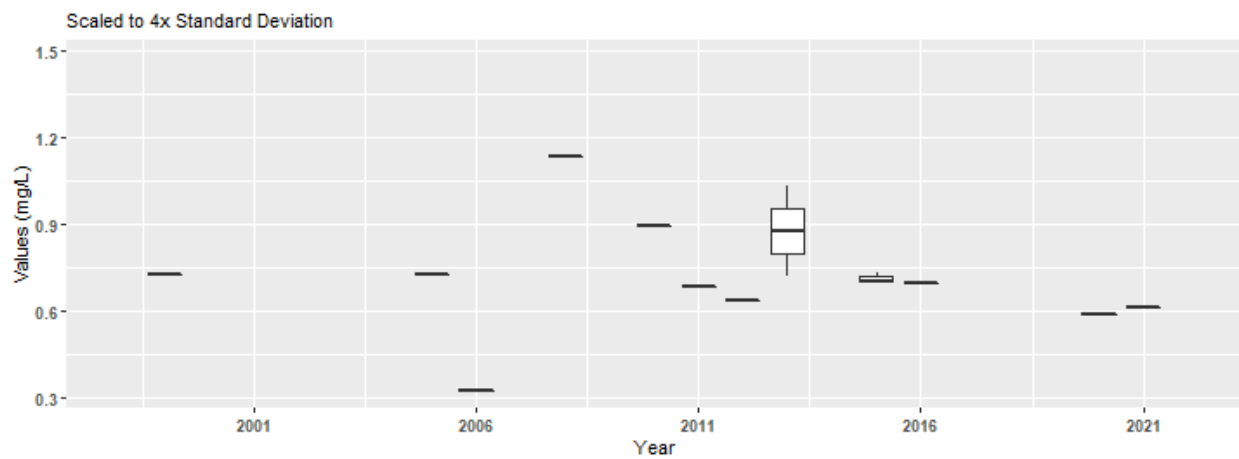
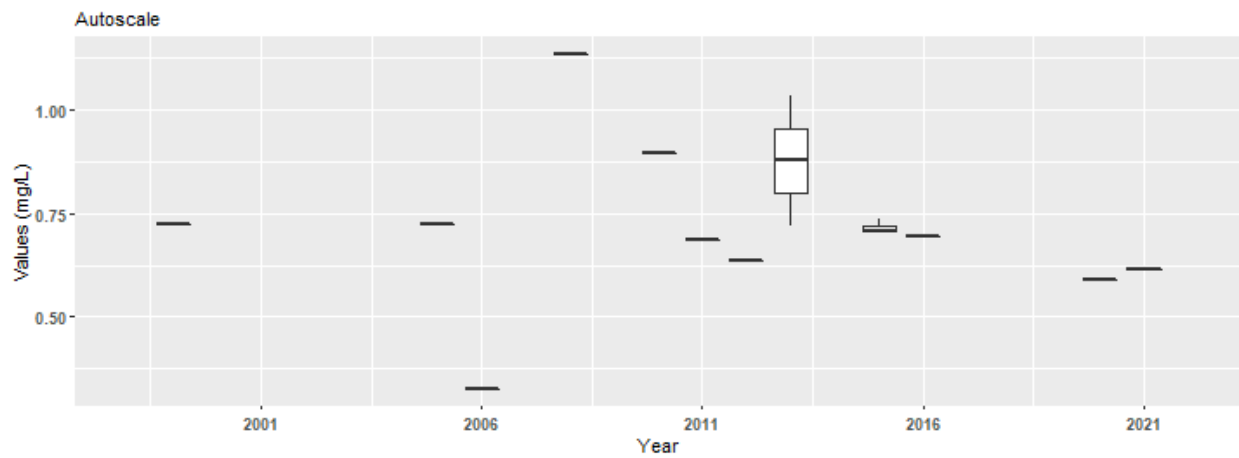


Summary Box Plots for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

By Month

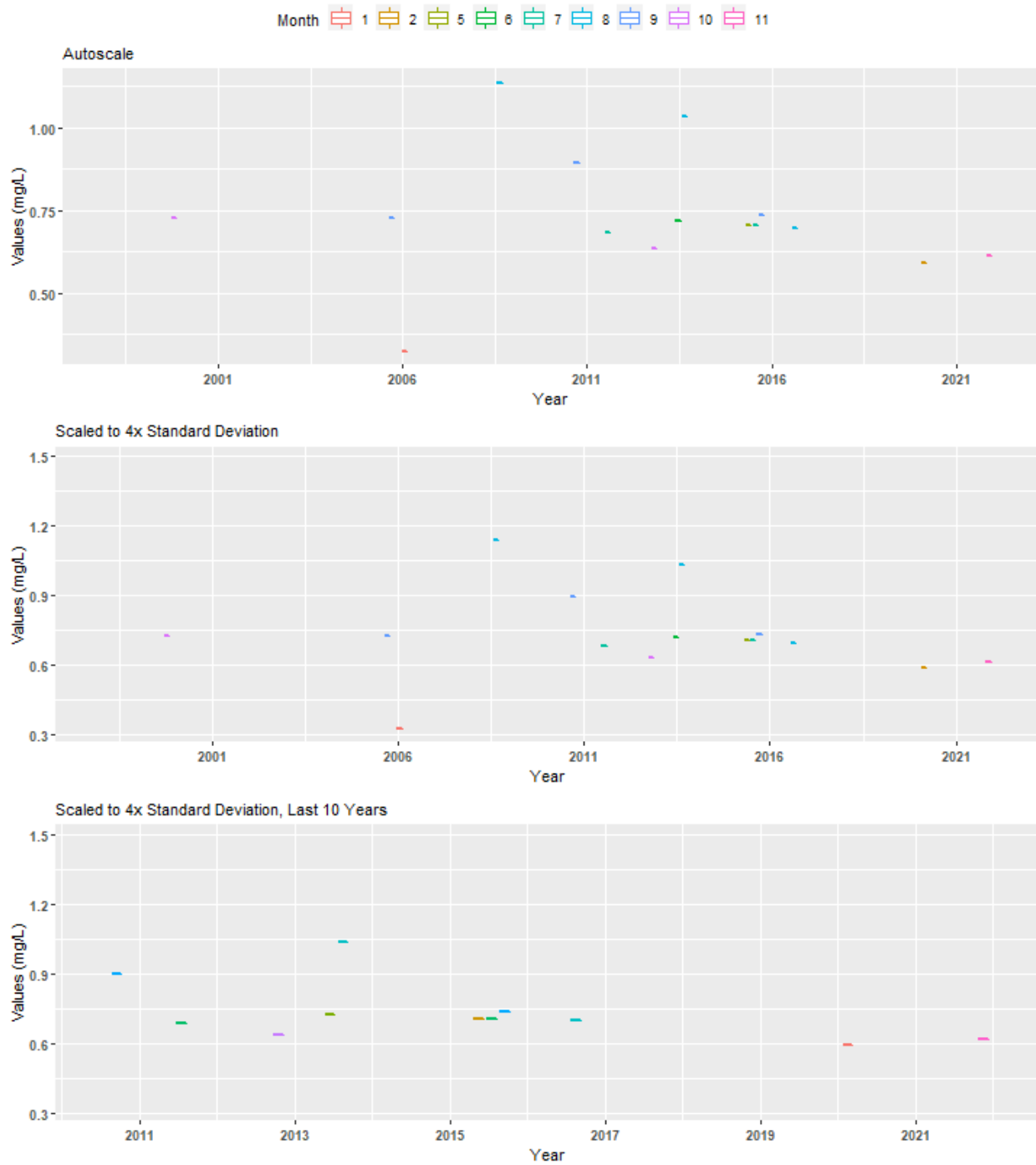


Summary Box Plots for Lemon Bay Aquatic Preserve By Year



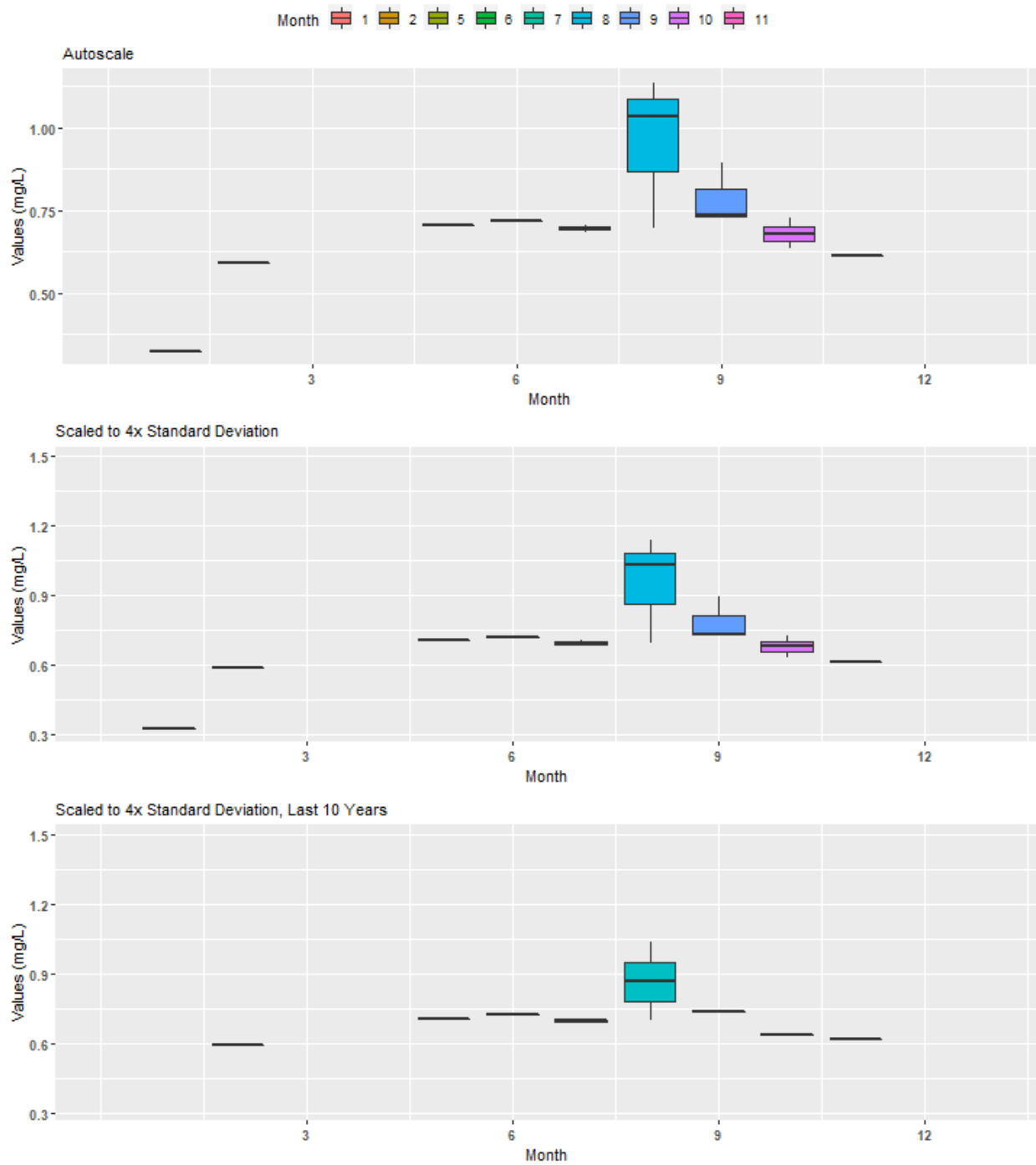
Summary Box Plots for Lemon Bay Aquatic Preserve

By Year & Month



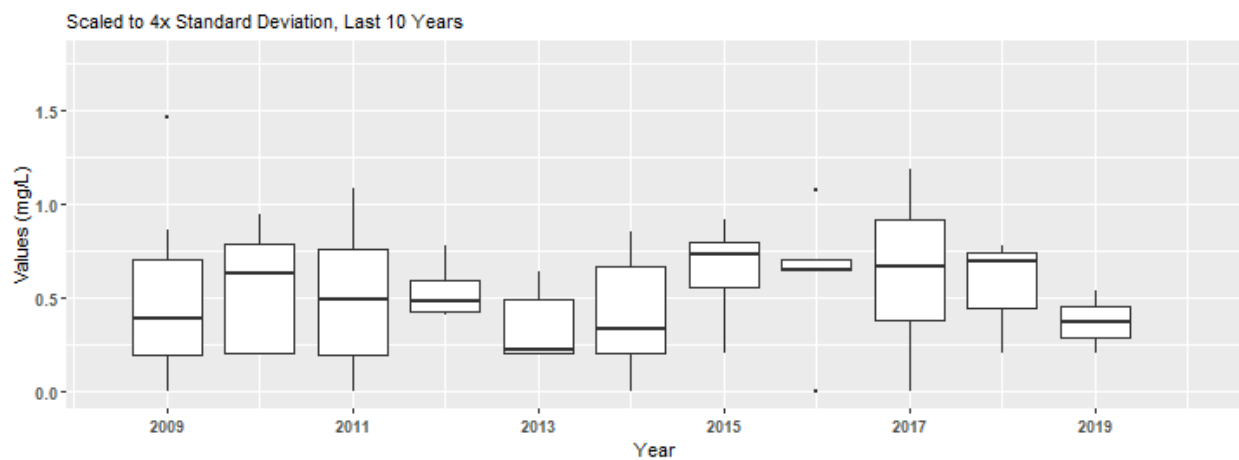
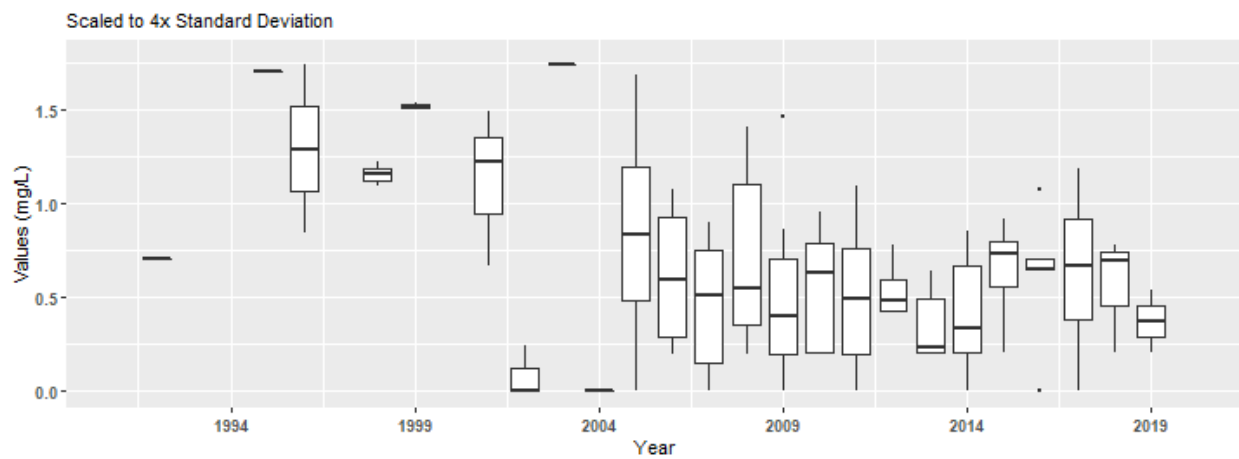
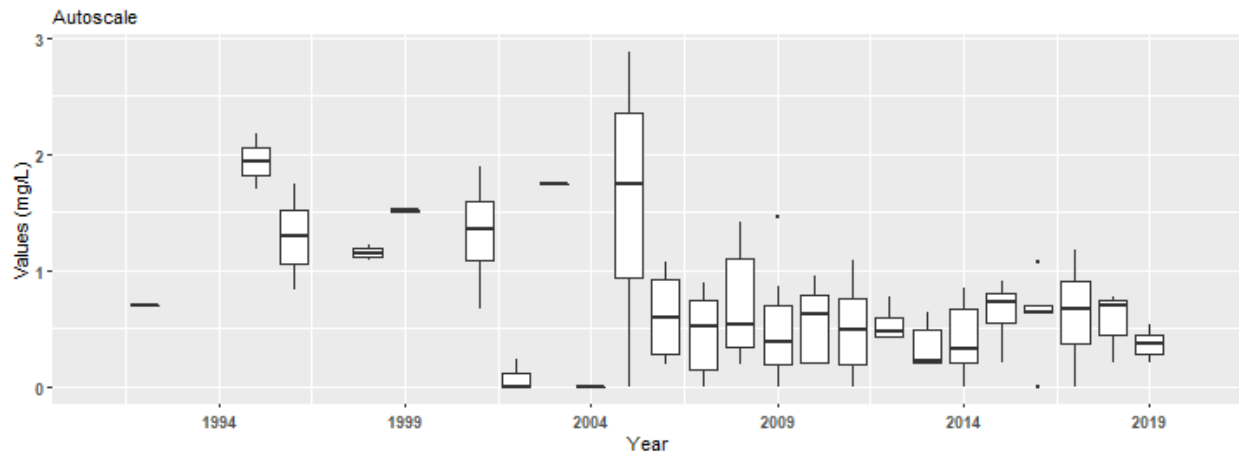
Summary Box Plots for Lemon Bay Aquatic Preserve

By Month



Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve

By Year



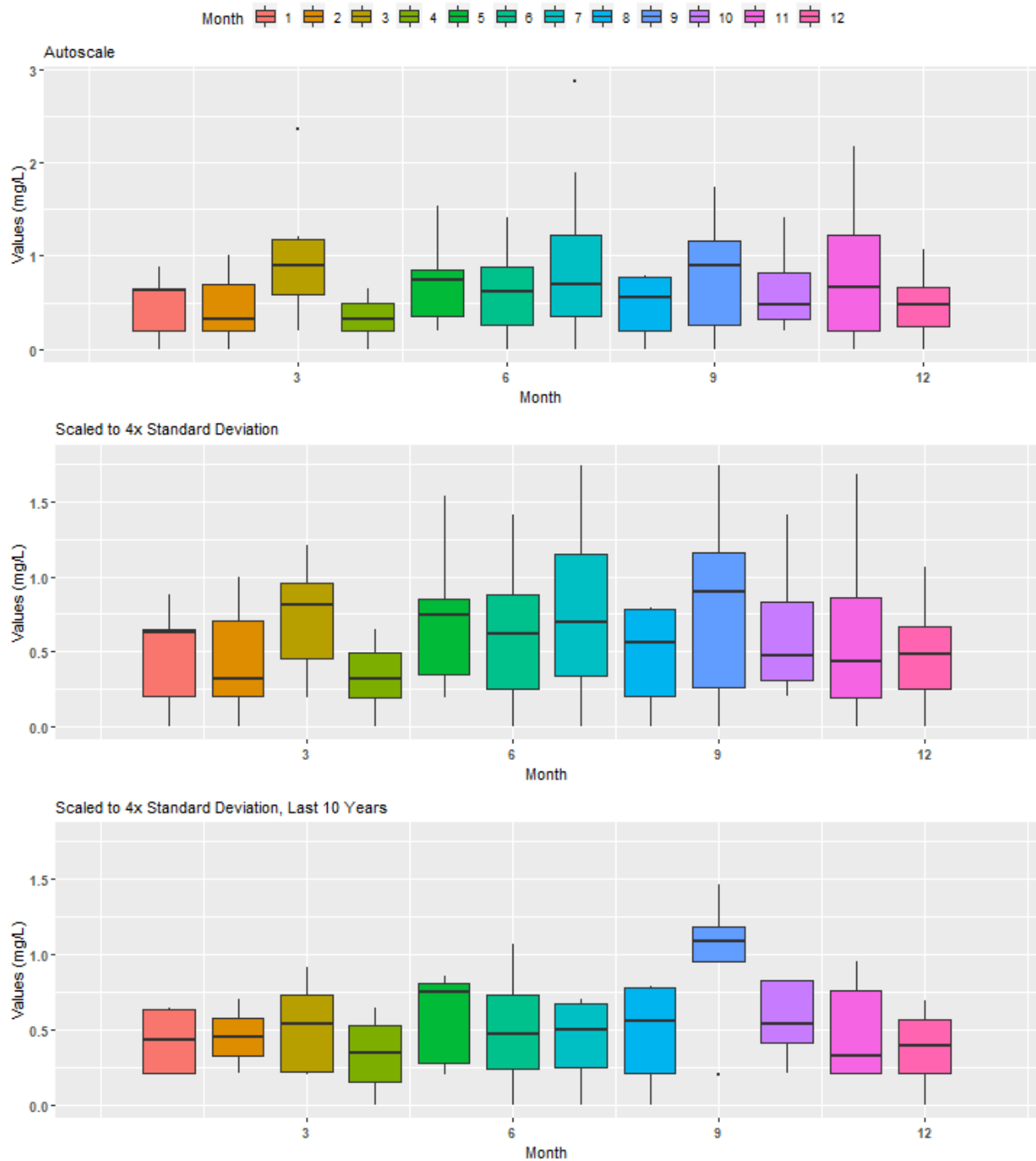
Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve

By Year & Month



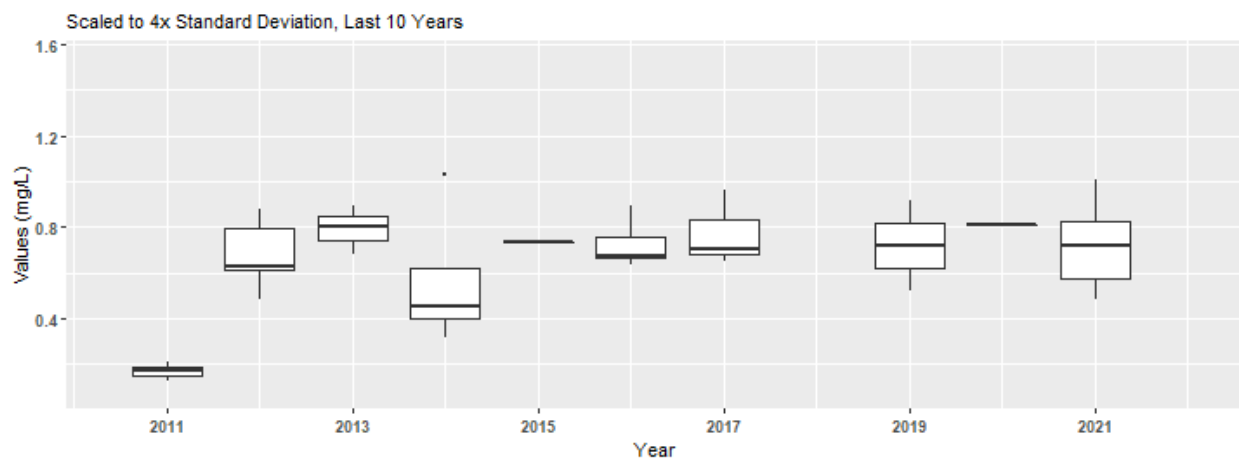
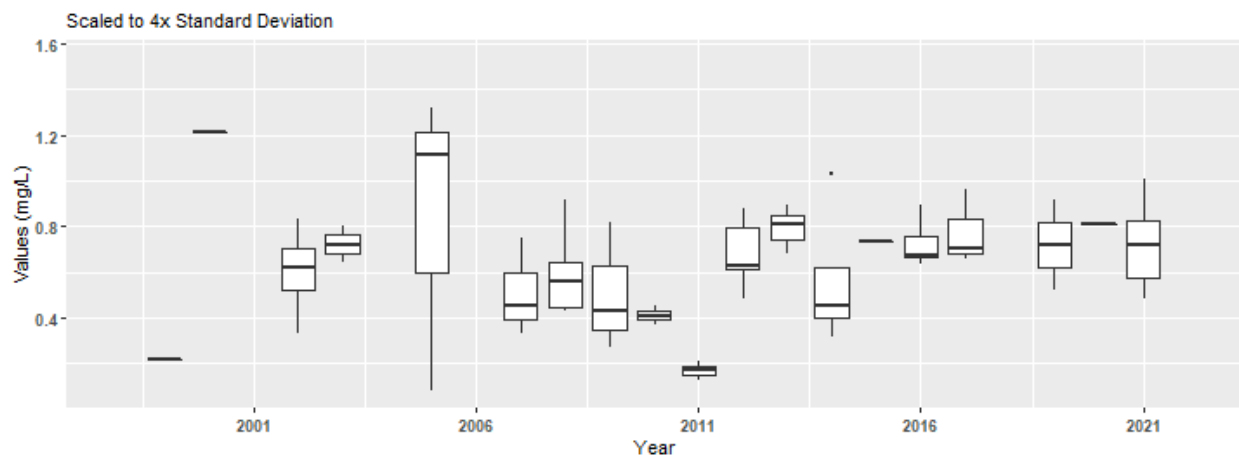
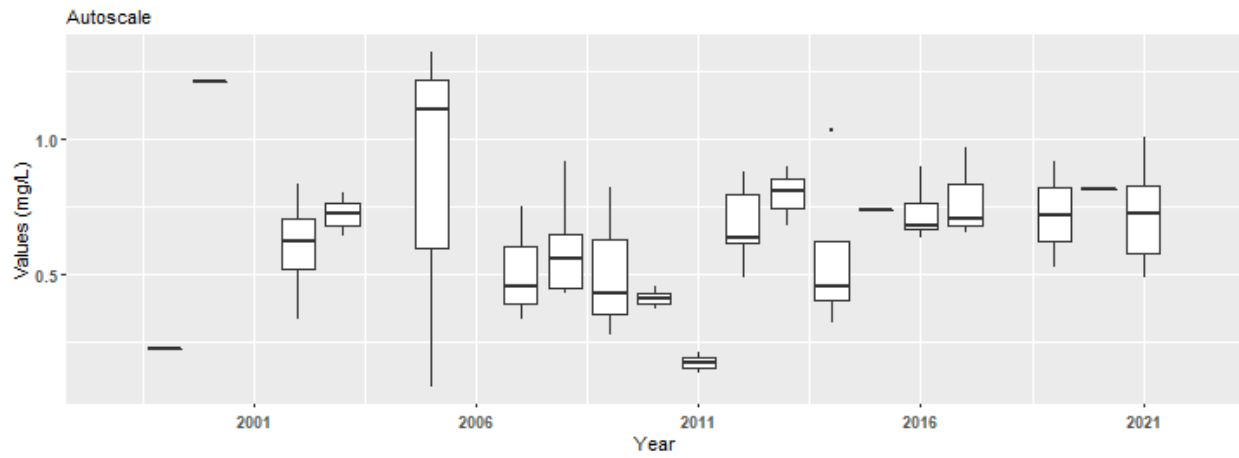
Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve

By Month



Summary Box Plots for Matlacha Pass Aquatic Preserve

By Year



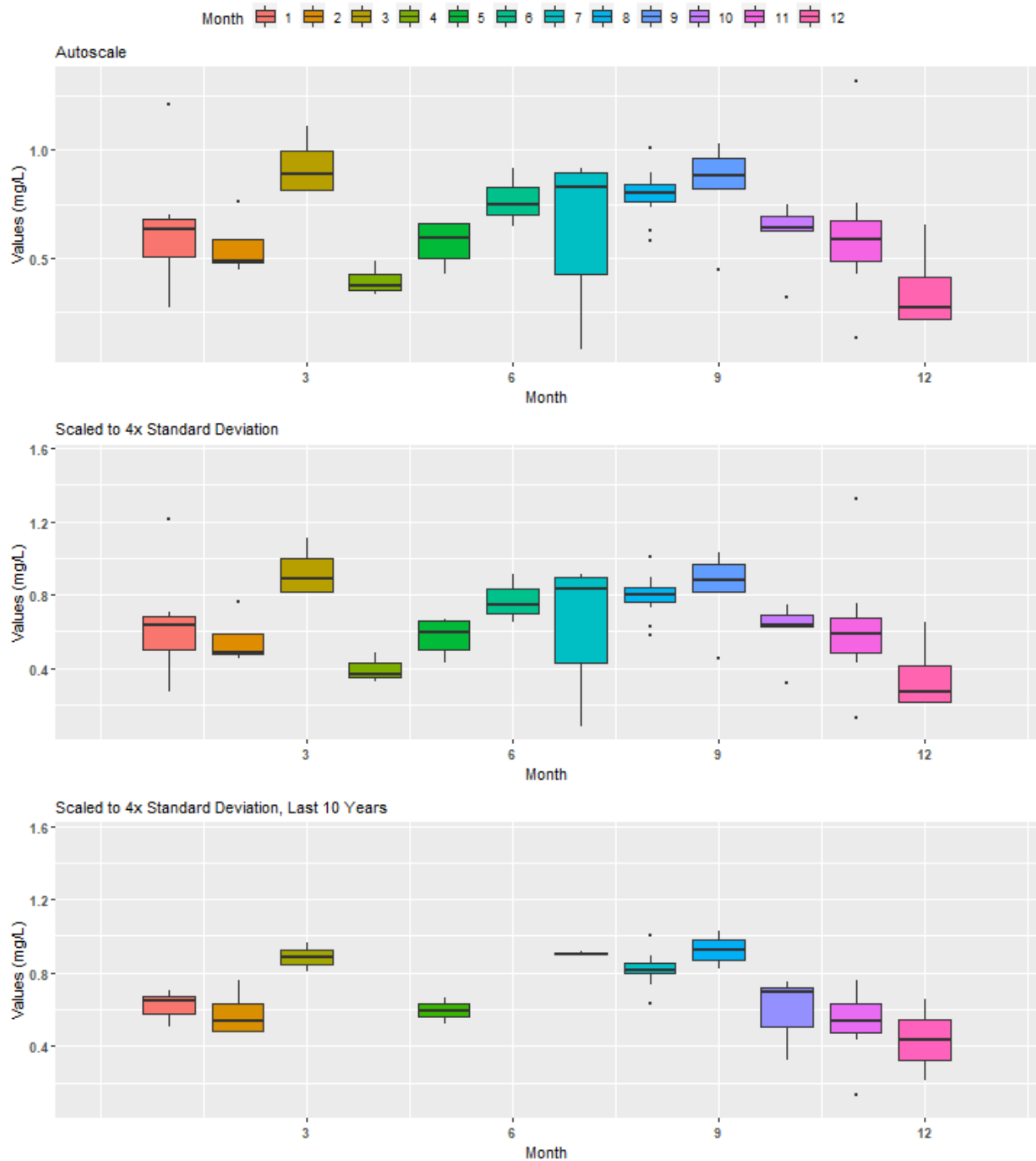
Summary Box Plots for Matlacha Pass Aquatic Preserve

By Year & Month

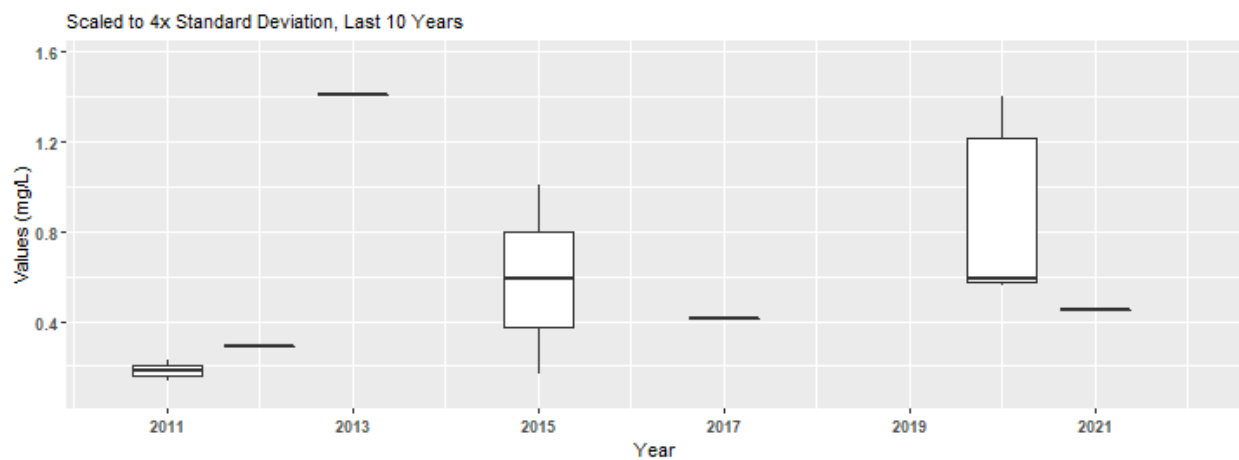
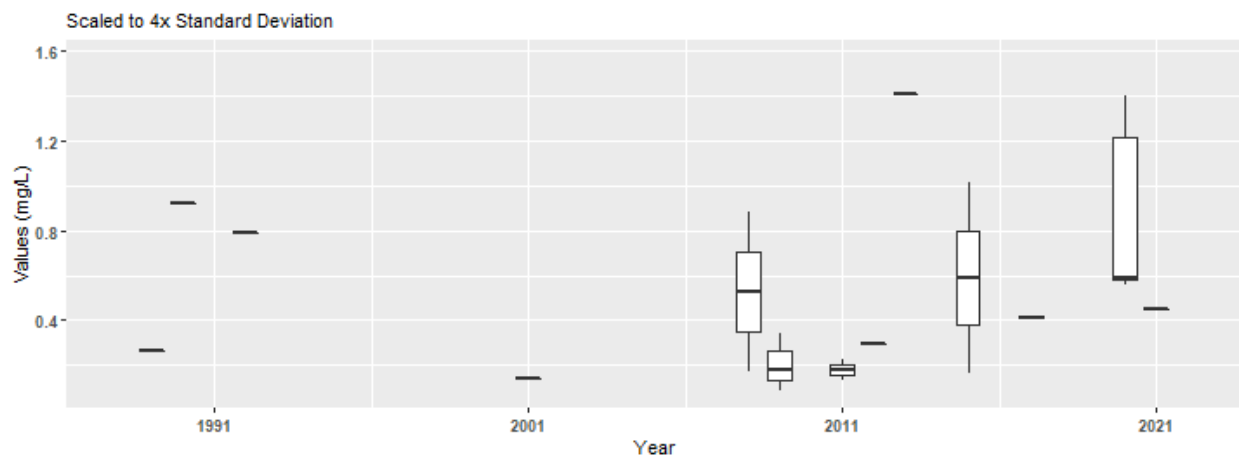
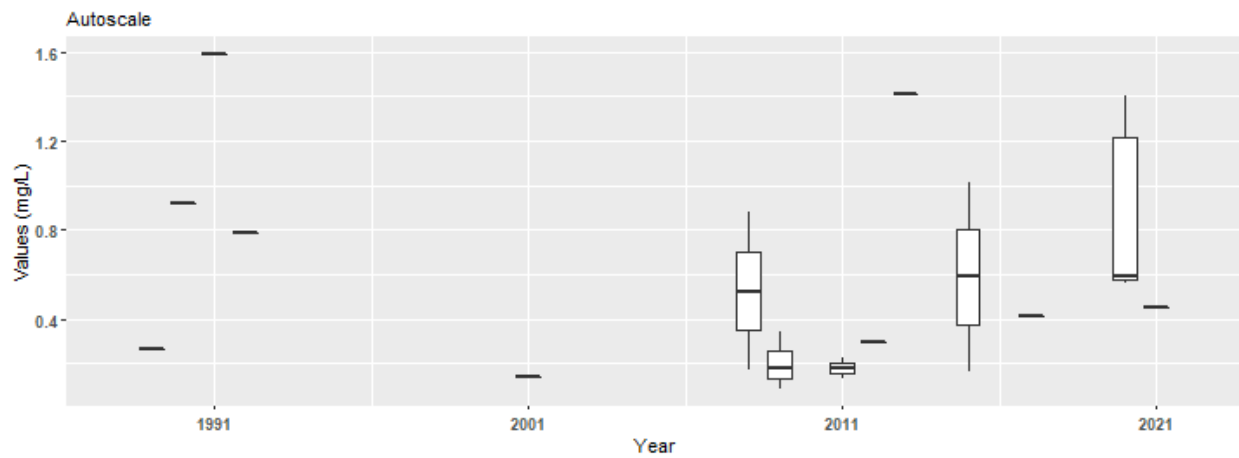


Summary Box Plots for Matlacha Pass Aquatic Preserve

By Month

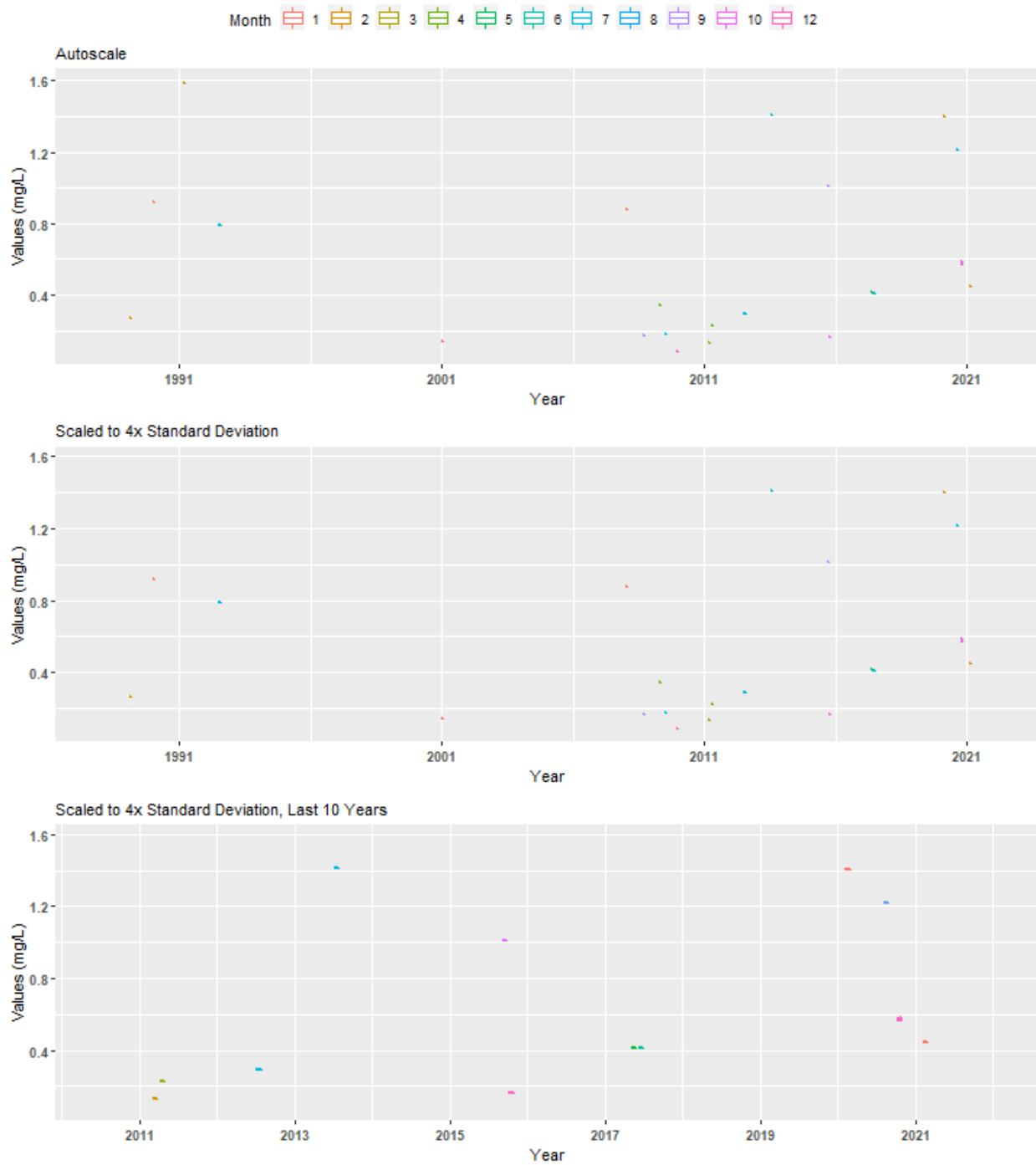


Summary Box Plots for Nassau River-St. Johns River Marshes Aquatic Preserve By Year

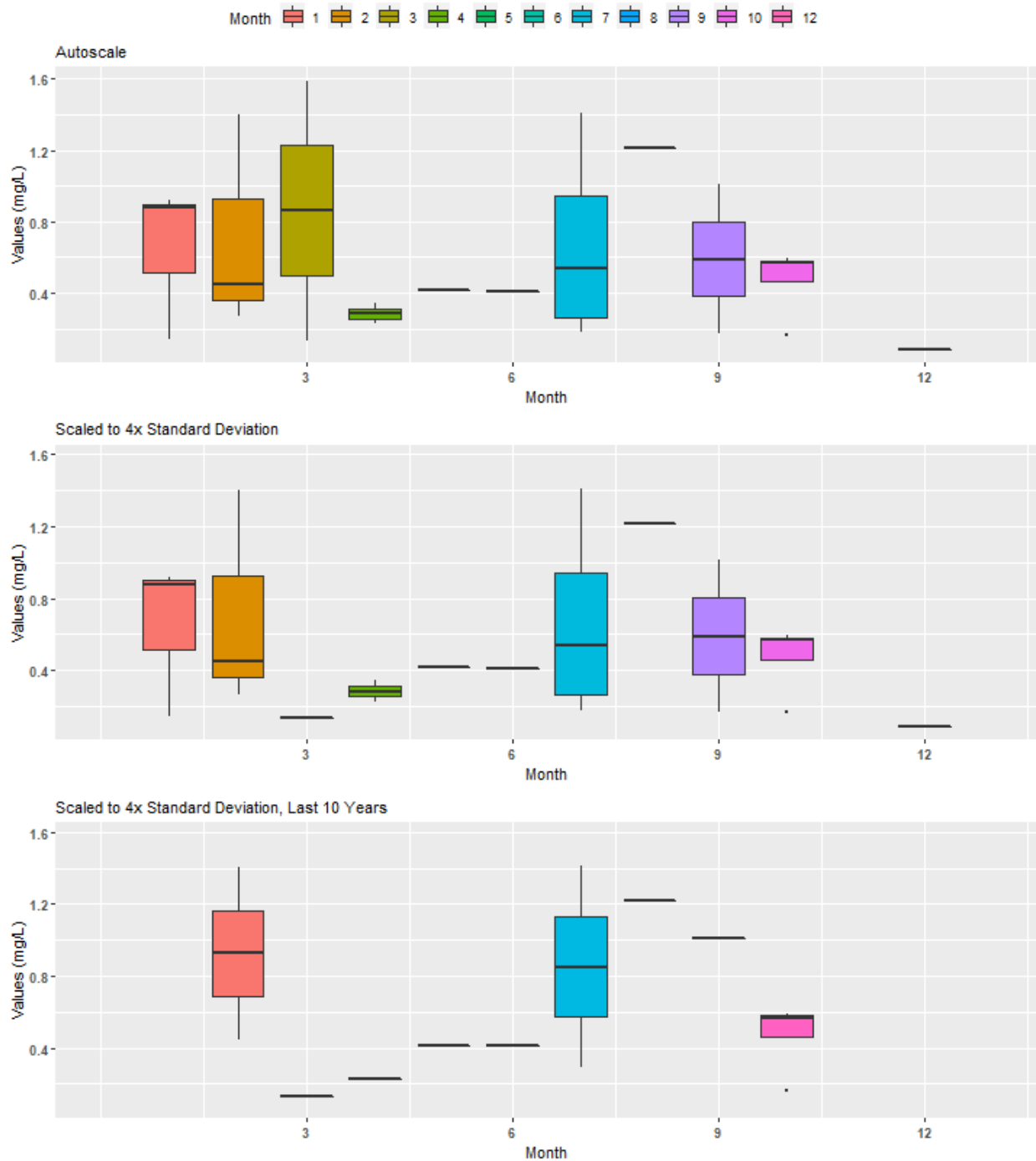


Summary Box Plots for Nassau River-St. Johns River Marshes Aquatic Preserve

By Year & Month

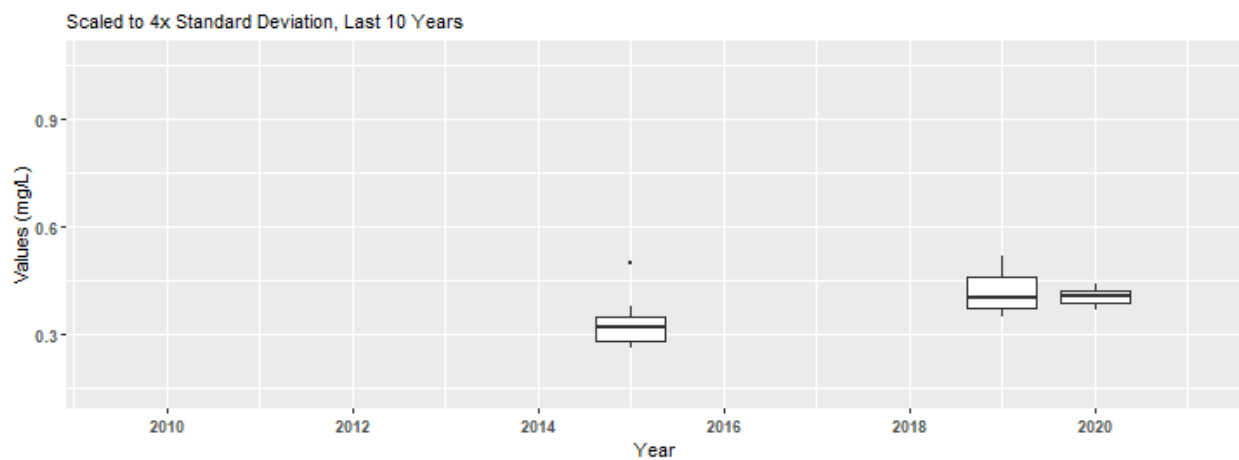
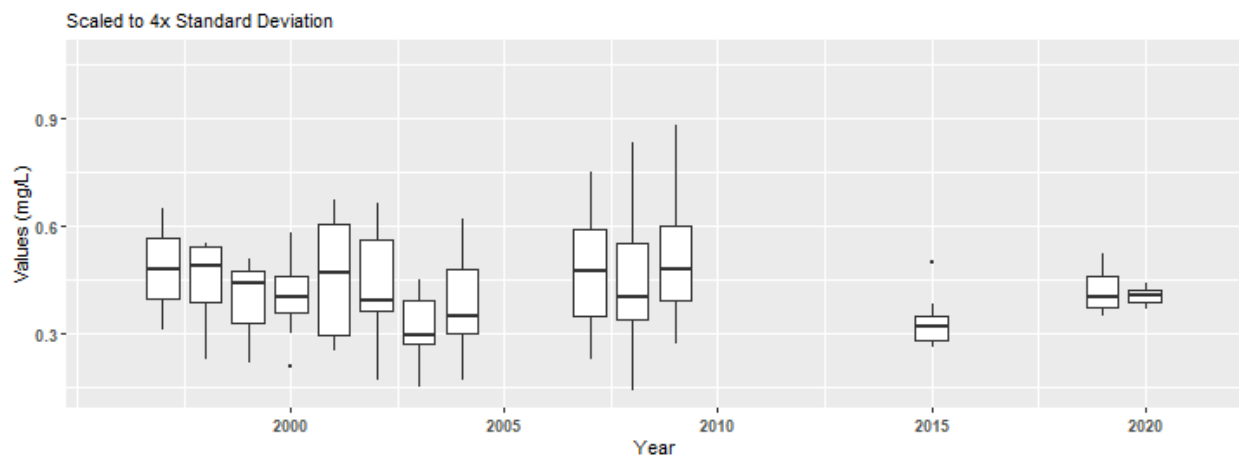
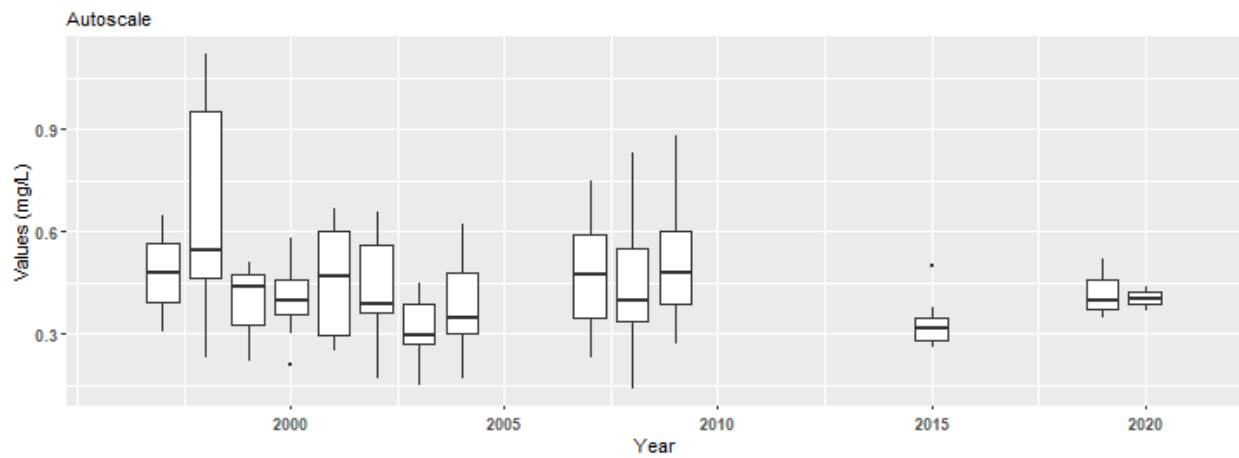


Summary Box Plots for Nassau River-St. Johns River Marshes Aquatic Preserve By Month



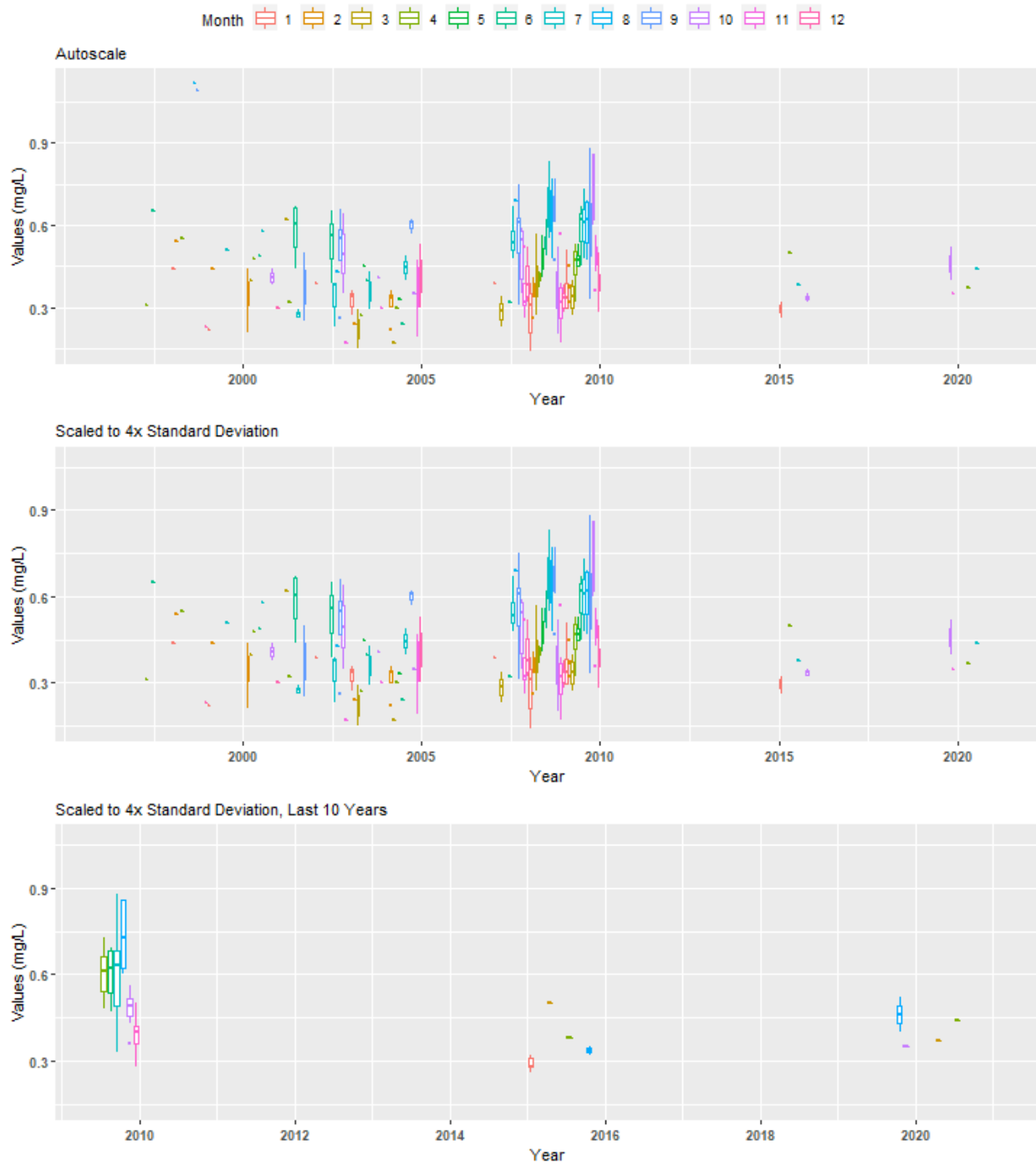
Summary Box Plots for Nature Coast Aquatic Preserve

By Year



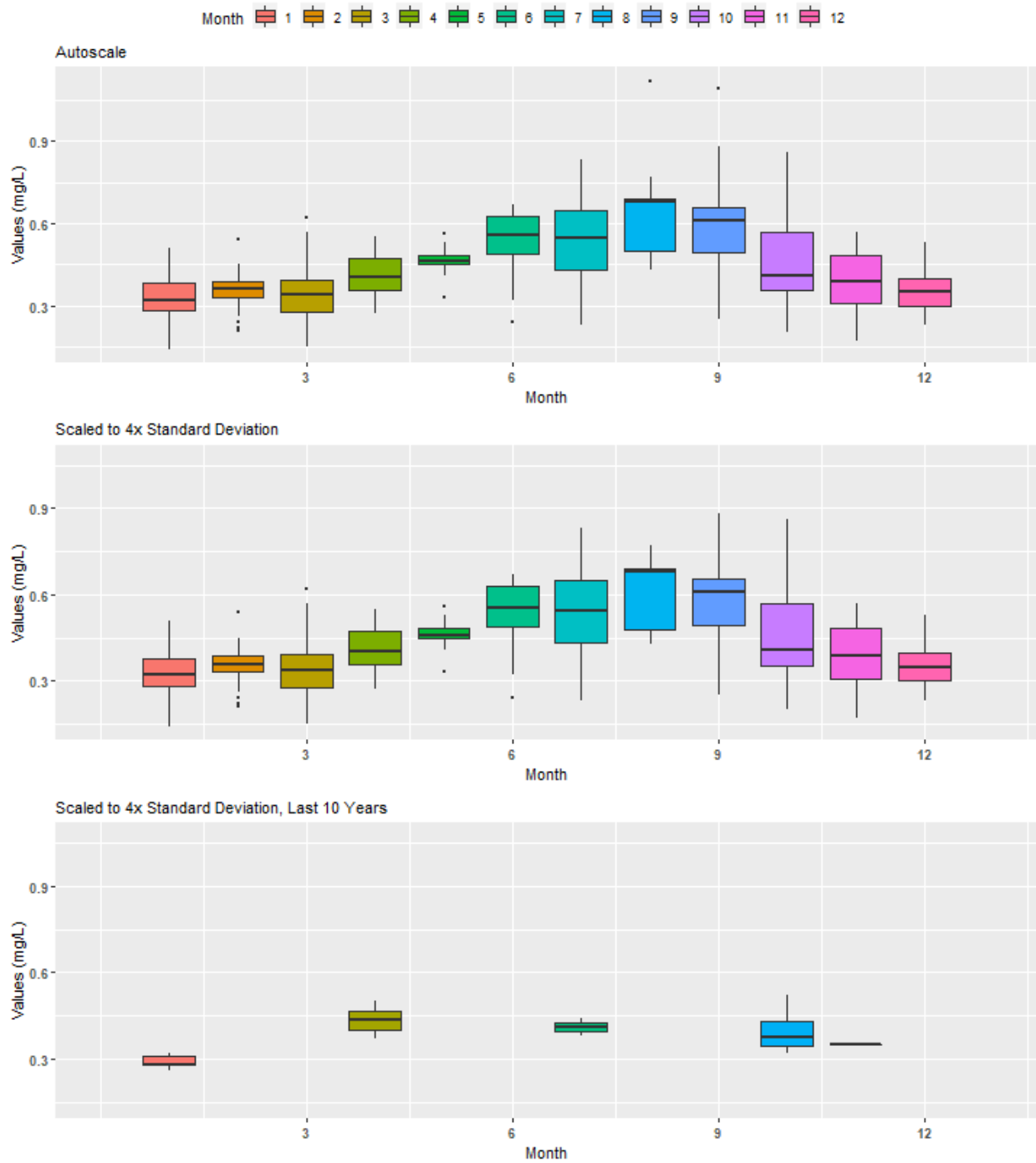
Summary Box Plots for Nature Coast Aquatic Preserve

By Year & Month



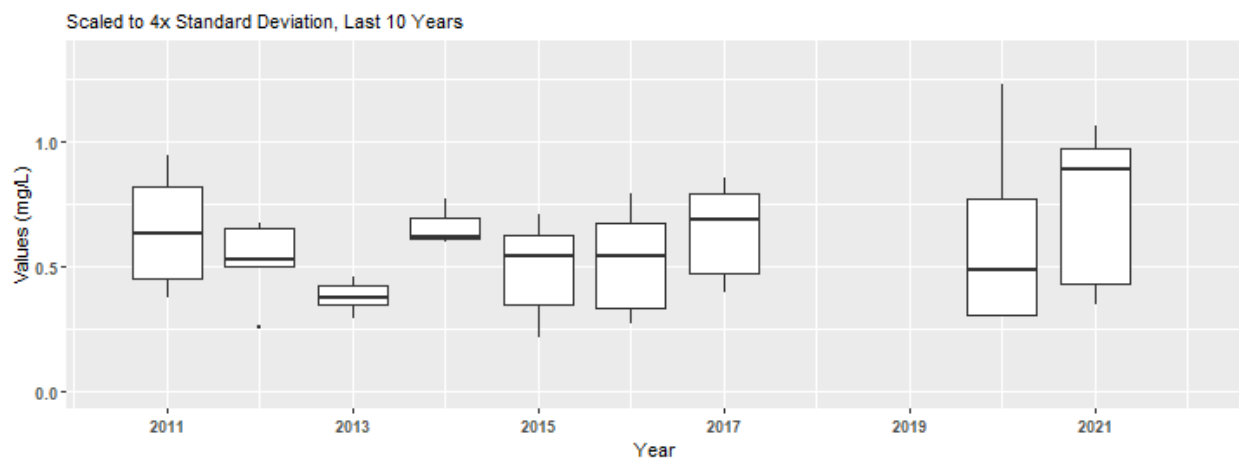
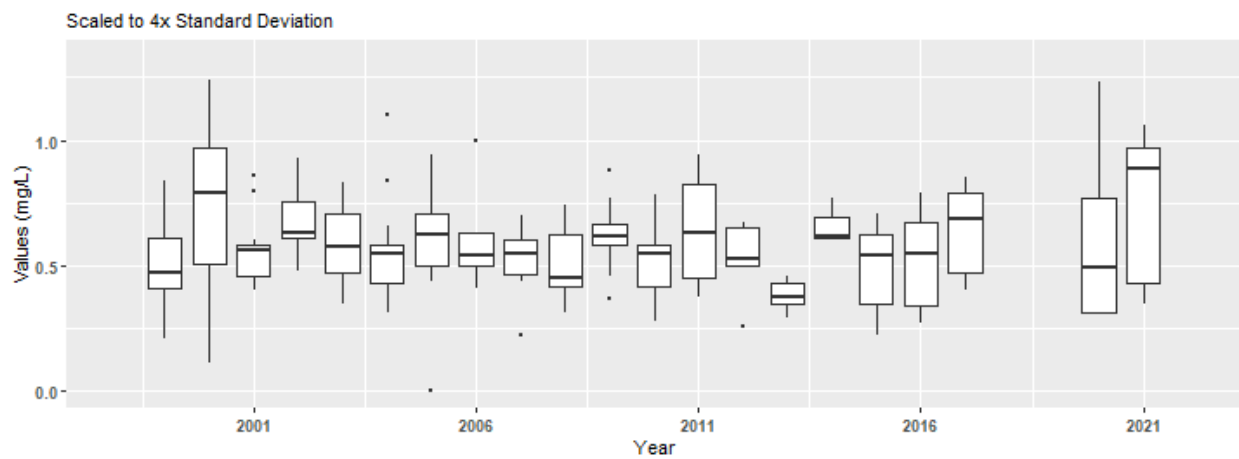
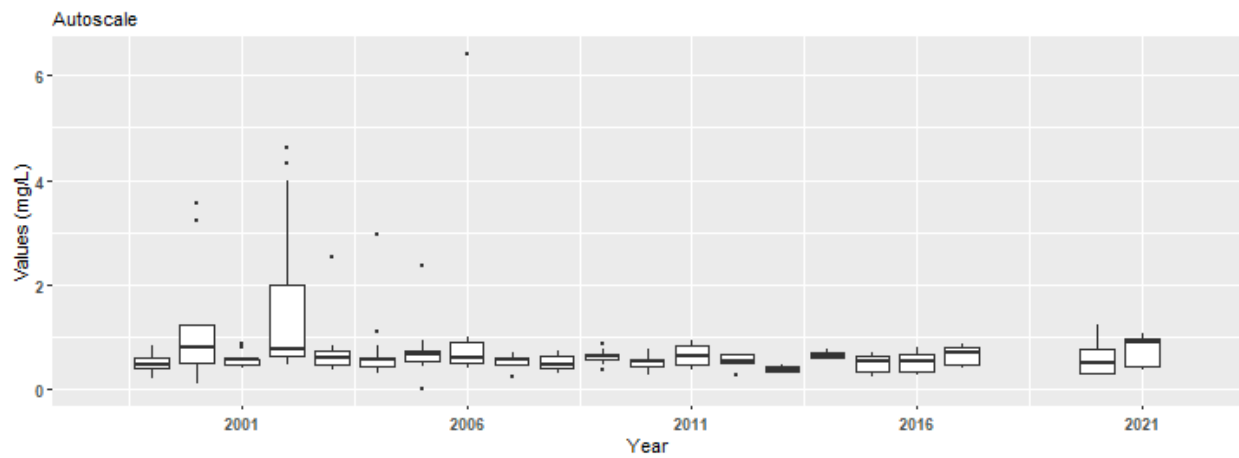
Summary Box Plots for Nature Coast Aquatic Preserve

By Month



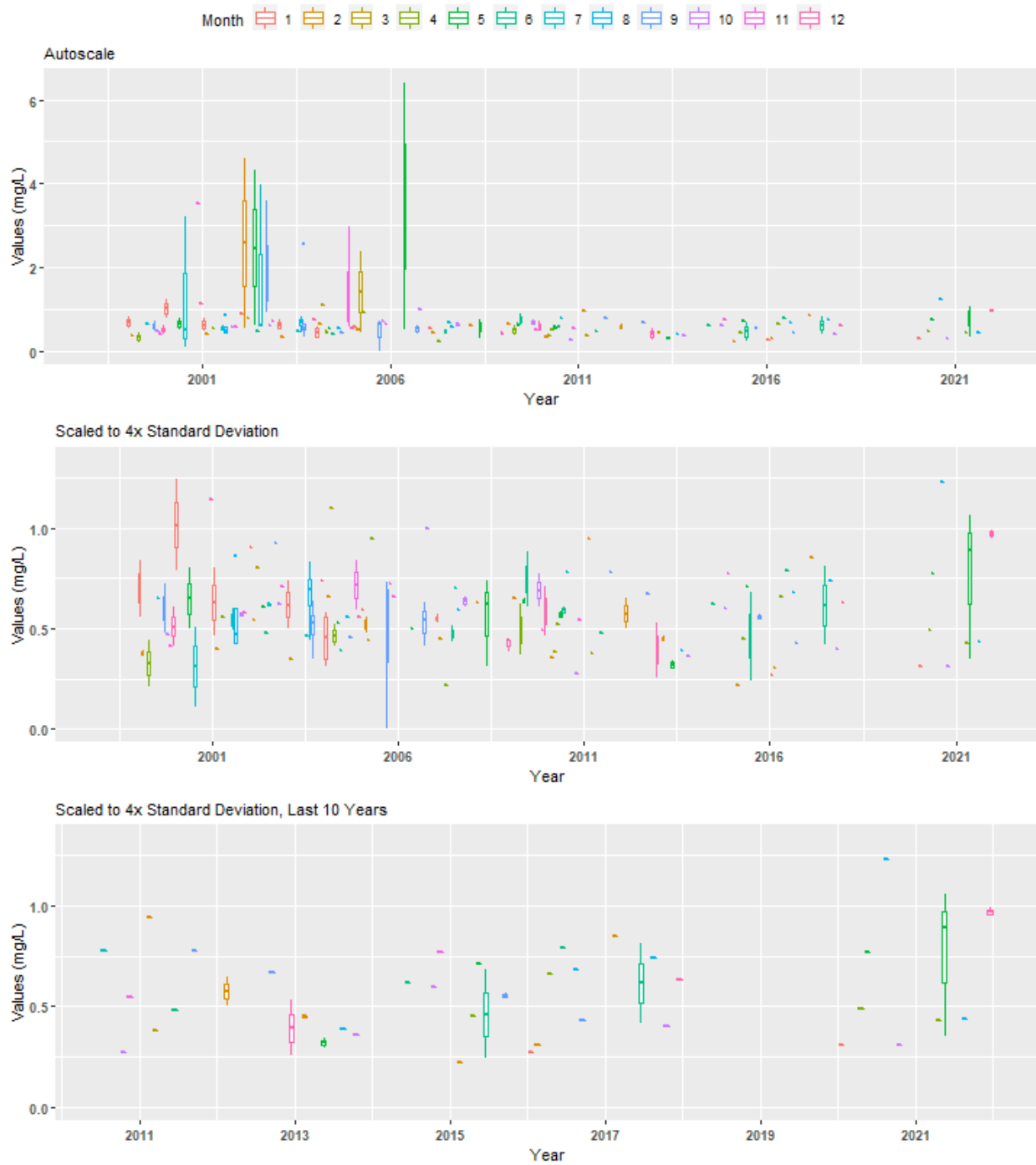
Summary Box Plots for Pinellas County Aquatic Preserve

By Year



Summary Box Plots for Pinellas County Aquatic Preserve

By Year & Month



Summary Box Plots for Pinellas County Aquatic Preserve

By Month

