SEACAR Discrete Water Quality Analysis: Sample Bottom Total Nitrogen

Last compiled on 30 May, 2022

Contents

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

```
library(EnvStats)
library(tidyr)
options(scipen=999)
opts_chunk$set(warning=FALSE, message=FALSE)
```

File Import

Imports file that is determined in the WC_Discrete_parameter_ReportCompile.R script.

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

The script then gets the name of the parameter as it appears in the data file, units of the parameter, sets the `SampleDate` as a date object, and creates various scales of the date to be used by plotting functions.

```
data <- fread(file_in, sep="|", header=TRUE, stringsAsFactors=FALSE,
              select=c("ManagedAreaName", "ProgramID", "ProgramName",
                       "ProgramLocationID", "SampleDate", "Year", "Month",
                       "RelativeDepth", "ActivityType", "ParameterName",
                       "ResultValue", "ParameterUnits", "ValueQualifier",
                       "SEACAR_QAQCFlagCode", "Include"), na.strings="")

parameter <- unique(data$ParameterName)
unit <- unique(data$ParameterUnits)
data$SampleDate <- as.Date(data$SampleDate)
data$YearMonth <- paste0(data$Month, "-", data$Year)
data$YearMonthDec <- data$Year + ((data$Month-0.5) / 12)
data$DecDate <- decimal_date(data$SampleDate)
```

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

After the initial filtering, a second filter variable is created to determine whether enough time is represented in the managed area, which is that each managed area has 10 year or more of unique year entries for observation that pass the initial filter. If data passes the first set of filtering criteria and the time criteria, they are used in the analysis.

After filtering, the amount of data impacted by the H (for dissolved oxygen & pH in program 476), I, Q, S (for Secchi depth), and U value qualifiers. A variable is also created that determines if scatter plot points should be a different color based on value qualifiers of interest.

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

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

if(param_name!="Secchi_Depth"){
   data <- data[!is.na(data$RelativeDepth),]
   data <- data[data$RelativeDepth==depth,]
}

if(length(grep("Blank", data$ActivityType))>0){
   data <- data[-grep("Blank", data$ActivityType),]
}

if(param_name=="Chlorophyll_a_uncorrected_for_pheophytin" |
   param_name=="Salinity" | param_name=="Turbidity"){
   data <- data[grep(activity, data$ActivityType[!is.na(data$ActivityType)]),]
}

if(param_name=="Water_Temperature"){
   data <- data[data$ResultValue>=-2,]
} else{
   data <- data[data$ResultValue>=0,]
}


data <- merge.data.frame(MA_All[,c("AreaID", "ManagedAreaName")],
                         data, by="ManagedAreaName", all=TRUE)

MA_Summ <- data %>%
   group_by(AreaID, ManagedAreaName) %>%
   summarize(ParameterName=parameter,
             RelativeDepth=depth,
             ActivityType=activity,
             N_Data=length(ResultValue[Include==TRUE & !is.na(ResultValue)]),
             N_Years=length(unique(Year[Include==TRUE & !is.na(Year)])),
             EarliestYear=min(Year[Include==TRUE]),
             LatestYear=max(Year[Include==TRUE]),
             SufficientData=ifelse(N_Data>0 & N_Years>=10, TRUE, FALSE))

data <- merge.data.frame(data, MA_Summ[,c("ManagedAreaName", "SufficientData")],
                         by="ManagedAreaName")

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

MA_Summ <- MA_Summ %>%
   select(AreaID, ManagedAreaName, ParameterName, RelativeDepth, ActivityType,
          SufficientData, everything())
```

```
MA_Summ <- as.data.frame(MA_Summ[order(MA_Summ$ManagedAreaName), ])


total <- length(data$Include)
pass_filter <- length(data$Include[data$Include==TRUE])

count_H <- length(grep("H", data$ValueQualifier[data$ProgramID==476]))
perc_H <- 100*count_H/length(data$ValueQualifier)

count_I <- length(grep("I", data$ValueQualifier))
perc_I <- 100*count_I/length(data$ValueQualifier)

count_Q <- length(grep("Q", data$ValueQualifier))
perc_Q <- 100*count_Q/length(data$ValueQualifier)

count_S <- length(grep("S", data$ValueQualifier))
perc_S <- 100*count_S/length(data$ValueQualifier)

count_U <- length(grep("U", data$ValueQualifier))
perc_U <- 100*count_U/length(data$ValueQualifier)


data$VQ_Plot <- data$ValueQualifier

inc_H <- ifelse(param_name=="pH" | param_name=="Dissolved_Oxygen" |
                   param_name=="Dissolved_Oxygen_Saturation", TRUE, FALSE)

if (inc_H==TRUE){
   data$VQ_Plot <- gsub("[^HU]+", "", data$VQ_Plot)
   data$VQ_Plot <- gsub("UH", "HU", data$VQ_Plot)
   data$VQ_Plot[na.omit(data$ProgramID!=476)] <- gsub("[^U]+", "",
                                              data$VQ_Plot[na.omit(data$ProgramID!=476)])
   data$VQ_Plot[data$VQ_Plot==""] <- NA


   cat(paste0("Number of Measurements: ", total,
              ", Number Passed Filter: ", pass_filter, "\n",
              "Program 476 H Codes: ", count_H, " (", round(perc_H, 6), "%)\n",
              "I Codes: ", count_I, " (", round(perc_I, 6), "%)\n",
              "Q Codes: ", count_Q, " (", round(perc_Q, 6), "%)\n",
              "U Codes: ", count_U, " (", round(perc_U, 6), "%)"))
} else if (param_name=="Secchi_Depth") {
   count_S <- length(grep("S", data$ValueQualifier))
   perc_S <- 100*count_S/length(data$ValueQualifier)
   data$VQ_Plot <- gsub("[^SU]+", "", data$VQ_Plot)
   data$VQ_Plot <- gsub("US", "SU", data$VQ_Plot)
   data$VQ_Plot[data$VQ_Plot==""] <- NA
   cat(paste0("Number of Measurements: ", total,
              ", Number Passed Filter: ", pass_filter, "\n",
              "I Codes: ", count_I, " (", round(perc_I, 6), "%)\n",
              "Q Codes: ", count_Q, " (", round(perc_Q, 6), "%)\n",
              "S Codes: ", count_S, " (", round(perc_S, 6), "%)\n",
              "U Codes: ", count_U, " (", round(perc_U, 6), "%)"))
```

```
} else{
   data$VQ_Plot <- gsub("[^U]+", "", data$VQ_Plot)
   data$VQ_Plot[data$VQ_Plot==""] <- NA
   cat(paste0("Number of Measurements: ", total,
              ", Number Passed Filter: ", pass_filter, "\n",
              "I Codes: ", count_I, " (", round(perc_I, 6), "%)\n",
              "Q Codes: ", count_Q, " (", round(perc_Q, 6), "%)\n",
              "U Codes: ", count_U, " (", round(perc_U, 6), "%)"))
}
```

```
## Number of Measurements: 11183, Number Passed Filter: 11149
## I Codes: 3 (0.026826%)
## Q Codes: 5 (0.044711%)
## U Codes: 86 (0.769024%)
```

```
data_summ <- data %>%
   group_by(AreaID, ManagedAreaName) %>%
   summarize(ParameterName=parameter,
             RelativeDepth=depth,
             ActivityType=activity,
             N_Total=length(ResultValue),
             N_AnalysisUse=length(ResultValue[SufficientData==TRUE]),
             N_H=length(grep("H", data$ValueQualifier[data$ProgramID==476])),
             perc_H=100*N_H/length(data$ValueQualifier),
             N_I=length(grep("I", data$ValueQualifier)),
             perc_I=100*N_I/length(data$ValueQualifier),
             N_Q=length(grep("Q", data$ValueQualifier)),
             perc_Q=100*N_Q/length(data$ValueQualifier),
             N_S=length(grep("S", data$ValueQualifier)),
             perc_S=100*N_S/length(data$ValueQualifier),
             N_U=length(grep("U", data$ValueQualifier)),
             perc_U=100*N_U/length(data$ValueQualifier))

data_summ <- as.data.table(data_summ[order(data_summ$ManagedAreaName), ])
fwrite(data_summ, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                         "_DataSummary.csv"), sep=",")
rm(data_summ)
MA_Include <- MA_Summ$ManagedAreaName[MA_Summ$SufficientData==TRUE &
                                         MA_Summ$N_Data<2000000]
n <- length(MA_Include)
MA_Exclude <- MA_Summ[MA_Summ$N_Years<10 & MA_Summ$N_Years>0,]
MA_Exclude <- MA_Exclude[,c("ManagedAreaName", "N_Years")]
z <- nrow(MA_Exclude)
```

Managed Area Statistics

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

1. Take the `data` variable and only include rows that have a `SufficientData` value of TRUE

2. Group data that have the same `ManagedAreaName`, `Year`, and `Month`.
    - Second summary statistics do not use the `Month` grouping and are only for `ManagedAreaName` and `Year`.
    - Third summary statistics do not use `Year` grouping and are only for `ManagedAreaName` and `Month`
3. For each group, provide the following information: Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, and a list of all Program IDs included in these measurements.
4. Sort the data in ascending (A to Z and 0 to 9) order based on `ManagedAreaName` then `Year` then `Month`
5. Write summary stats to a pipe-delimited .txt file in the output directory
    - Click this text to open Git directory with output files

```
MA_YM_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, Year, Month) %>%
   summarize(ParameterName=parameter,
             RelativeDepth=depth,
             ActivityType=activity,
             N_Data=length(ResultValue),
             Min=min(ResultValue),
             Max=max(ResultValue),
             Median=median(ResultValue),
             Mean=mean(ResultValue),
             StandardDeviation=sd(ResultValue),
             ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                              collapse=', '))
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                               MA_YM_Stats$Year,
                                               MA_YM_Stats$Month), ])
fwrite(MA_YM_Stats, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                           "_ManagedArea_YearMonth_Stats.txt"), sep="|")
rm(MA_YM_Stats)

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

MA_M_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, Month) %>%
```

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

Monitoring Location Statistics

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

1. Take the `data` variable and only include rows that have a `SufficientData` value of `TRUE`
2. Group data that have the same `ManagedAreaName`, `ProgramID`, `ProgramName`, and `ProgramLocationID`.
3. For each group, provide the following information: Earliest Sample Date (EarliestSampleDate), Latest Sample Date (LastSampleDate), Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, and Standard Deviation.
4. Sort the data in ascending (A to Z and 0 to 9) order based on `ManagedAreaName` then `ProgramName` then `ProgramID` then `ProgramLocationID`
5. Write summary stats to a pipe-delimited .txt file in the output directory
   - Click this text to open Git directory with output files

```
Mon_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
  group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID) %>%
  summarize(ParameterName=parameter,
            RelativeDepth=depth,
            ActivityType=activity,
            EarliestSampleDate=min(SampleDate),
            LastSampleDate=max(SampleDate),
            N=length(ResultValue),
            Min=min(ResultValue),
            Max=max(ResultValue),
            Median=median(ResultValue),
            Mean=mean(ResultValue),
            StandardDeviation=sd(ResultValue))

Mon_Stats <- as.data.table(Mon_Stats[order(Mon_Stats$ManagedAreaName,
                                           Mon_Stats$ProgramName,
                                           Mon_Stats$ProgramID,
                                           Mon_Stats$ProgramLocationID), ])
```

```
fwrite(Mon_Stats, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                         "_MonitoringLoc_Stats.txt"), sep="|")
rm(Mon_Stats)
```

Seasonal Kendall Tau Analysis

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

The following steps are performed:

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

```
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)
   rm(ken)
   }, warning = function(w) {
      print(w)
   }, error = function(e) {
      print(e)
```

```r
  }, 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
    }
  })
  KT <-c(unique(data$AreaID),
         unique(data$ManagedAreaName),
         stats.median,
         independent,
         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[9])) {
    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$ManagedAreaName
c_names <- c("AreaID", "ManagedAreaName", "Median", "Independent",
             "tau", "p", "SennSlope", "SennIntercept", "Trend")
if(n==0){
   KT.Stats <- data.frame(matrix(ncol=length(c_names),
                                 nrow=length(MA_Summ$ManagedAreaName)))
   colnames(KT.Stats) <- c_names
   KT.Stats[, c("AreaID", "ManagedAreaName")] <-
      MA_Summ[, c("AreaID", "ManagedAreaName")]
} else{
   for (i in 1:n) {
      x <- nrow(data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_Include[i], ])
      if (x>0) {
         KT.Stats <- runStats(data[data$Use_In_Analysis == TRUE &
                                      data$ManagedAreaName ==
                                      MA_Include[i], ])
      }
   }
   KT.Stats <- as.data.frame(KT.Stats)
   c_names <- c("AreaID", "ManagedAreaName", "Median", "Independent",
                "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$tau <- round(as.numeric(KT.Stats$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)
}
```

```r
KT.Stats <-  merge.data.frame(MA_Summ, KT.Stats,
                              by=c("AreaID", "ManagedAreaName"), all=TRUE)

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

fwrite(KT.Stats, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                        "_KendallTau_Stats.txt"), sep="|")
data <- data[!is.na(data$ResultValue),]
```

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.

```r
plot_theme <- theme_bw() + theme(text=element_text(family="Segoe UI"),
                                 title=element_text(face="bold"),
                                 plot.title=element_text(hjust=0.5, size=14, color="#314963"),
                                 plot.subtitle=element_text(hjust=0.5, size=10, color="#314963"),
                                 axis.text.x=element_text(face="bold"),
                                 axis.text.y=element_text(face="bold"))

year_lower <- min(data$Year)
year_upper <- max(data$Year)
min_RV <- min(data$ResultValue)
mn_RV <- mean(data$ResultValue[data$ResultValue <
                                  quantile(data$ResultValue, 0.98)])
sd_RV <- sd(data$ResultValue[data$ResultValue <
                                quantile(data$ResultValue, 0.98)])
x_scale <- ifelse(year_upper - year_lower > 30, 10, 5)
y_scale <- mn_RV + 4 * sd_RV


p1 <- ggplot(data=data[data$Include==TRUE,],
             aes(x=SampleDate, y=ResultValue, fill=VQ_Plot)) +
   geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
   labs(subtitle="Autoscale",
        x="Year", y=paste0("Values (", unit, ")"),
        fill="Value Qualifier") +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal",
         legend.justification="right") +
   scale_x_date(labels=date_format("%Y")) +
   {if(inc_H==TRUE){
      scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                                 "HU"="#7CAE00"), na.value="#cccccc")
   } else if(param_name=="Secchi_Depth"){
      scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                                 "SU"="#7CAE00"), na.value="#cccccc")
   } else {
```

```
    scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
  }}


p2 <- ggplot(data=data[data$Include==TRUE,],
             aes(x=SampleDate, y=ResultValue, fill=VQ_Plot)) +
  geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
  ylim(min_RV, y_scale) +
  labs(subtitle="Scaled to 4x Standard Deviation",
       x="Year", y=paste0("Values (", unit, ")")) +
  plot_theme +
  theme(legend.position="none") +
  scale_x_date(labels=date_format("%Y")) +
  {if(inc_H==TRUE){
    scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                               "HU"="#7CAE00"), na.value="#cccccc")
  } else if(param_name=="Secchi_Depth"){
    scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                               "SU"="#7CAE00"), na.value="#cccccc")
  } else {
    scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
  }}

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

p0 <- ggplot() + labs(title="Scatter Plot for Entire Dataset") +
  plot_theme + theme(panel.border=element_blank(),
                     panel.grid.major=element_blank(),
                     panel.grid.minor=element_blank(),
                     axis.line=element_blank())

ggarrange(p0, pset, ncol=1, heights=c(0.1, 1))
```

Appendix II: Dataset Summary Box Plots

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

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

This set of box plots are grouped by year.

```
min_RV <- min(data$ResultValue[data$Include==TRUE])
mn_RV <- mean(data$ResultValue[data$Include==TRUE &
                                  data$ResultValue <
                                  quantile(data$ResultValue, 0.98)])
sd_RV <- sd(data$ResultValue[data$Include==TRUE &
                                data$ResultValue <
                                quantile(data$ResultValue, 0.98)])
y_scale <- mn_RV + 4 * sd_RV

p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=Year, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Year",
        y=paste0("Values (", unit, ")")) +
   plot_theme

p2 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=Year, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation", x="Year",
        y=paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   plot_theme

p3 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=as.integer(Year), y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",
        x="Year", y=paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+1),
                      breaks=seq(max(data$Year) - 10, max(data$Year), 2)) +
   plot_theme

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

p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",
```

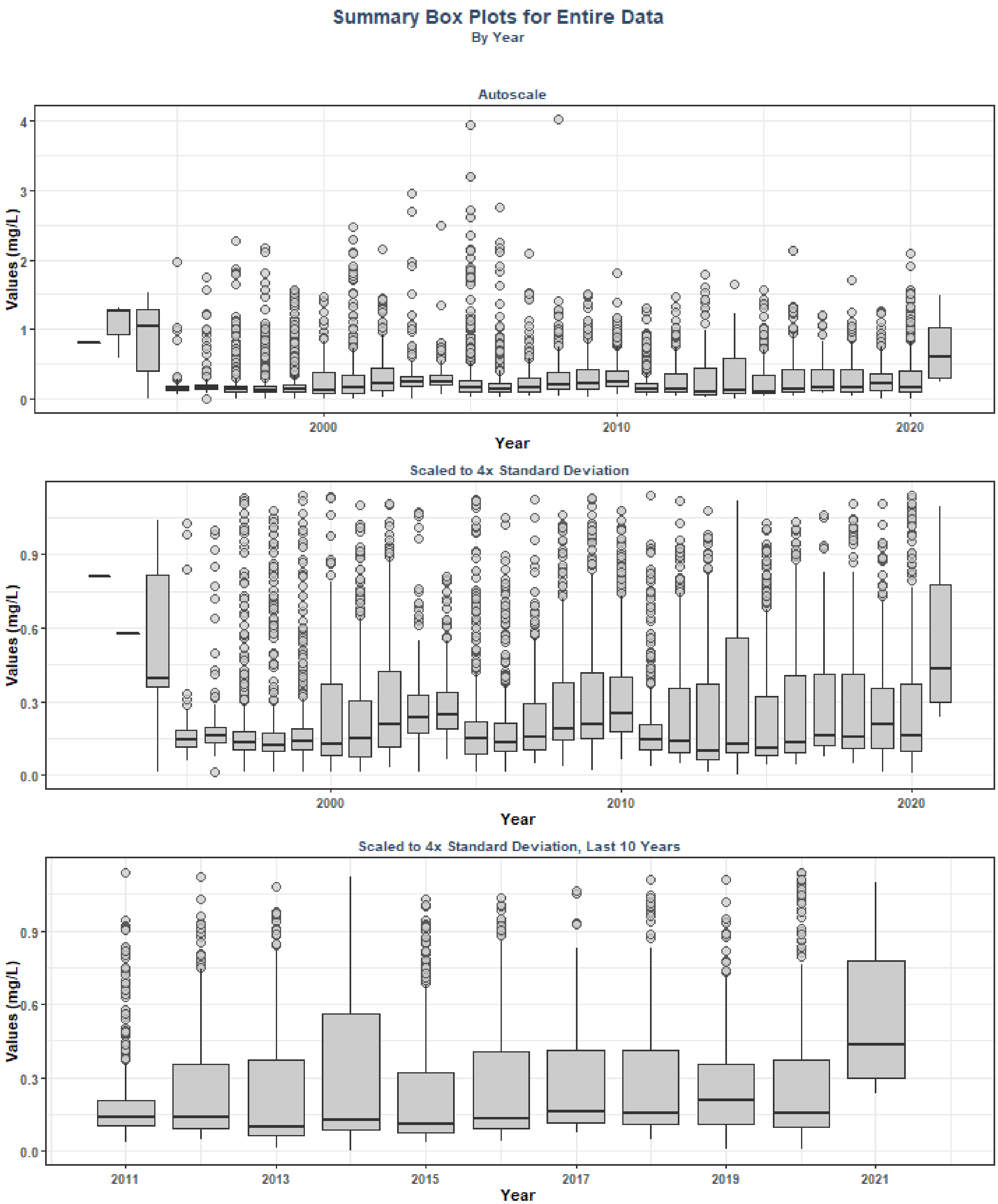
```
                      subtitle="By Year") + plot_theme +
   theme(panel.border=element_blank(), panel.grid.major=element_blank(),
         panel.grid.minor=element_blank(), axis.line=element_blank())

Yset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))
```

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

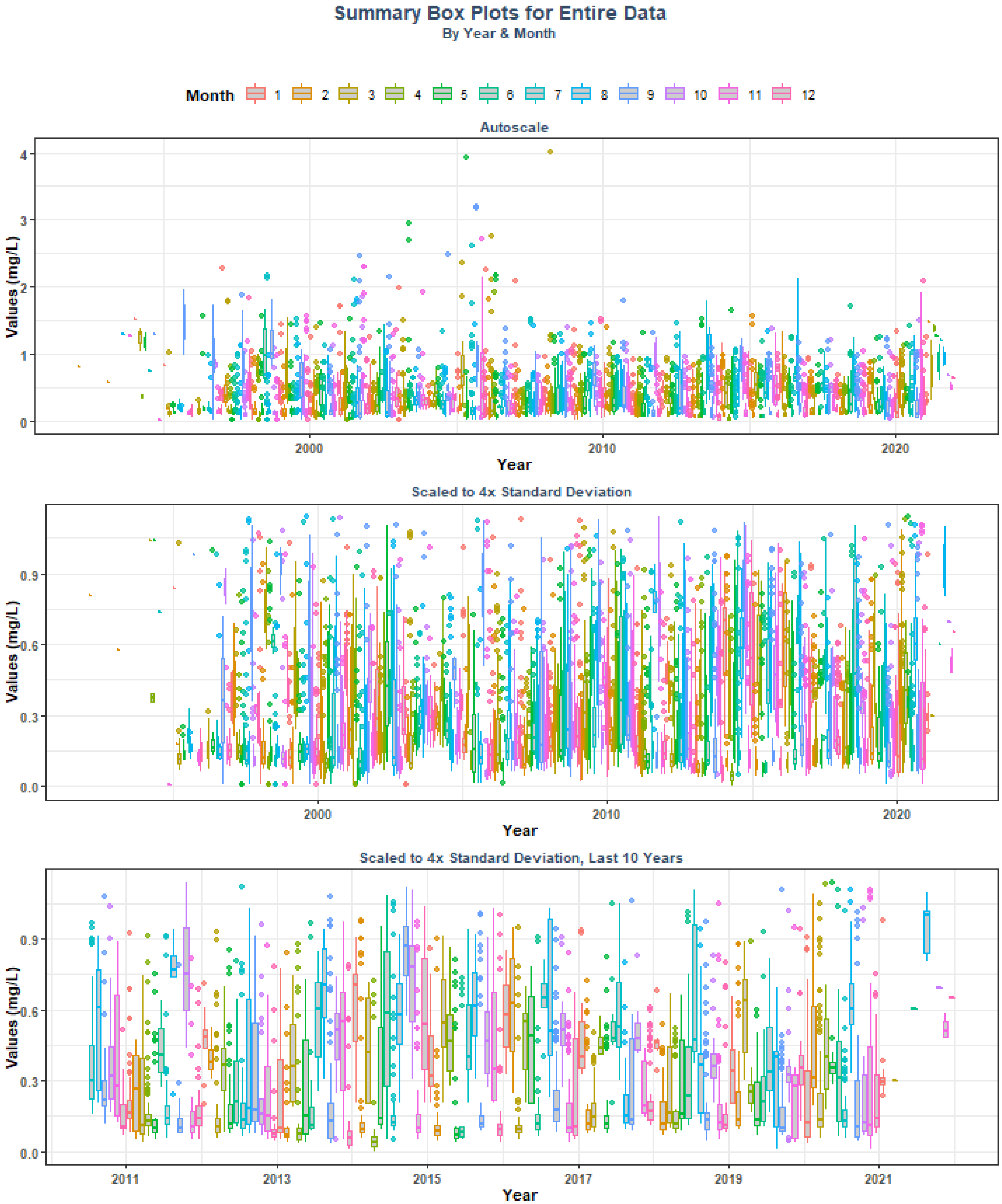
```
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Year",
        y=paste0("Values (", unit, ")"), color="Month") +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal") +
   guides(color=guide_legend(nrow=1))

p2 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation",
        x="Year", y=paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   plot_theme +
   theme(legend.position="none", axis.text.x=element_text(face="bold"),
         axis.text.y=element_text(face="bold"))

p3 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",
        x="Year", y=paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+1),
                      breaks=seq(max(data$Year) - 10, max(data$Year), 2)) +
   plot_theme +
   theme(legend.position="none")
leg <- get_legend(p1)
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,
                 heights=c(0.1, 1, 1, 1))

p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",
                      subtitle="By Year & Month") + plot_theme +
   theme(panel.border=element_blank(), panel.grid.major=element_blank(),
         panel.grid.minor=element_blank(), axis.line=element_blank())

YMset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))
```

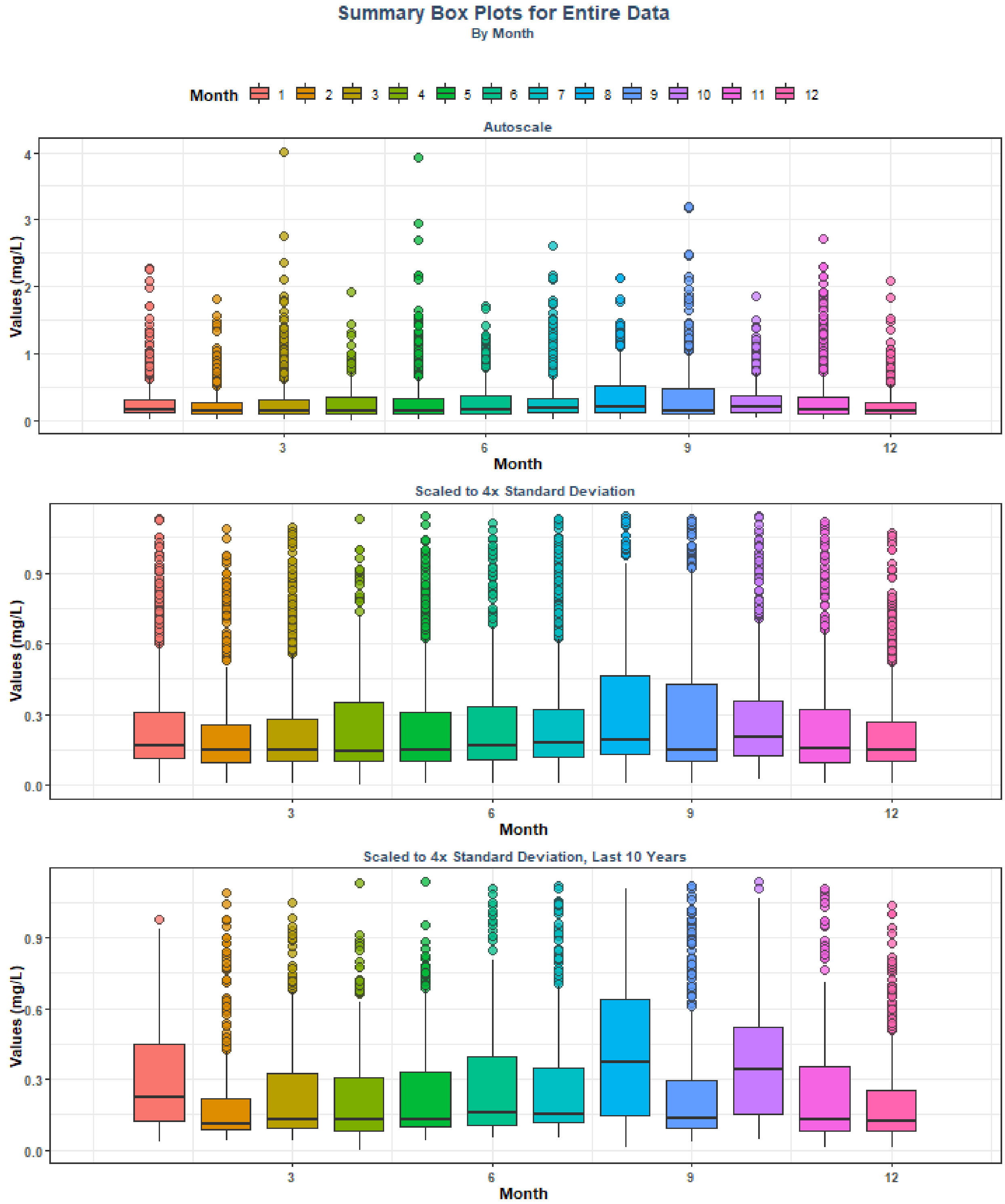
The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

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

p2 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
   geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
                outlier.color="#333333", outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation",
        x="Month", y=paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +
   plot_theme +
   theme(legend.position="none")

p3 <- ggplot(data=data[data$Include==TRUE &
                          data$Year >= max(data$Year) - 10, ],
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
   geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
                outlier.color="#333333", outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",
        x="Month", y=paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +
   plot_theme +
   theme(legend.position="none")
leg <- get_legend(p1)
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,
                 heights=c(0.1, 1, 1, 1))

p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",
                      subtitle="By Month") + plot_theme +
   theme(panel.border=element_blank(), panel.grid.major=element_blank(),
         panel.grid.minor=element_blank(), axis.line=element_blank())

Mset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))
```



Summary Box Plots for Entire Data
By Year
Autoscale
Values (mg/L)
0
1
2
3
4
2000
2010
2020
Year
Scaled to 4x Standard Deviation
Values (mg/L)
0.0
0.3
0.6
0.9
2000
2010
2020
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.0
0.3
0.6
0.9
2011
2013
2015
2017
2019
2021
Year

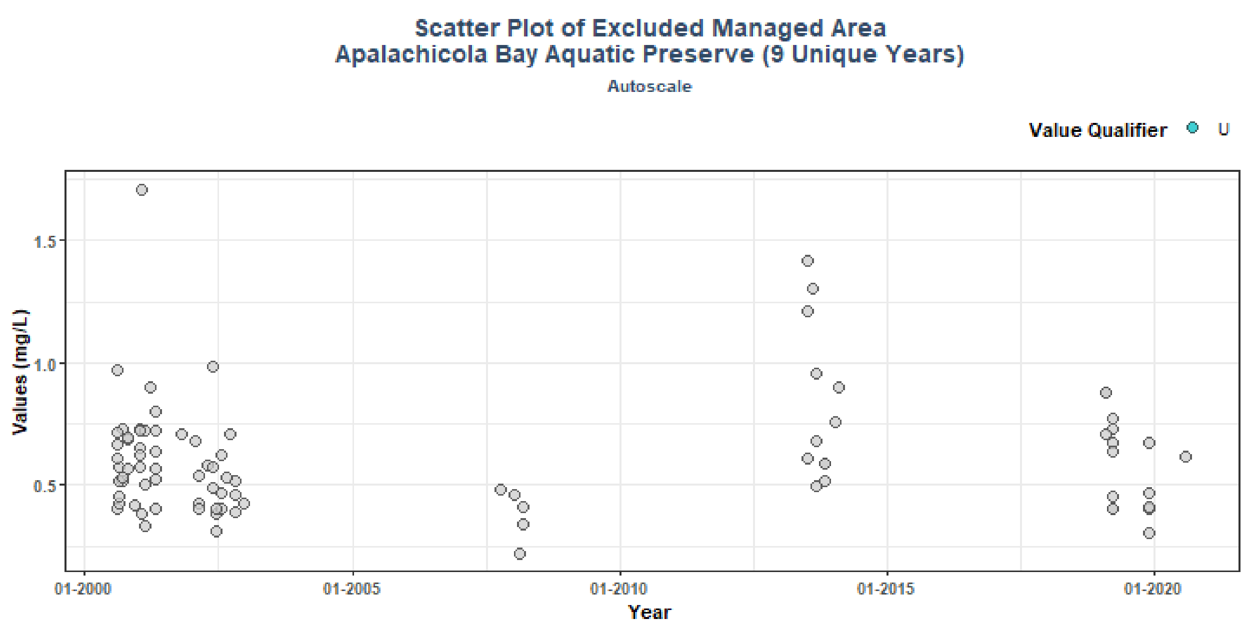

Summary Box Plots for Entire Data
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year

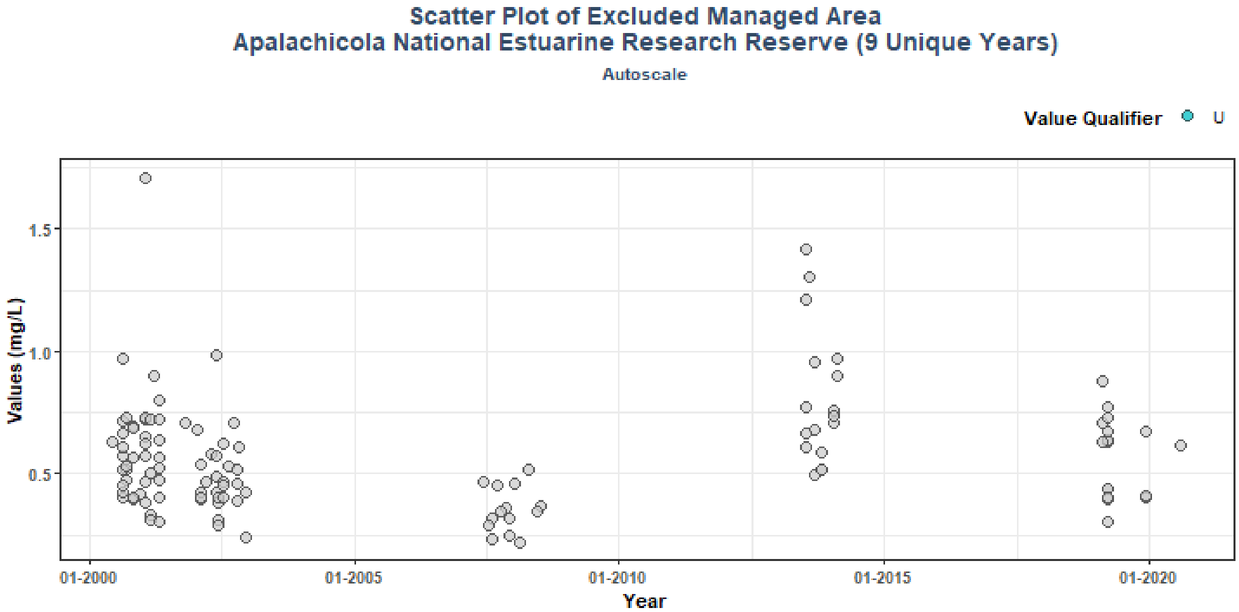


Appendix III: Excluded Managed Areas

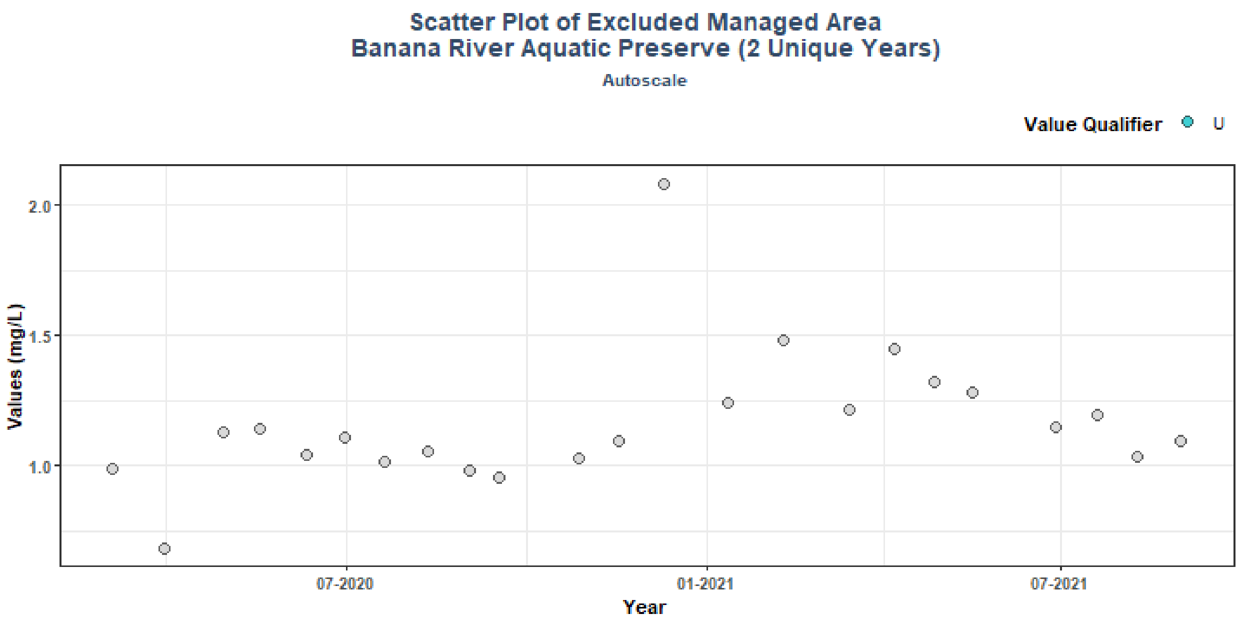
Scatter plots of data values are created for managed areas that have fewer than 10 separate years of data entries. Data points are colored based on specific value qualifiers of interest.

```r
if(z==0){
   print("There are no managed areas that qualify.")
} else {
   for(i in 1:z){
      p1<-ggplot(data=data[data$ManagedAreaName==MA_Exclude$ManagedAreaName[i]&
                              data$Include==TRUE, ],
                 aes(x=SampleDate, y=ResultValue, fill=VQ_Plot)) +
         geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
         labs(title=paste0("Scatter Plot of Excluded Managed Area\n",
                           MA_Exclude$ManagedAreaName[i], " (",
                           MA_Exclude$N_Years[i], " Unique Years)"),
              subtitle="Autoscale", x="Year",
              y=paste0("Values (", unit, ")"), fill="Value Qualifier") +
         plot_theme +
         theme(legend.position="top", legend.box="horizontal",
               legend.justification="right") +
         scale_x_date(labels=date_format("%m-%Y")) +
         {if(inc_H==TRUE){
            scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                                       "HU"="#7CAE00"), na.value="#cccccc")
         } else if(param_name=="Secchi_Depth"){
            scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                                       "SU"="#7CAE00"), na.value="#cccccc")
         } else {
            scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
         }}
      print(p1)
   }
}
```

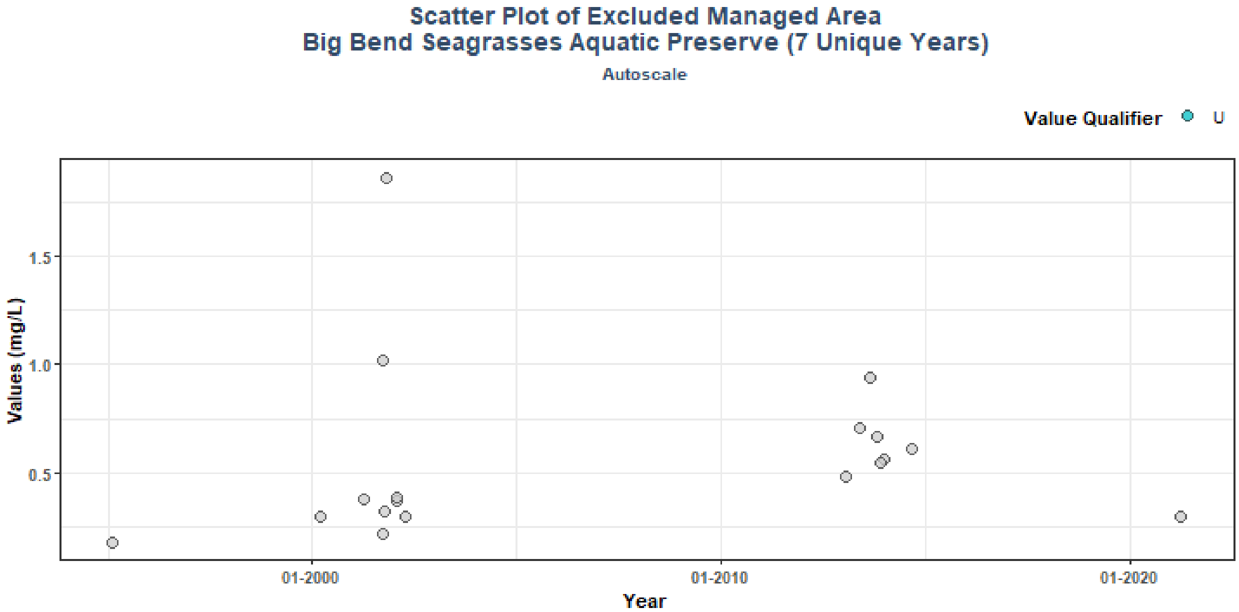


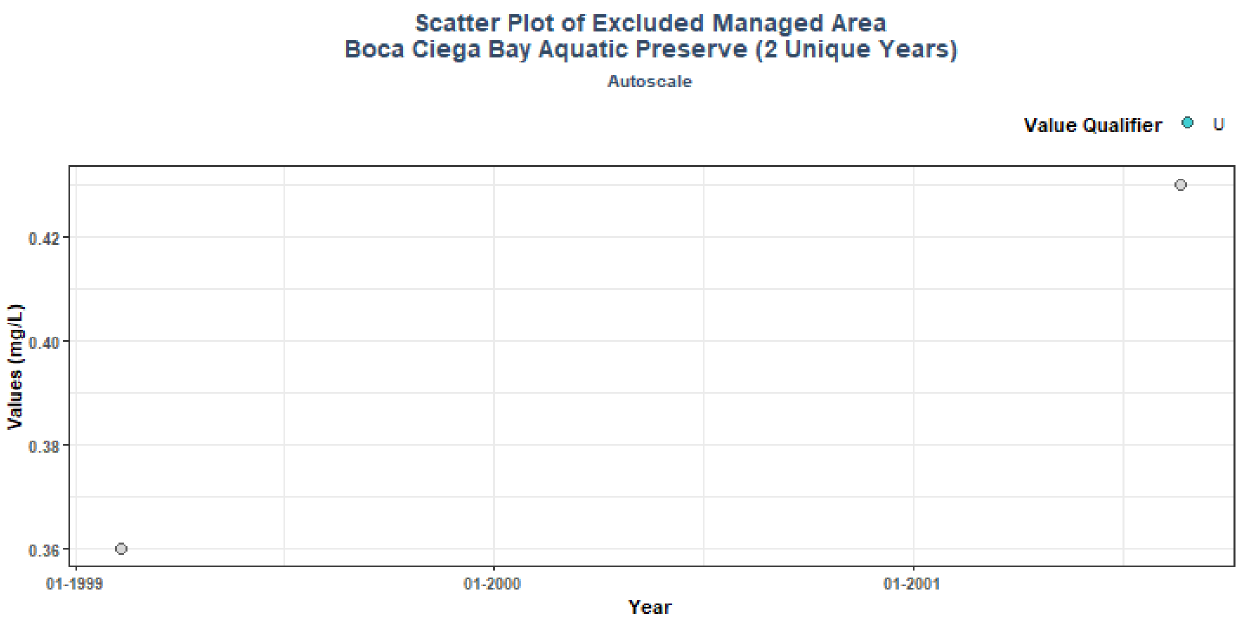


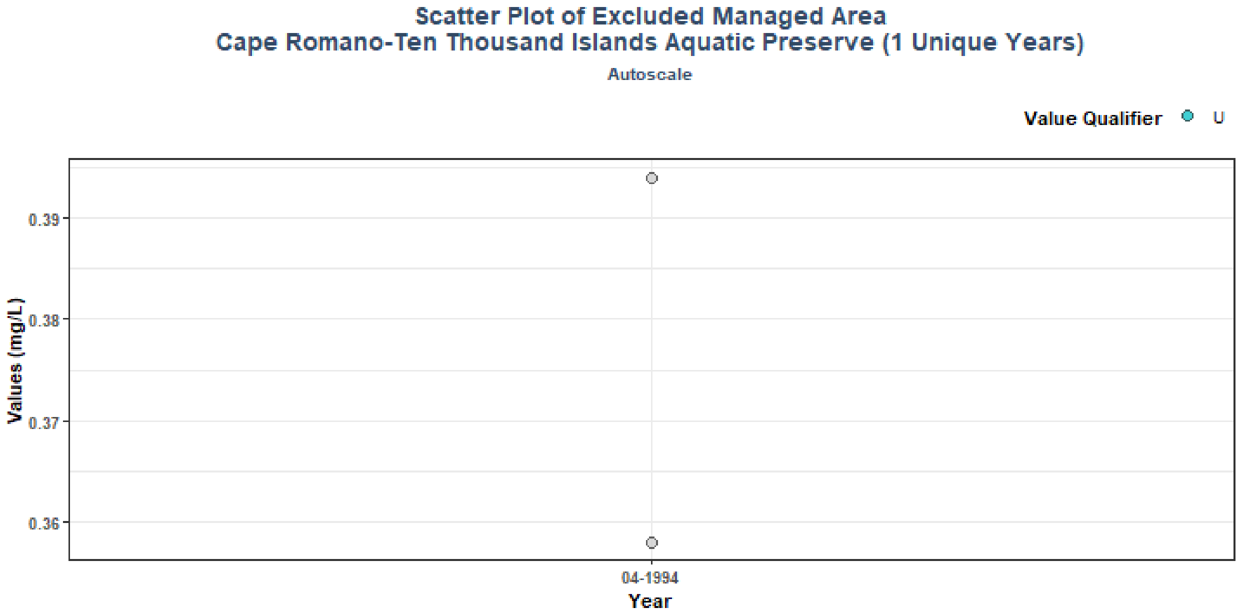
Scatter Plot of Excluded Managed Area
Apalachicola National Estuarine Research Reserve (9 Unique Years)
Autoscale
Value Qualifier
U
1.5
1.0
0.5
Values (mg/L)
01-2000
01-2005
01-2010
01-2015
01-2020
Year

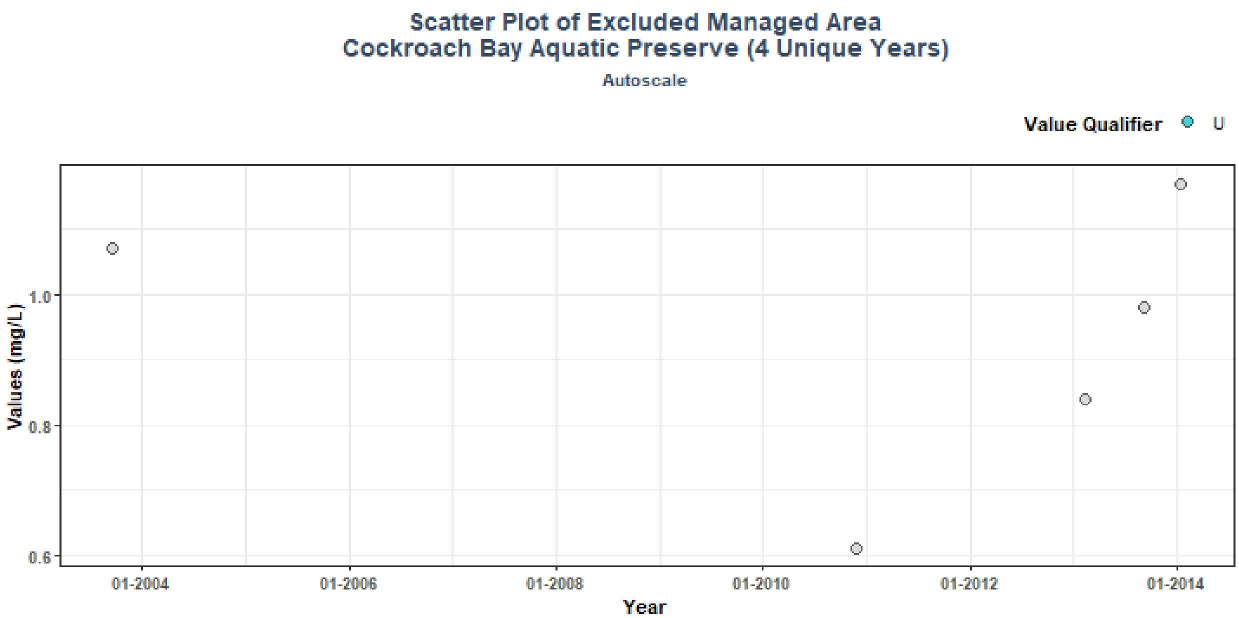


Scatter Plot of Excluded Managed Area
Banana River Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
2.0
1.5
1.0
Values (mg/L)
07-2020
01-2021
07-2021
Year

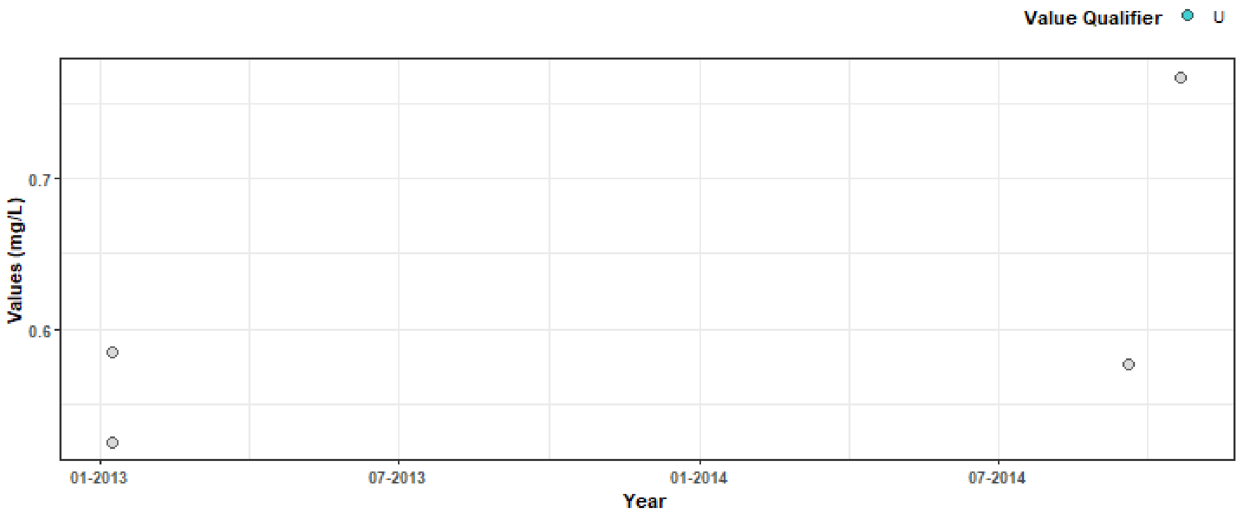

Scatter Plot of Excluded Managed Area
Big Bend Seagrasses Aquatic Preserve (7 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
1.5
1.0
0.5
01-2000
01-2010
01-2020
Year


Scatter Plot of Excluded Managed Area
Boca Ciega Bay Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
0.42
0.40
0.38
0.36
01-1999
01-2000
01-2001
Year

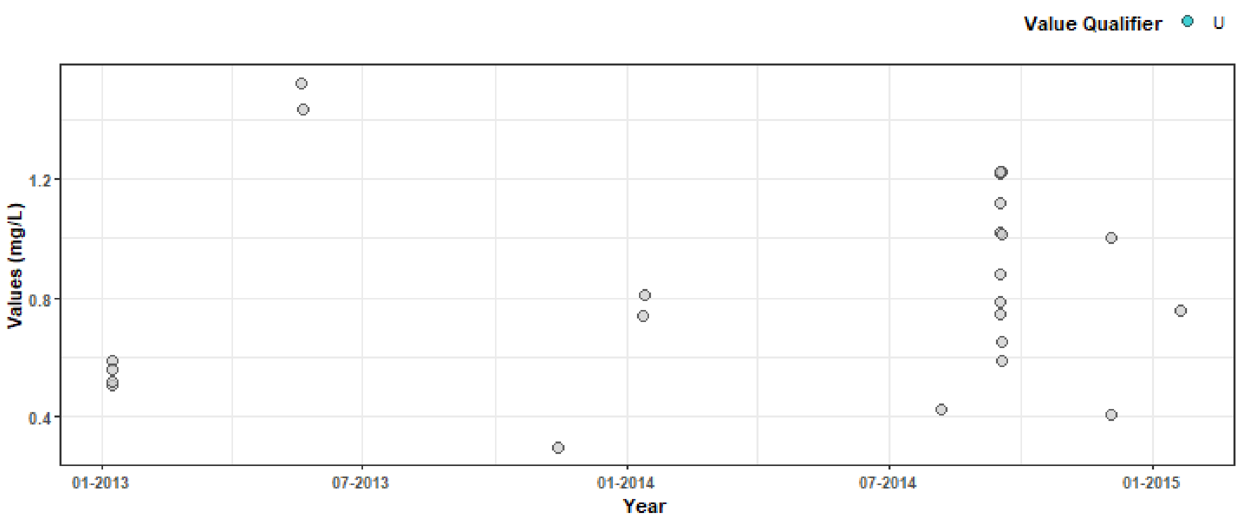

Scatter Plot of Excluded Managed Area
Cape Romano-Ten Thousand Islands Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
0.39
0.38
0.37
0.36
04-1994
Year

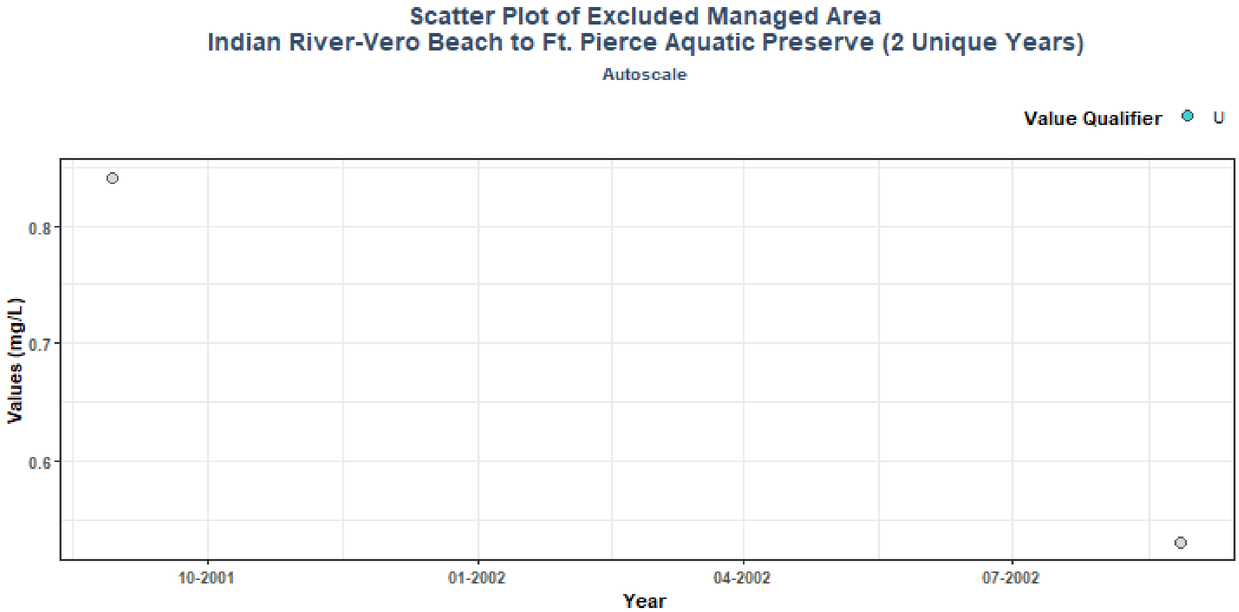

Scatter Plot of Excluded Managed Area
Cockroach Bay Aquatic Preserve (4 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
1.0
0.8
0.6
01-2004
01-2006
01-2008
01-2010
01-2012
01-2014
Year

Scatter Plot of Excluded Managed Area
Guana River Marsh Aquatic Preserve (2 Unique Years)
Autoscale

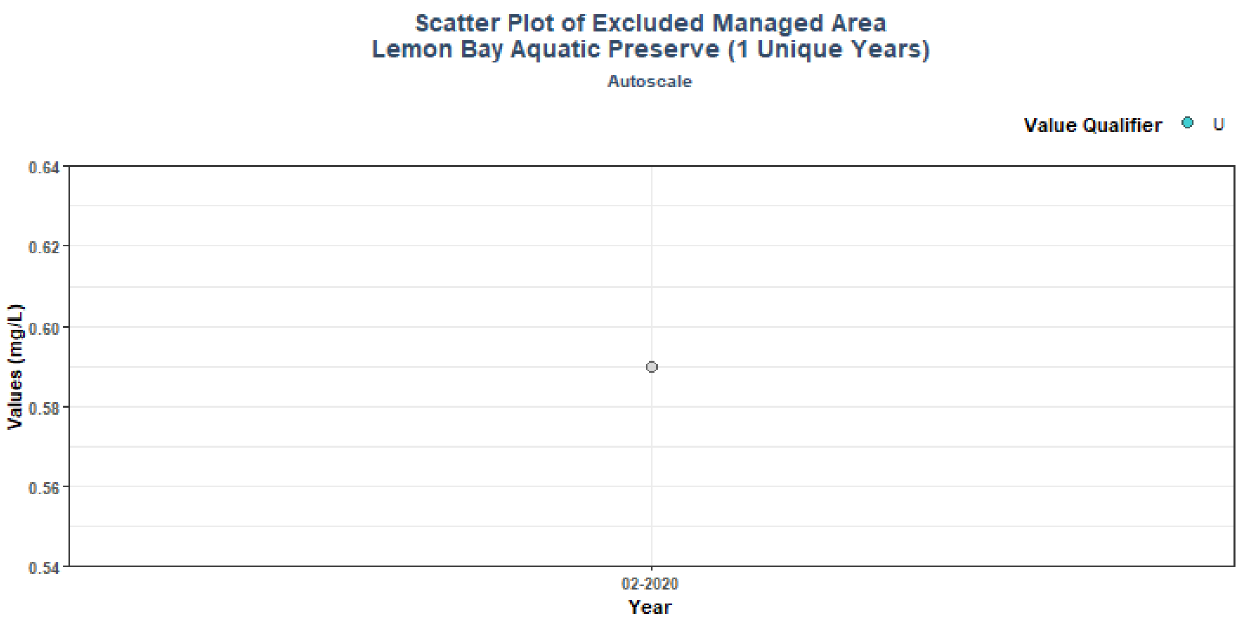


Scatter Plot of Excluded Managed Area
Guana Tolomato Matanzas National Estuarine Research Reserve (3 Unique Years)
Autoscale

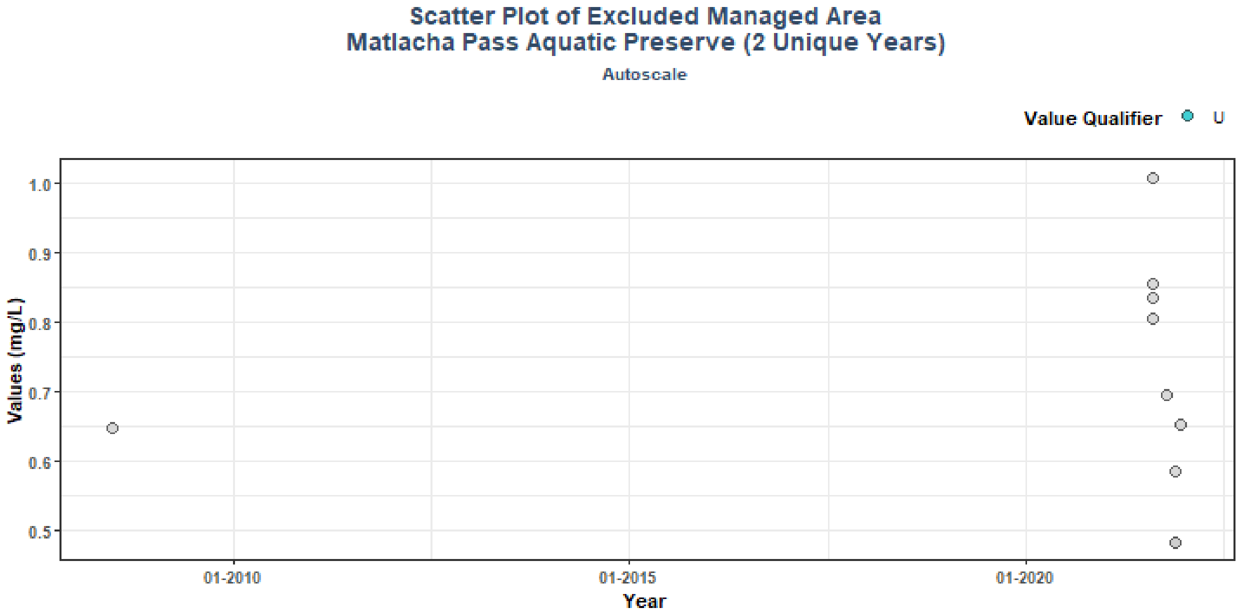


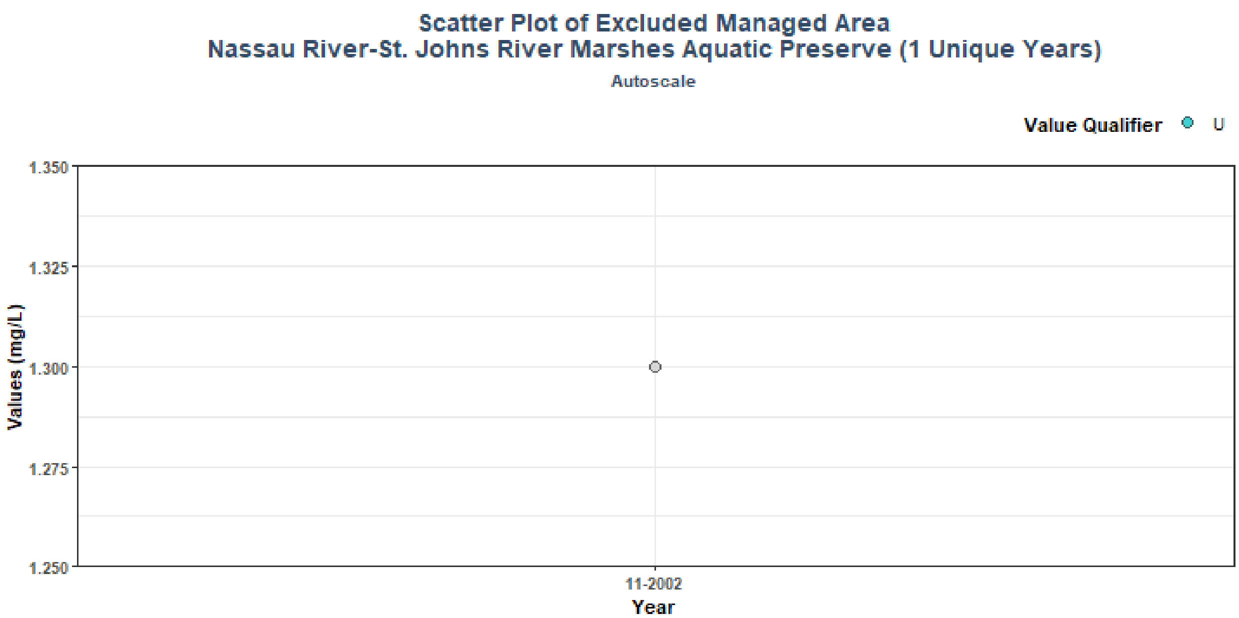


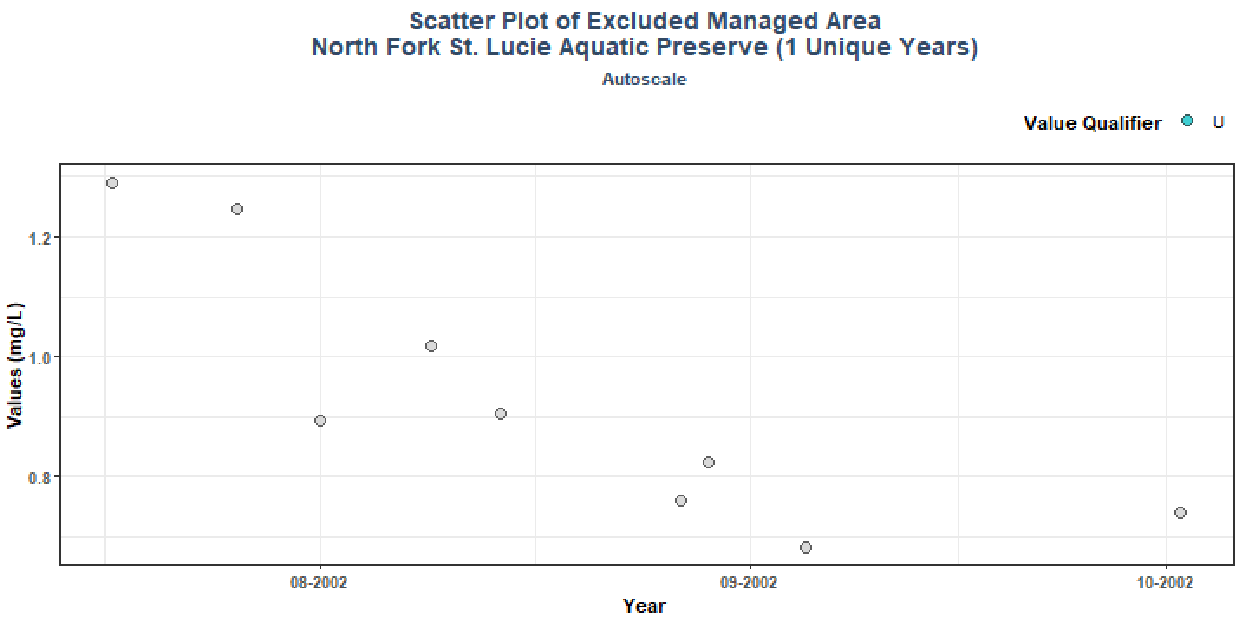
Scatter Plot of Excluded Managed Area
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
0.8
0.7
0.6
Values (mg/L)
10-2001
01-2002
04-2002
07-2002
Year



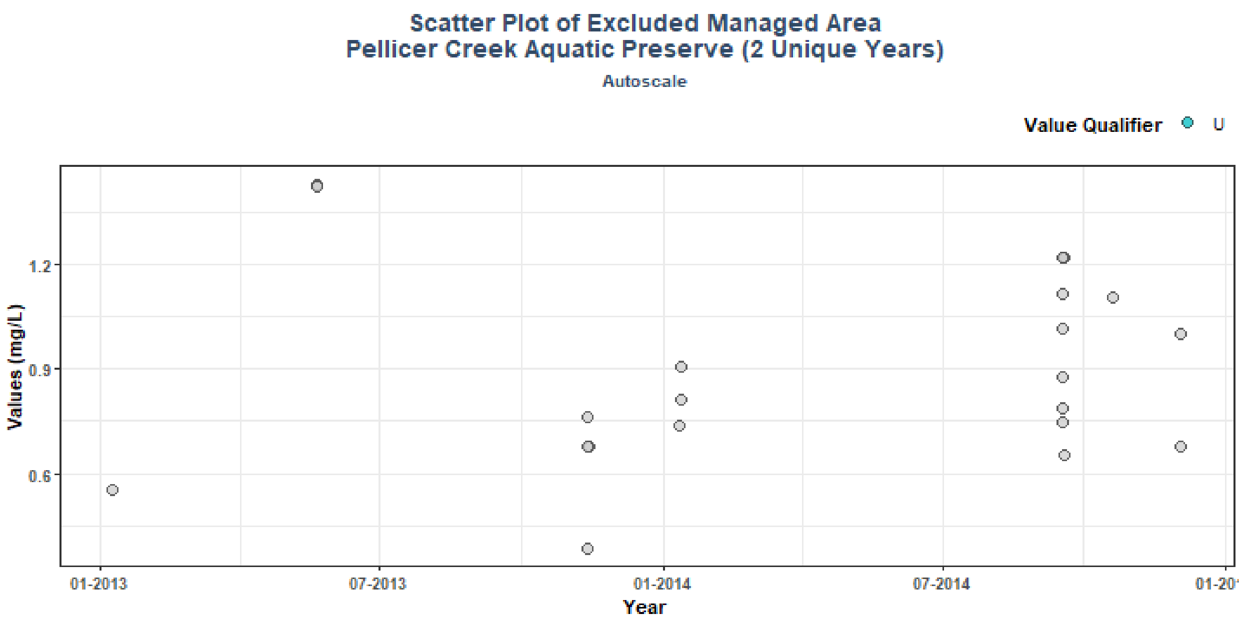
Scatter Plot of Excluded Managed Area
Lemon Bay Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier
U
0.64
0.62
0.60
0.58
0.56
0.54
Values (mg/L)
02-2020
Year


Scatter Plot of Excluded Managed Area
Matlacha Pass Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
1.0
0.9
0.8
0.7
0.6
0.5
01-2010
01-2015
01-2020
Year

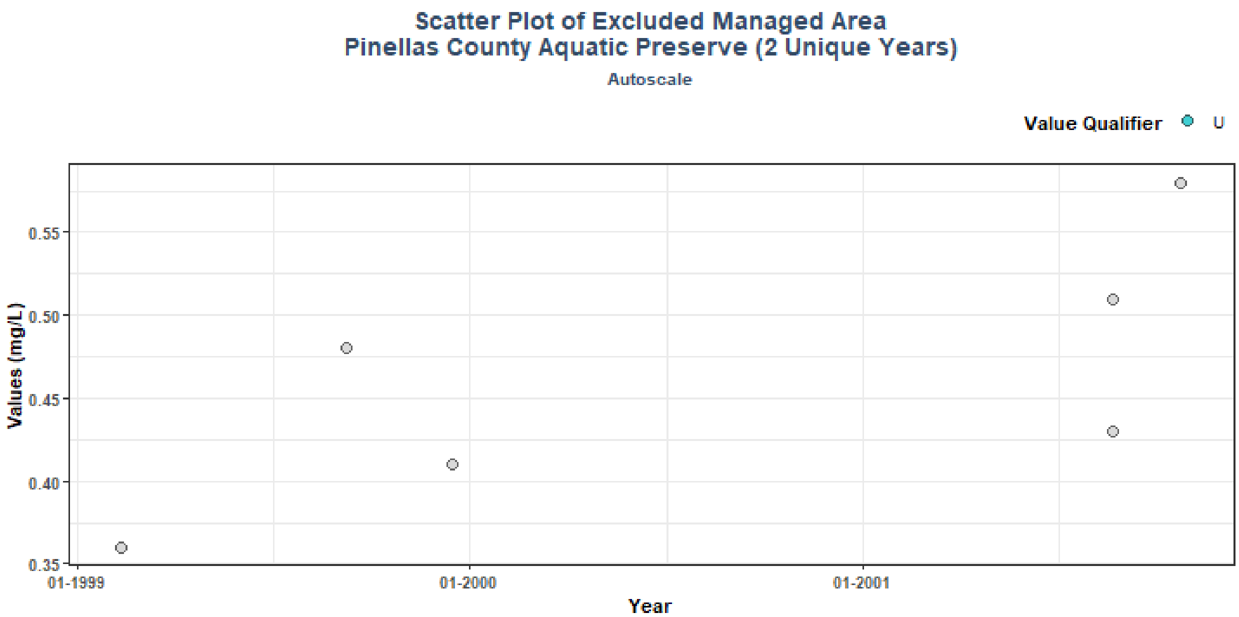

Scatter Plot of Excluded Managed Area
Nassau River-St. Johns River Marshes Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
1.350
1.325
1.300
1.275
1.250
11-2002
Year

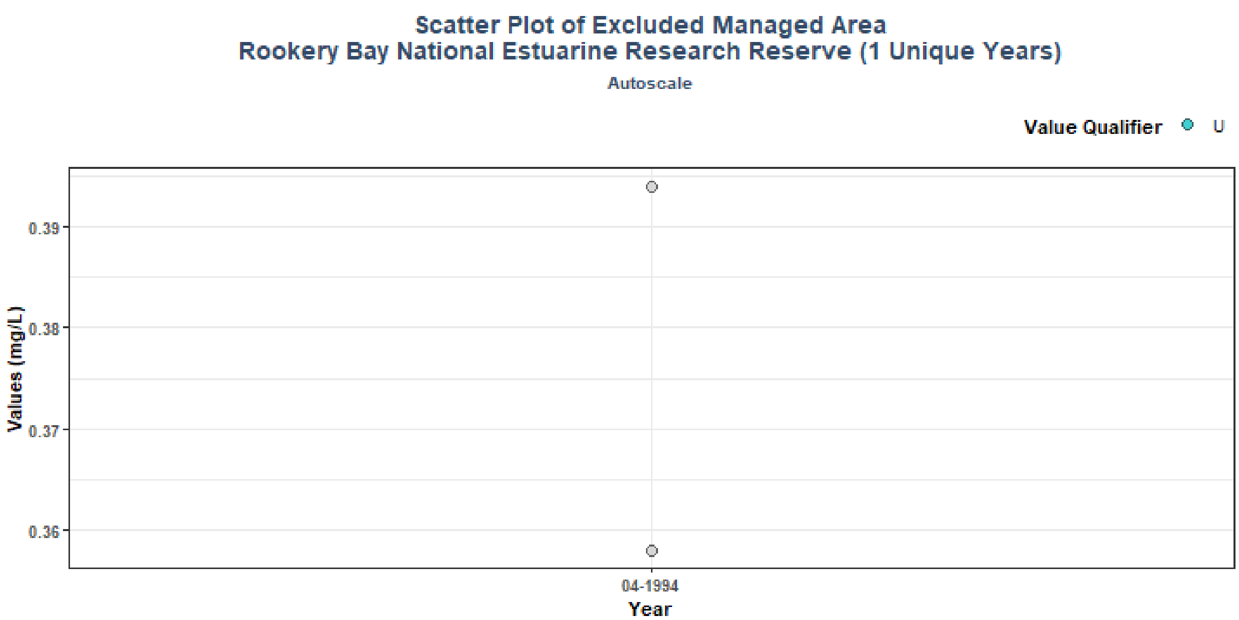


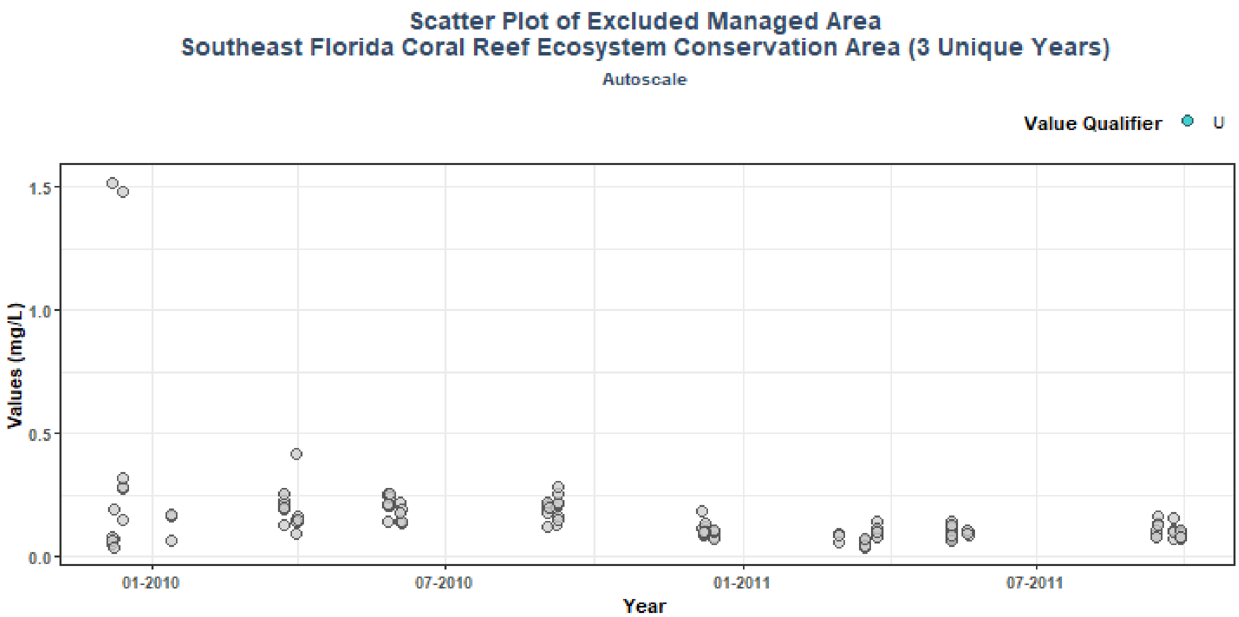
Scatter Plot of Excluded Managed Area
North Fork St. Lucie Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
1.2
1.0
0.8
08-2002
09-2002
10-2002
Year



Scatter Plot of Excluded Managed Area
Pellicer Creek Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
1.2
0.9
0.6
01-2013
07-2013
01-2014
07-2014
01-201
Year


Scatter Plot of Excluded Managed Area
Pinellas County Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
0.55
0.50
0.45
0.40
0.35
01-1999
01-2000
01-2001
Year


Scatter Plot of Excluded Managed Area
Rookery Bay National Estuarine Research Reserve (1 Unique Years)
Autoscale
Value Qualifier
U
Values (mg/L)
0.39
0.38
0.37
0.36
04-1994
Year



Appendix IV: Managed Area Trendlines

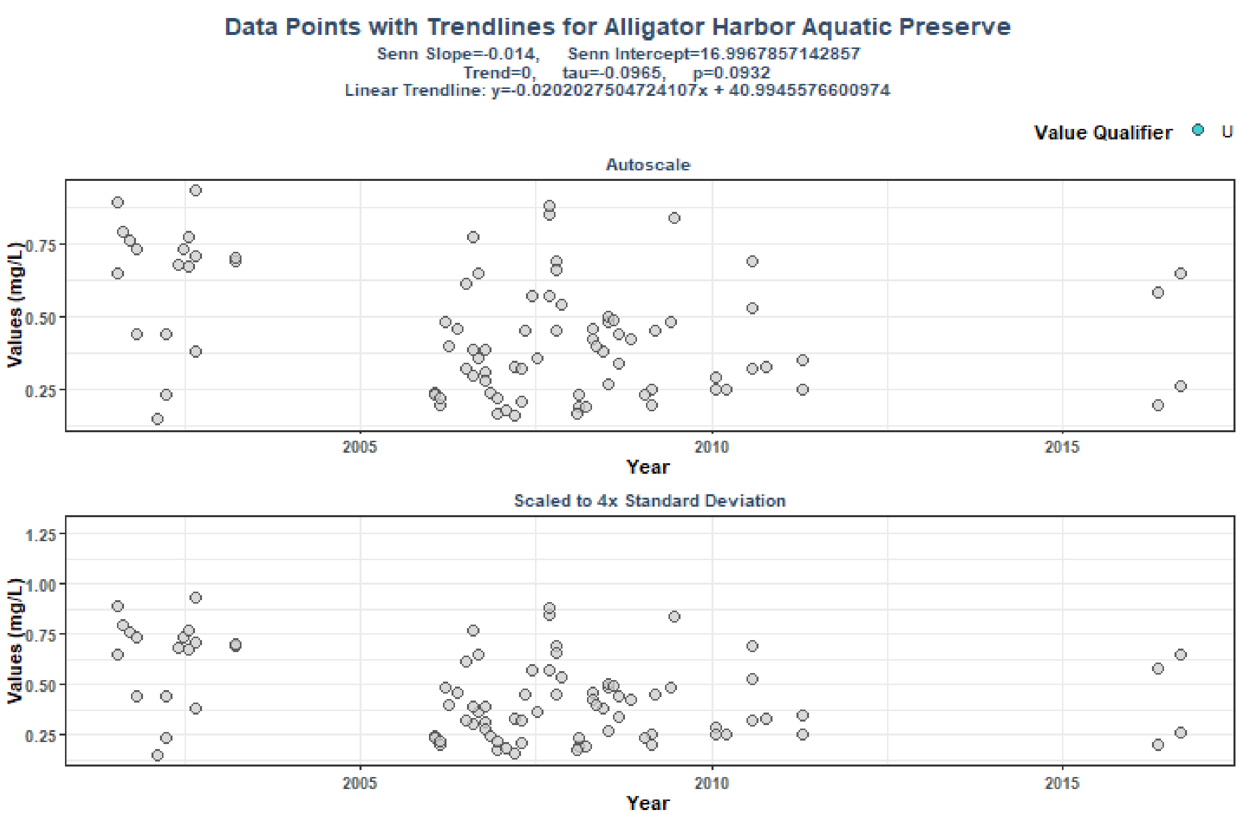
The plots created in this section are designed to show the general trend of the data. Data is taken and grouped by `ManagedAreaName`. The trendlines on the plots are created using the Senn slope and intercept from the seasonal Kendall Tau analysis. The scripts that create plots follow this format

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

```
if(n==0){
   print("There are no managed areas that qualify.")
} else {
   for (i in 1:n) {
      plot_data <- data[data$SufficientData==TRUE &
                           data$ManagedAreaName==MA_Include[i],]
      year_lower <- min(plot_data$Year)
      year_upper <- max(plot_data$Year)
      min_RV <- min(plot_data$ResultValue)
      mn_RV <- mean(plot_data$ResultValue[plot_data$ResultValue <
                                             quantile(data$ResultValue, 0.98)])
```

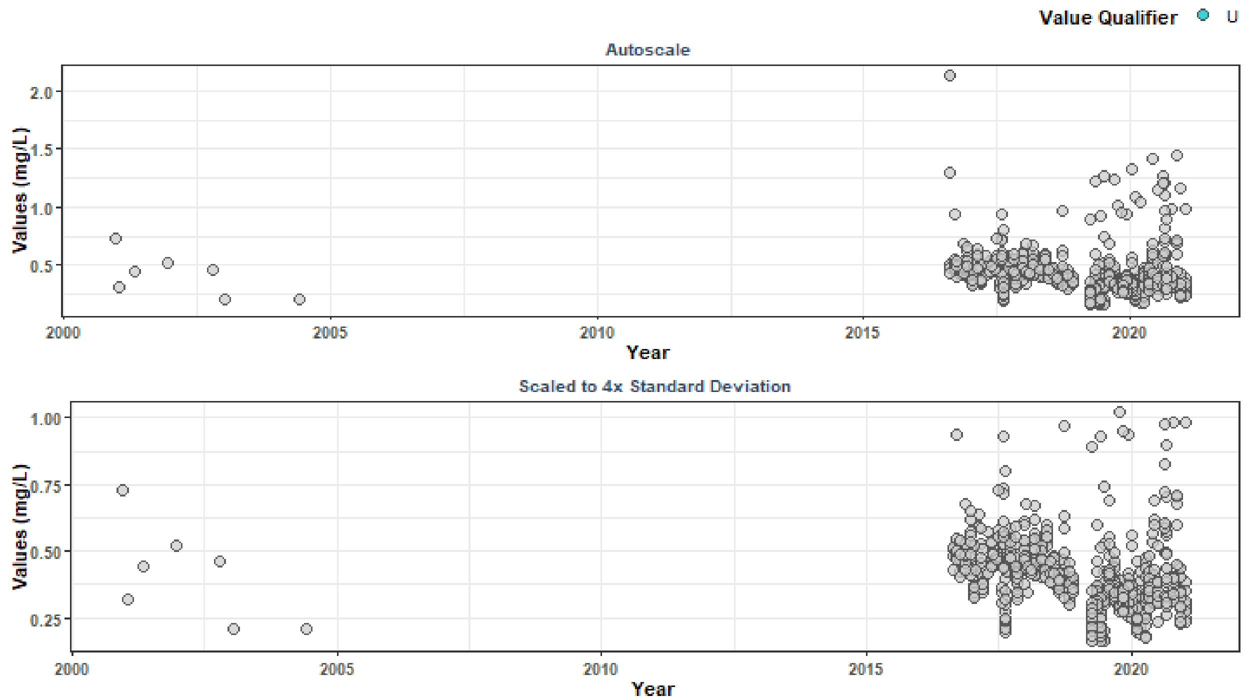
```
sd_RV <- sd(plot_data$ResultValue[plot_data$ResultValue <
                                    quantile(data$ResultValue, 0.98)])
x_scale <- ifelse(year_upper - year_lower > 30, 10, 5)
y_scale <- mn_RV + 4 * sd_RV

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

model <- lm(ResultValue ~ DecDate,
            data=plot_data)
m_int <- coef(model)[[1]]
m_slope <- coef(model)[[2]]
rm(model)
p1 <- ggplot(data=plot_data,
             aes(x=DecDate, y=ResultValue, fill=VQ_Plot)) +
   geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
   geom_abline(aes(slope=s_slope, intercept=s_int),
               color="blue", size=1.2) +
   labs(subtitle="Autoscale",
        x="Year", y=paste0("Values (", unit, ")"),
        fill="Value Qualifier") +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal",
         legend.justification="right") +
   {if(inc_H==TRUE){
      scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                                 "HU"="#7CAE00"), na.value="#cccccc")
   } else if(param_name=="Secchi_Depth"){
      scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                                 "SU"="#7CAE00"), na.value="#cccccc")
   } else {
      scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
   }}

p2 <- ggplot(data=plot_data,
             aes(x=DecDate, y=ResultValue, fill=VQ_Plot)) +
   geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
   geom_abline(aes(slope=s_slope, intercept=s_int),
               color="blue", size=1.2) +
   ylim(min_RV, y_scale) +
   labs(subtitle="Scaled to 4x Standard Deviation",
        x="Year", y=paste0("Values (", unit, ")")) +
   plot_theme +
   theme(legend.position="none") +
   {if(inc_H==TRUE){
      scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                                 "HU"="#7CAE00"), na.value="#cccccc")
   } else if(param_name=="Secchi_Depth"){
      scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                                 "SU"="#7CAE00"), na.value="#cccccc")
```

```
         } else {
            scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
         }}
      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_Include[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)) +
         plot_theme + theme(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)))
      rm(plot_data)
      rm(KTset, leg)
   }
}
```



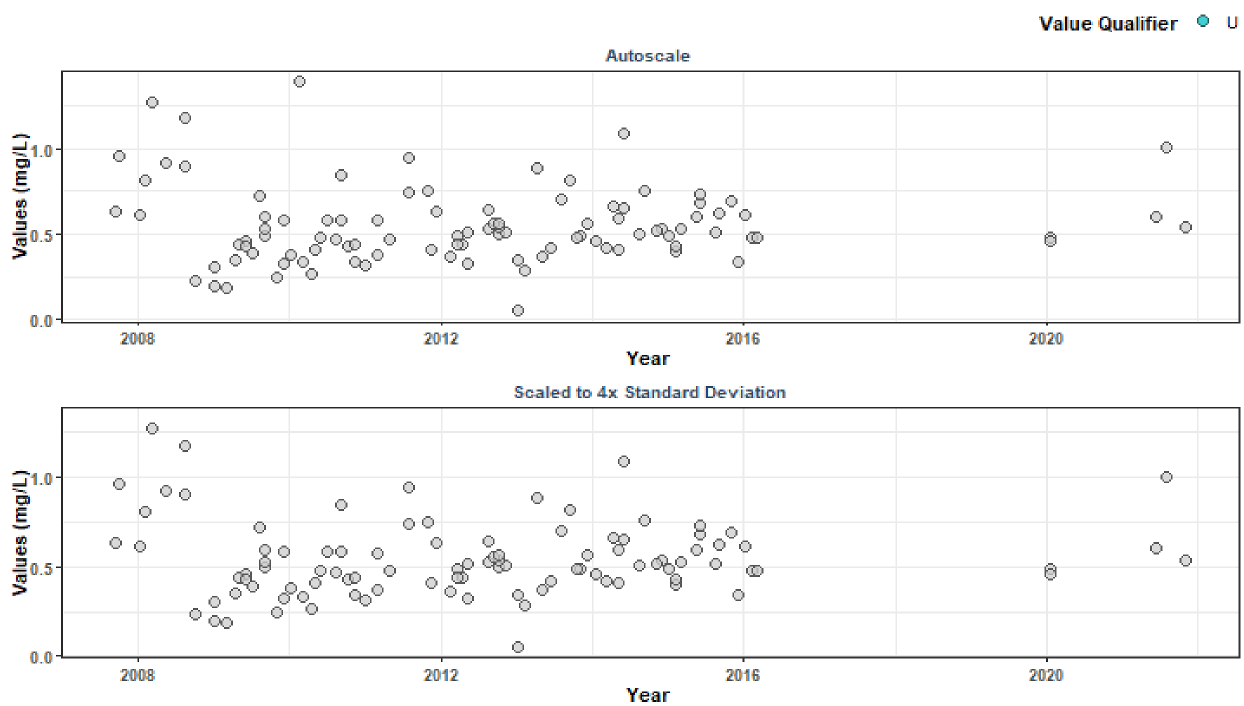
Data Points with Trendlines for Biscayne Bay Aquatic Preserve

Senn Slope=-0.031, Senn Intercept=83.956375
Trend=-1, tau=-0.2443, p=0
Linear Trendline: y=-0.00778934430820457x + 16.1645635301792



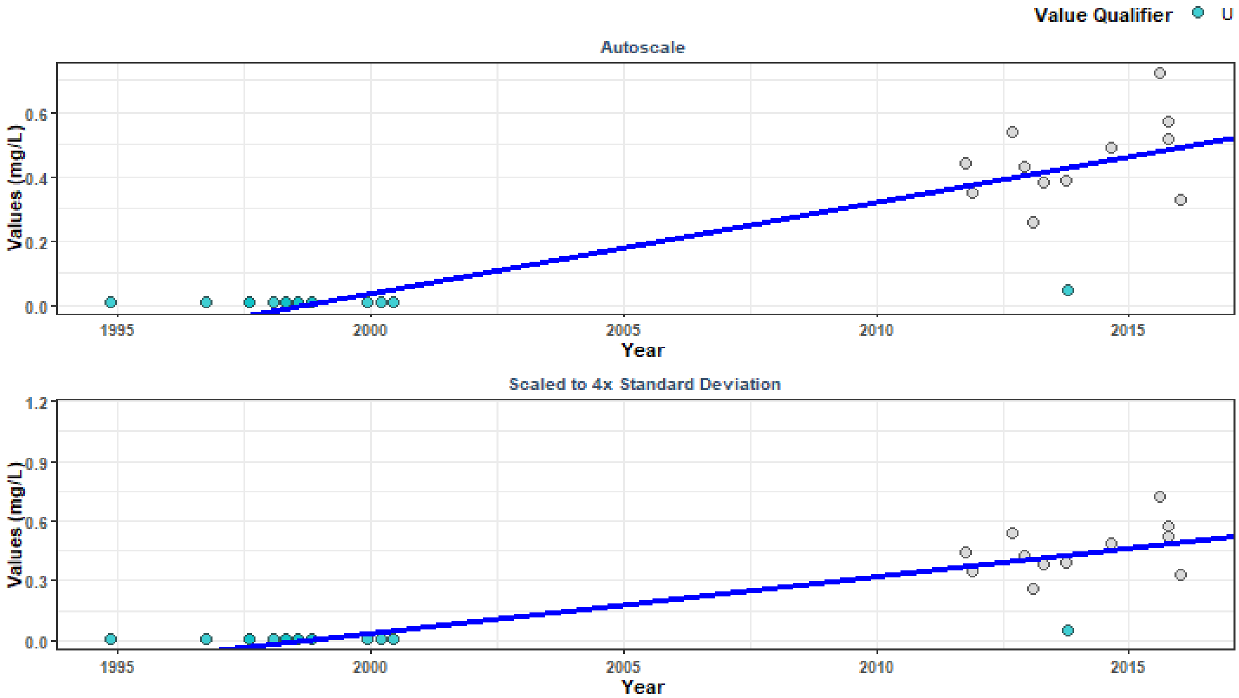
Data Points with Trendlines for Cape Haze Aquatic Preserve

Senn Slope=0.015, Senn Intercept=-33.824
Trend=0, tau=0.1779, p=0.0551
Linear Trendline: y=-0.000326836898200628x + 1.20791682471351



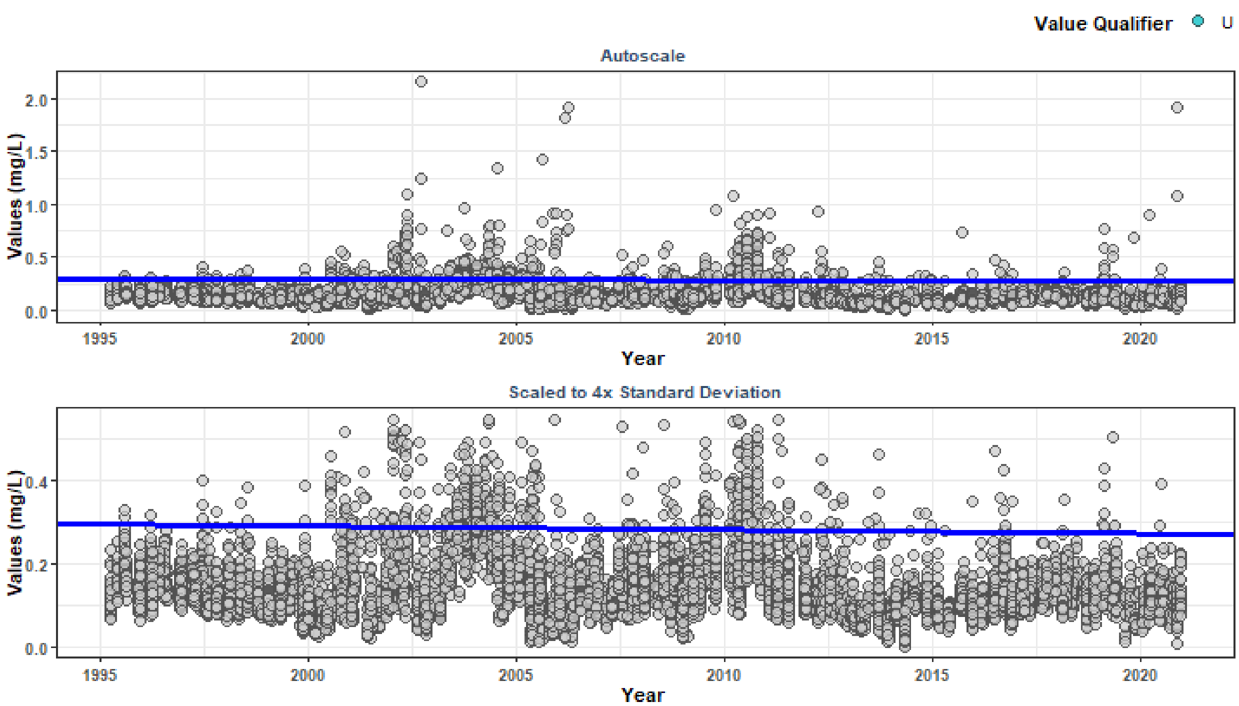
Data Points with Trendlines for Estero Bay Aquatic Preserve

Senn Slope=0.0283333333333333, Senn Intercept=-56.6283333333333
Trend=2, tau=0.697, p=0.0041
Linear Trendline: y=0.026095625880579x + -52.1360864224668



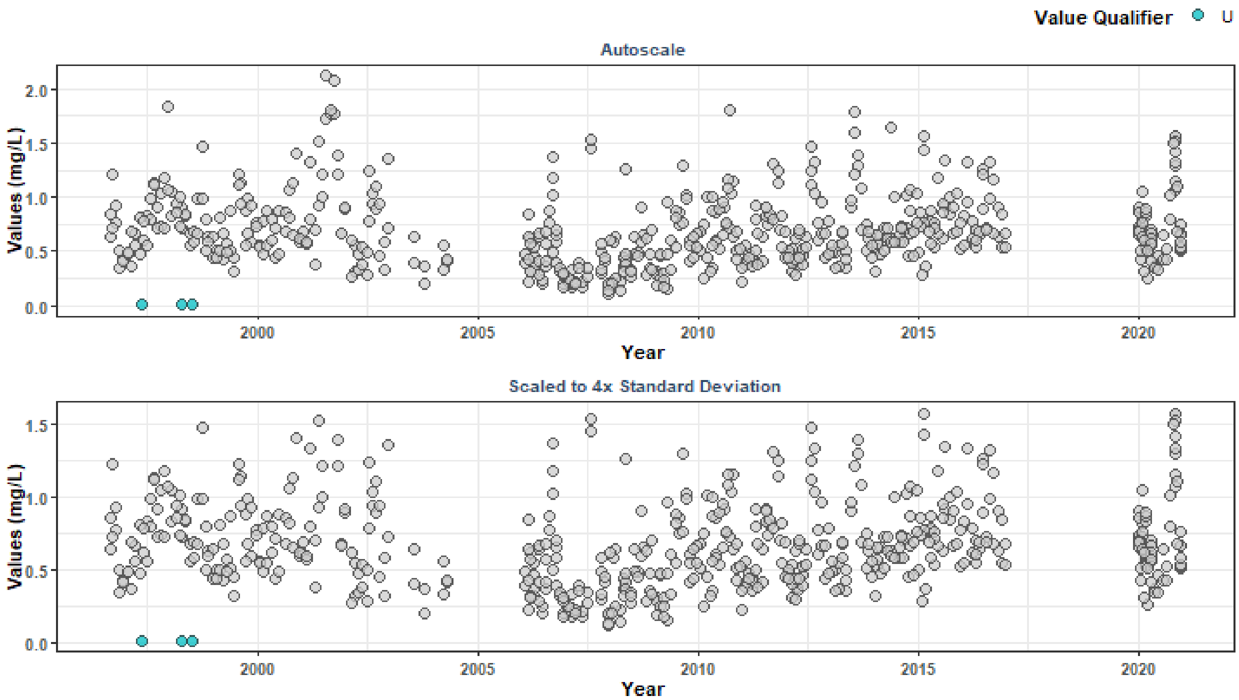
Data Points with Trendlines for Florida Keys National Marine Sanctuary

Senn Slope=-0.000899393333333333, Senn Intercept=2.08823015753846
Trend=-1, tau=-0.0715, p=0
Linear Trendline: y=-0.00148595098890679x + 3.13792200732661



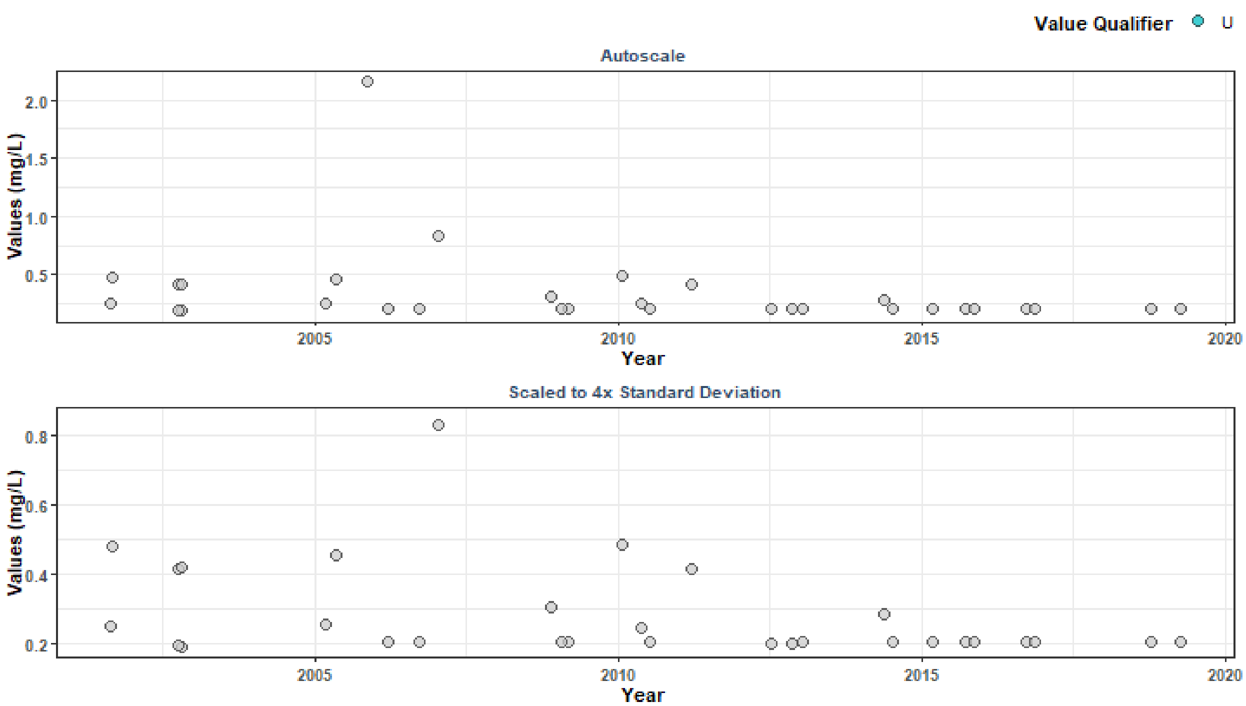
Data Points with Trendlines for Gasparilla Sound-Charlotte Harbor Aquatic Preserve

Senn Slope=0.00563961038961039, Senn Intercept=-5.41799999999999
Trend=1, tau=0.0915, p=0.0011
Linear Trendline: y=0.000755902446971553x + -0.831141821664503



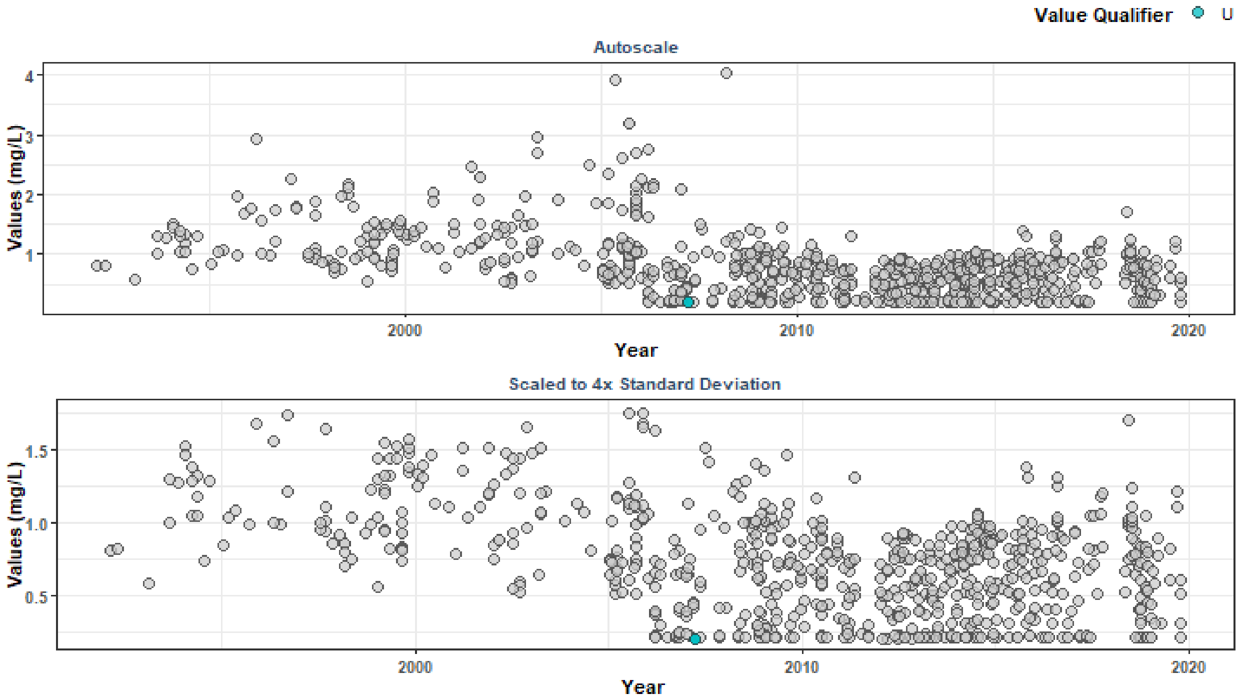
Data Points with Trendlines for Jensen Beach to Jupiter Inlet Aquatic Preserve

Senn Slope=-0.001, Senn Intercept=12.9051875
Trend=0, tau=-0.3036, p=0.112
Linear Trendline: y=-0.0183824183677122x + 37.2902750889013



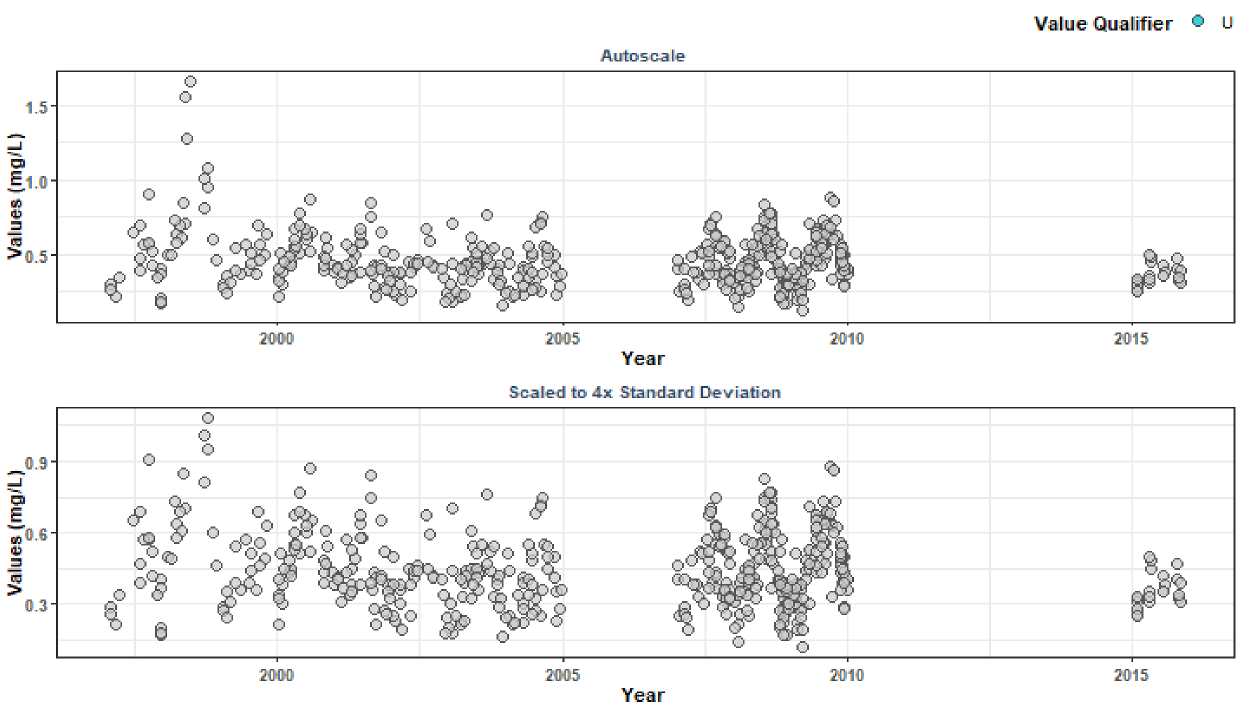
Data Points with Trendlines for Loxahatchee River-Lake Worth Creek Aquatic Preserve

Senn Slope=-0.0301666666666667, Senn Intercept=54.5074578088578
Trend=-1, tau=-0.2493, p=0
Linear Trendline: y=-0.0377049325466528x + 76.5785836322848



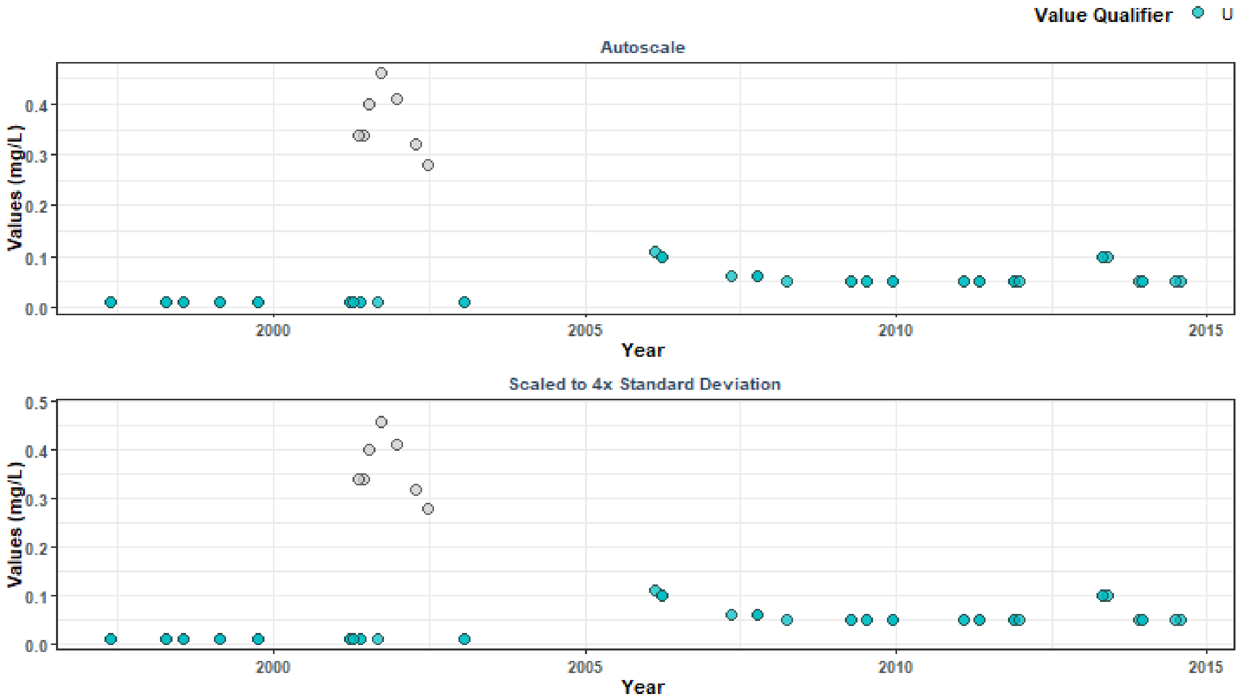
Data Points with Trendlines for Nature Coast Aquatic Preserve

Senn Slope=0, Senn Intercept=2.09875
Trend=0, tau=-0.0287, p=0.5363
Linear Trendline: y=-0.00556554701937702x + 11.6171794891473



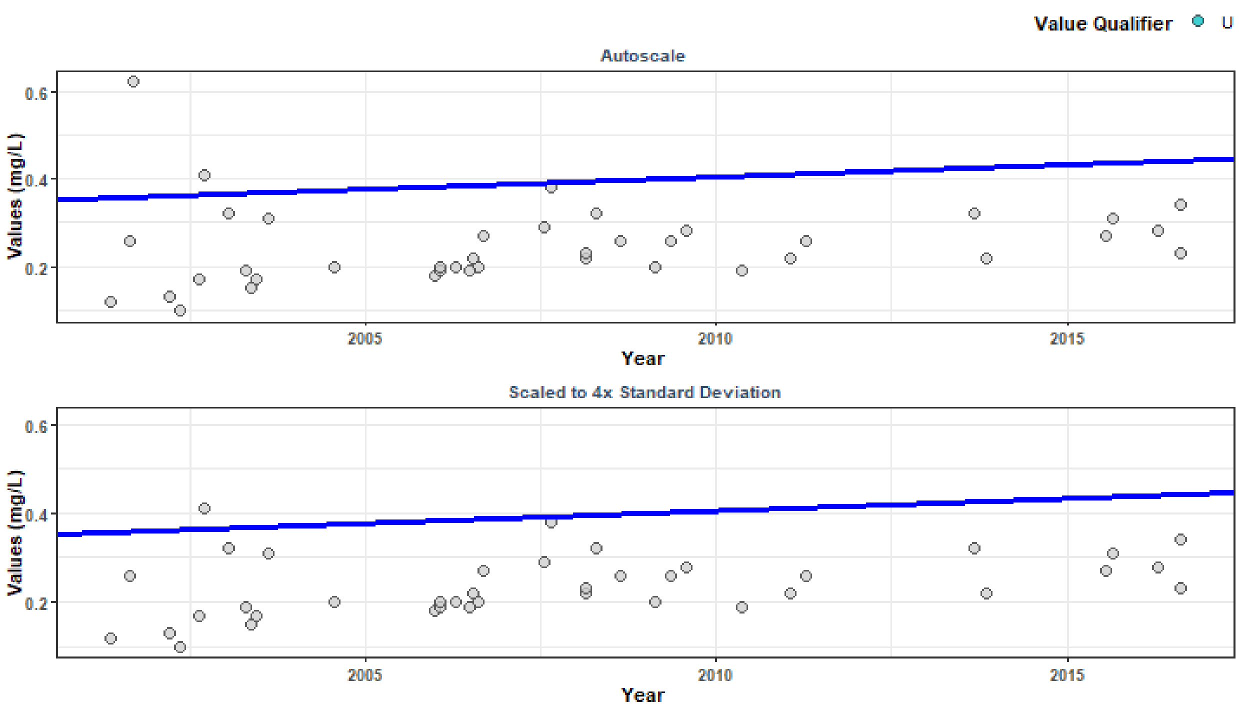
Data Points with Trendlines for Pine Island Sound Aquatic Preserve

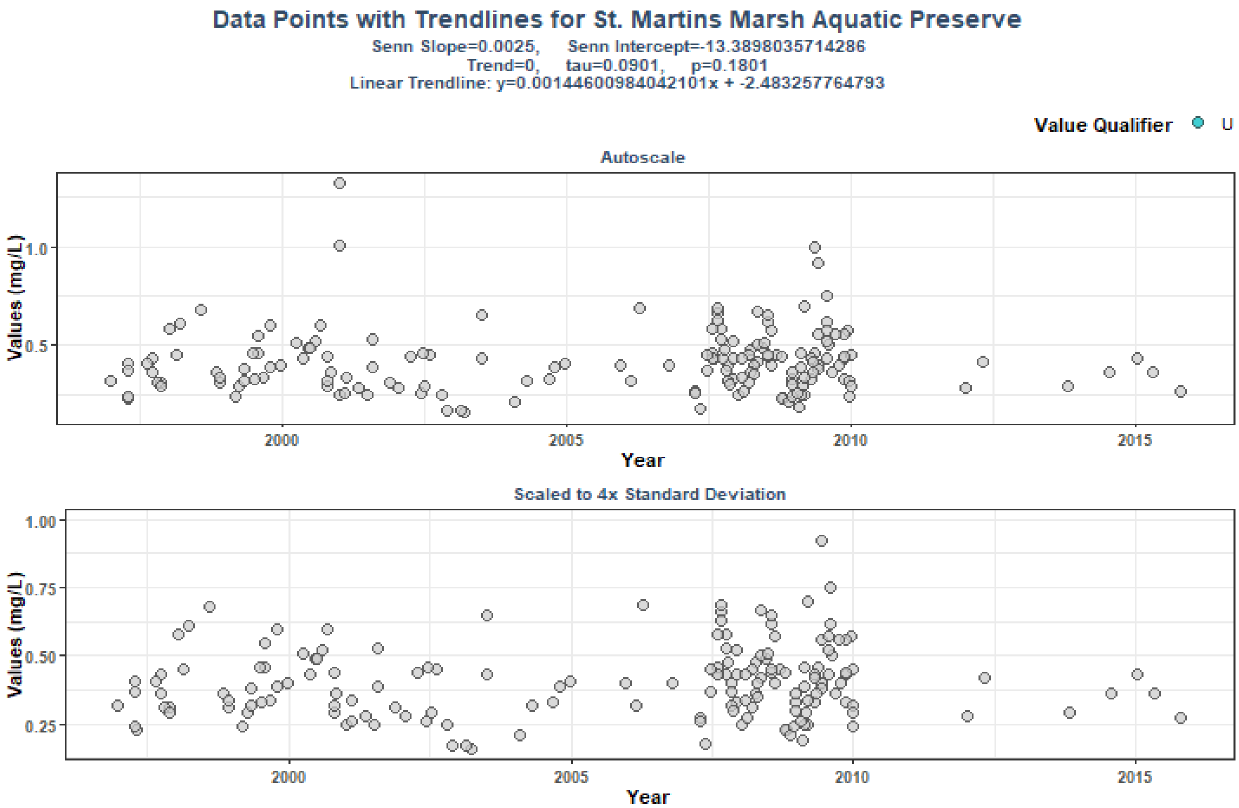
Senn Slope=0.00363636363636364, Senn Intercept=-6.65333333333333
Trend=1, tau=0.2777, p=0
Linear Trendline: y=0.000583029337021343x + -1.09935281162516



Data Points with Trendlines for St. Joseph Bay Aquatic Preserve

Senn Slope=0.00555555555555556, Senn Intercept=-10.762380952381
Trend=0, tau=0.1869, p=0.1535
Linear Trendline: y=0.00280021831974189x + -5.37473397094402





Appendix V: Managed Area Summary Box Plots

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

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

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

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

```r
if(n==0){
  print("There are no managed areas that qualify.")
} else {
  for (i in 1:n) {
    plot_data <- data[data$SufficientData==TRUE &
                        data$ManagedAreaName==MA_Include[i],]
    year_lower <- min(plot_data$Year)
    year_upper <- max(plot_data$Year)
    min_RV <- min(plot_data$ResultValue)
    mn_RV <- mean(plot_data$ResultValue[plot_data$ResultValue <
                                          quantile(data$ResultValue, 0.98)])
    sd_RV <- sd(plot_data$ResultValue[plot_data$ResultValue <
                                        quantile(data$ResultValue, 0.98)])
    x_scale <- ifelse(year_upper - year_lower > 30, 10, 5)
    y_scale <- mn_RV + 4 * sd_RV

    ##Year plots
    p1 <- ggplot(data=plot_data,
                 aes(x=Year, y=ResultValue, group=Year)) +
      geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                   outlier.size=3, outlier.color="#333333",
                   outlier.fill="#cccccc", outlier.alpha=0.75) +
      labs(subtitle="Autoscale",
           x="Year", y=paste0("Values (", unit, ")")) +
      scale_x_continuous(limits=c(year_lower - 1, year_upper + 1),
                         breaks=rev(seq(year_upper,
                                        year_lower, -x_scale))) +
      plot_theme

    p2 <- ggplot(data=plot_data,
                 aes(x=Year, y=ResultValue, group=Year)) +
      geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                   outlier.size=3, outlier.color="#333333",
                   outlier.fill="#cccccc", outlier.alpha=0.75) +
      labs(subtitle="Scaled to 4x Standard Deviation",
           x="Year", y=paste0("Values (", unit, ")")) +
      ylim(min_RV, y_scale) +
      scale_x_continuous(limits=c(year_lower - 1, year_upper + 1),
                         breaks=rev(seq(year_upper,
                                        year_lower, -x_scale))) +
      plot_theme

    p3 <- ggplot(data=plot_data[plot_data$Year >= year_upper - 10, ],
                 aes(x=Year, y=ResultValue, group=Year)) +
      geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                   outlier.size=3, outlier.color="#333333",
                   outlier.fill="#cccccc", outlier.alpha=0.75) +
      labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",
           x="Year", y=paste0("Values (", unit, ")")) +
      ylim(min_RV, y_scale) +
      scale_x_continuous(limits=c(year_upper - 10.5, year_upper + 1),
                         breaks=rev(seq(year_upper, year_upper - 10,-2))) +
      plot_theme
```

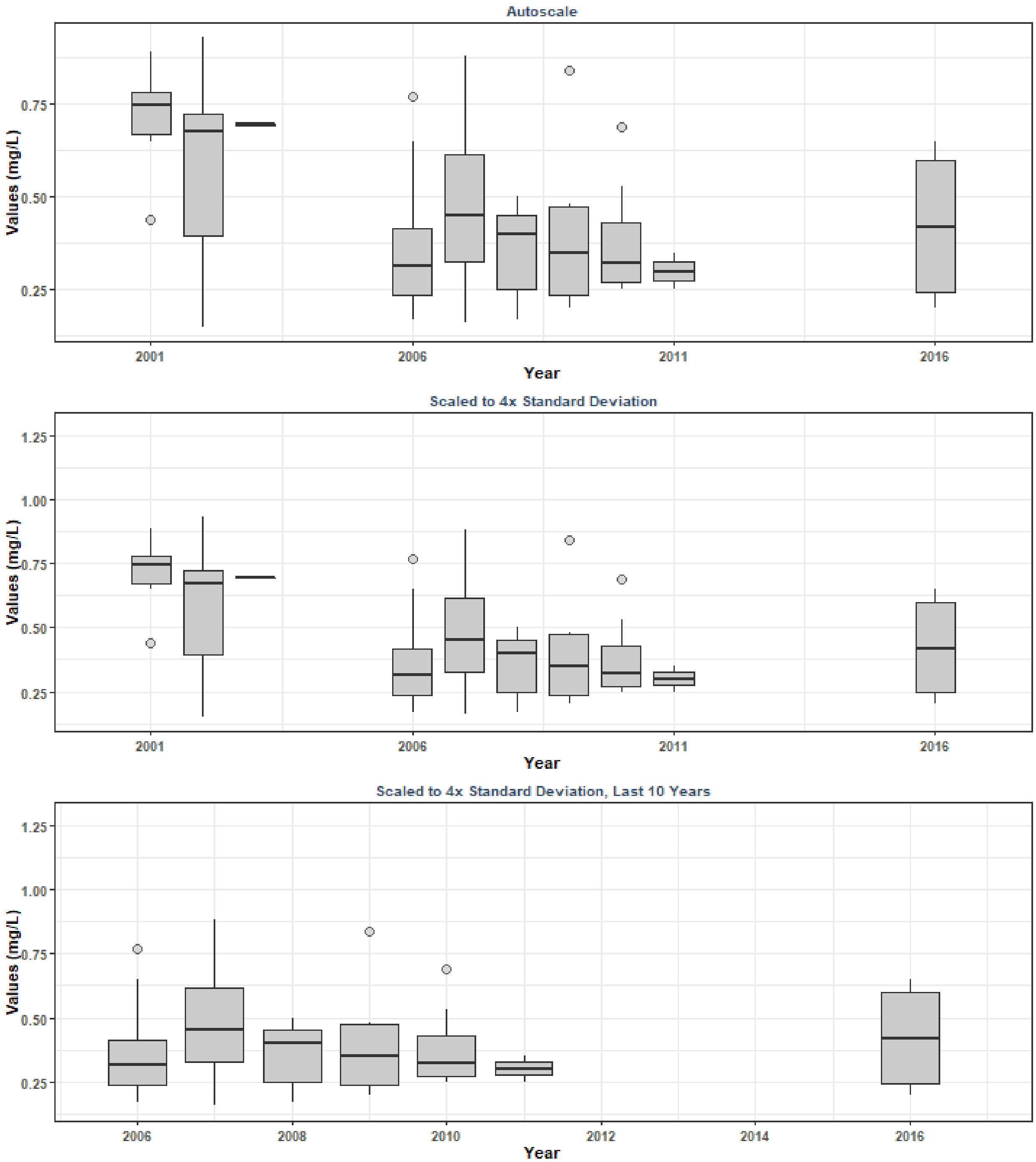
```r
Yset <- ggarrange(p1, p2, p3, ncol=1)

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


## Year & Month Plots
p4 <- ggplot(data=plot_data,
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Autoscale",
        x="Year", y=paste0("Values (", unit, ")"), color="Month") +
   scale_x_continuous(limits=c(year_lower - 1, year_upper + 1),
                      breaks=rev(seq(year_upper,
                                     year_lower, -x_scale))) +
   plot_theme +
   theme(legend.position="none")

p5 <- ggplot(data=plot_data,
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation",
        x="Year", y=paste0("Values (", unit, ")"), color="Month") +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(year_lower - 1, year_upper + 1),
                      breaks=rev(seq(year_upper,
                                     year_lower, -x_scale))) +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal") +
   guides(color=guide_legend(nrow=1))

p6 <- ggplot(data=plot_data[plot_data$Year >= year_upper - 10, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",
        x="Year", y=paste0("Values (", unit, ")"), color="Month") +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(year_upper - 10.5, year_upper + 1),
                      breaks=rev(seq(year_upper, year_upper - 10,-2))) +
   plot_theme +
   theme(legend.position="none")

leg1 <- get_legend(p5)
YMset <- ggarrange(leg1, p4, p5 + theme(legend.position="none"), p6,
                   ncol=1, heights=c(0.1, 1, 1, 1))
```

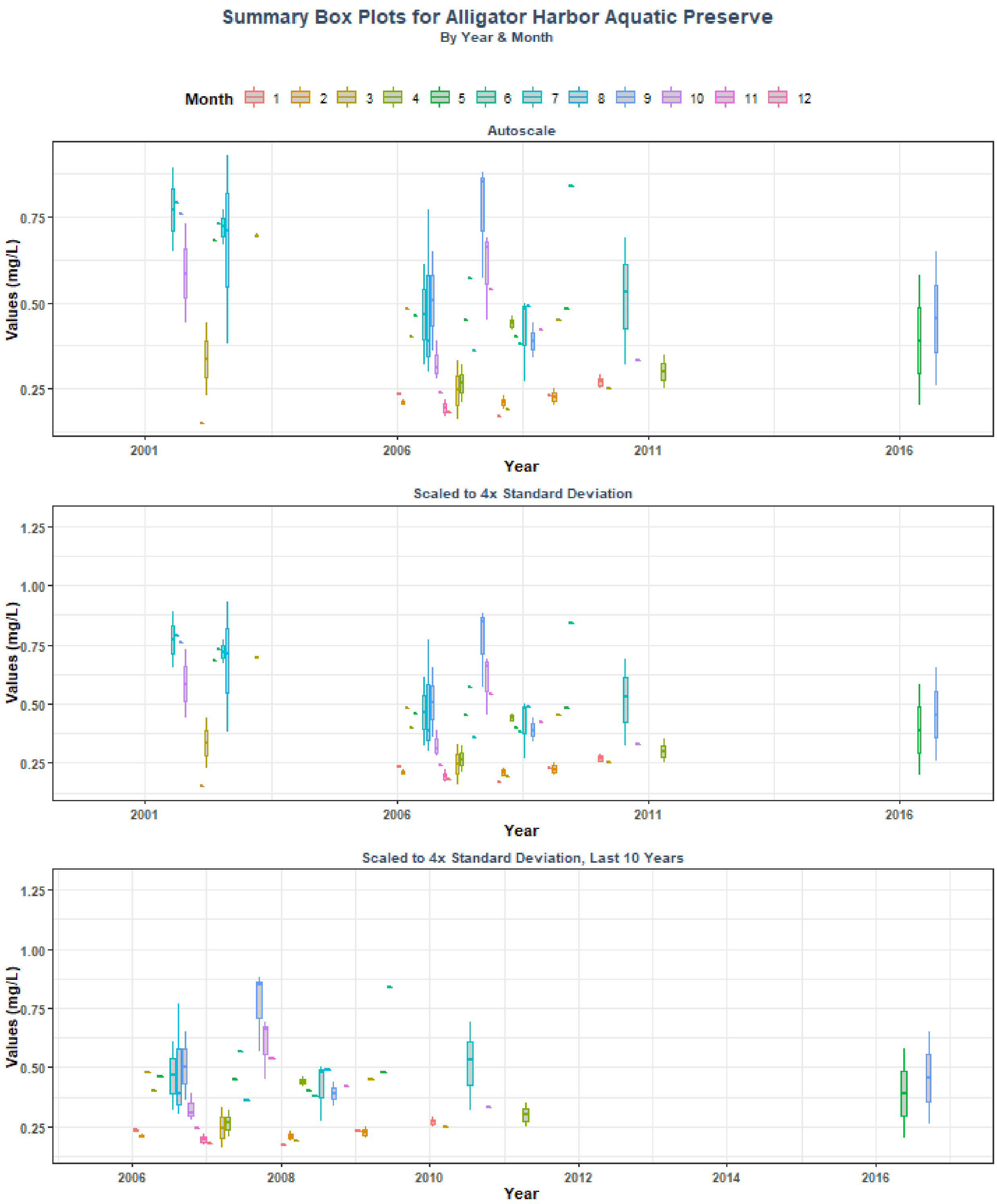
```
p00 <- ggplot() + labs(title=paste0("Summary Box Plots for ",
                                    MA_Include[i]),
                       subtitle="By Year & Month") + plot_theme +
  theme(panel.border=element_blank(),
        panel.grid.major=element_blank(),
        panel.grid.minor=element_blank(), axis.line=element_blank())

## Month Plots
p7 <- ggplot(data=plot_data,
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
  geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
               outlier.color="#333333", outlier.alpha=0.75) +
  labs(subtitle="Autoscale",
       x="Month", y=paste0("Values (", unit, ")"), fill="Month") +
  scale_x_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +
  plot_theme +
  theme(legend.position="none")

p8 <- ggplot(data=plot_data,
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
  geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
               outlier.color="#333333", outlier.alpha=0.75) +
  labs(subtitle="Scaled to 4x Standard Deviation",
       x="Month", y=paste0("Values (", unit, ")"), fill="Month") +
  ylim(min_RV, y_scale) +
  scale_x_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +
  plot_theme +
  theme(legend.position="top", legend.box="horizontal") +
  guides(fill=guide_legend(nrow=1))

p9 <- ggplot(data=plot_data[plot_data$Year >= year_upper - 10, ],
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
  geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
               outlier.color="#333333", outlier.alpha=0.75) +
  labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",
       x="Month", y=paste0("Values (", unit, ")"), fill="Month") +
  ylim(min_RV, y_scale) +
  scale_x_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +
  plot_theme +
  theme(legend.position="none")

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

p000 <- ggplot() + labs(title=paste0("Summary Box Plots for ",
                                     MA_Include[i]),
                        subtitle="By Month") + plot_theme +
  theme(panel.border=element_blank(),
        panel.grid.major=element_blank(),
```

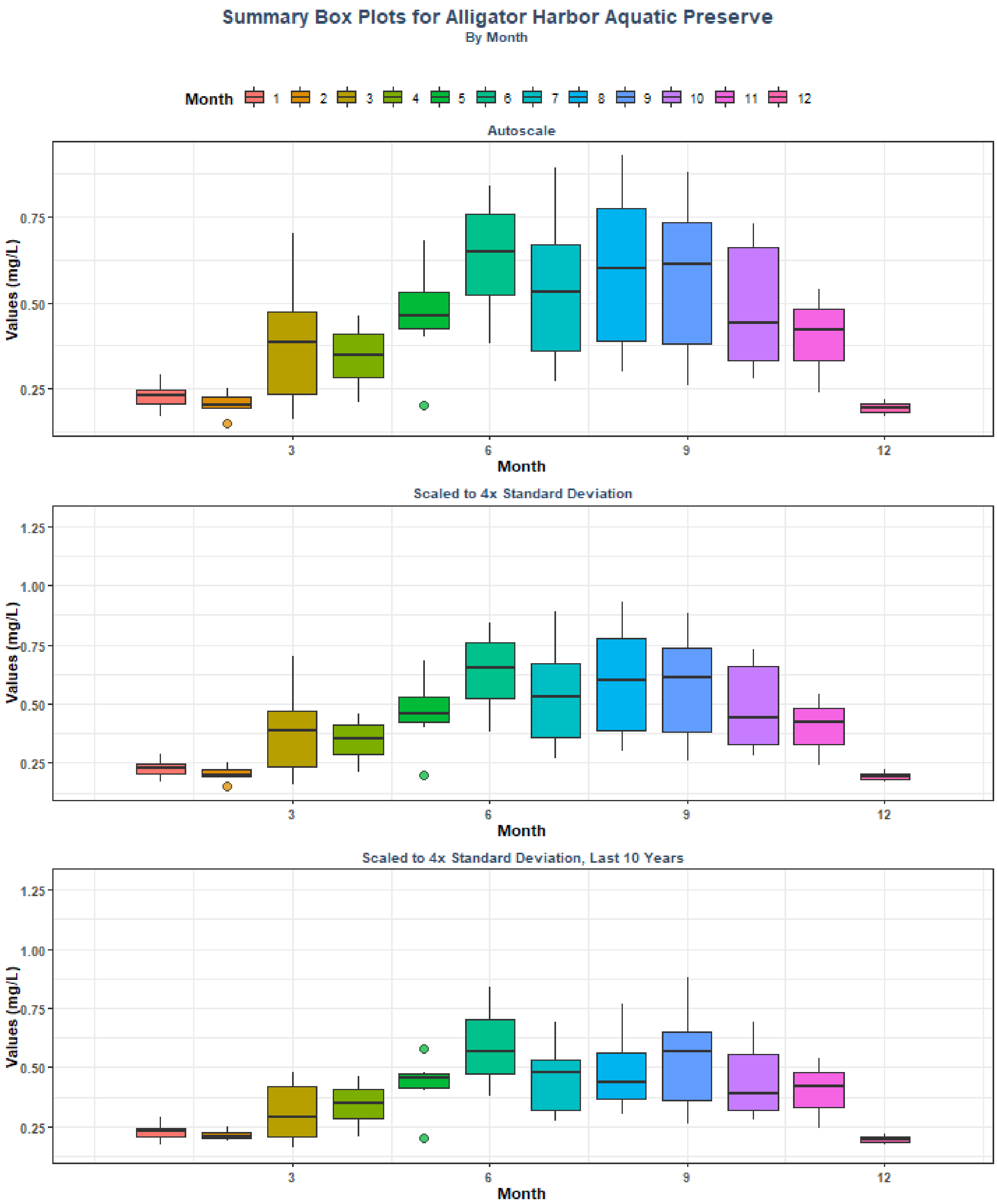
```
             panel.grid.minor=element_blank(), axis.line=element_blank())

      print(ggarrange(p0, Yset, ncol=1, heights=c(0.07, 1)))
      print(ggarrange(p00, YMset, ncol=1, heights=c(0.07, 1)))
      print(ggarrange(p000, Mset, ncol=1, heights=c(0.07, 1, 0.7)))

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

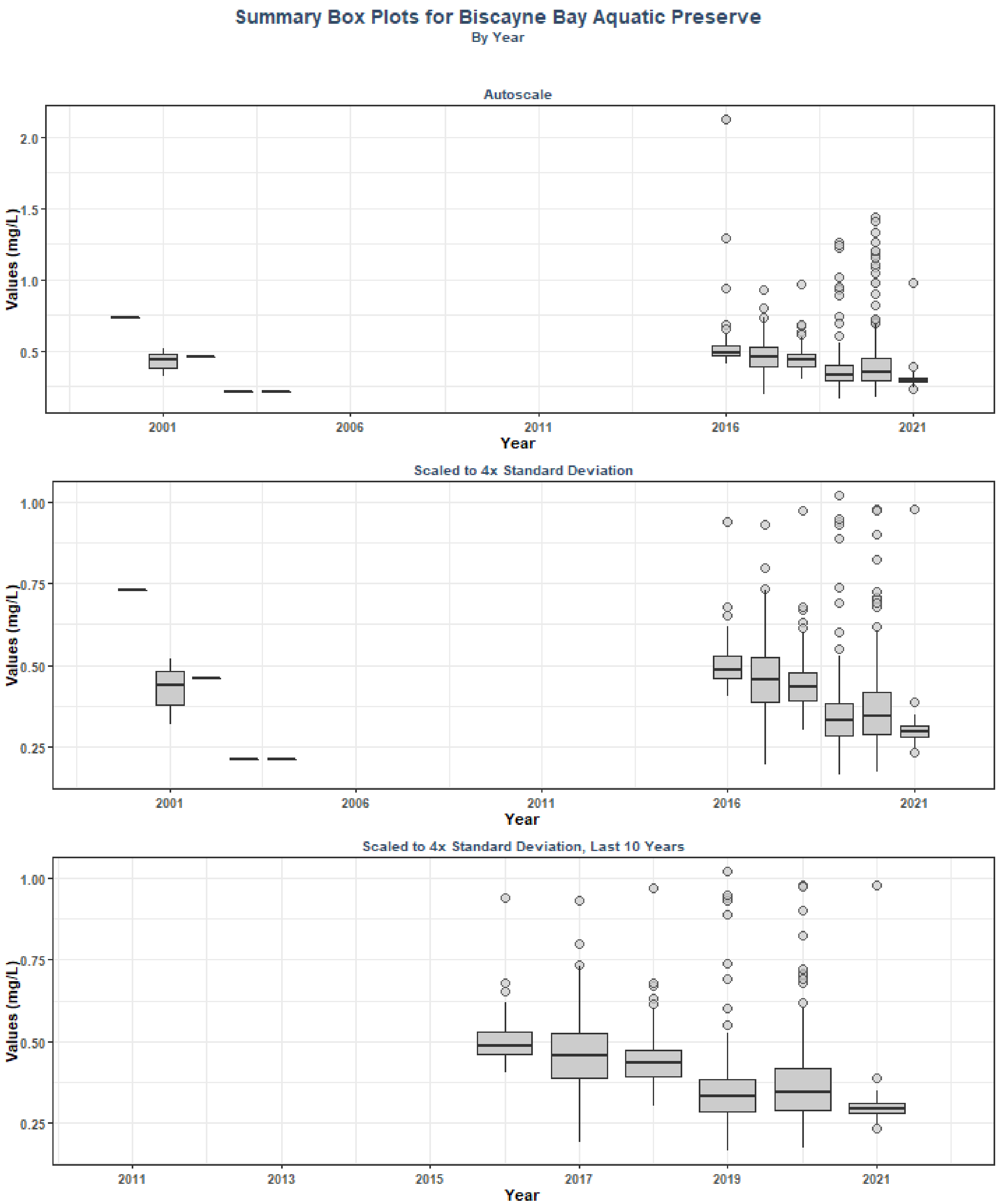
Summary Box Plots for Alligator Harbor Aquatic Preserve

By Year

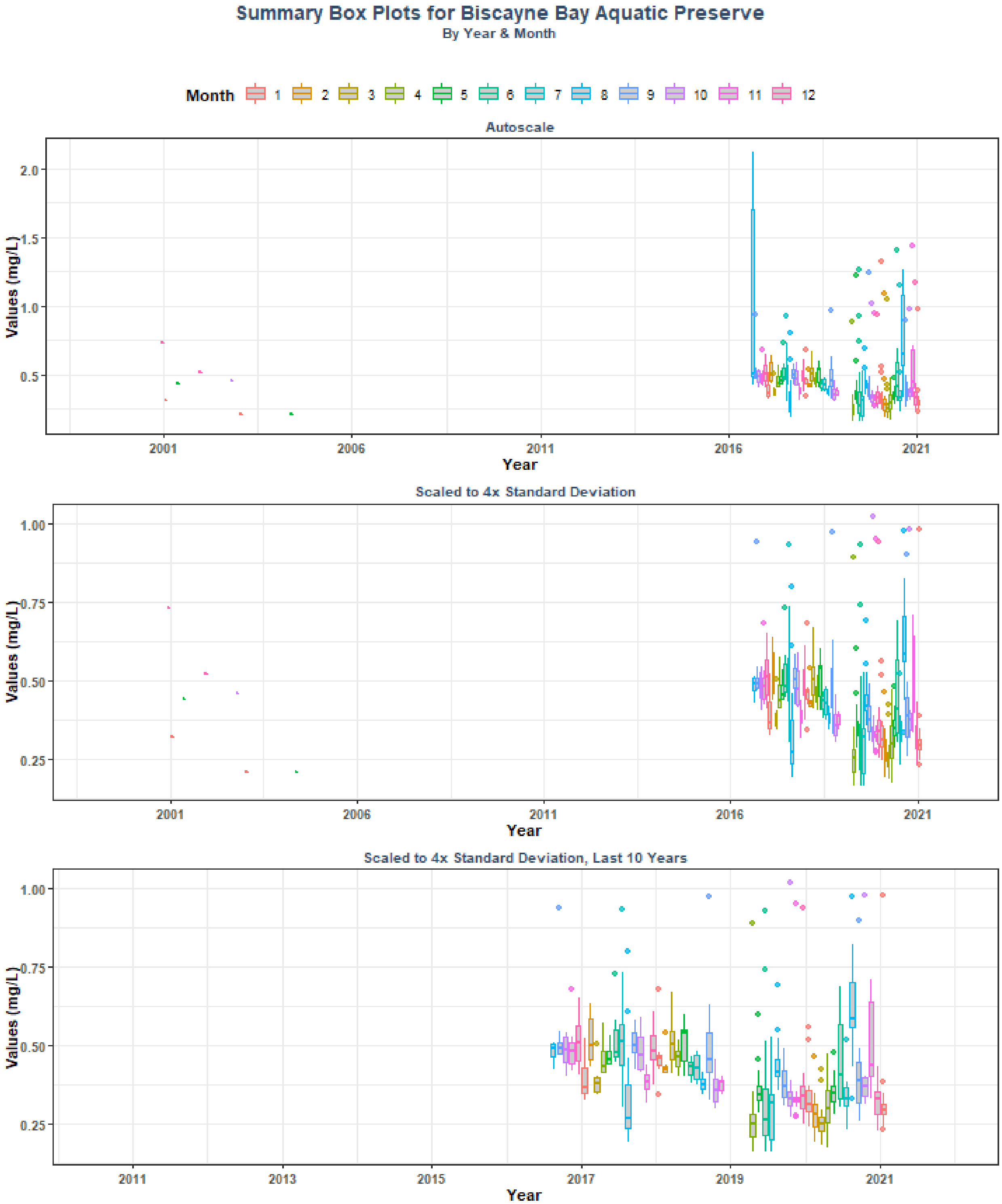


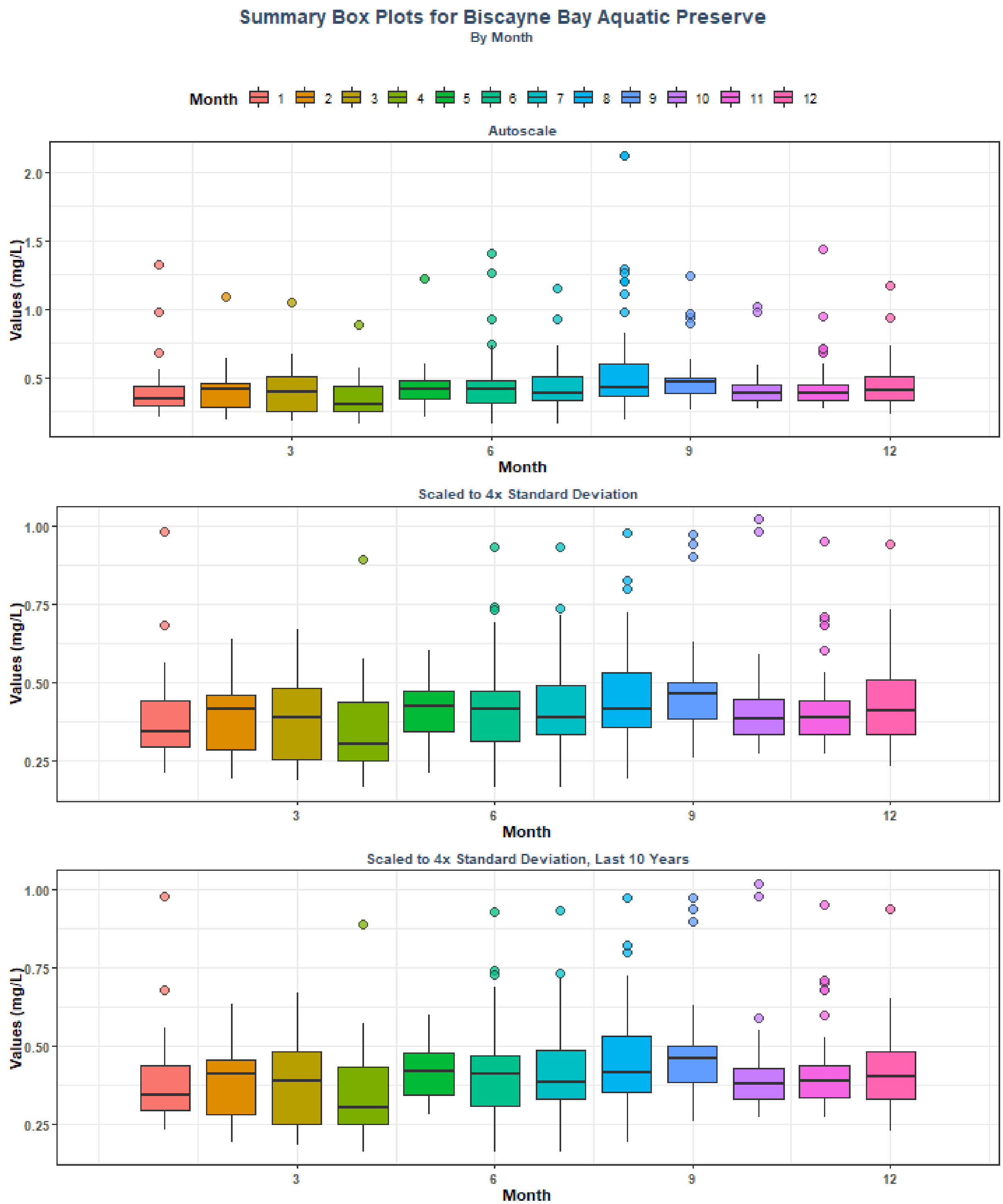

Summary Box Plots for Alligator Harbor Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
0.75
0.50
0.25
2001
2006
2011
2016
Year
Scaled to 4x Standard Deviation
Values (mg/L)
1.25
1.00
0.75
0.50
0.25
2001
2006
2011
2016
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
1.25
1.00
0.75
0.50
0.25
2006
2008
2010
2012
2014
2016
Year


Summary Box Plots for Alligator Harbor Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
0.25
0.50
0.75
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
0.25
0.50
0.75
1.00
1.25
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.25
0.50
0.75
1.00
1.25
3
6
9
12
Month

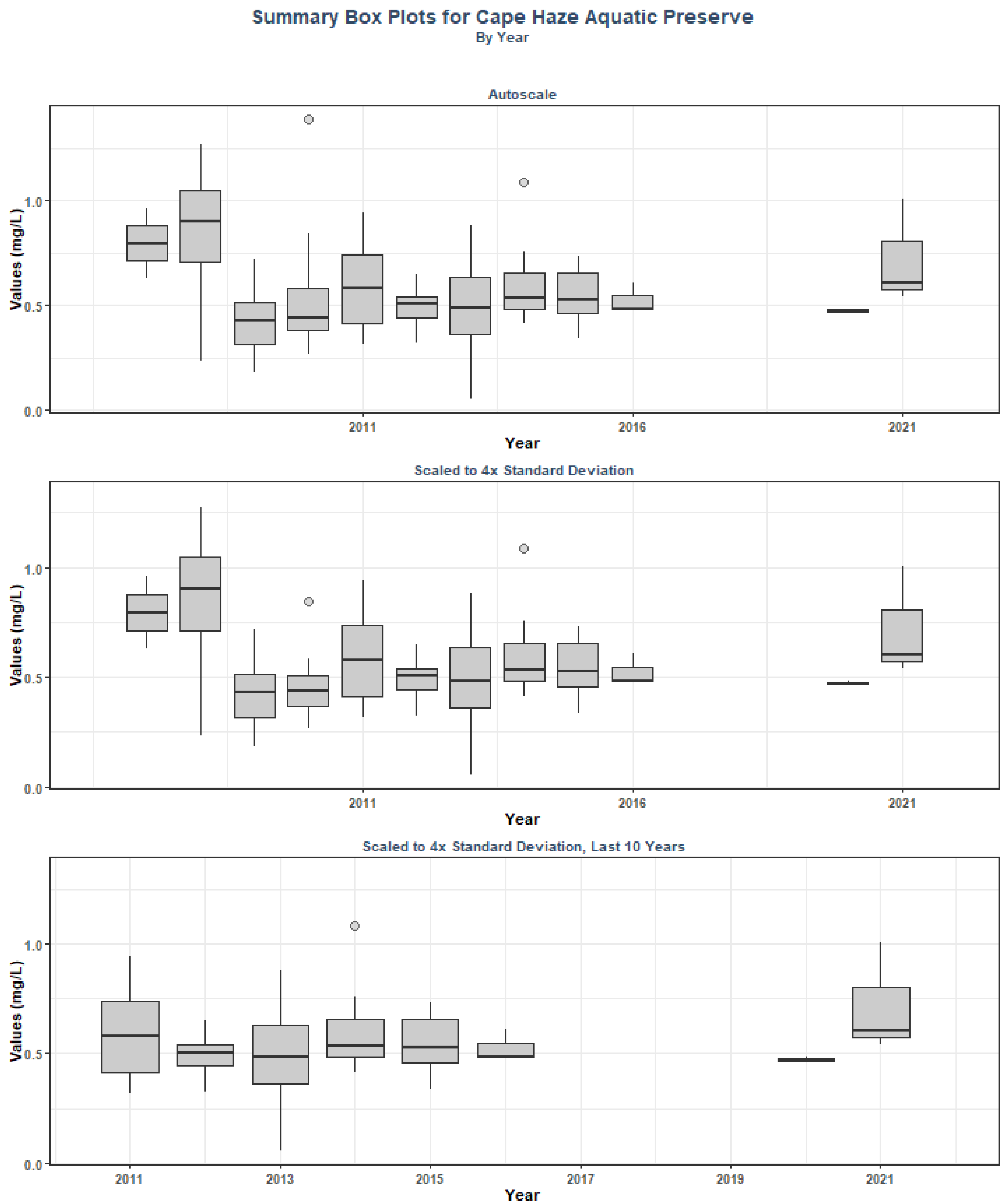


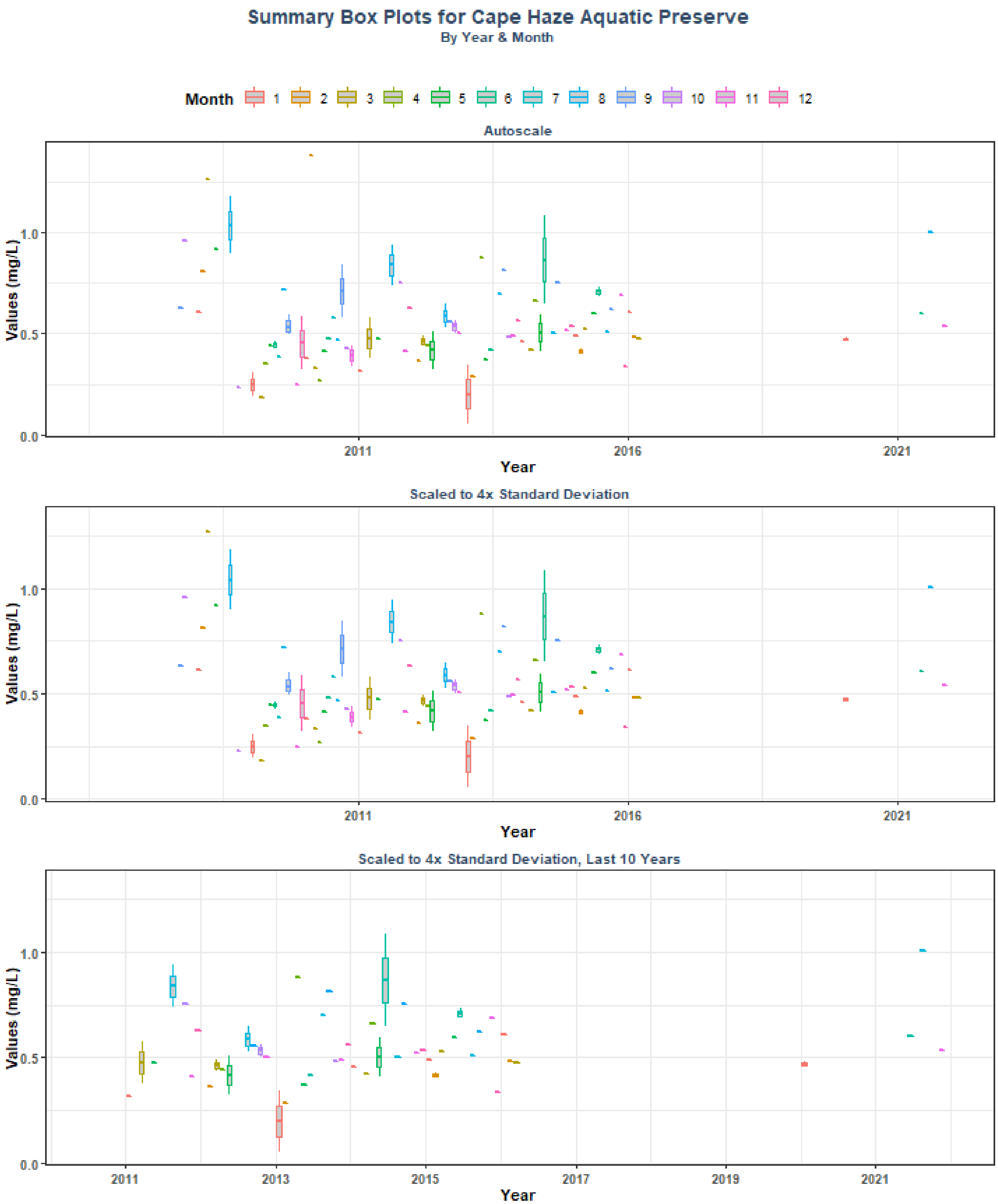
Summary Box Plots for Biscayne Bay Aquatic Preserve
By Year
Autoscale
Values (mg/L)
2.0
1.5
1.0
0.5
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (mg/L)
1.00
0.75
0.50
0.25
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
1.00
0.75
0.50
0.25
2011
2013
2015
2017
2019
2021
Year

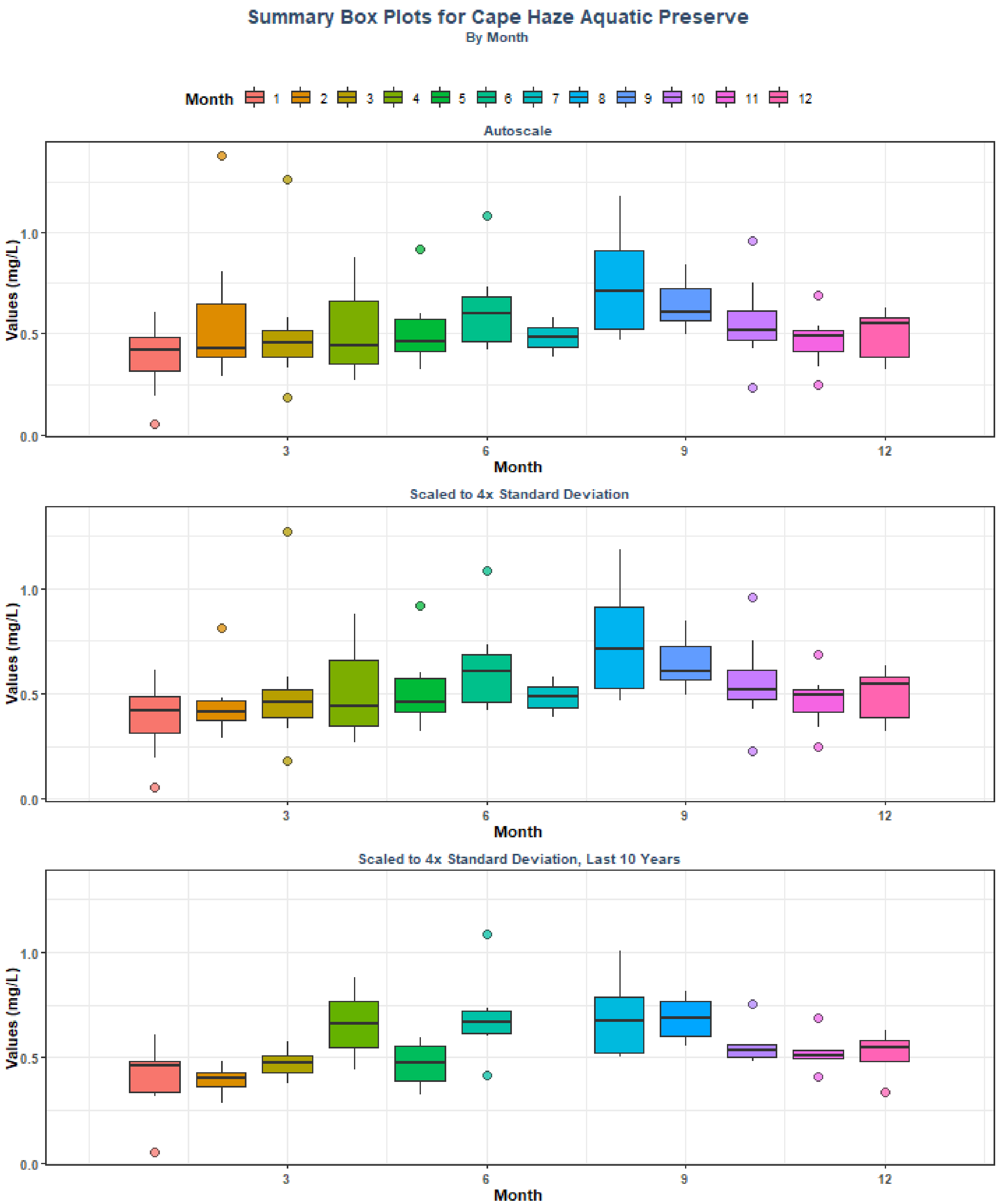

Summary Box Plots for Biscayne Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year



Summary Box Plots for Biscayne Bay Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
2.0
1.5
1.0
0.5
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
1.00
0.75
0.50
0.25
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
1.00
0.75
0.50
0.25
3
6
9
12
Month

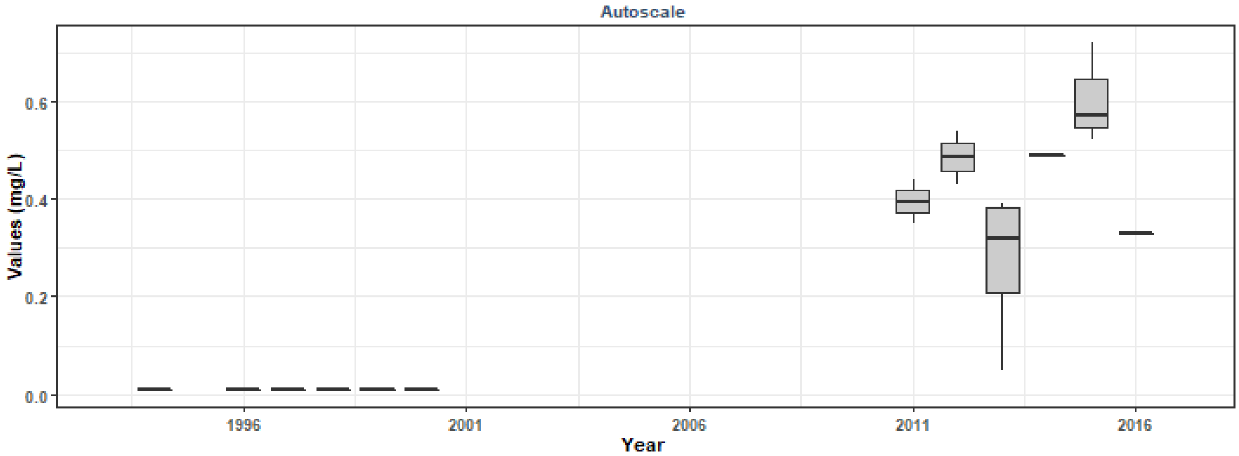

Summary Box Plots for Cape Haze Aquatic Preserve
By Year
Autoscale
Values (mg/L)
0.0
0.5
1.0
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (mg/L)
0.0
0.5
1.0
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.0
0.5
1.0
2011
2013
2015
2017
2019
2021
Year

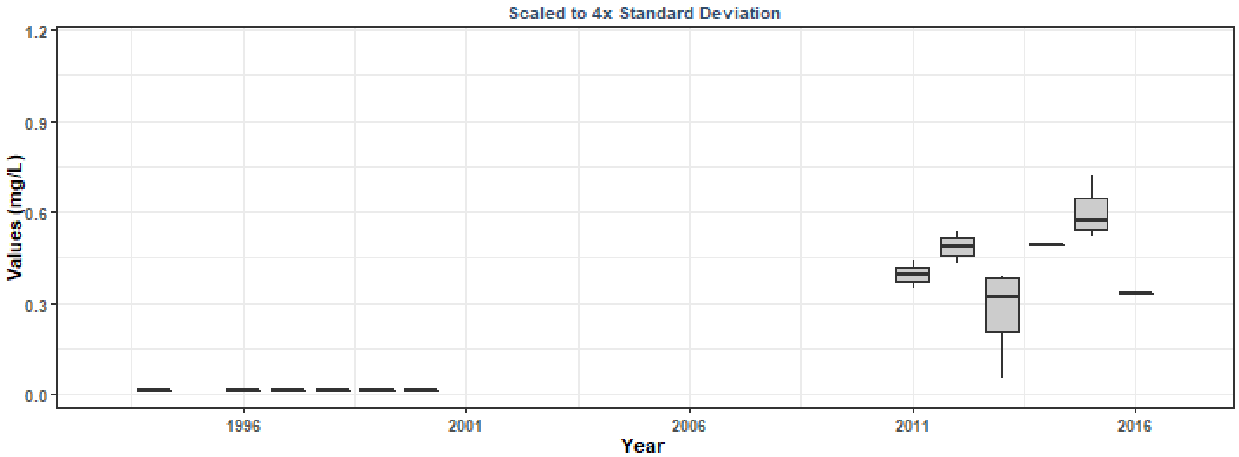

Summary Box Plots for Cape Haze Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Scaled to 4x Standard Deviation
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year
0.0
0.5
1.0
2011
2013
2015
2016
2017
2019
2021

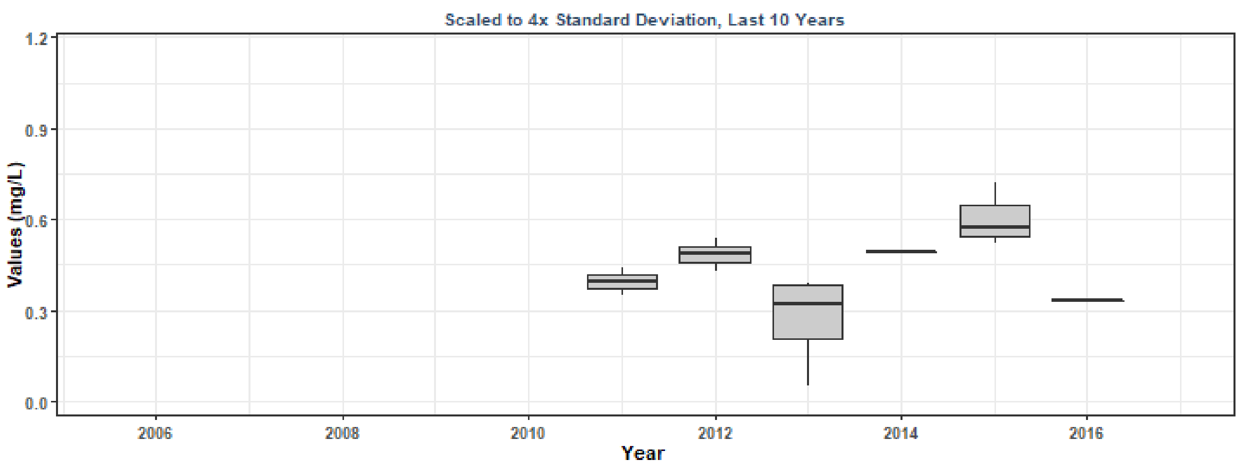

Summary Box Plots for Cape Haze Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
0.0
0.5
1.0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
0.0
0.5
1.0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.0
0.5
1.0
3
6
9
12
Month

Summary Box Plots for Estero Bay Aquatic Preserve

By Year

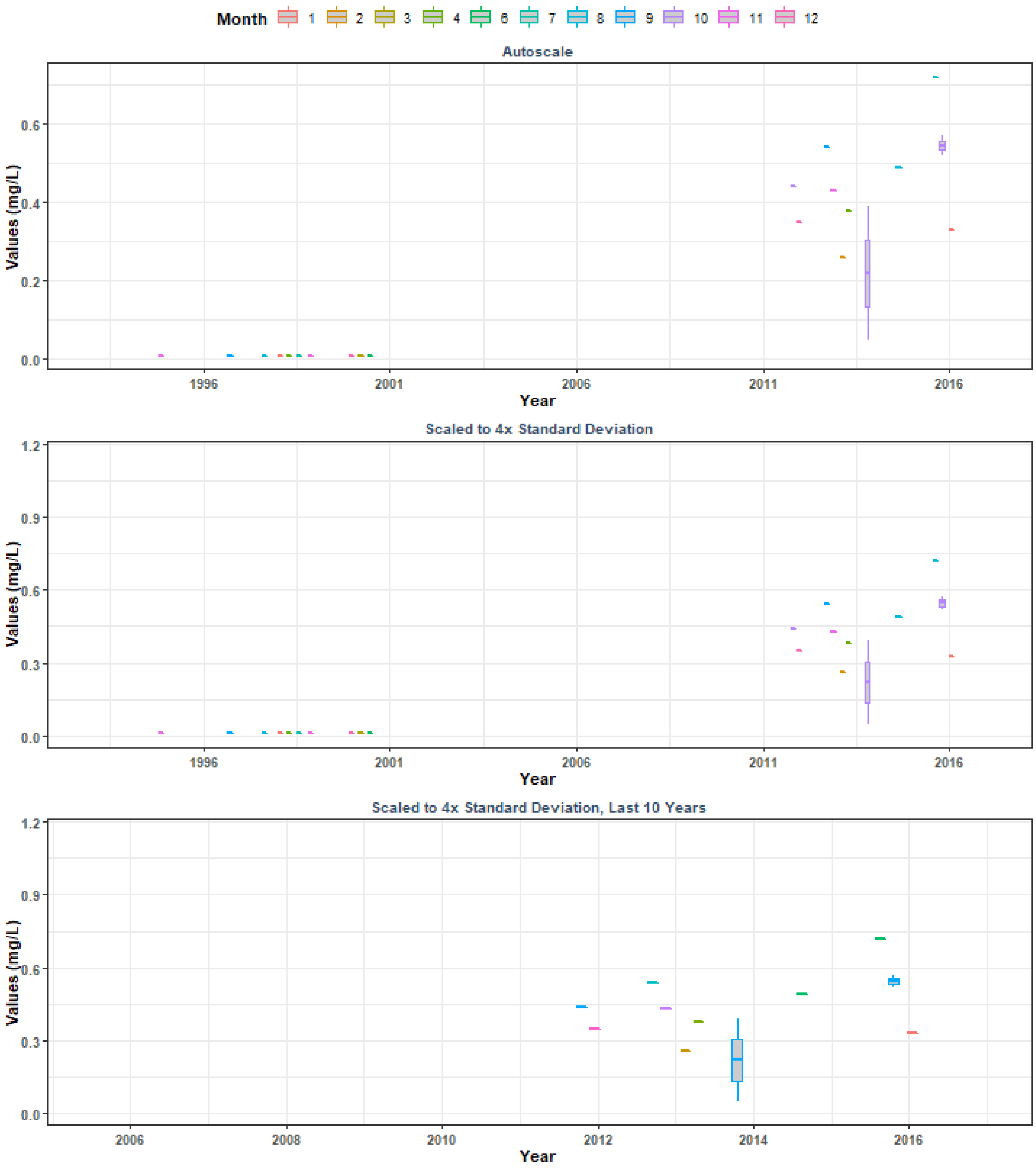


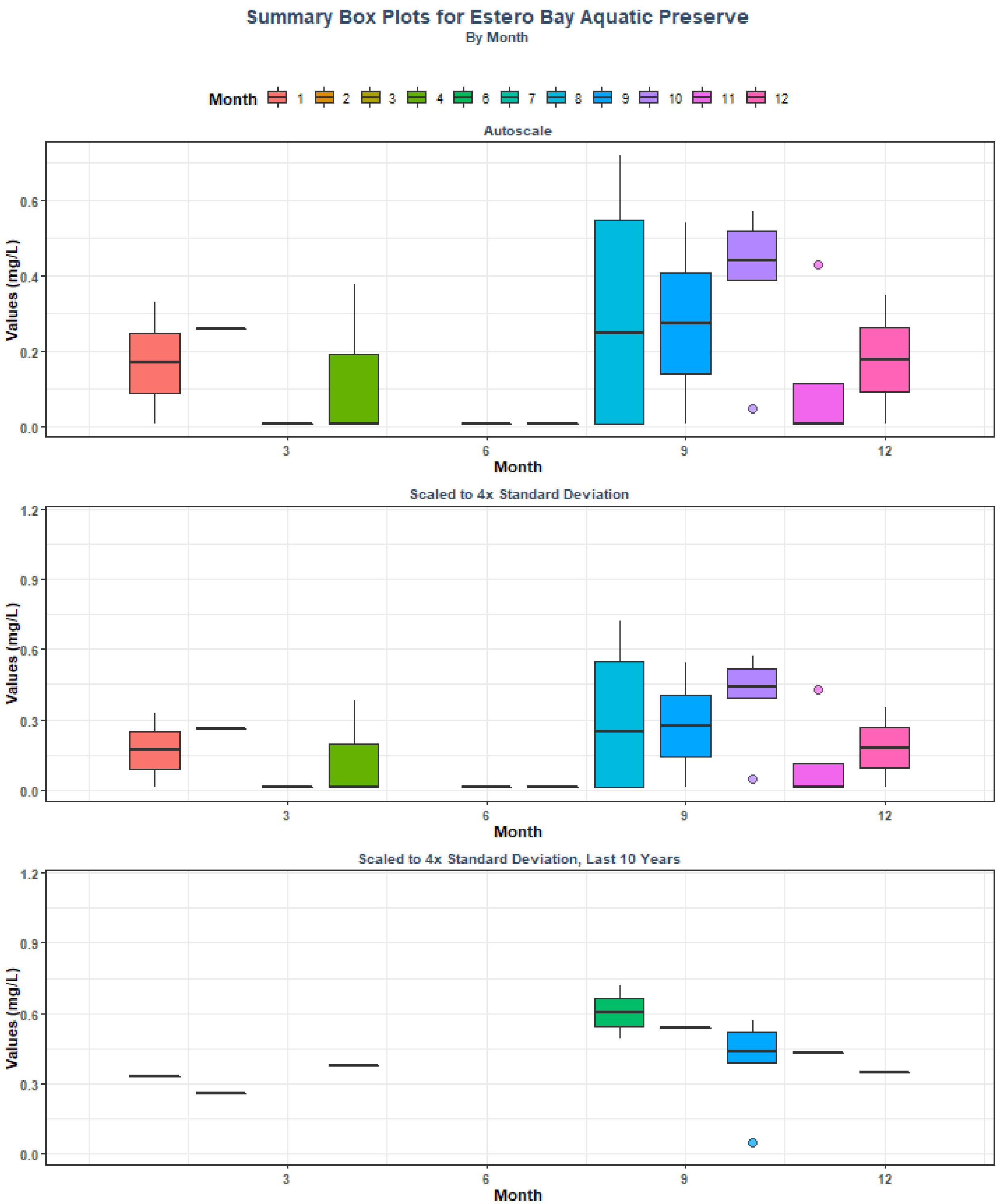




Summary Box Plots for Estero Bay Aquatic Preserve

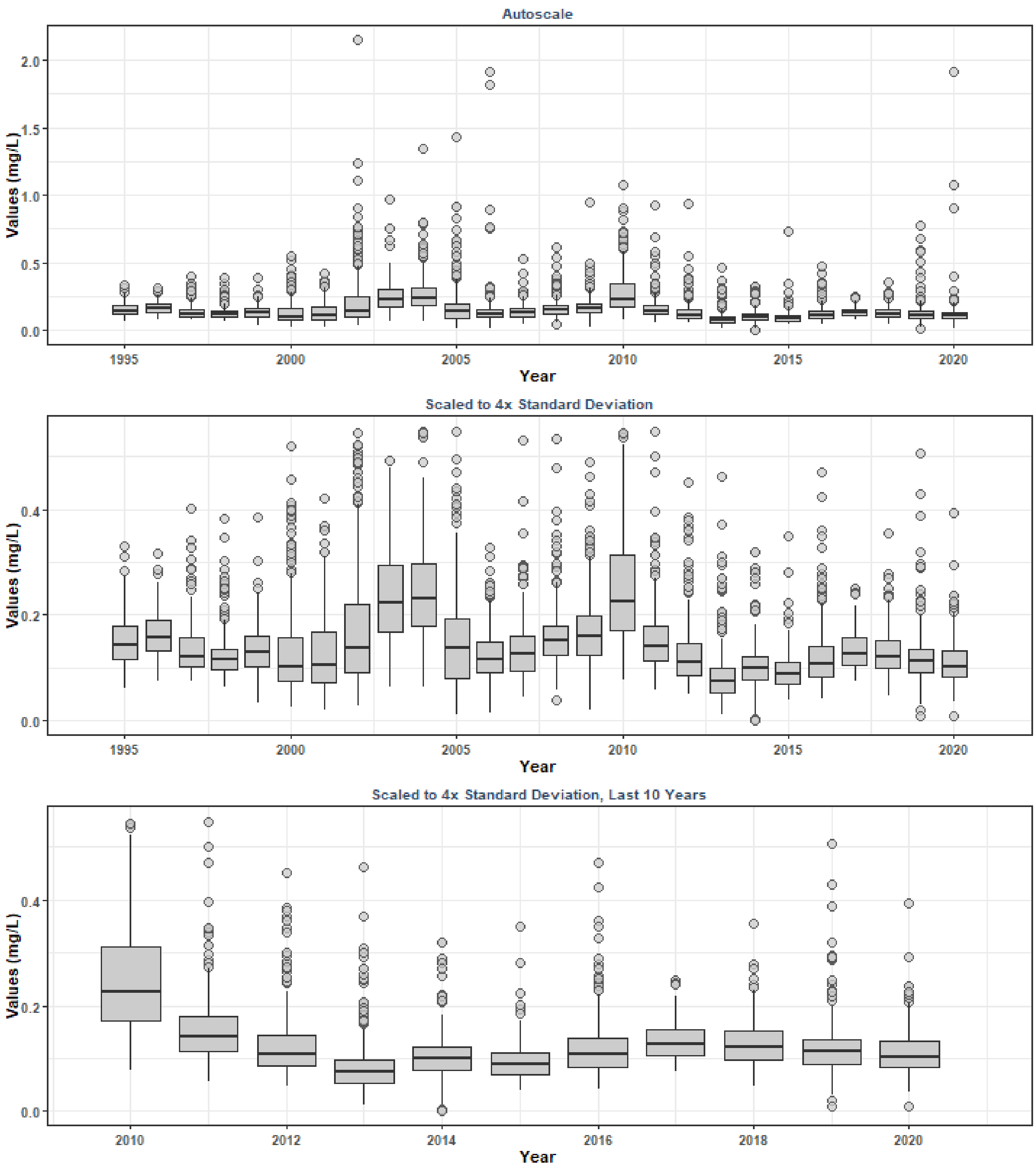
By Year & Month

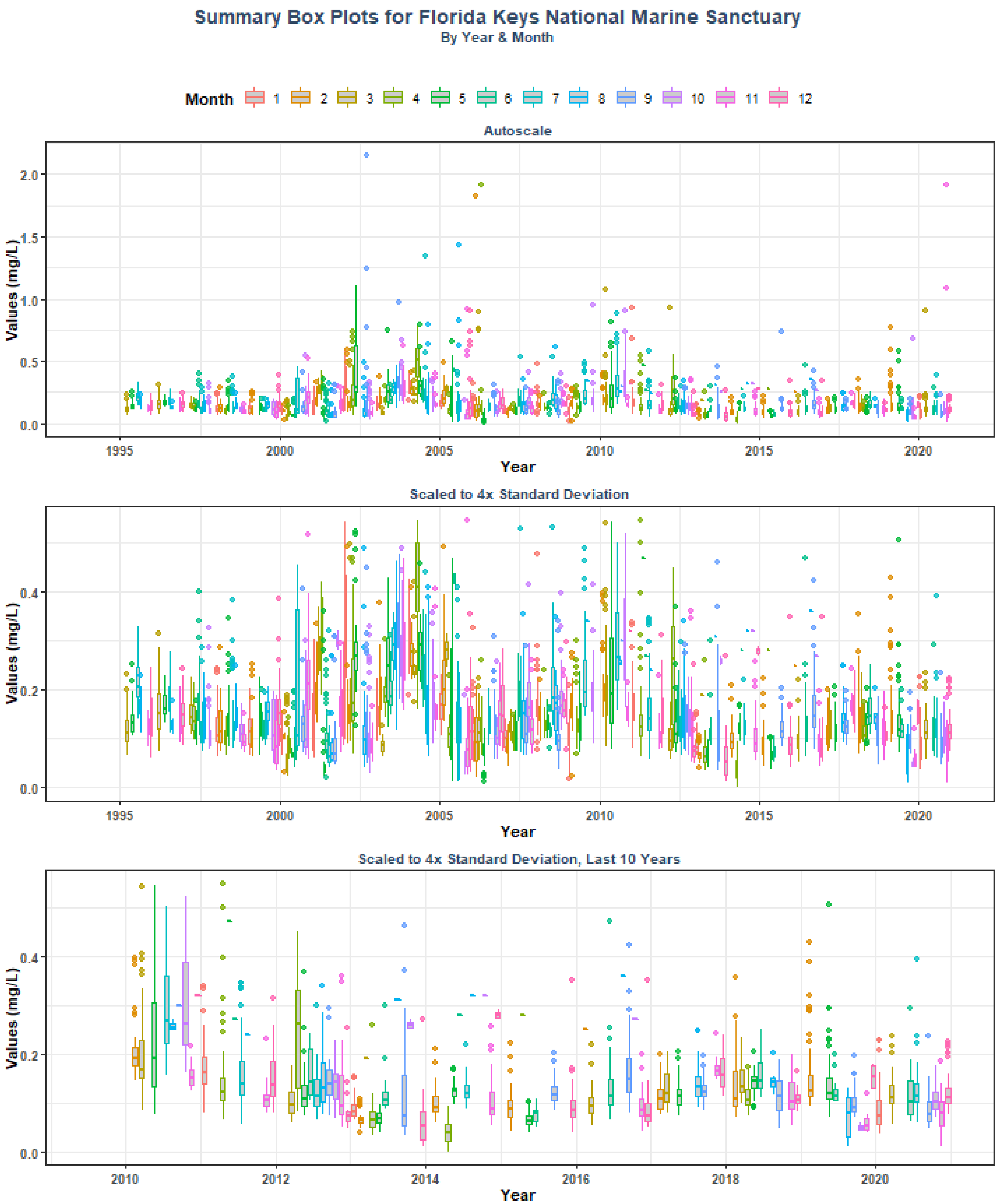


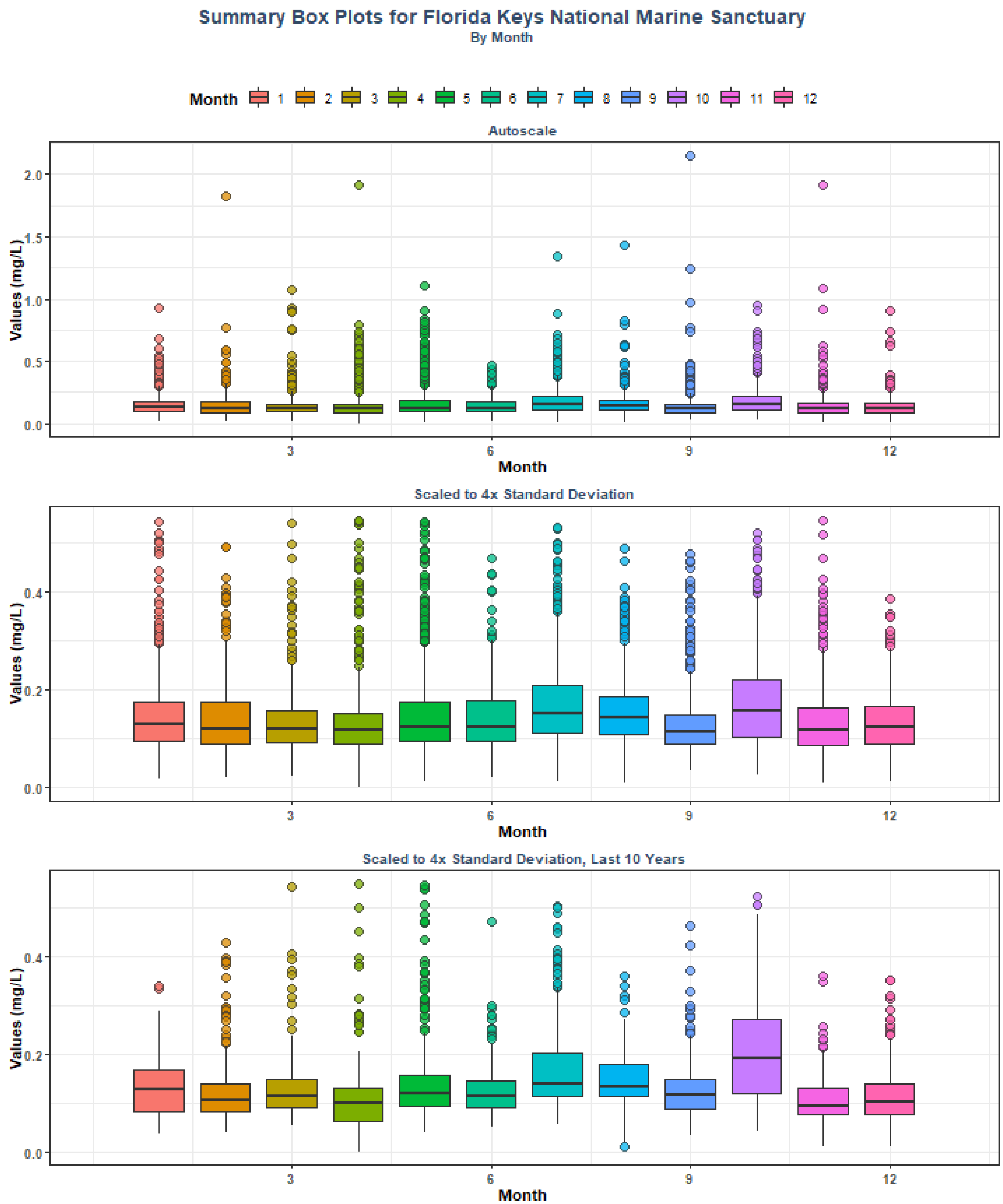

Summary Box Plots for Estero Bay Aquatic Preserve
By Month
Month 1 2 3 4 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
0.0
0.2
0.4
0.6
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
0.0
0.3
0.6
0.9
1.2
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.0
0.3
0.6
0.9
1.2
3
6
9
12
Month

Summary Box Plots for Florida Keys National Marine Sanctuary

By Year

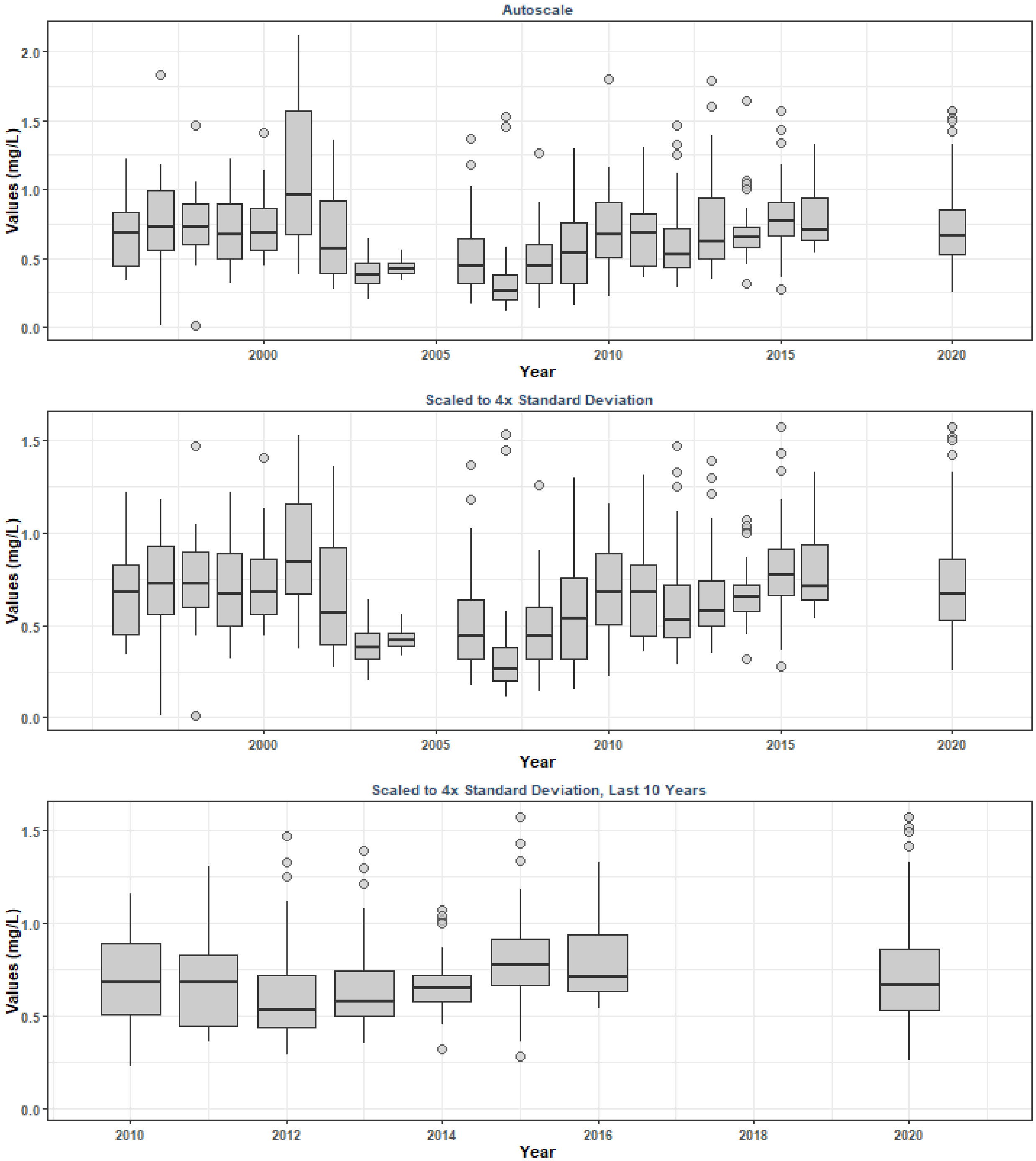


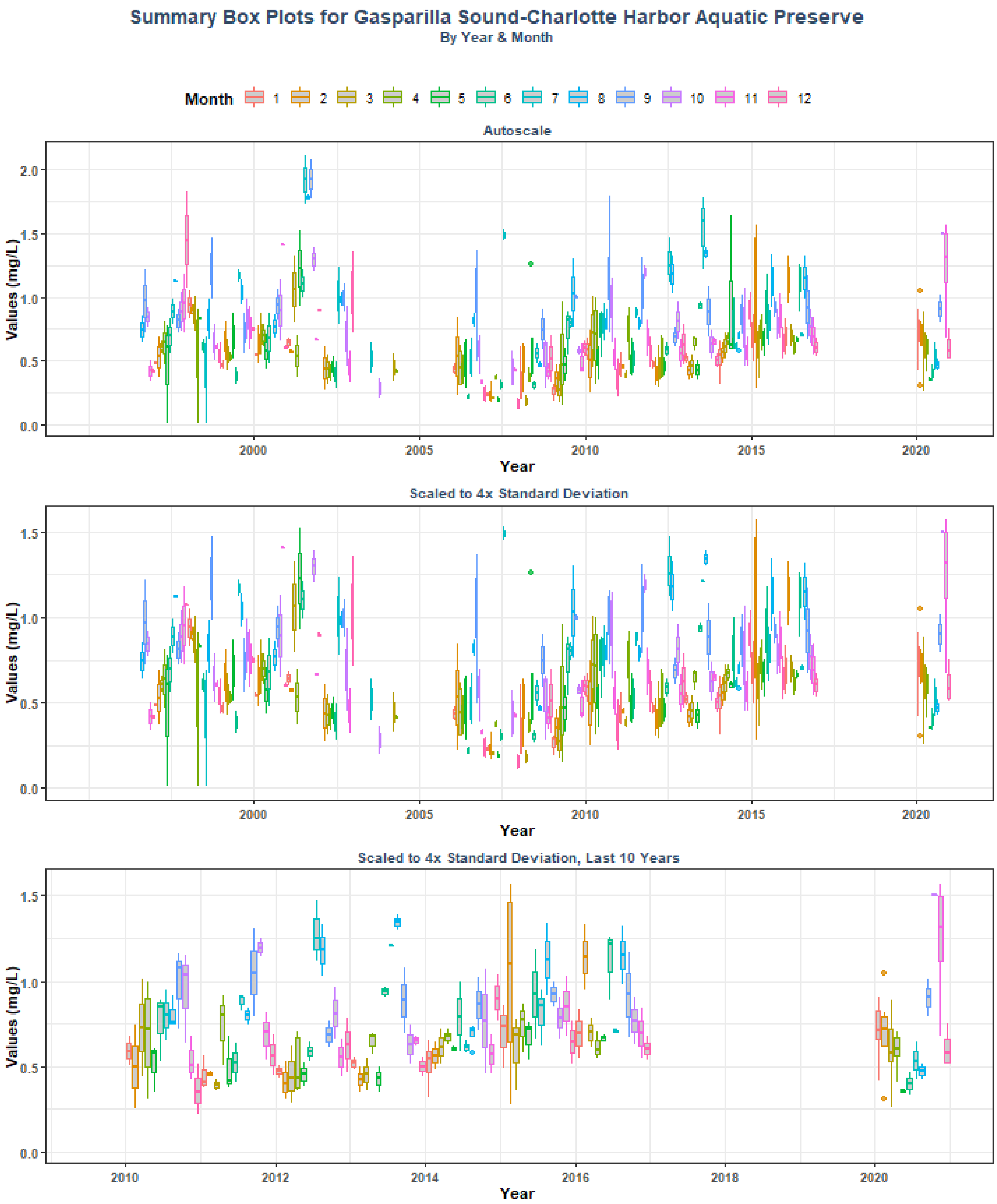

Summary Box Plots for Florida Keys National Marine Sanctuary
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year

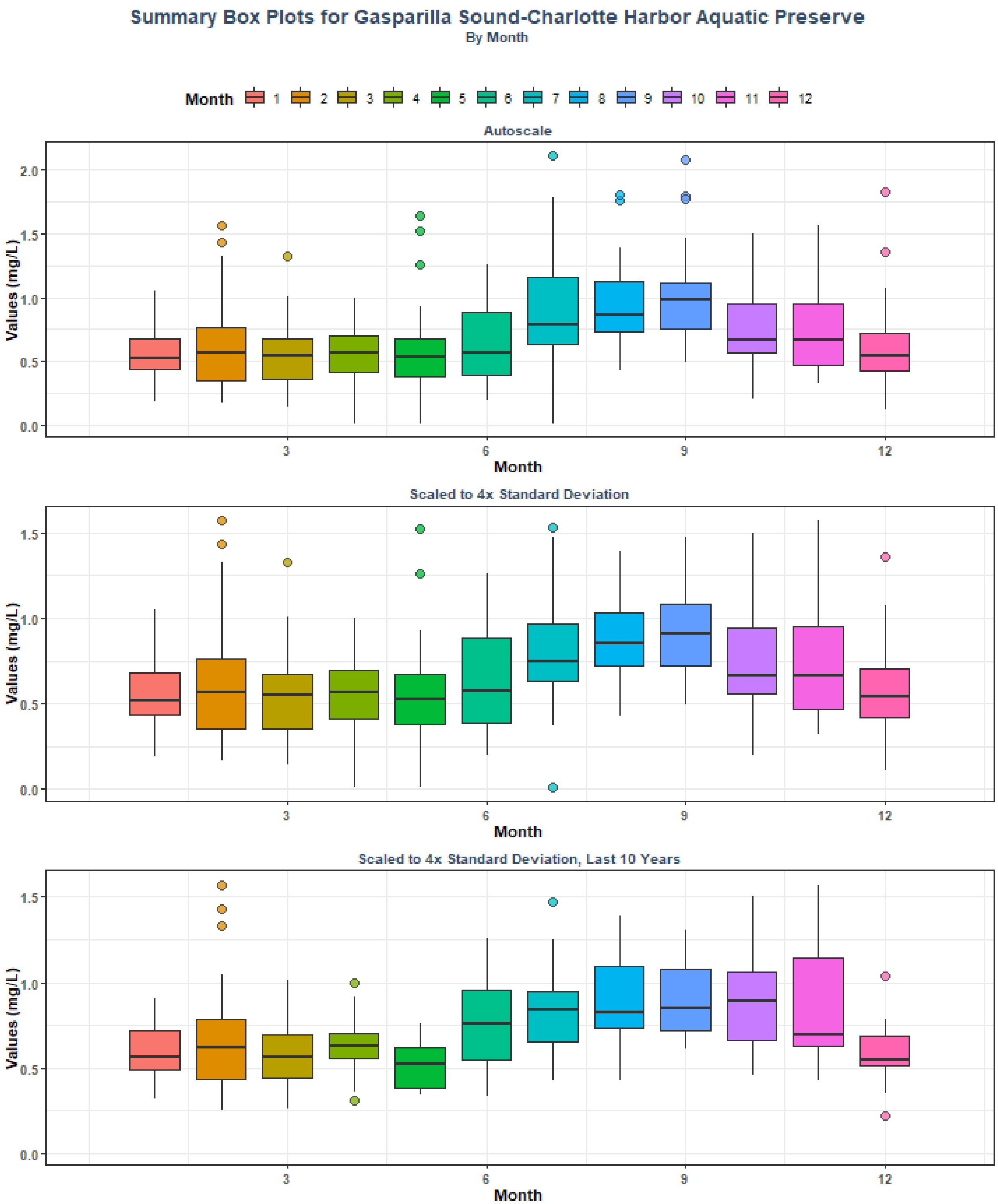

Summary Box Plots for Florida Keys National Marine Sanctuary
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
2.0
1.5
1.0
0.5
0.0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
0.4
0.2
0.0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.4
0.2
0.0
3
6
9
12
Month

Summary Box Plots for Gasparilla Sound-Charlotte Harbor Aquatic Preserve

By Year



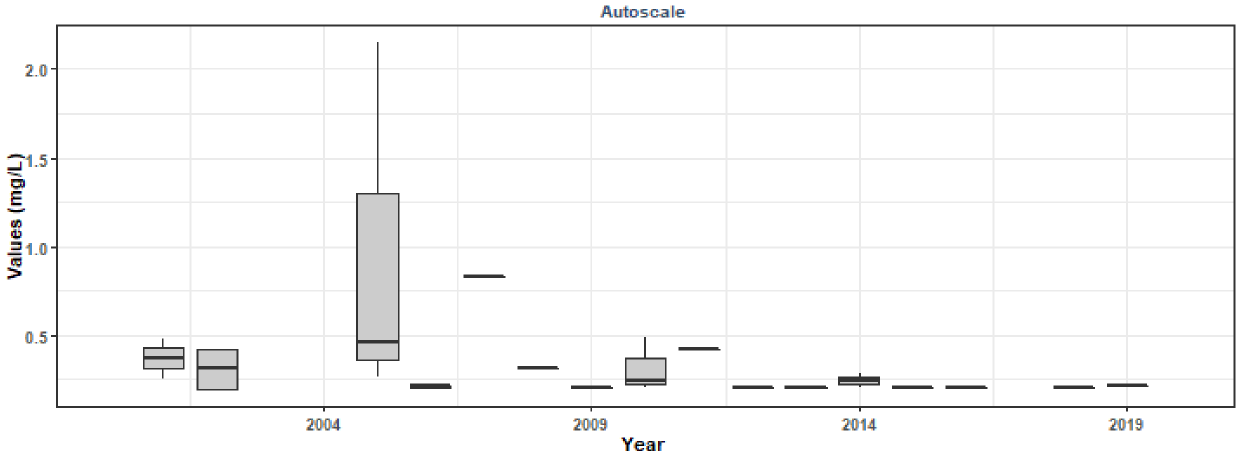

Summary Box Plots for Gasparilla Sound-Charlotte Harbor Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year

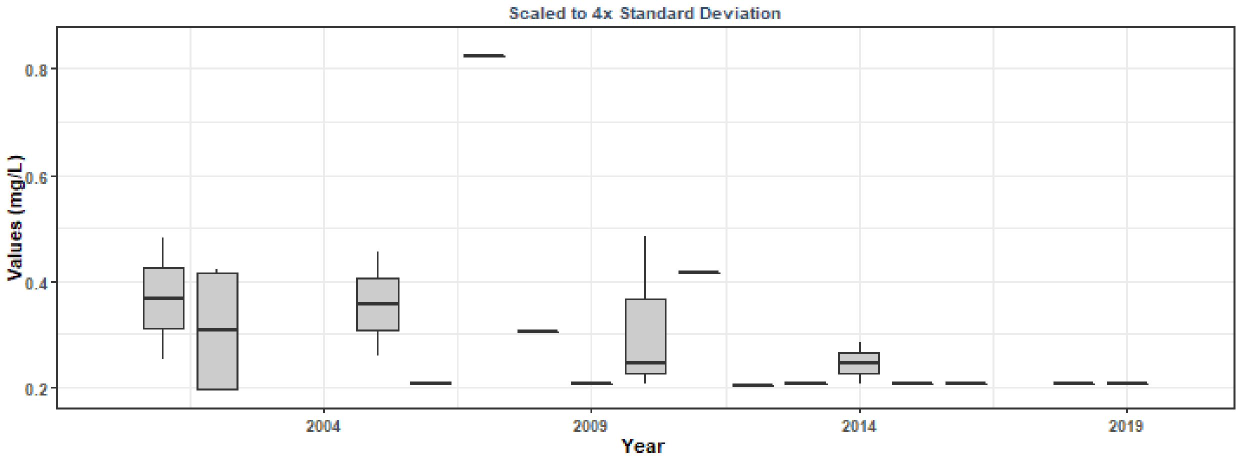


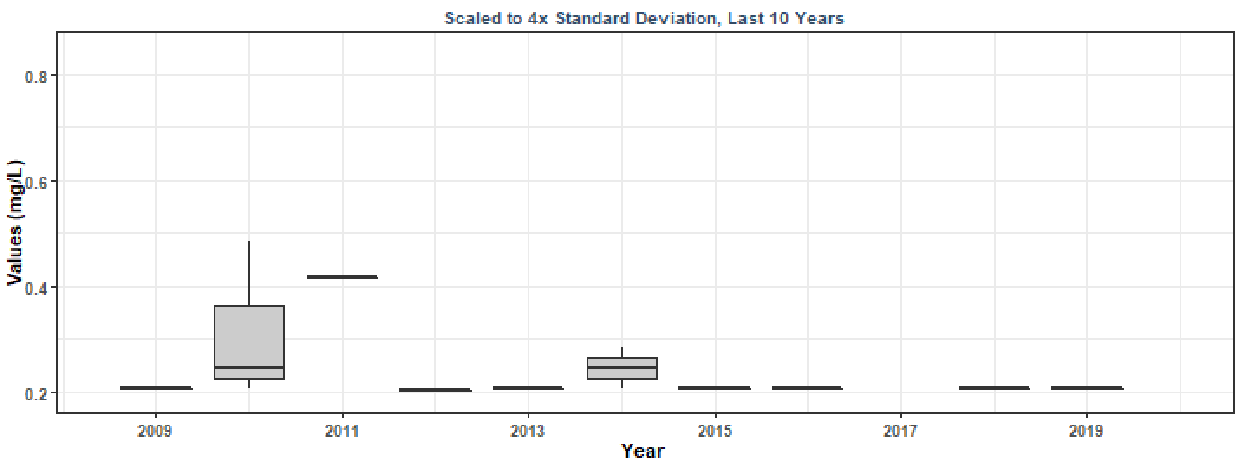
Summary Box Plots for Gasparilla Sound-Charlotte Harbor Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
2.0
1.5
1.0
0.5
0.0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
1.5
1.0
0.5
0.0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
1.5
1.0
0.5
0.0
3
6
9
12
Month

Summary Box Plots for Jensen Beach to Jupiter Inlet Aquatic Preserve

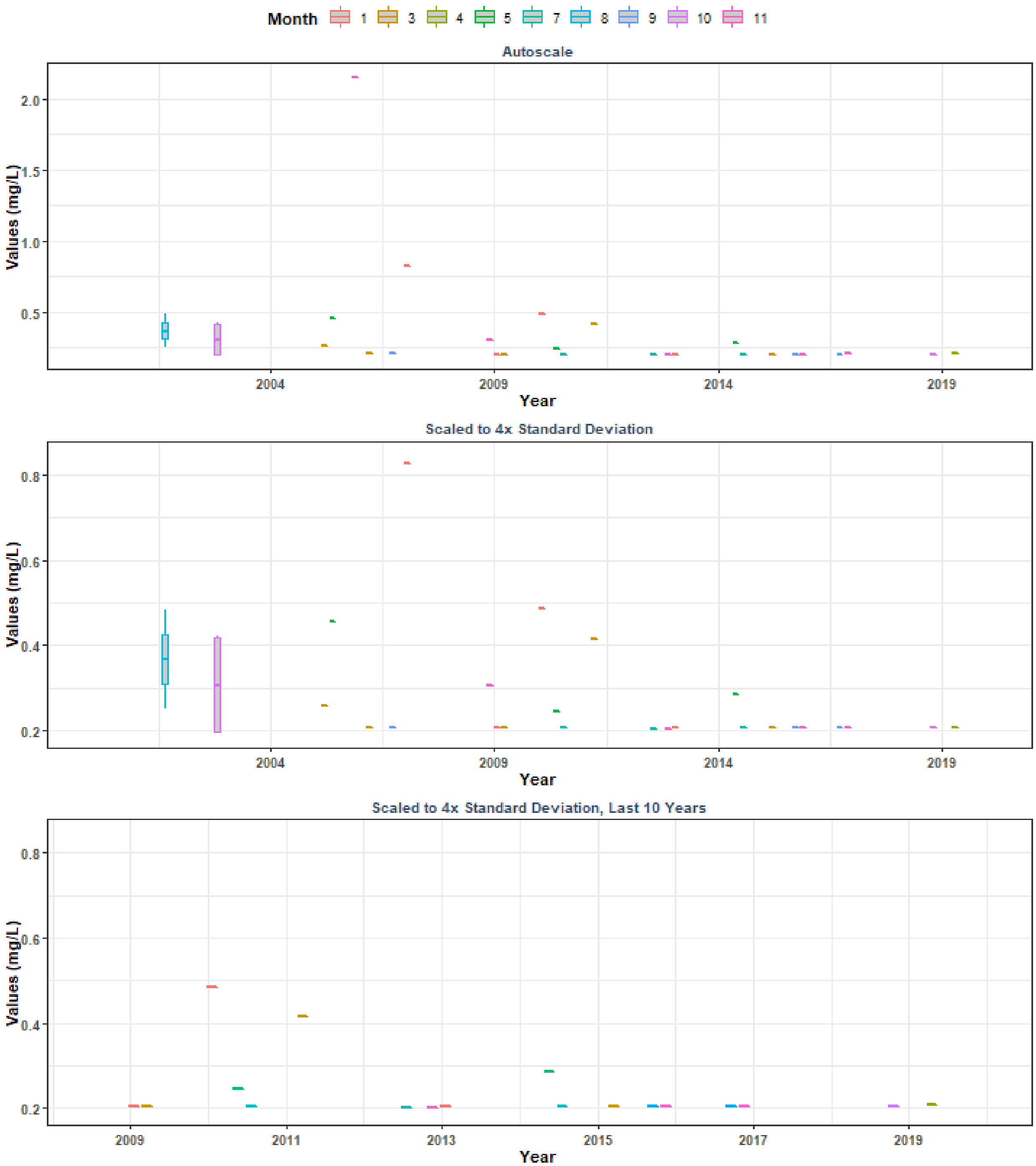
By Year





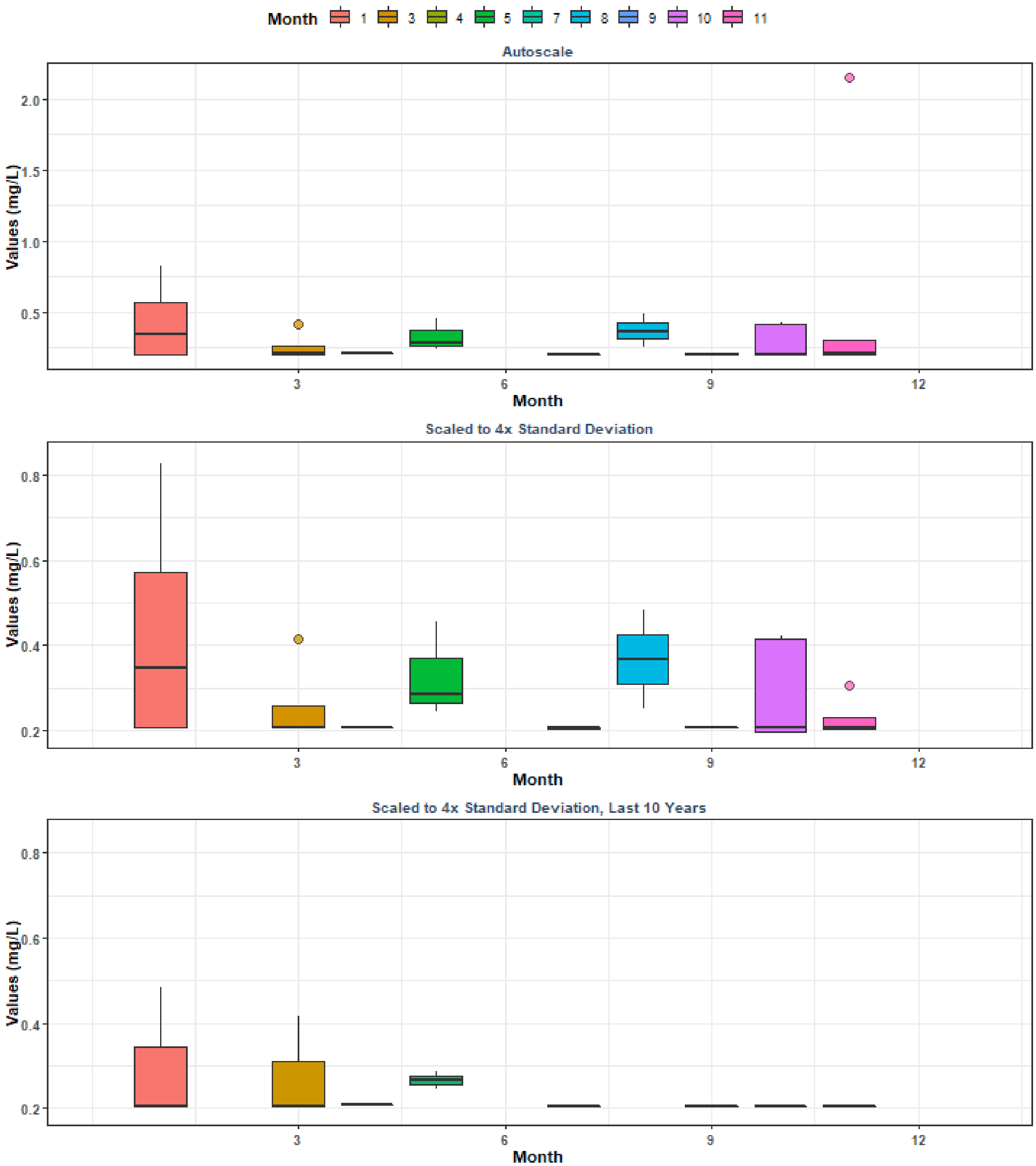


Summary Box Plots for Jensen Beach to Jupiter Inlet Aquatic Preserve
By Year & Month



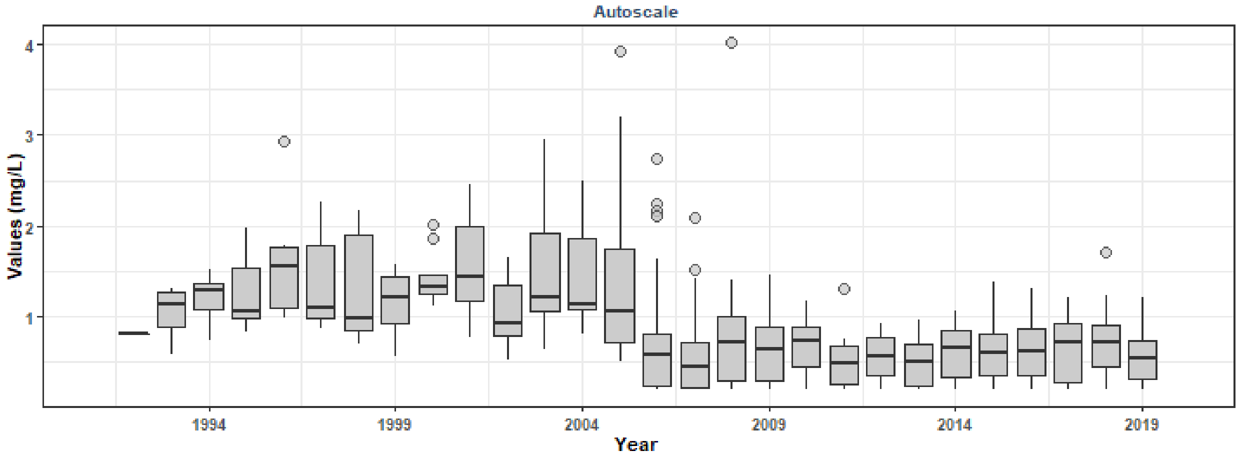
Summary Box Plots for Jensen Beach to Jupiter Inlet Aquatic Preserve

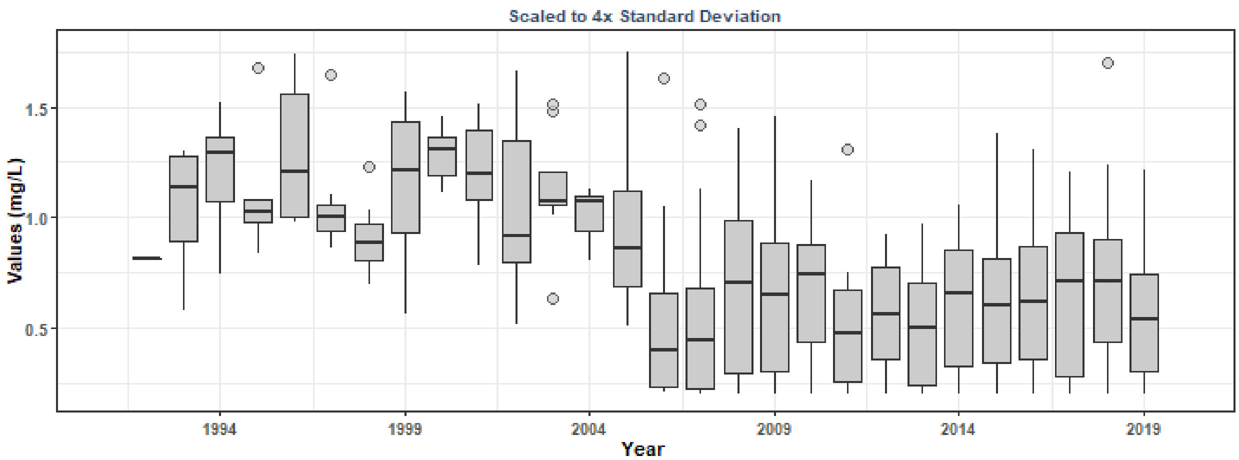
By Month

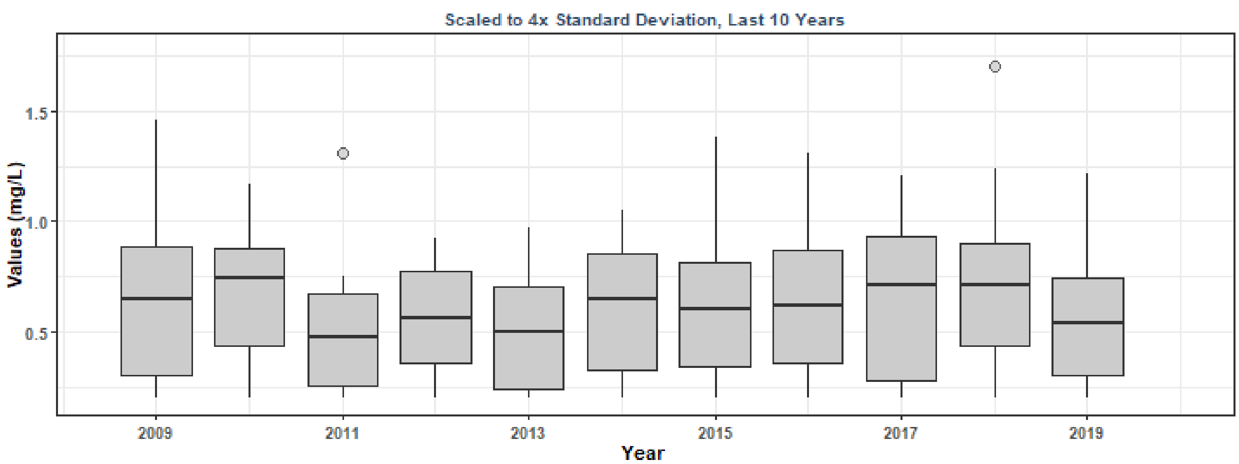


Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve

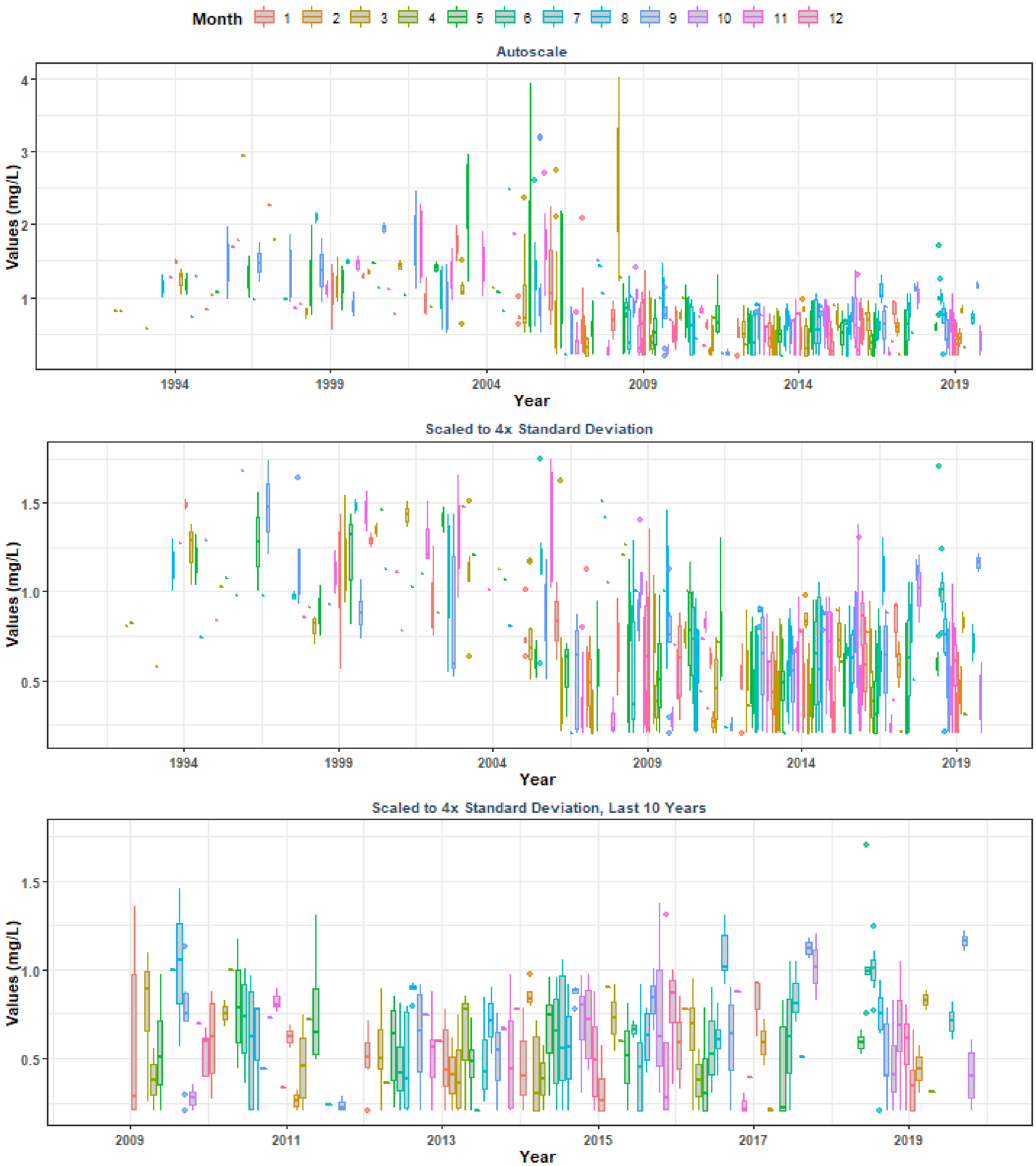
By Year

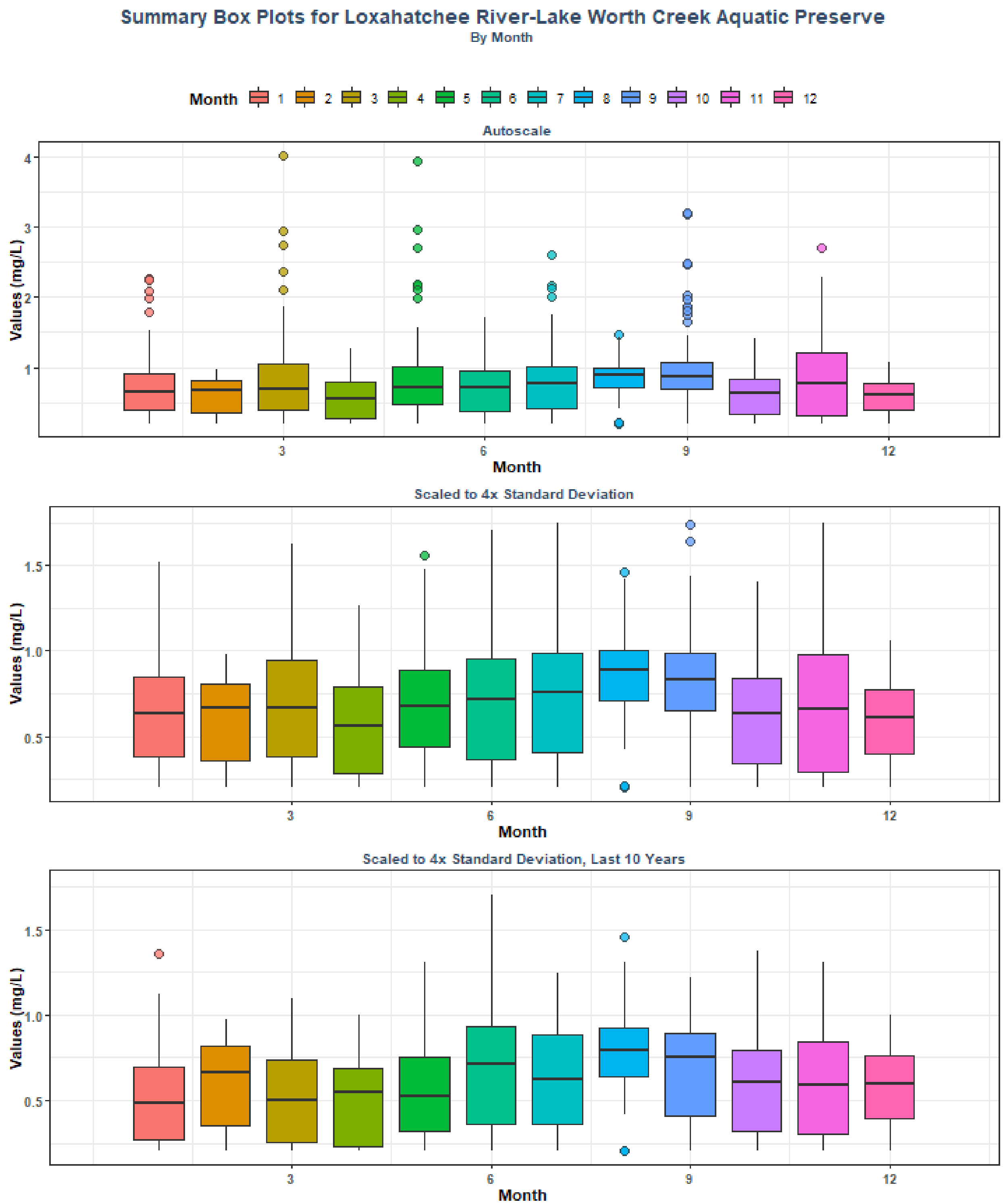






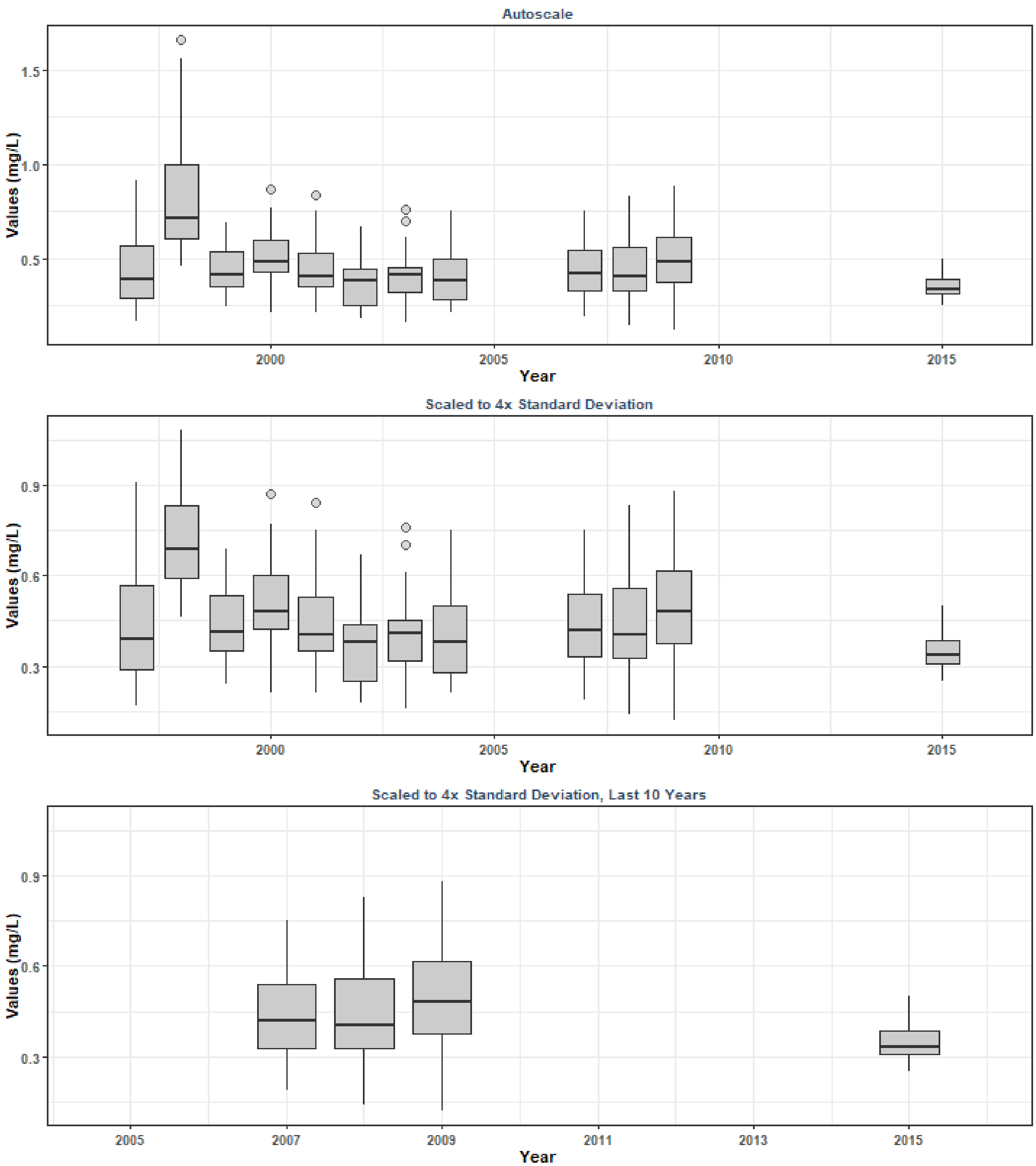
Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve
By Year & Month

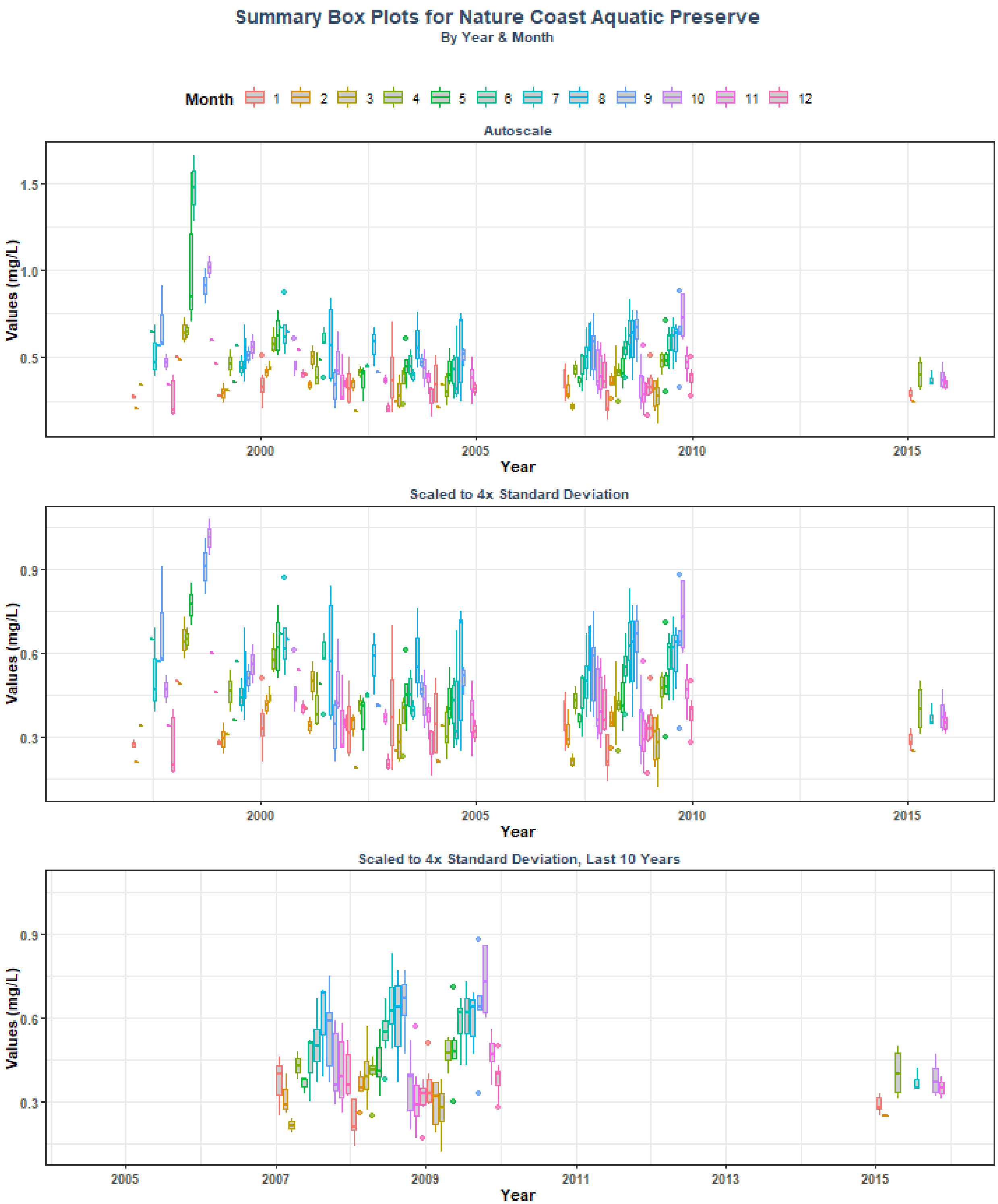


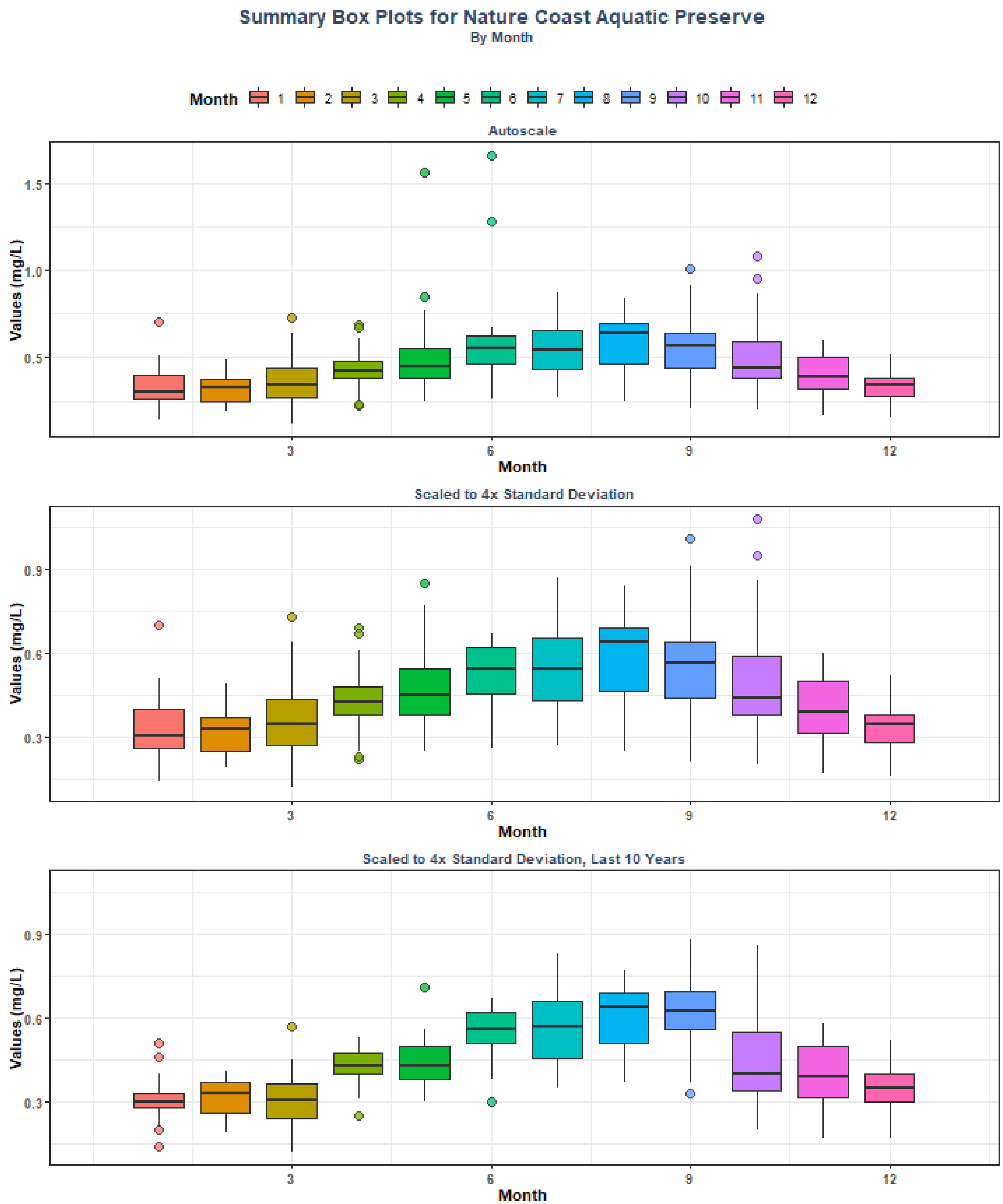


Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Month
Scaled to 4x Standard Deviation
Values (mg/L)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Month

Summary Box Plots for Nature Coast Aquatic Preserve
By Year

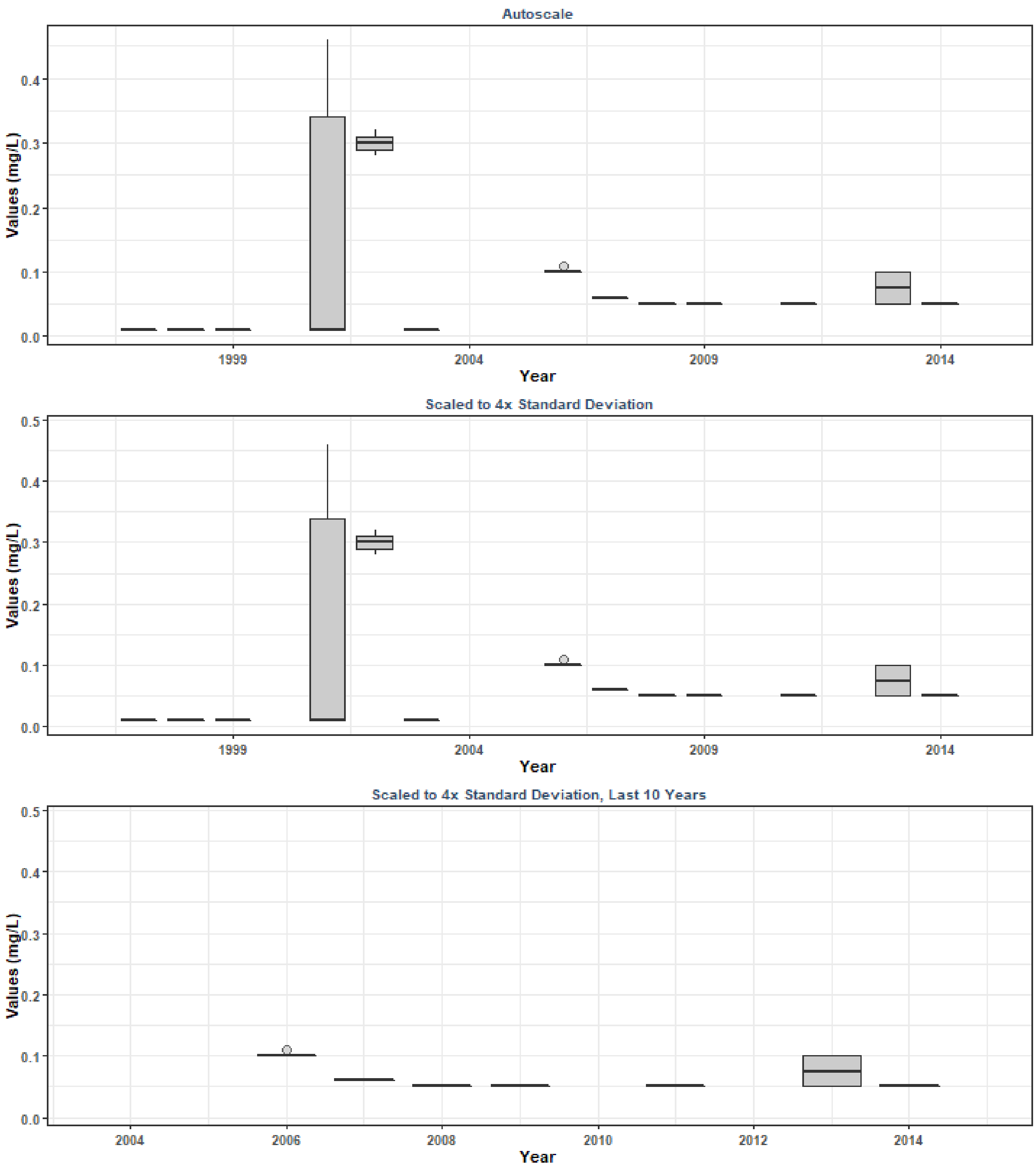


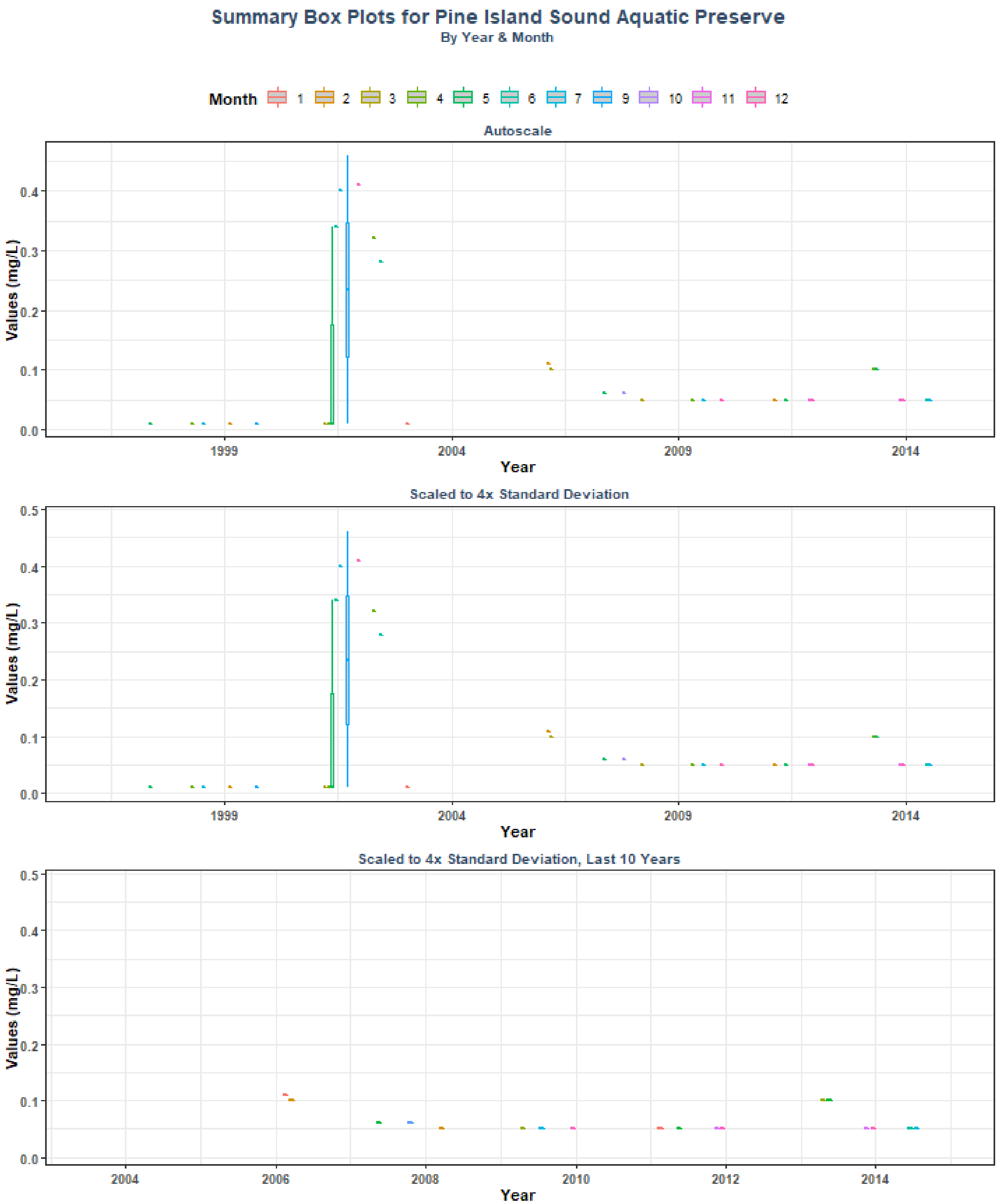

Summary Box Plots for Nature Coast Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
1.5
1.0
0.5
2000
2005
2010
2015
Year
Scaled to 4x Standard Deviation
Values (mg/L)
0.9
0.6
0.3
2000
2005
2010
2015
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.9
0.6
0.3
2005
2007
2009
2011
2013
2015
Year


Summary Box Plots for Nature Coast Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
1.5
1.0
0.5
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
0.9
0.6
0.3
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.9
0.6
0.3
3
6
9
12
Month

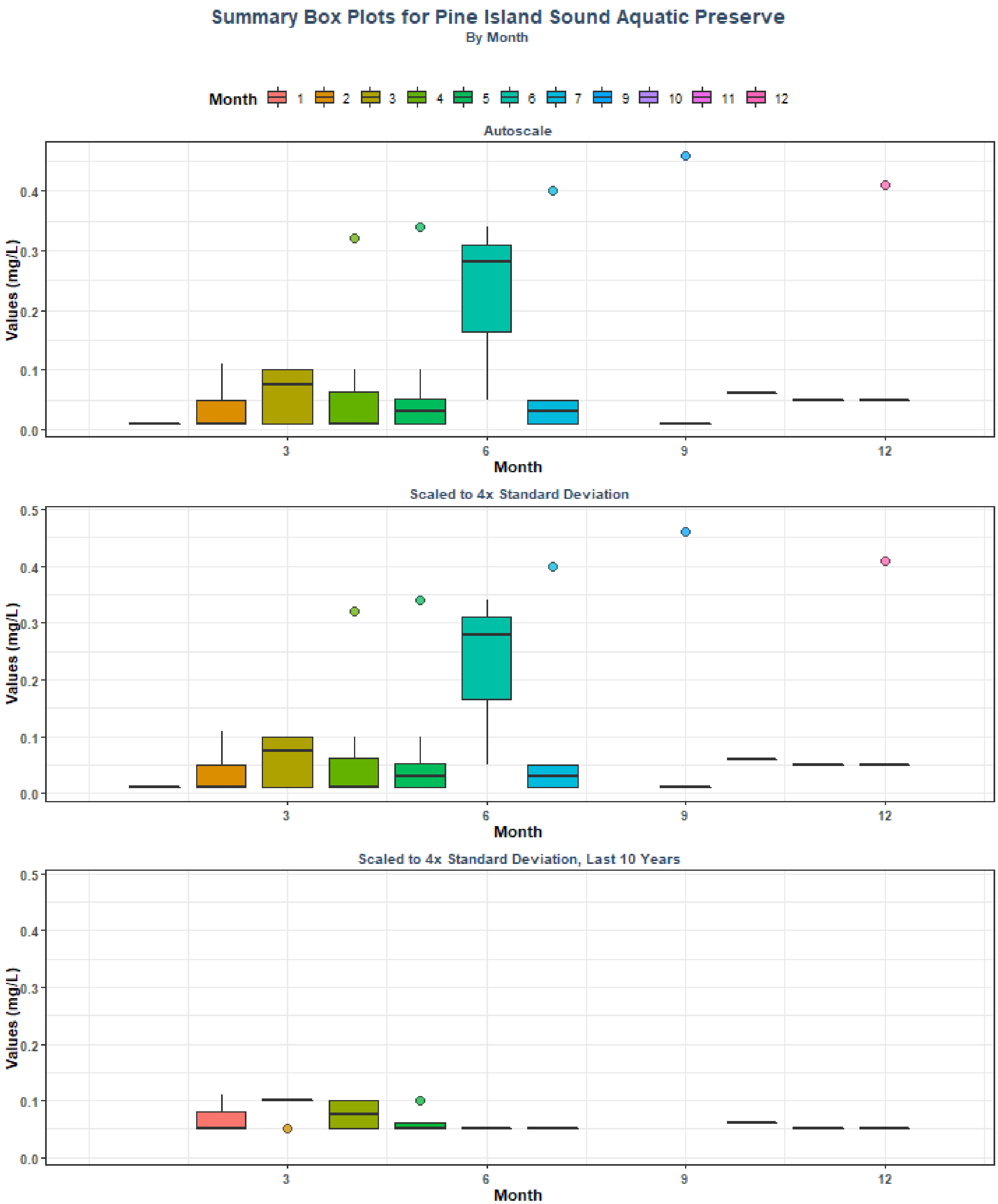
Summary Box Plots for Pine Island Sound Aquatic Preserve

By Year





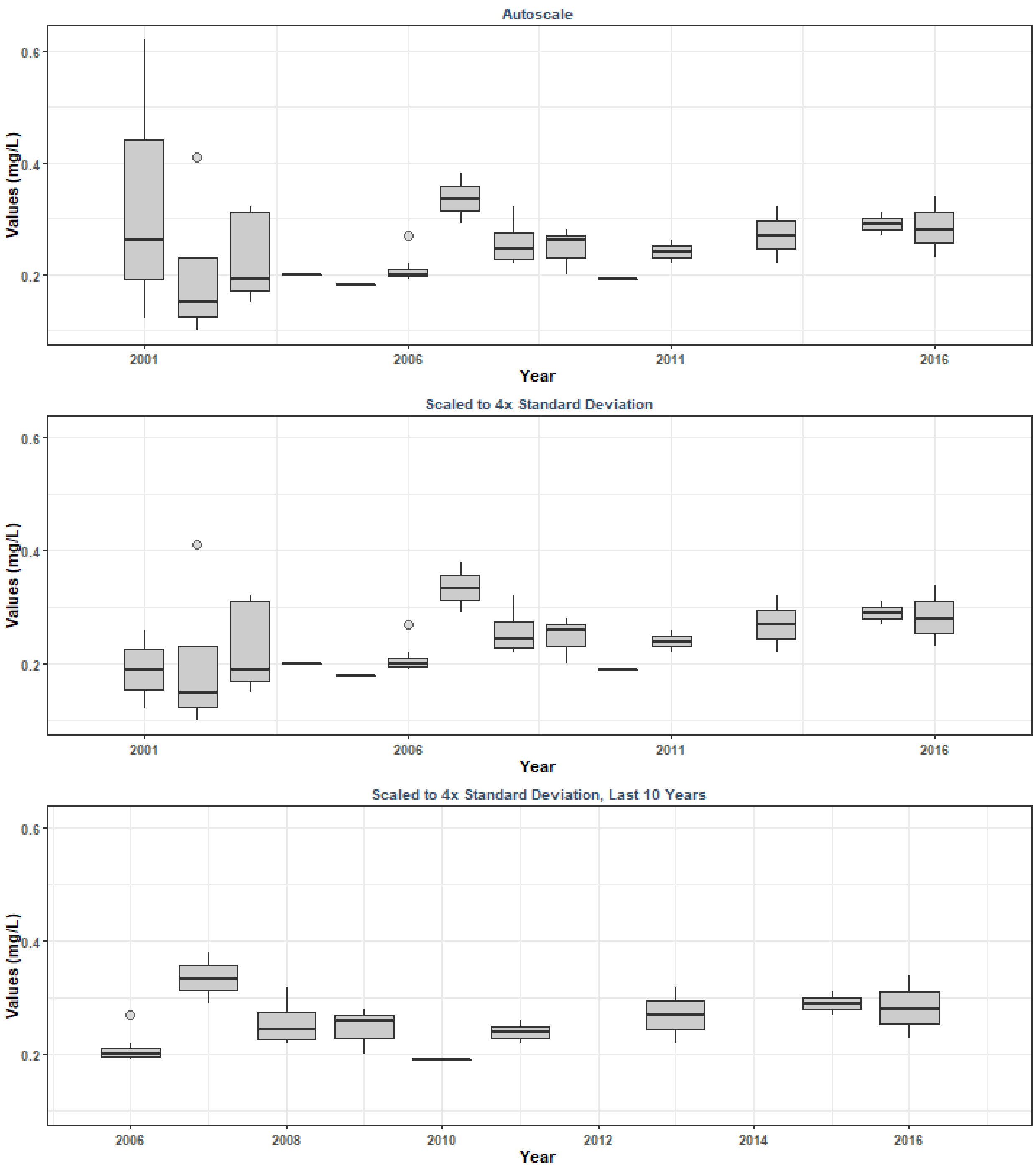
Summary Box Plots for Pine Island Sound Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 9 10 11 12
Autoscale
Values (mg/L)
0.0
0.1
0.2
0.3
0.4
1999
2004
2009
2014
Year
Scaled to 4x Standard Deviation
0.5
Scaled to 4x Standard Deviation, Last 10 Years
2006
2008
2010
2012

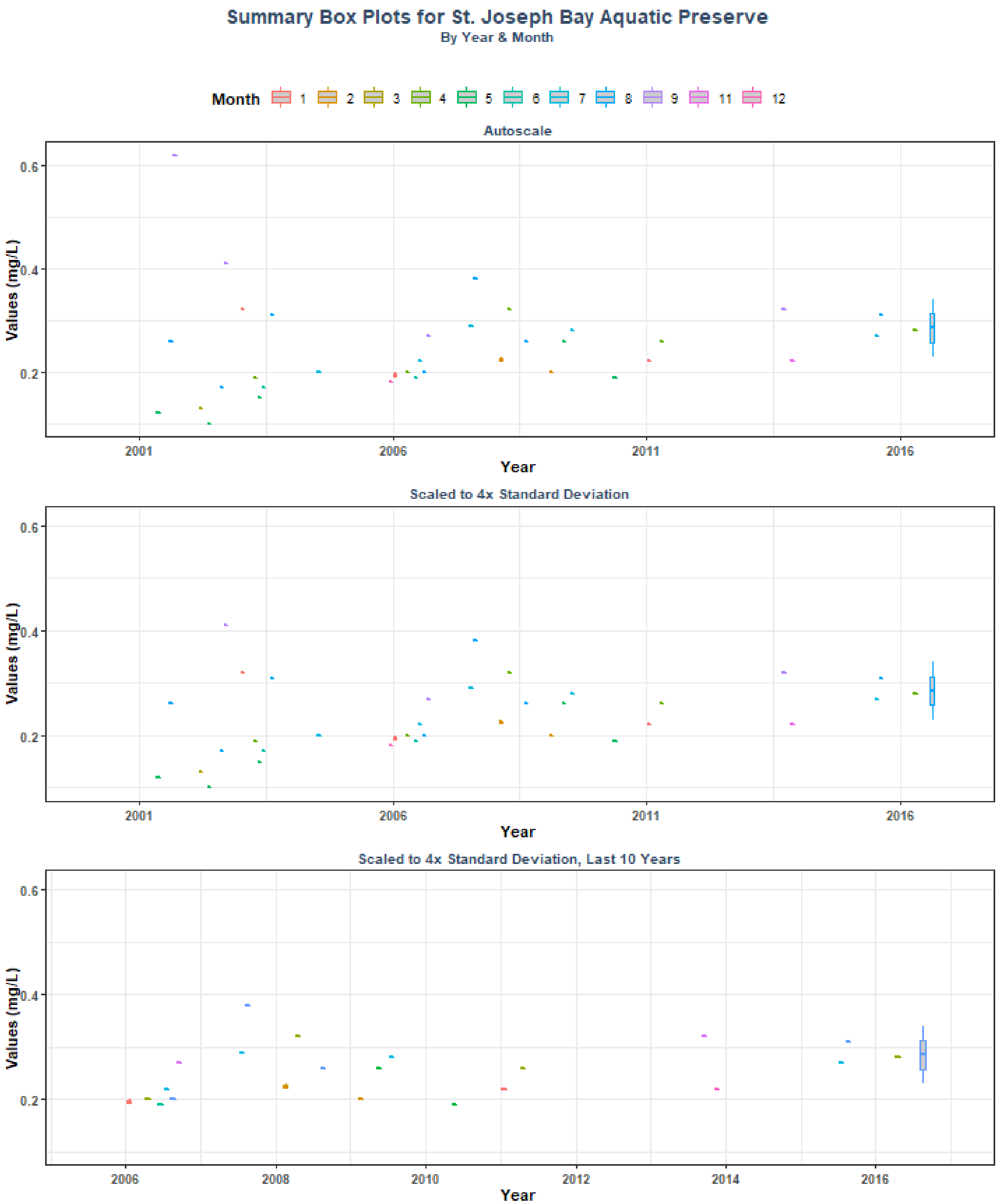


Summary Box Plots for Pine Island Sound Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 9 10 11 12
Autoscale
Values (mg/L)
0.0
0.1
0.2
0.3
0.4
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
0.0
0.1
0.2
0.3
0.4
0.5
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.0
0.1
0.2
0.3
0.4
0.5
3
6
9
12
Month

Summary Box Plots for St. Joseph Bay Aquatic Preserve

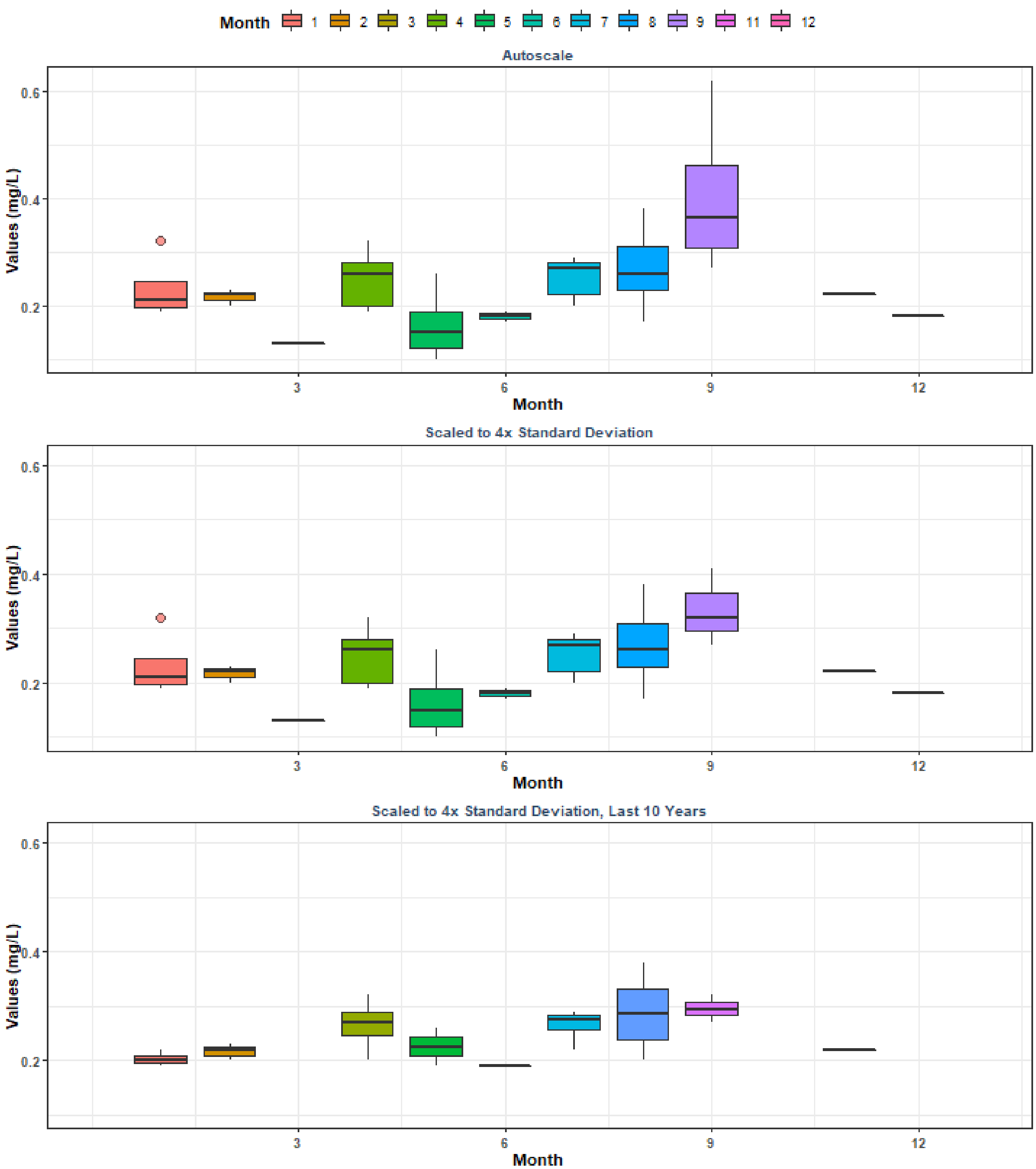
By Year




Summary Box Plots for St. Joseph Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 11 12
Autoscale
Scaled to 4x Standard Deviation
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0.2
0.4
0.6
2001
2006
2011
2016
2008
2010
2012
2014
Year

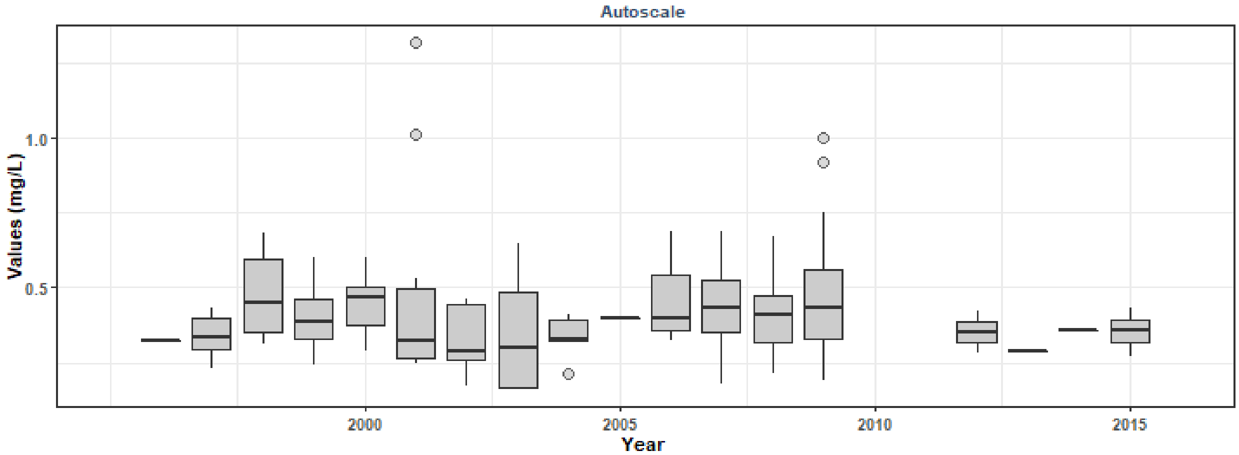
Summary Box Plots for St. Joseph Bay Aquatic Preserve

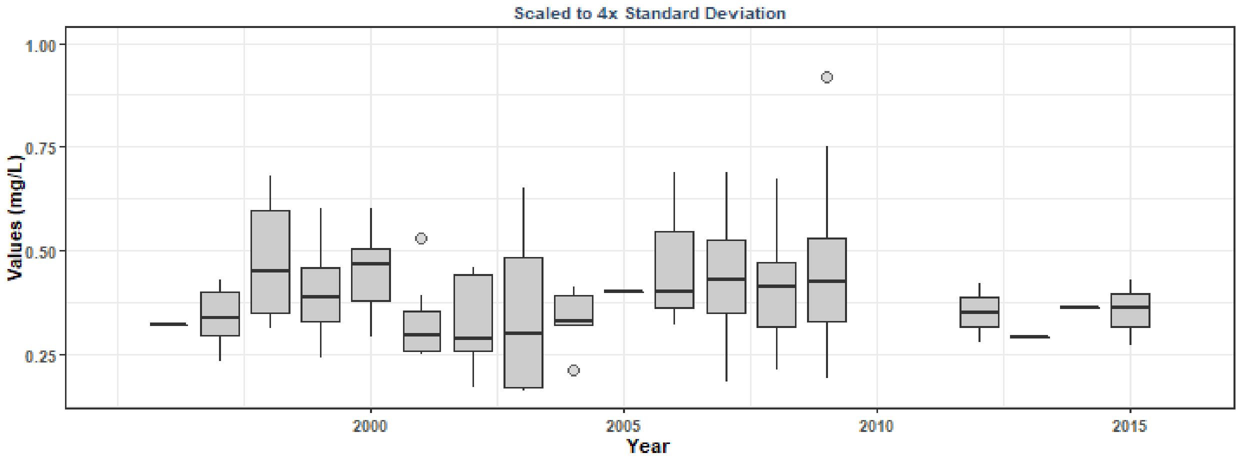
By Month

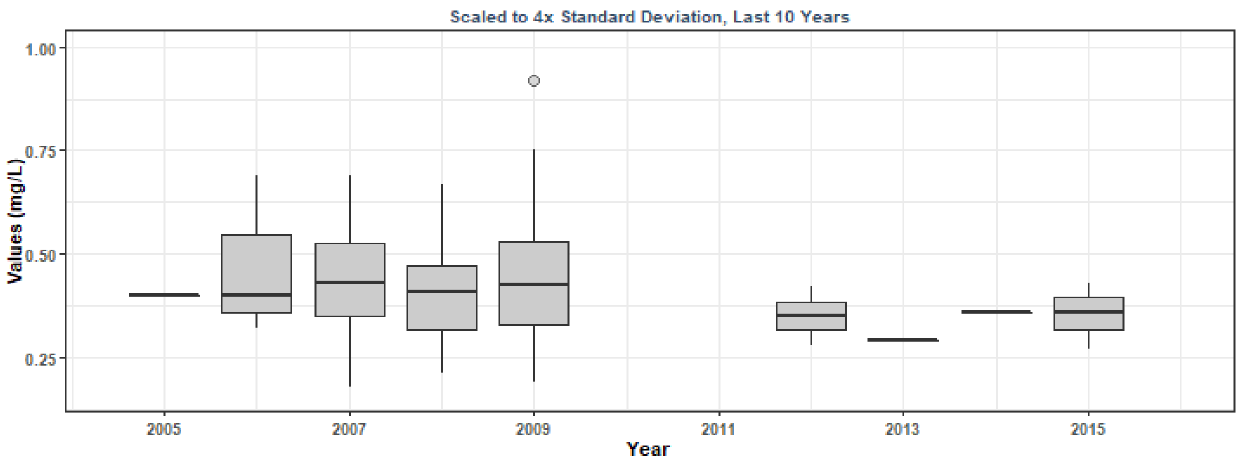


Summary Box Plots for St. Martins Marsh Aquatic Preserve

By Year

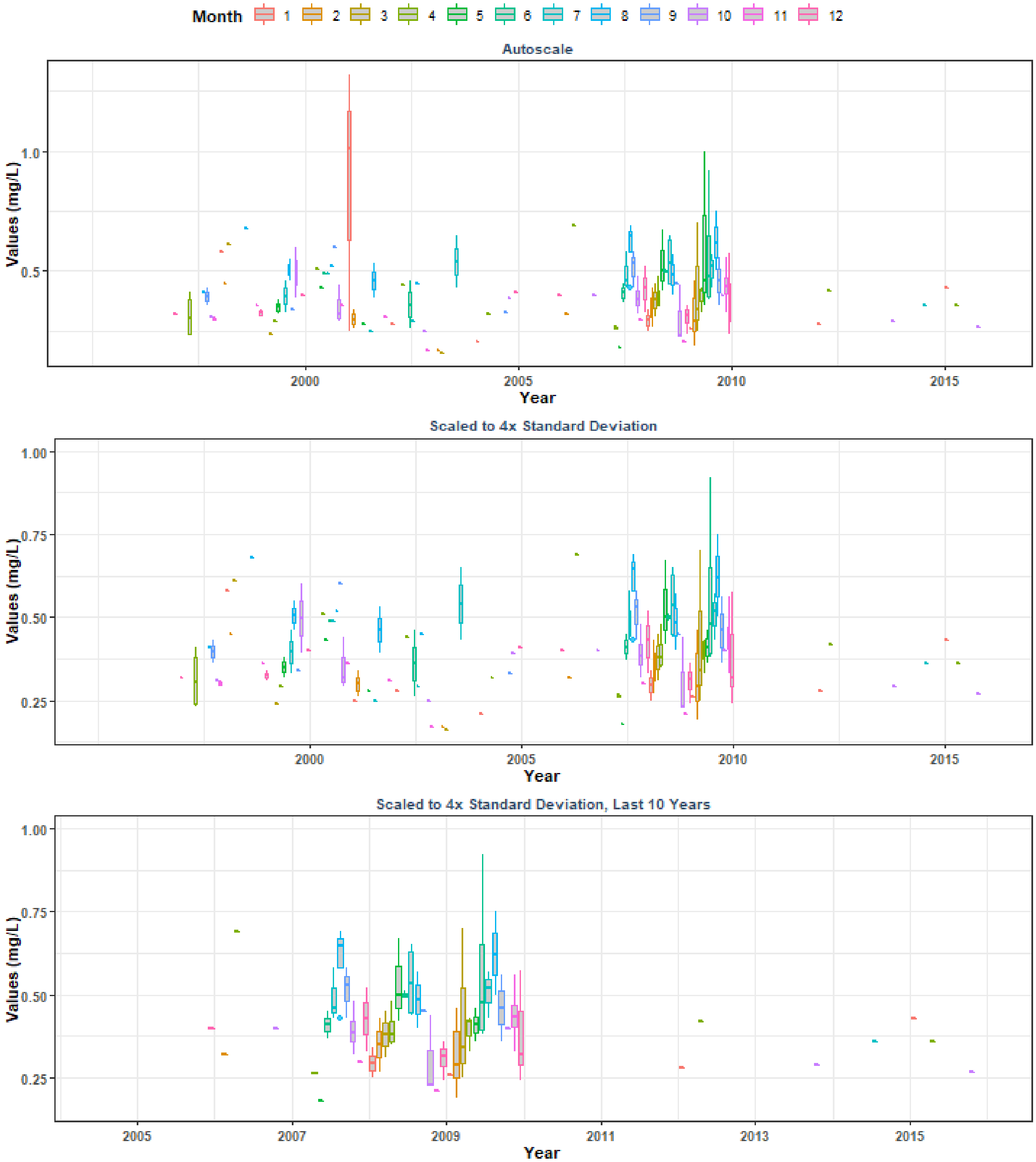


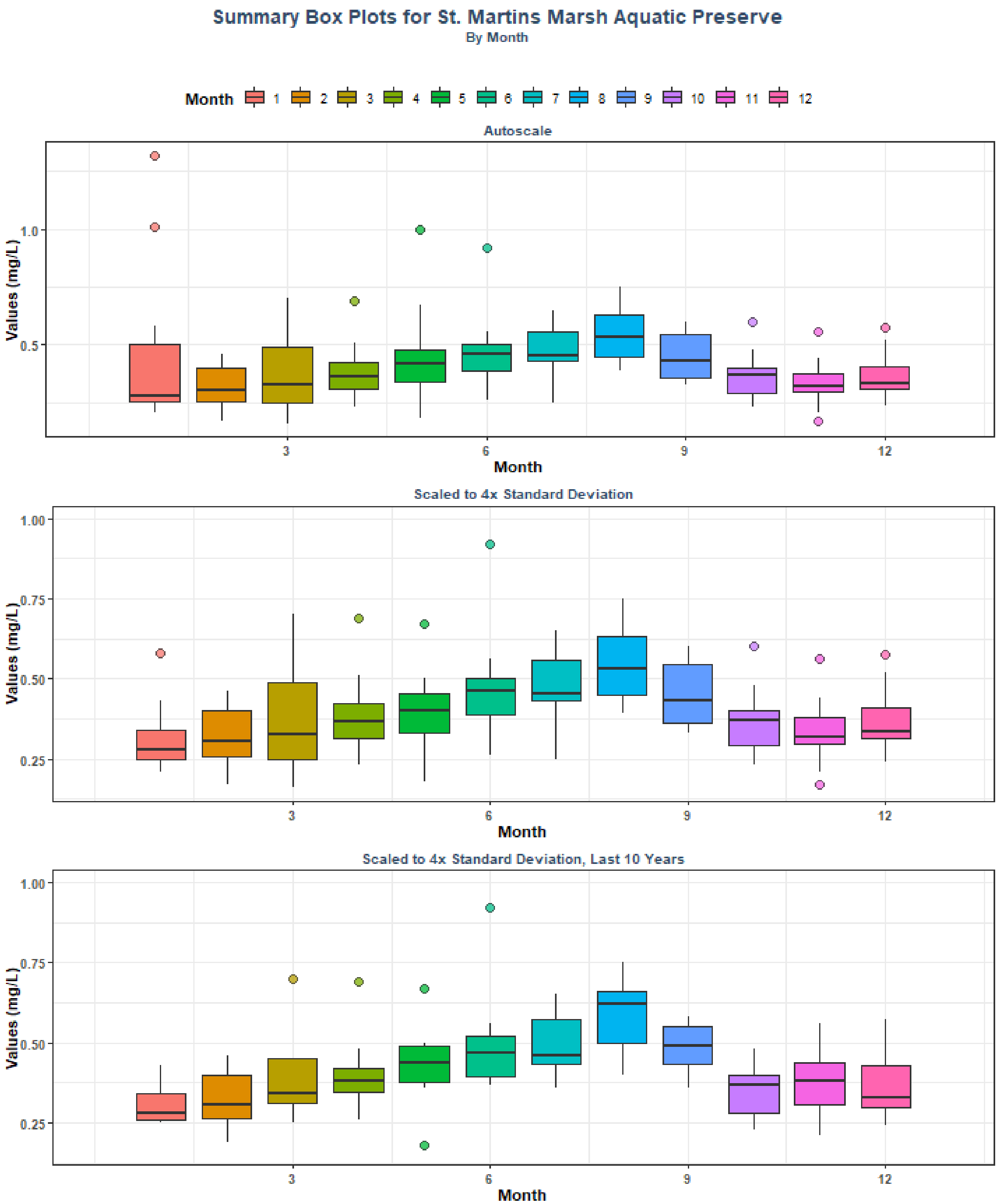




Summary Box Plots for St. Martins Marsh Aquatic Preserve

By Year & Month





Summary Box Plots for St. Martins Marsh Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
1.0
0.5
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
1.00
0.75
0.50
0.25
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
1.00
0.75
0.50
0.25
3
6
9
12
Month