SEACAR Continuous Water Quality Analysis: NW Region for Salinity

Last compiled on 01 June, 2022

Contents

Important Notes

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

https://github.com/FloridaSEACAR/SEACAR_Panzik

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

Libraries

Loads libraries used in the script. 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_Continuous_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 and units of the parameter.

```
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 Filtering

Most data filtering is performed on export from the database, and is indicated by the `Include` variable. `Include` values of 1 indicate the data should be used for analysis, values of 0 indicate the data should not be used for analysis. Documentation on the database filtering is provided here: SEACAR Documentation-Analysis Filters and Calculations.docx

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

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.

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

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

After the initial filtering, a second filter variable is created to determine whether enough time is represented in the managed area, which is that each managed area has 5 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.

```
data$Include <- as.logical(data$Include)
data <- data[data$Include==TRUE,]
data <- data[!is.na(data$ResultValue),]
data <- data[!is.na(data$RelativeDepth),]
data <- data[!grep("Blank", data$ActivityType),]

if(param_name == "Water_Temperature"){
   data <- data[data$ResultValue>=-5,]
} else{
   data <- data[data$ResultValue>=0,]
}



data <- data %>%
   group_by(ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
            SampleDate) %>%
   summarise(Year = unique(Year), Month = unique(Month),
             RelativeDepth = unique(RelativeDepth),
             ResultValue = mean(ResultValue), Include = unique(Include))

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

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 <- data %>%
    group_by(ManagedAreaName, ProgramID, ProgramName, ProgramLocationID) %>%
    mutate(MonitoringID = cur_group_id())
# data <- data %>%
#    mutate(MonitoringID = group_indices(., ManagedAreaName, ProgramID,
#                                        ProgramName, ProgramLocationID))

Mon_Summ <- data %>%
   group_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,
            ProgramLocationID) %>%
   summarize(ParameterName=parameter,
             RelativeDepth=unique(RelativeDepth),
             N_Data=length(ResultValue[Include==TRUE & !is.na(ResultValue)]),
             N_Years=length(unique(Year[Include==TRUE & !is.na(Year)])),
             EarliestYear=min(Year[Include==TRUE]),
             LatestYear=max(Year[Include==TRUE]),
             SufficientData=ifelse(N_Data>0 & N_Years>=10, TRUE, FALSE))
#
# Mon_Years <- data[data$Include == TRUE, ] %>%
#    group_by(MonitoringID) %>%
```

```r
#    summarize(AreaID = unique(AreaID),
#              ManagedAreaName = unique(ManagedAreaName),
#              ProgramID = unique(ProgramID),
#              ProgramName = unique(ProgramName),
#              ProgramLocationID = unique(ProgramLocationID),
#              ParameterName = parameter,
#              RelativeDepth = unique(RelativeDepth),
#              Y = length(unique(Year)))
Mon_Summ <- as.data.table(Mon_Summ[order(Mon_Summ$MonitoringID), ])

data <- merge.data.frame(data, Mon_Summ[,c("MonitoringID", "SufficientData")],
                         by = "MonitoringID")
# data$Exclude_MonitoringID <- is.element(data$MonitoringID,
#                                          Mon_Years$MonitoringID[
#                                            Mon_Years$Enough_Time == FALSE])
data$Use_In_Analysis <- ifelse(data$Include == TRUE &
                                  data$SufficientData == TRUE, TRUE, FALSE)

Mon_IDs <- unique(data$MonitoringID[data$Use_In_Analysis == TRUE])
Mon_IDs <- Mon_IDs[order(Mon_IDs)]
n <- length(Mon_IDs)
```

Monitoring Location Statistics

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

1. Take the `data` variable and only include rows that have a `Use_In_Analysis` value of TRUE
2. Group data that have the same `ManagedAreaName`, `ProgramID`, `ProgramName`, `ProgramLocationID`, `Year`, and `Month`.
   - Second summary statistics consider the monitoring location grouping and `Year`.
   - Third summary statistics consider the monitoring location grouping and `Month`.
3. For each group, provide the following information: Earliest Sample Date (EarliestSampleDate), Latest Sample Date (LastSampleDate), Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, and a list of all Program IDs included in these measurements.
4. Sort the data in ascending (A to Z and 0 to 9) order based on `ManagedAreaName`, `ProgramID`, `ProgramName`, `ProgramLocationID`, `Year`, and `Month` in that order.
5. Write summary stats to a pipe-delimited .txt file in the output directory

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

```
Mon_YM_Stats <- as.data.table(Mon_YM_Stats[order(Mon_YM_Stats$ManagedAreaName,
                                                  Mon_YM_Stats$ProgramID,
                                                  Mon_YM_Stats$ProgramName,
                                                  Mon_YM_Stats$ProgramLocationID,
                                                  Mon_YM_Stats$Year,
                                                  Mon_YM_Stats$Month), ])
fwrite(Mon_YM_Stats, paste0(out_dir,"/", param_name, "_", region,
                            "_MonitoringLoc_YearMonth_Stats.txt"), sep = "|")

Mon_Y_Stats <- data[data$Use_In_Analysis == TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
            Year) %>%
   summarize(ParameterName = parameter,
             RelativeDepth = unique(RelativeDepth),
             EarliestSampleDate = min(SampleDate),
             LastSampleDate = max(SampleDate), N = length(ResultValue),
             Min = min(ResultValue), Max = max(ResultValue),
             Median = median(ResultValue), Mean = mean(ResultValue),
             StandardDeviation = sd(ResultValue))
Mon_Y_Stats <- as.data.table(Mon_Y_Stats[order(Mon_Y_Stats$ManagedAreaName,
                                                Mon_Y_Stats$ProgramID,
                                                Mon_Y_Stats$ProgramName,
                                                Mon_Y_Stats$ProgramLocationID,
                                                Mon_Y_Stats$Year), ])
fwrite(Mon_Y_Stats, paste0(out_dir,"/", param_name, "_", region,
                           "_MonitoringLoc_Year_Stats.txt"), sep = "|")

Mon_M_Stats <- data[data$Use_In_Analysis == TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
            Month) %>%
   summarize(ParameterName = parameter,
             RelativeDepth = unique(RelativeDepth),
             EarliestSampleDate = min(SampleDate),
             LastSampleDate = max(SampleDate), N = length(ResultValue),
             Min = min(ResultValue), Max = max(ResultValue),
             Median = median(ResultValue), Mean = mean(ResultValue),
             StandardDeviation = sd(ResultValue))
Mon_M_Stats <- as.data.table(Mon_M_Stats[order(Mon_M_Stats$ManagedAreaName,
                                                Mon_M_Stats$ProgramID,
                                                Mon_M_Stats$ProgramName,
                                                Mon_M_Stats$ProgramLocationID,
                                                Mon_M_Stats$Month), ])

fwrite(Mon_M_Stats, paste0(out_dir,"/", param_name, "_", region,
                           "_MonitoringLoc_Month_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 that performed at The Water Atlas: https://sarasota.wateratlas.usf.edu/water-quality-trends/#analysis-overview

The following steps are performed:

1. Define the trend function.
2. Take the `data` variable and only include rows that have a `Use_In_Analysis` value of `TRUE`
3. Group data that have the same `ManagedAreaName`, `ProgramID`, `ProgramName`, and `ProgramLocationID`.
4. For each group, provides the following information: Earliest Sample Date (EarliestSampleDate), Latest Sample Date (LastSampleDate), Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation,
5. For each group, a temporary variable is created to run the `kendallSeasonalTrendTest` function using the `Year` values for year, and `Month` as the seasonal qualifier, and Trend.
   - An independent.obs value of `TRUE` indicates that the data should be treated as not being serially auto-correlated. An independent.obs value of `FALSE` indicates that it is treated as being serially auto-correlated, but also requires one observation per season per year for the full time of observation.
   - tau, Senn Slope (SennSlope), Senn Intercept (SennIntercept), and p are extracted from the model results.
6. The two stats tables are merged based on similar groups, and then Trend is determined from the user-defined function.
7. Write summary stats to a pipe-delimited .txt file in the output directory
   - Click this text to open Git directory with output files
8. Add the Monitoring IDS to `KT.Stats` for easier use while plotting.

```
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)
   }, 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$MonitoringID),
         stats.median,
         independent,
         tau,
         p,
         slope,
         intercept,
         trend)
  return(KT)
}
runStats <- function(data) {
  data$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[8])) {
    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.) {
```

```r
      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$ManagedAreaName
c_names <- c("MonitoringID", "Median", "Independent", "tau", "p",
             "SennSlope", "SennIntercept", "Trend")
if(n==0){
   KT.Stats <- data.frame(matrix(ncol=length(c_names),
                                 nrow=nrow(Mon_Summ)))
   colnames(KT.Stats) <- c_names
   KT.Stats[, c("MonitoringID")] <- Mon_Summ[, c("MonitoringID")]
} else{
   for (i in 1:n) {
      x <- nrow(data[data$Use_In_Analysis == TRUE &
                     data$MonitoringID == Mon_IDs[i], ])
      if (x>0) {
         KT.Stats <- runStats(data[data$Use_In_Analysis == TRUE &
                                   data$MonitoringID == Mon_IDs[i], ])
      }
   }
   KT.Stats <- as.data.frame(KT.Stats)

   if(dim(KT.Stats)[2]==1){
      KT.Stats <- as.data.frame(t(KT.Stats))
   }
   colnames(KT.Stats) <- c_names
   rownames(KT.Stats) <- seq(1:nrow(KT.Stats))
   KT.Stats$tau <- round(as.numeric(KT.Stats$tau), digits=4)
   KT.Stats$p <- round(as.numeric(KT.Stats$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)
}

KT.Stats <-  merge.data.frame(Mon_Summ, KT.Stats,
                              by=c("MonitoringID"), all=TRUE)

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

KT.Stats$MonitoringID <- NULL
fwrite(KT.Stats, paste0(out_dir,"/", param_name, "_", region,
                        "_KendallTau_Stats.txt"), sep = "|")
```

```
KT.Stats$MonitoringID <- Mon_Summ$MonitoringID
data <- data[!is.na(data$ResultValue),]
```

Appendix I: Dataset Summary Box Plots

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

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

This set of box plots are grouped by year.

```
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"))

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(0, y_scale) +
   plot_theme

p3 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = as.integer(Year), y = ResultValue, group = Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle = "Scaled to 4x Standard Deviation, Last 10 Years",
        x = "Year", y = paste0("Values (", unit, ")")) +
   ylim(0, y_scale) +
   scale_x_continuous(limits = c(max(data$Year) - 10.5, max(data$Year)+0.5),
                      breaks = seq(max(data$Year) - 10, max(data$Year), 2)) +
   plot_theme

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 5x Standard Deviation",
        x = "Year", y = paste0("Values (", unit, ")")) +
   ylim(0, y_scale) +
   plot_theme +
   theme(legend.position = "none")

p3 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = YearMonthDec, y = ResultValue,
                 group = YearMonth, color = as.factor(Month))) +
```

```
  geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
  labs(subtitle = "Scaled to 5x Standard Deviation, Last 10 Years",
       x = "Year", y = paste0("Values (", unit, ")")) +
  ylim(0, y_scale) +
  scale_x_continuous(limits = c(max(data$Year) - 10.5, max(data$Year)+0.5),
                     breaks = seq(max(data$Year) - 10, max(data$Year), 2)) +
  plot_theme +
  theme(legend.position = "none")
leg <- get_legend(p1)
set <- ggarrange(leg, p1 + theme(legend.position = "none"), p2, p3, ncol = 1,
                 heights = c(0.1, 1, 1, 1))

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

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

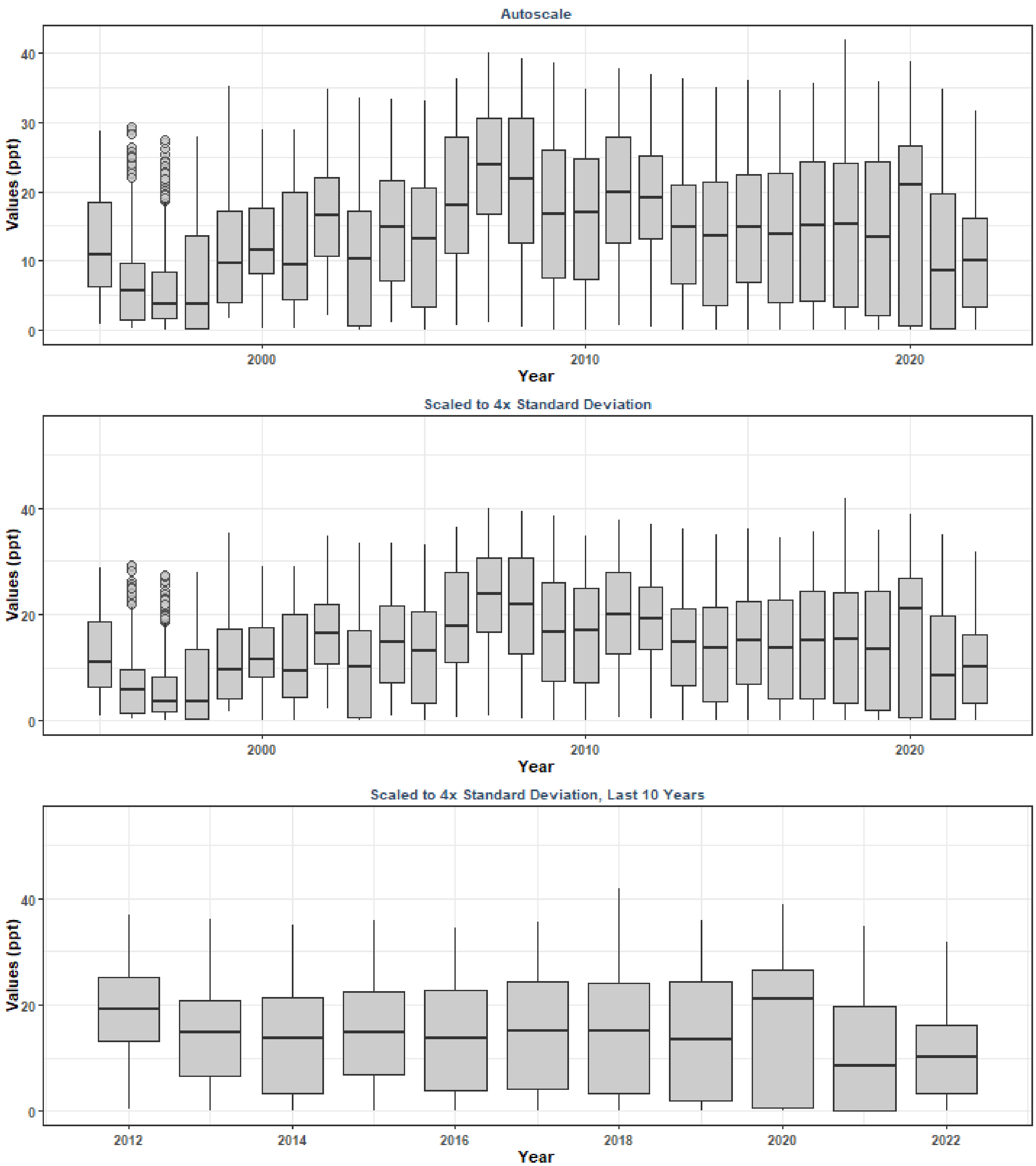
The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

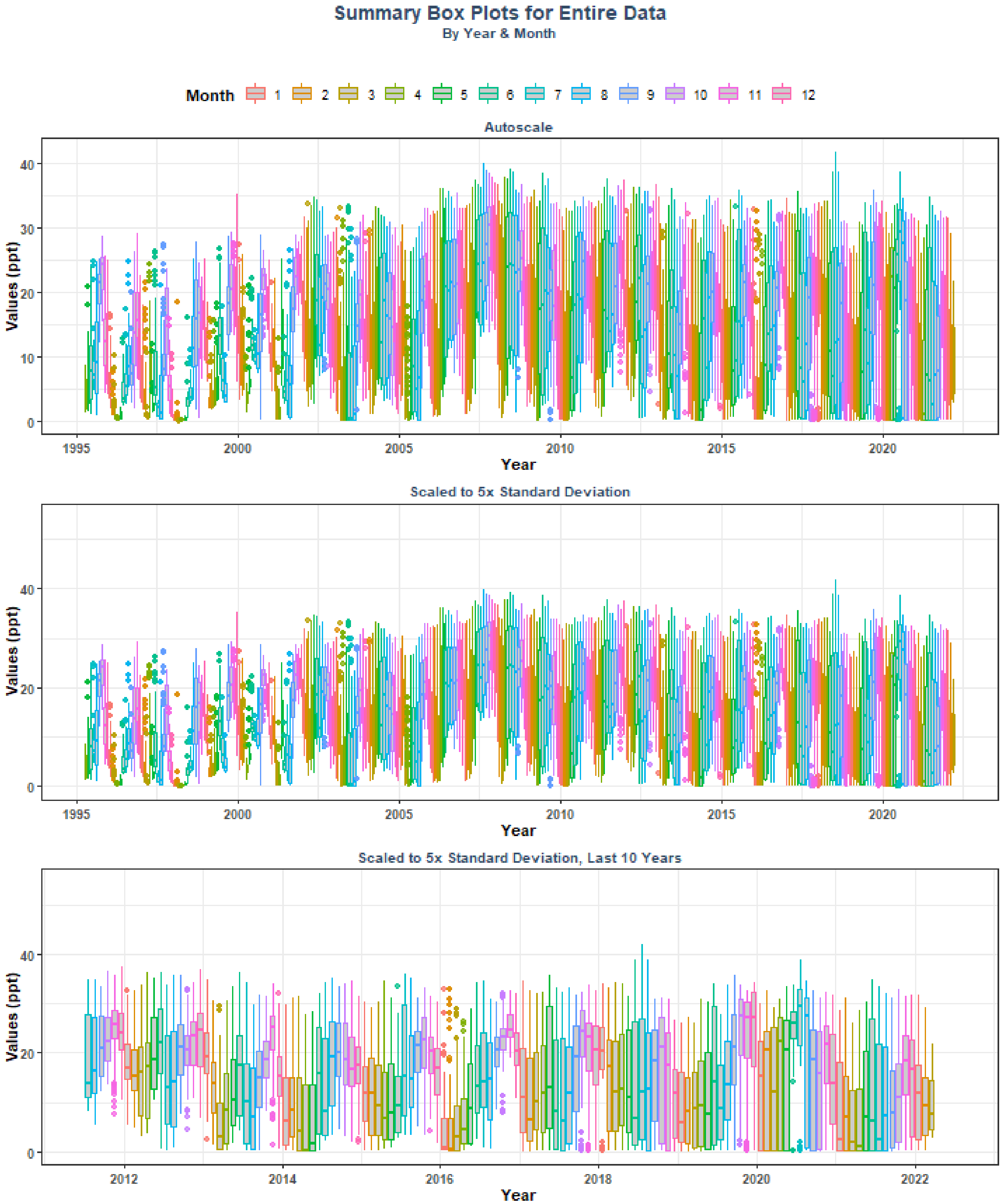
```
p1 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
  geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
               outlier.color="#333333", outlier.alpha=0.75) +
  labs(subtitle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

p2 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
  geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
               outlier.color="#333333", outlier.alpha=0.75) +
  labs(subtitle = "Scaled to 5x Standard Deviation",
       x = "Month", y = paste0("Values (", unit, ")")) +
  ylim(0, y_scale) +
  scale_x_continuous(limits = c(0, 13), breaks = seq(3, 12, 3)) +
  plot_theme +
  theme(legend.position = "none")

p3 <- ggplot(data = data[data$Include == TRUE &
                            data$Year >= max(data$Year) - 10, ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
  geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
               outlier.color="#333333", outlier.alpha=0.75) +
  labs(subtitle = "Scaled to 5x Standard Deviation, Last 10 Years",
```

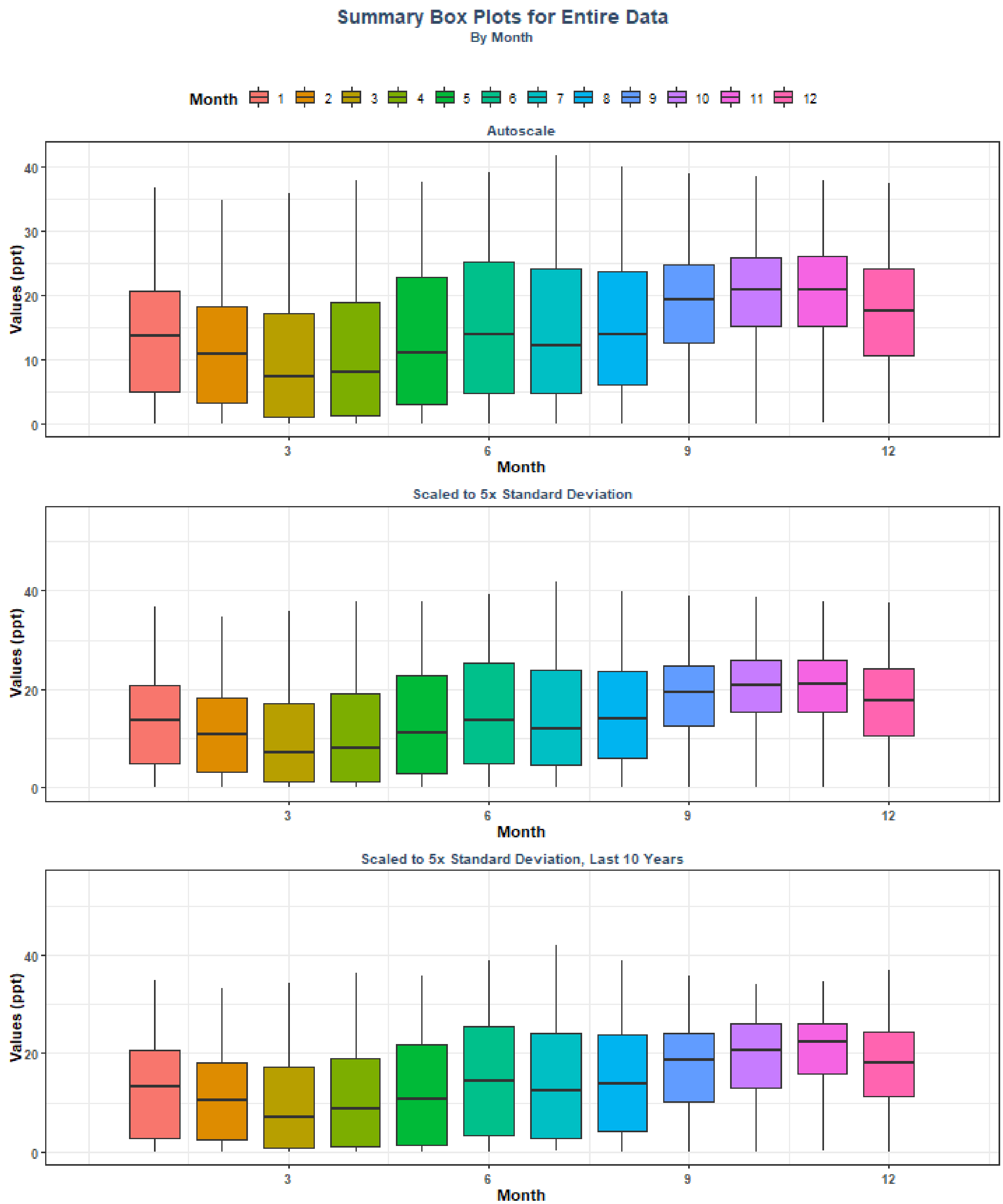
```r
       x = "Month", y = paste0("Values (", unit, ")")) +
  ylim(0, y_scale) +
  scale_x_continuous(limits = c(0, 13), breaks = seq(3, 12, 3)) +
  plot_theme +
  theme(legend.position = "none")
leg <- get_legend(p1)
set <- ggarrange(leg, p1 + theme(legend.position = "none"), p2, p3, ncol = 1,
                 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




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



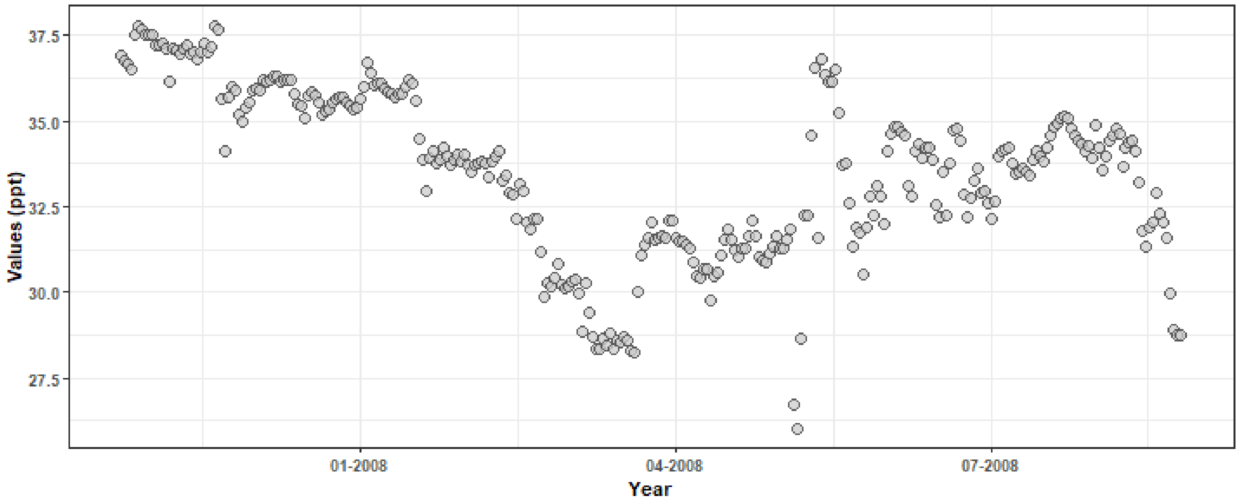
Appendix II: Excluded Monitoring Locations

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

```
Mon_Exclude <- Mon_Summ[Mon_Summ$N_Years<5 & Mon_Summ$N_Years>0,]
Mon_Exclude <- Mon_Exclude[order(Mon_Exclude$MonitoringID),]
z=nrow(Mon_Exclude)

if(z==0){
   print("There are no monitoring locations that qualify.")
} else {
   for(i in 1:z){
      MA_name <- unique(data$ManagedAreaName[
         data$MonitoringID==Mon_Exclude$MonitoringID[i]])
      Mon_name <- paste(unique(data$ProgramID[
         data$MonitoringID==Mon_Exclude$MonitoringID[i]]),
         unique(data$ProgramName[
            data$MonitoringID==Mon_Exclude$MonitoringID[i]]),
         unique(data$ProgramLocationID[
            data$MonitoringID==Mon_Exclude$MonitoringID[i]]),
         sep = " | ")

      p1<-ggplot(data=data[data$MonitoringID==Mon_Exclude$MonitoringID[i]&
                              data$Include == TRUE, ],
                 aes(x = SampleDate, y = ResultValue)) +
         geom_point(shape=21, size=3, color="#333333", fill="#cccccc",
                    alpha=0.75) +
         labs(title=
                 paste0("Scatter Plot of Excluded Monitoring Location ",
                        MA_name, "\n", Mon_name, "\n(", Mon_Exclude$Y[i],
                        " Unique Years)"),
              subtitle="Autoscale", x = "Year",
              y = paste0("Values (", unit, ")")) +
         plot_theme +
         scale_x_date(labels = date_format("%m-%Y"))
      print(p1)
   }
}
```

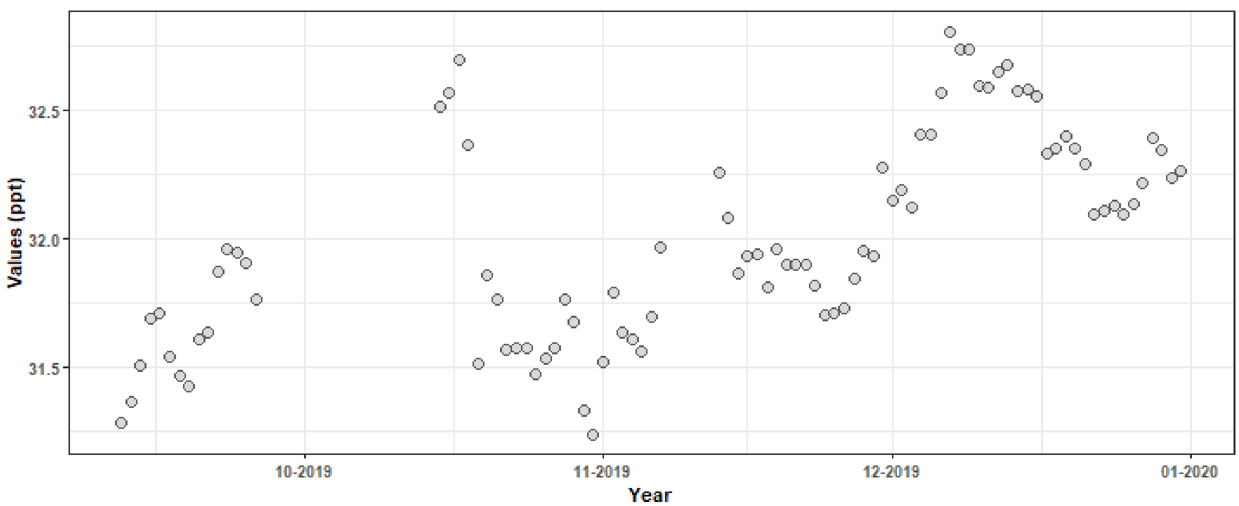
Scatter Plot of Excluded Monitoring Location Alligator Harbor Aquatic Preserve
468 | Central Panhandle Aquatic Preserves Continuous Water Quality Monitoring | CPAH
(Unique Years)

Autoscale



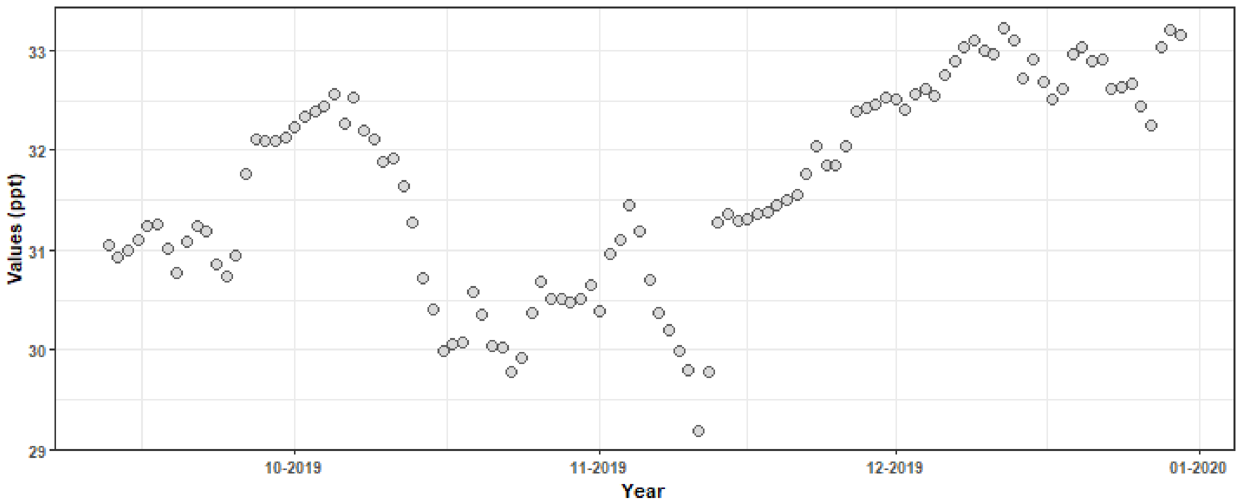
Scatter Plot of Excluded Monitoring Location Alligator Harbor Aquatic Preserve
468 | Central Panhandle Aquatic Preserves Continuous Water Quality Monitoring | CPAH2
(Unique Years)

Autoscale



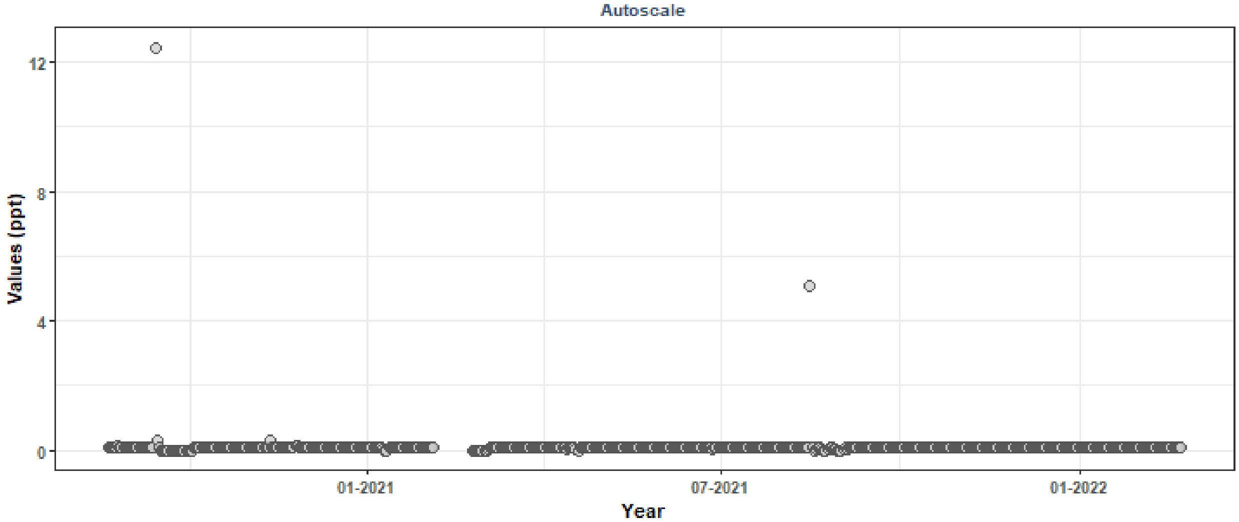
Scatter Plot of Excluded Monitoring Location Alligator Harbor Aquatic Preserve
468 | Central Panhandle Aquatic Preserves Continuous Water Quality Monitoring | CPFS
(Unique Years)

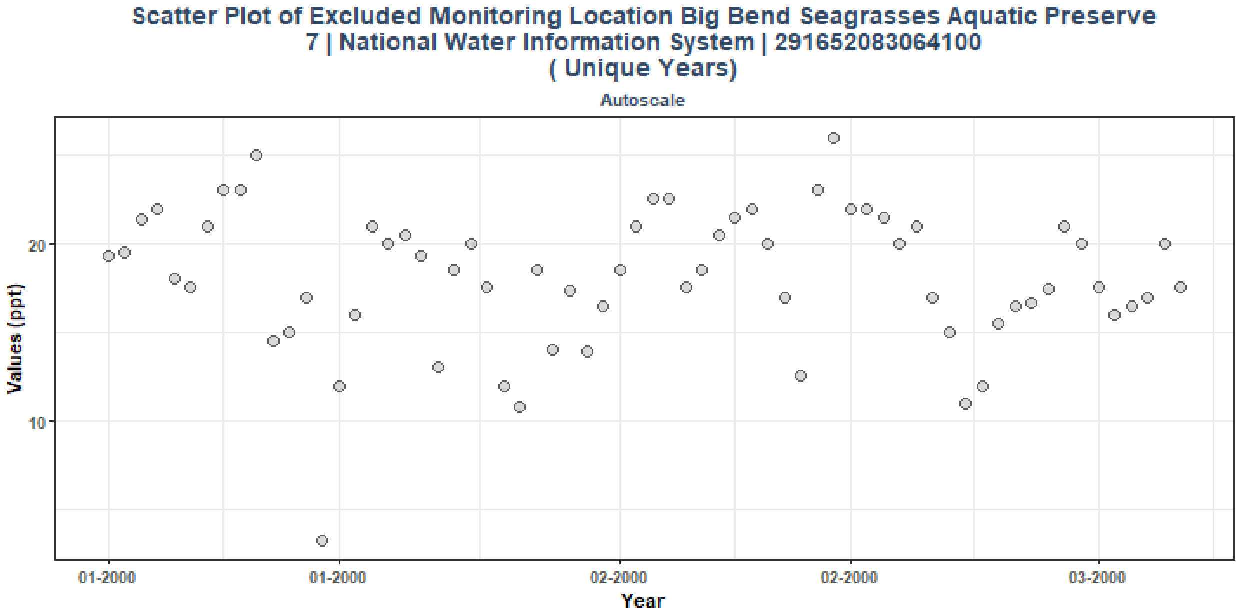
Autoscale

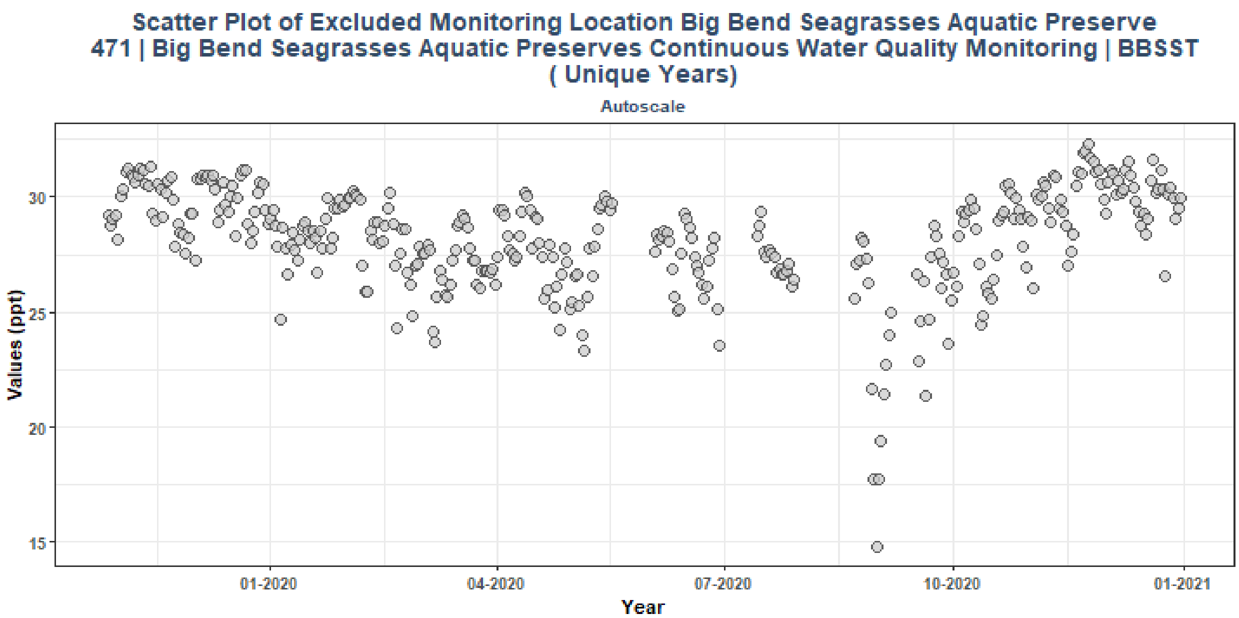


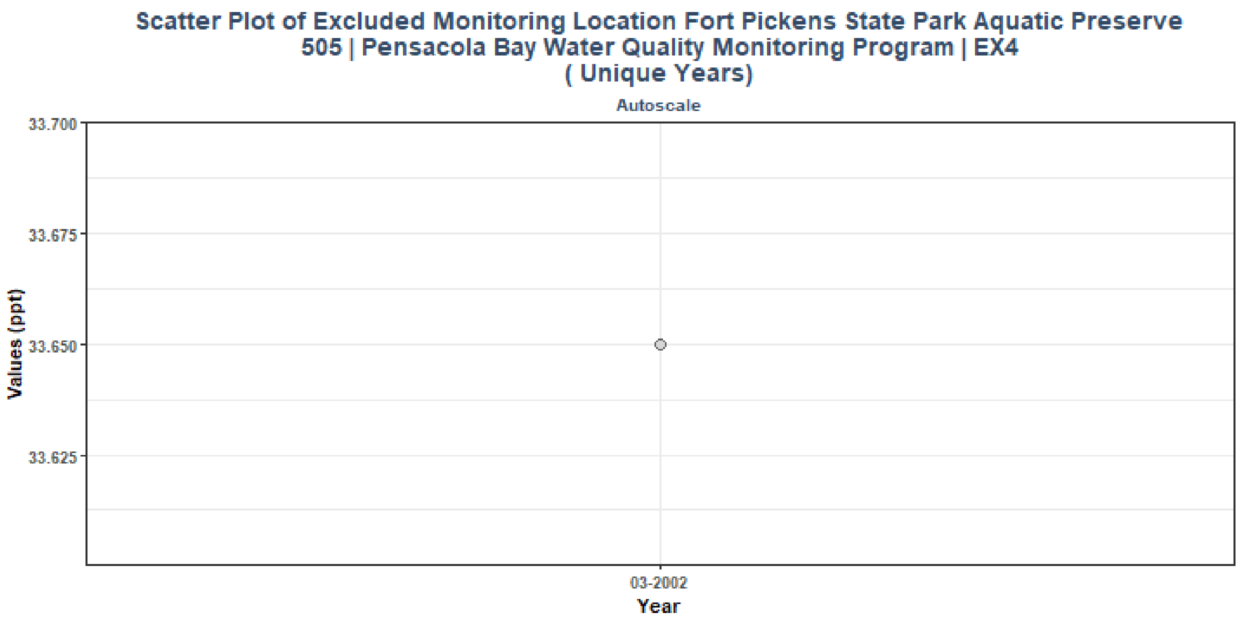
Scatter Plot of Excluded Monitoring Location Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apabpwq
(Unique Years)

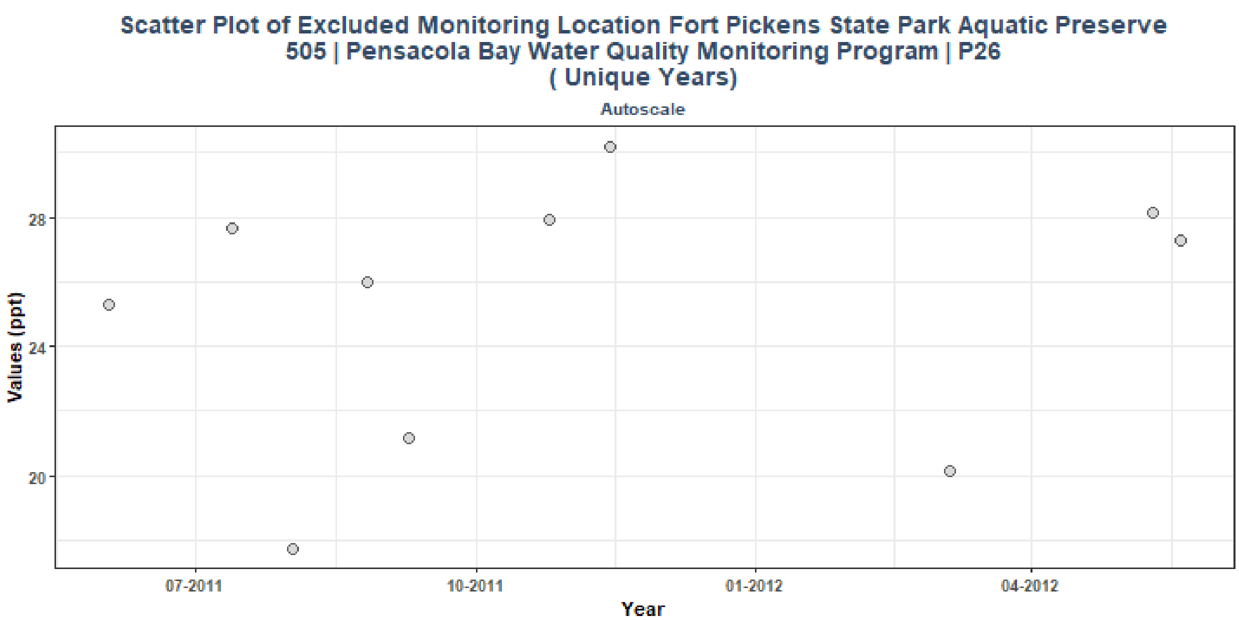
Autoscale

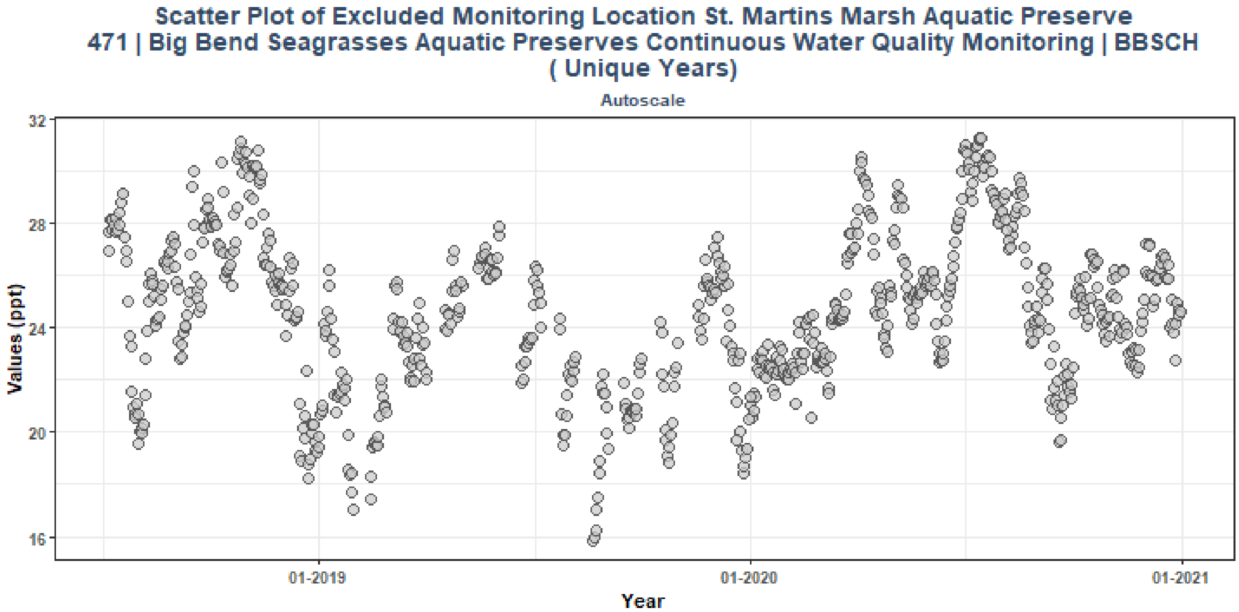



Scatter Plot of Excluded Monitoring Location Big Bend Seagrasses Aquatic Preserve
7 | National Water Information System | 291652083064100
(Unique Years)
Autoscale
Values (ppt)
20
10
01-2000
01-2000
02-2000
02-2000
03-2000
Year


Scatter Plot of Excluded Monitoring Location Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSST
(Unique Years)
Autoscale
Values (ppt)
30
25
20
15
01-2020
04-2020
07-2020
10-2020
01-2021
Year


Scatter Plot of Excluded Monitoring Location Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program | EX4
(Unique Years)
Autoscale
Values (ppt)
33.700
33.675
33.650
33.625
03-2002
Year


Scatter Plot of Excluded Monitoring Location Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program | P26
(Unique Years)
Autoscale
Values (ppt)
28
24
20
07-2011
10-2011
01-2012
04-2012
Year



Appendix III: Monitoring Location Trendlines

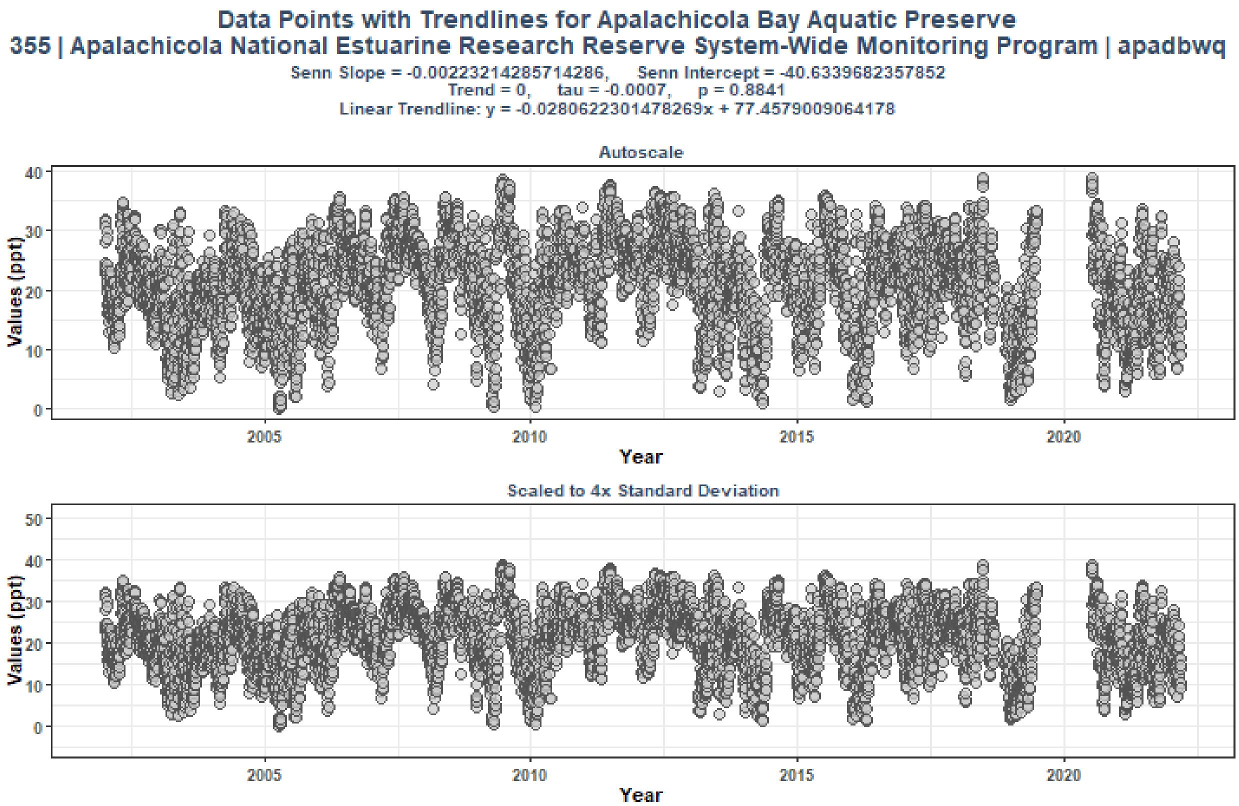
The plots created in this section are designed to show the general trend of the data. Data is taken and grouped by `MonitoringID`. The trendlines on the plots are created using the Senn slope and intercept from the seasonal Kendall Tau analysis. The scripts that create plots follow this format

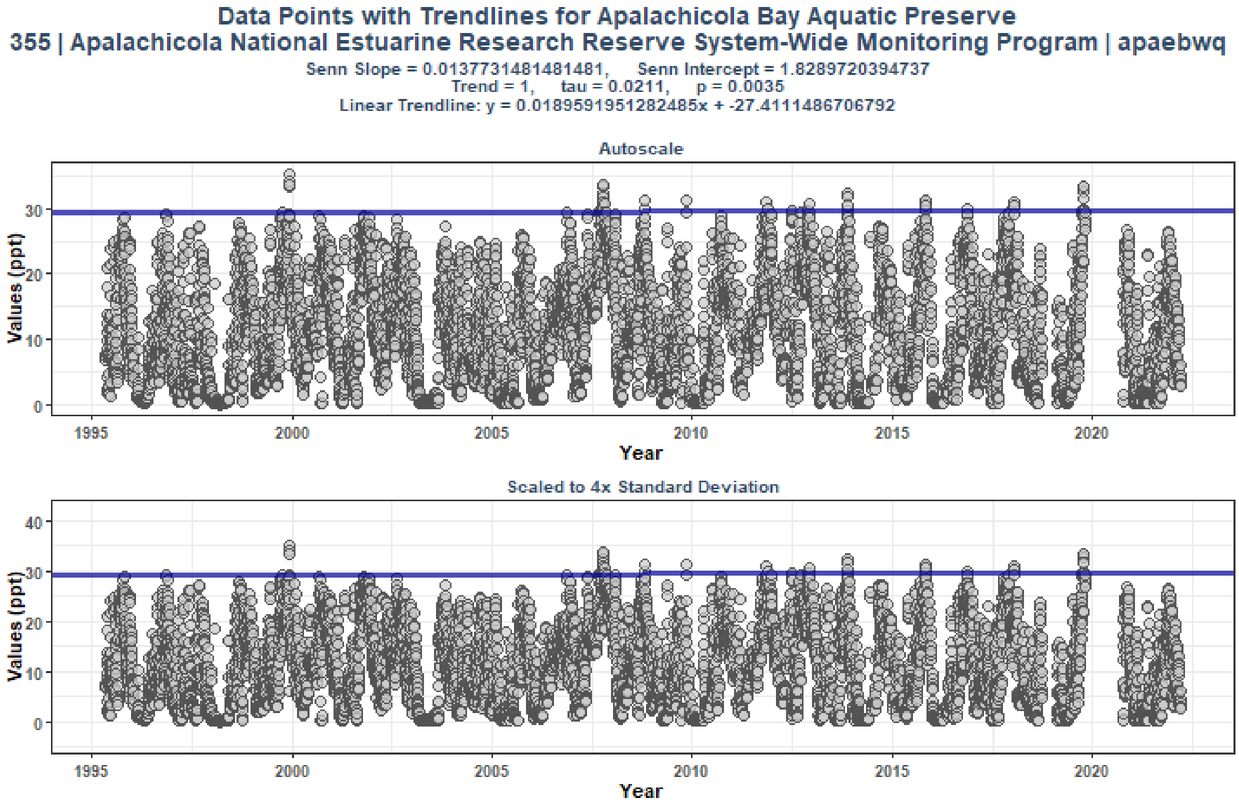
1. Use the data set that only has `Use_In_Analysis` of `TRUE` for the desired monitoring location
2. Determine the earliest and latest year of the data to create x-axis scale and intervals
3. Determine the minimum, mean, and standard deviation for the data to be used for y-axis scales
   - Excludes the top 2% of values to reduce the impact of extreme outliers on the y-axis scale
4. Set what values are to be used for the x-axis, y-axis, and the variable that should determine groups for the 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

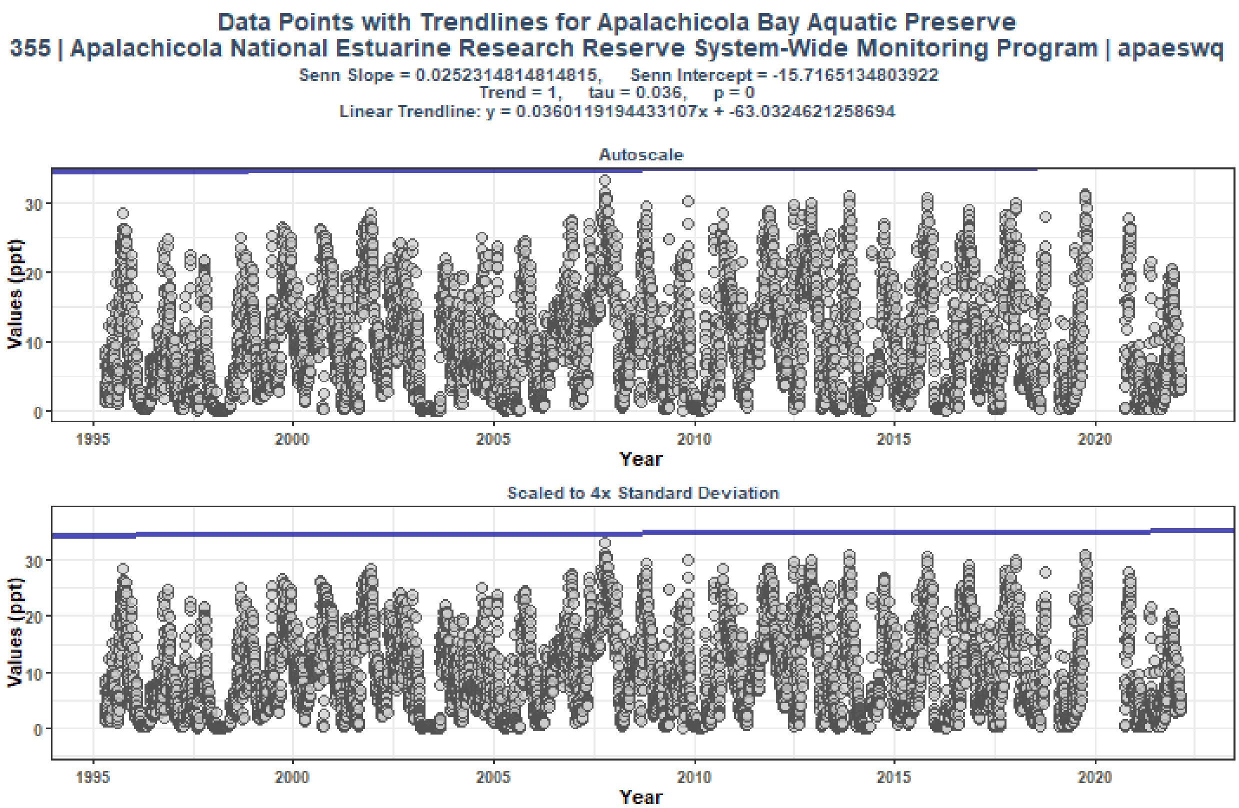
```
if(n==0){
   print("There are no monitoring locations that qualify.")
} else {
   for (i in 1:n) {
      plot_data <- data[data$Use_In_Analysis == TRUE &
                              data$MonitoringID == Mon_IDs[i],]
      year_lower <- min(plot_data$Year)
      year_upper <- max(plot_data$Year)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

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

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

model <- lm(ResultValue ~ DecDate,
            data = plot_data)
m_int <- coef(model)[[1]]
m_slope <- coef(model)[[2]]
rm(model)
MA_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID == Mon_IDs[i]]
Mon_name <- paste(KT.Stats$ProgramID[KT.Stats$MonitoringID == Mon_IDs[i]],
   KT.Stats$ProgramName[KT.Stats$MonitoringID == Mon_IDs[i]],
   KT.Stats$ProgramLocationID[KT.Stats$MonitoringID == Mon_IDs[i]],
   sep = " | ")

p1 <- ggplot(data = plot_data,
             aes(x = DecDate, y = ResultValue)) +
   geom_point(shape=21, size=3, color="#333333", fill="#cccccc",
              alpha=0.75) +
   geom_abline(aes(slope=s_slope, intercept=s_int),
               color="#000099", size=1.2, alpha=0.7) +
   labs(subtitle = "Autoscale",
        x = "Year", y = paste0("Values (", unit, ")")) +
   plot_theme

p2 <- ggplot(data = plot_data,
             aes(x = DecDate, y = ResultValue)) +
   geom_point(shape=21, size=3, color="#333333", fill="#cccccc",
              alpha=0.75) +
   geom_abline(aes(slope=s_slope, intercept=s_int),
               color="#000099", size=1.2, alpha=0.7) +
   ylim(min_RV-0.1*y_scale, y_scale) +
   labs(subtitle = "Scaled to 4x Standard Deviation",
        x = "Year", y = paste0("Values (", unit, ")")) +
   plot_theme

KTset <- ggarrange(p1, p2, ncol = 1, heights = c(1, 1))

p0 <- ggplot() + labs(title = paste0("Data Points with Trendlines for ",
                                     MA_name, "\n", Mon_name),
                      subtitle =paste0("Senn Slope = ", s_slope,
                                       ",     Senn Intercept = ", s_int,
                                       "\nTrend = ", trend,
                                       ",     tau = ", tau,
                                       ",     p = ", p,
                                       "\nLinear Trendline: ",
```

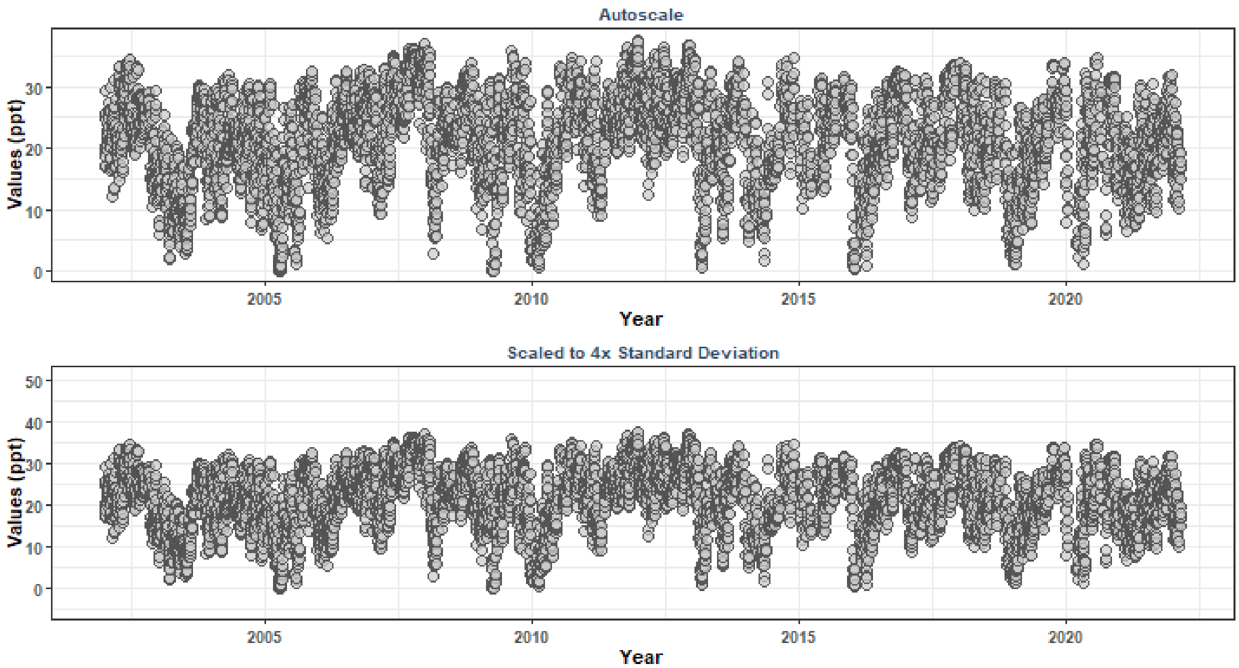
```
                                             "y = ", m_slope,"x + ",m_int)) +
         plot_theme + theme(panel.border = element_blank(),
                            panel.grid.major = element_blank(),
                            panel.grid.minor = element_blank(),
                            axis.line = element_blank())

      print(ggarrange(p0, KTset, ncol = 1, heights = c(0.20, 1)))
      rm(plot_data)
      rm(KTset, leg)
   }
}
```



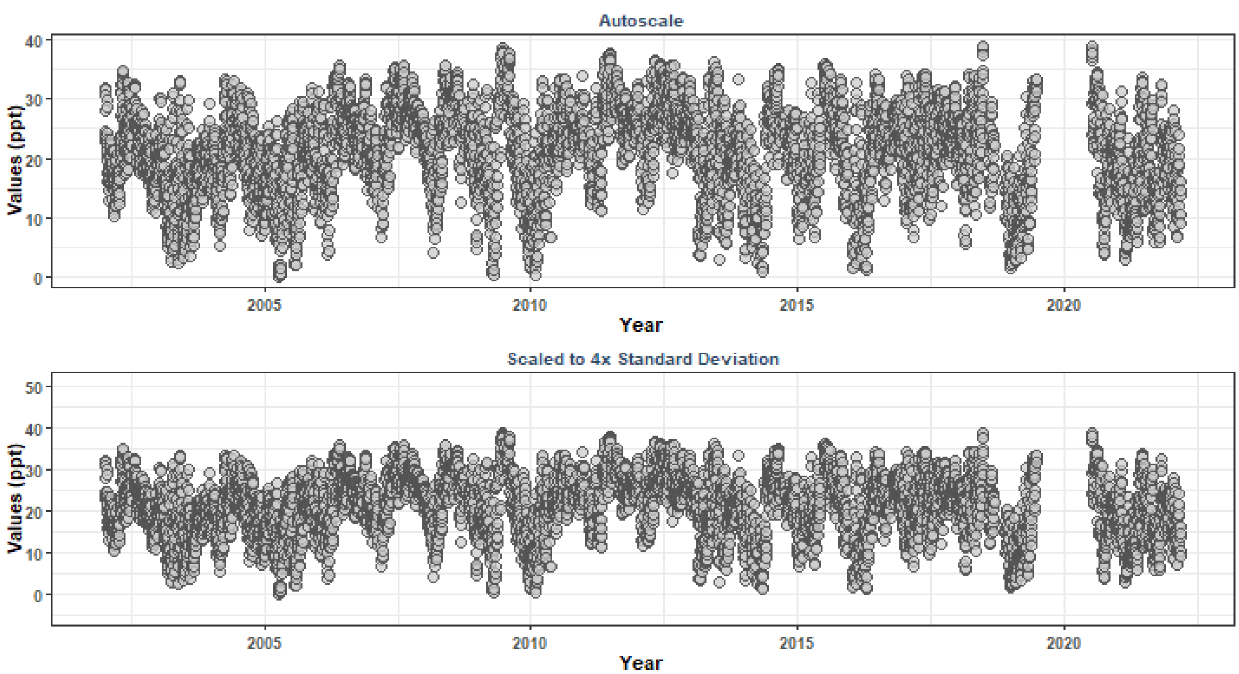

Data Points with Trendlines for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq
Senn Slope = 0.0137731481481481, Senn Intercept = 1.8289720394737
Trend = 1, tau = 0.0211, p = 0.0035
Linear Trendline: y = 0.0189591951282485x + -27.4111486706792
Autoscale
Values (ppt)
Year
Scaled to 4x Standard Deviation
Values (ppt)
Year

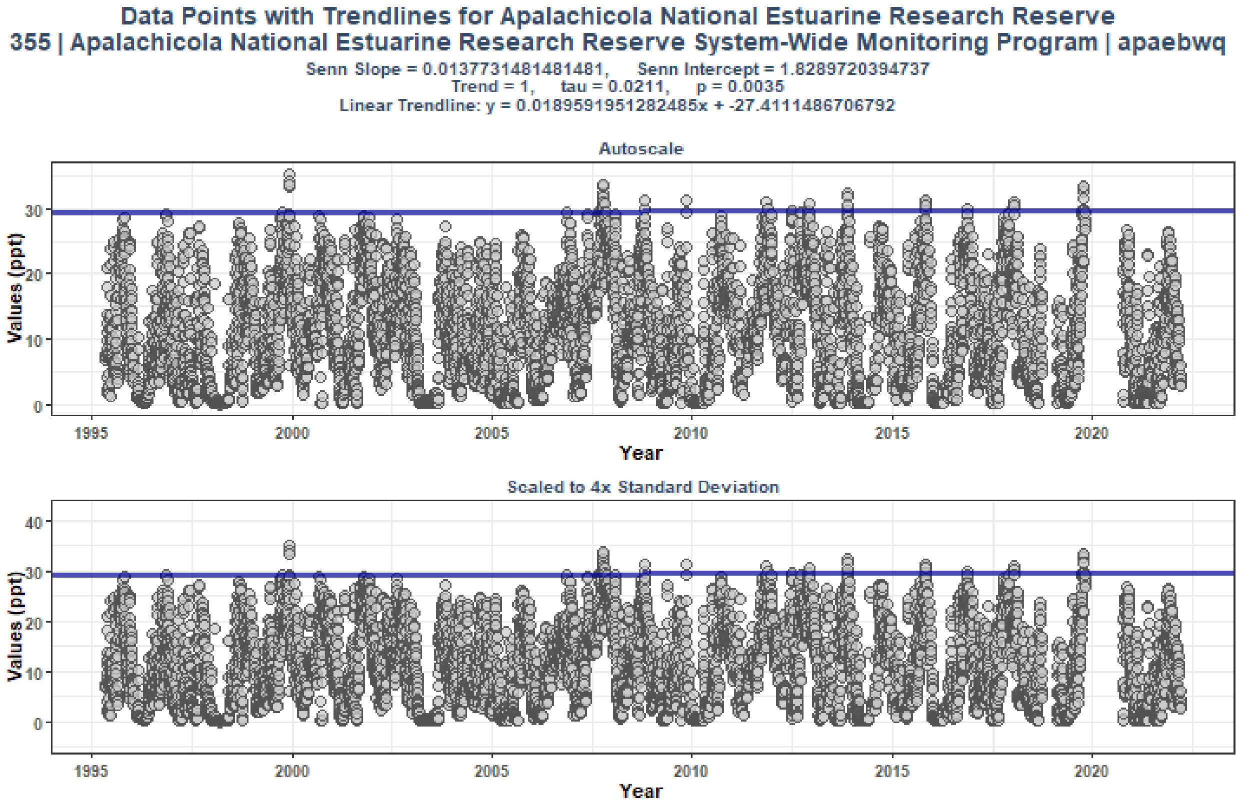

Data Points with Trendlines for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
Senn Slope = 0.0252314814814815, Senn Intercept = -15.7165134803922
Trend = 1, tau = 0.036, p = 0
Linear Trendline: y = 0.0360119194433107x + -63.0324621258694
Autoscale
Values (ppt)
Year
Scaled to 4x Standard Deviation
Values (ppt)
Year

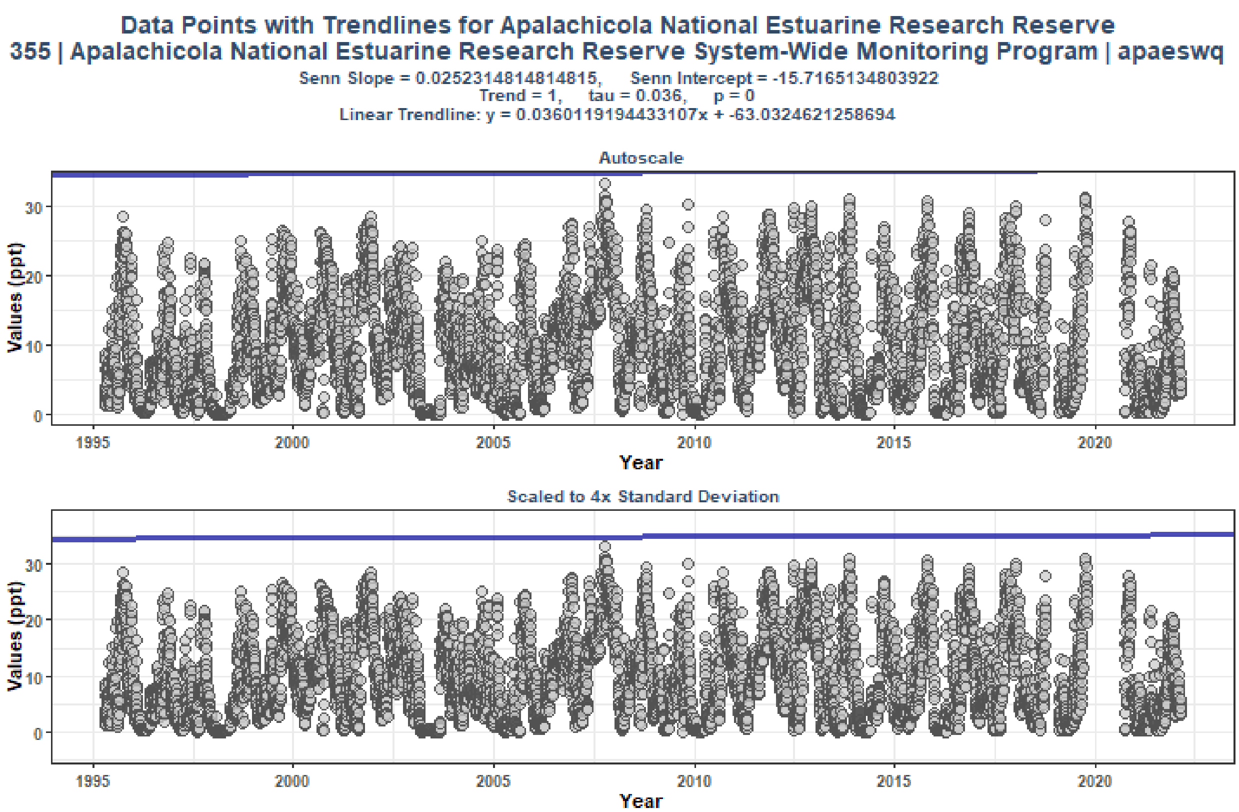
Data Points with Trendlines for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apacpwq
Senn Slope = -0.0404513888888888, Senn Intercept = 165.890916546306
Trend = -1, tau = -0.0233, p = 0.0046
Linear Trendline: y = -0.0201118059215334x + 61.7556544783784



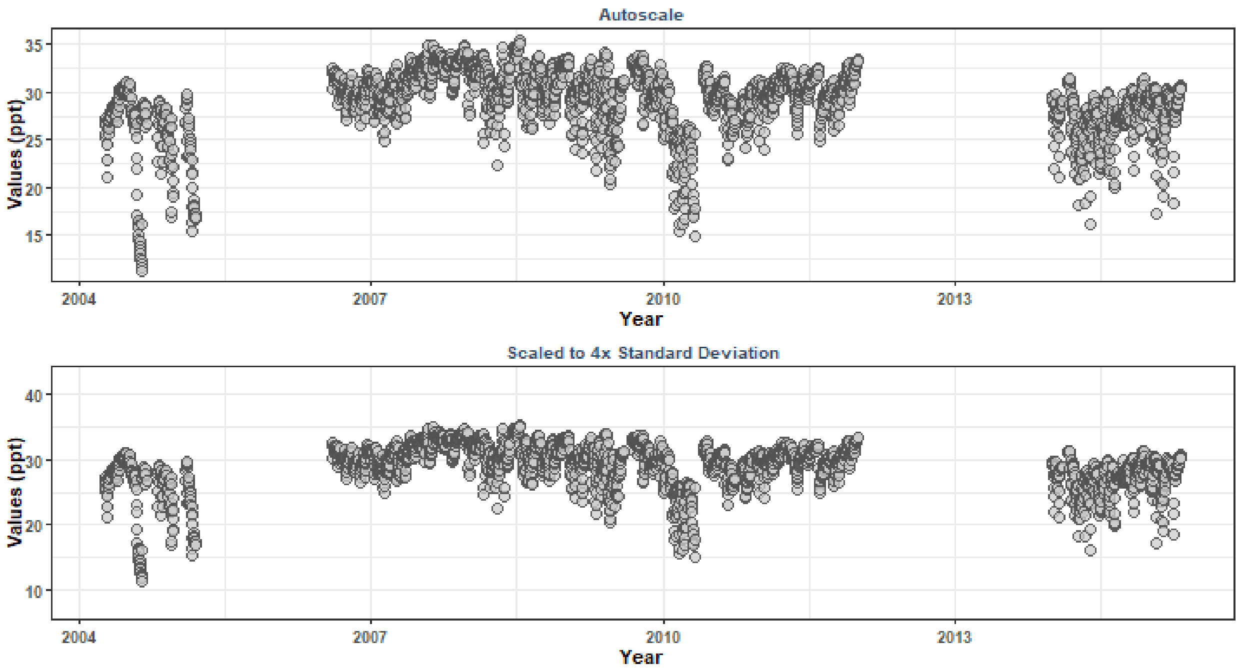
Data Points with Trendlines for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apadbwq
Senn Slope = -0.00223214285714286, Senn Intercept = -40.6339682357852
Trend = 0, tau = -0.0007, p = 0.8841
Linear Trendline: y = -0.0280622301478269x + 77.4579009064178



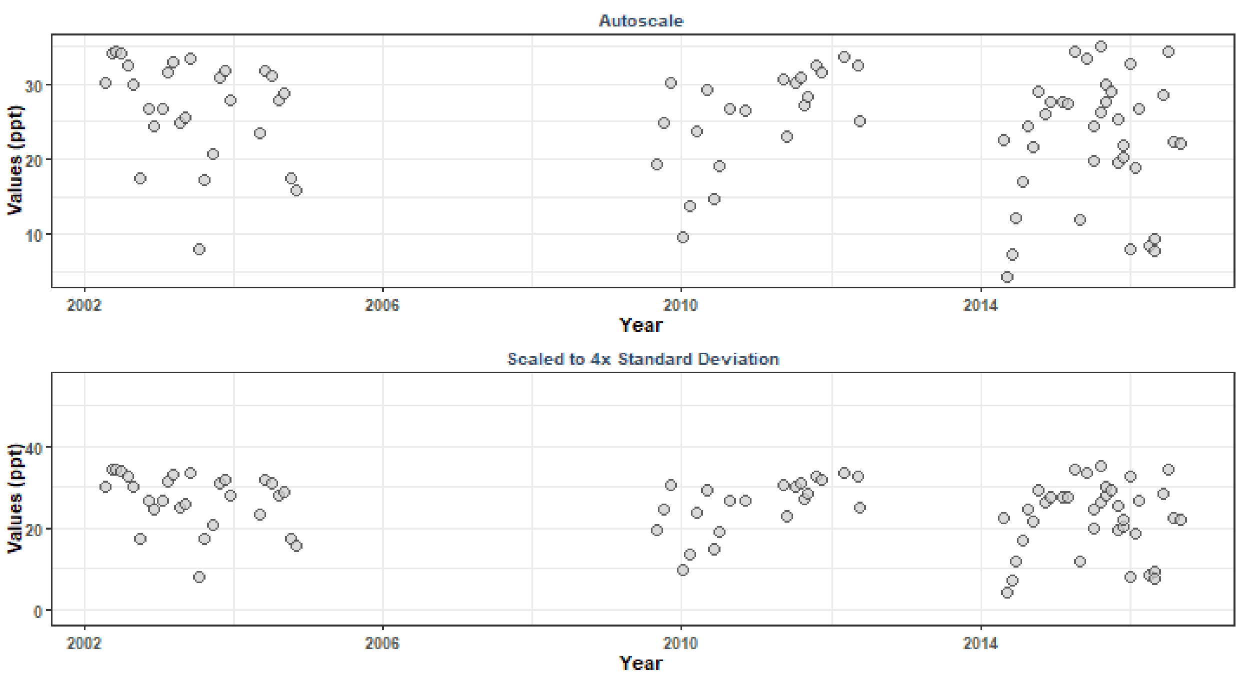

Data Points with Trendlines for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq
Senn Slope = 0.0137731481481481, Senn Intercept = 1.8289720394737
Trend = 1, tau = 0.0211, p = 0.0035
Linear Trendline: y = 0.0189591951282485x + -27.4111486706792
Autoscale
Values (ppt)
Year
Scaled to 4x Standard Deviation
Values (ppt)
Year

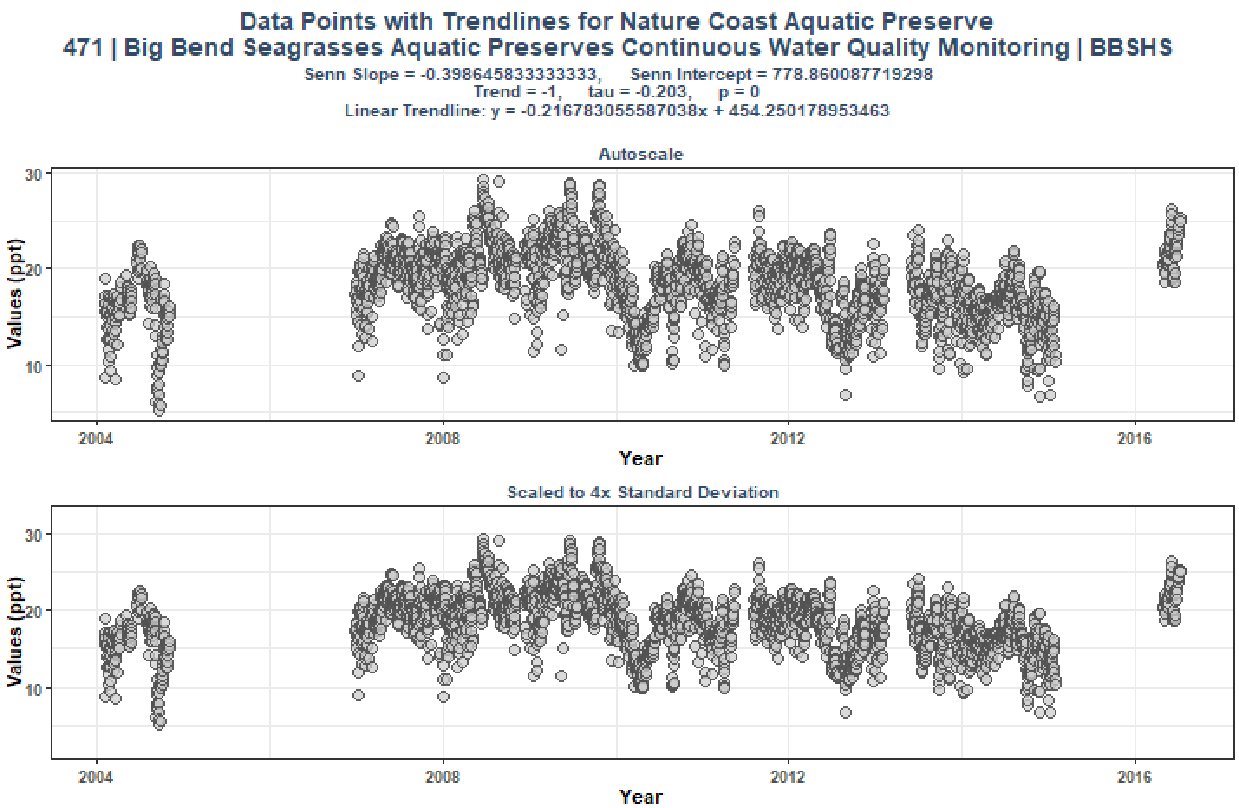

Data Points with Trendlines for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
Senn Slope = 0.0252314814814815, Senn Intercept = -15.7165134803922
Trend = 1, tau = 0.036, p = 0
Linear Trendline: y = 0.0360119194433107x + -63.0324621258694
Autoscale
Values (ppt)
Year
Scaled to 4x Standard Deviation
Values (ppt)
Year

Data Points with Trendlines for Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSSK
Senn Slope = -0.195427693927694, Senn Intercept = 487.988191558982
Trend = -1, tau = -0.1301, p = 0
Linear Trendline: y = -0.112337829680224x + 254.865257950786



Data Points with Trendlines for Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program | P09
Senn Slope = -0.288327943376068, Senn Intercept = 564.543315625
Trend = -1, tau = -0.1812, p = 0.0365
Linear Trendline: y = -0.341635249480331x + 711.473489771031





Appendix IV: Monitoring Location Summary Box Plots

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

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

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

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

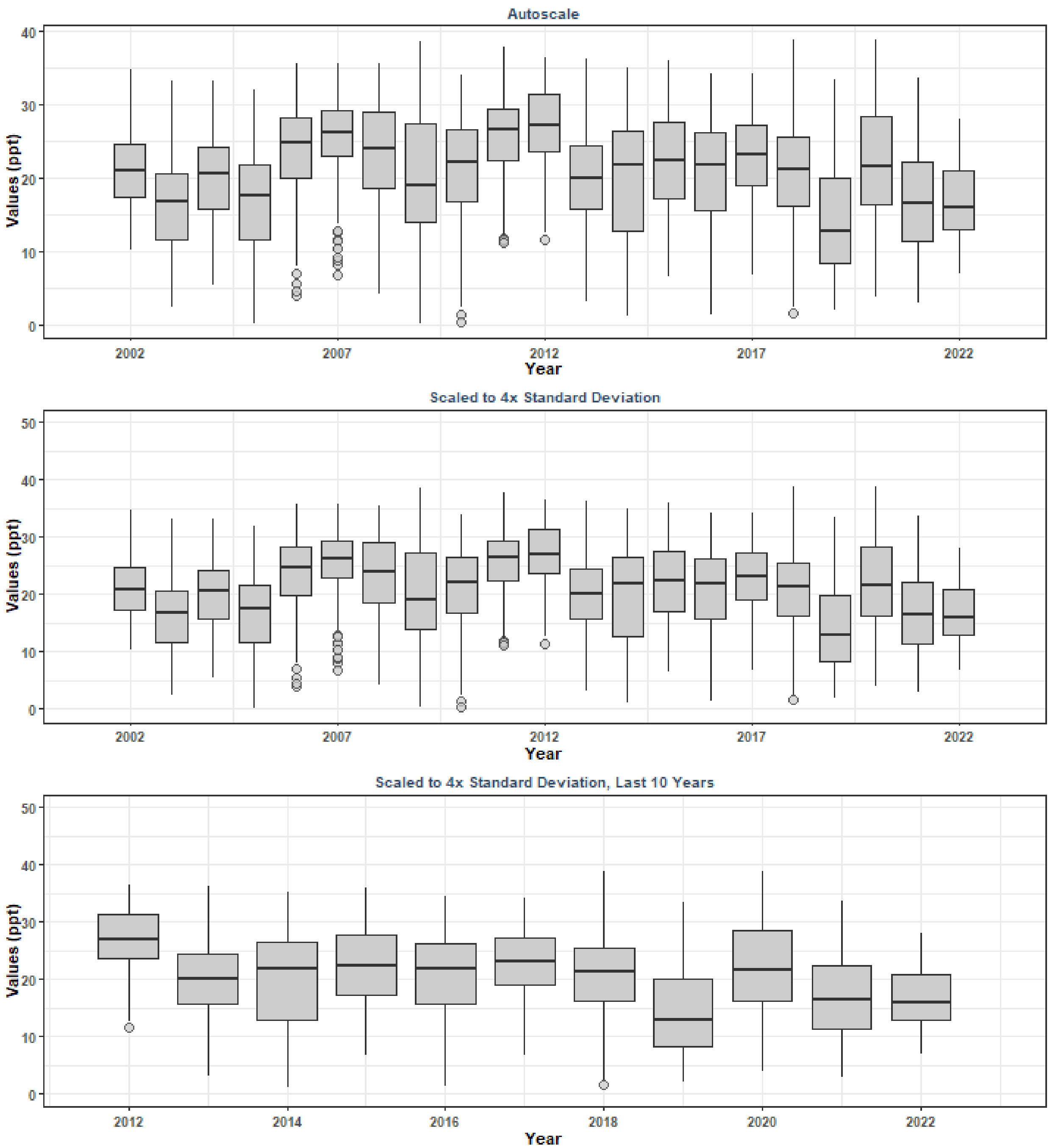
```r
if(n==0){
   print("There are no monitoring locations that qualify.")
} else {
   for (i in 1:n) {
      year_lower <- min(data$Year[data$Use_In_Analysis == TRUE &
                                 data$MonitoringID == Mon_IDs[i]])
      year_upper <- max(data$Year[data$Use_In_Analysis == TRUE &
                                 data$MonitoringID == Mon_IDs[i]])
      min_RV <- min(data$ResultValue[data$Use_In_Analysis == TRUE &
                                    data$MonitoringID == Mon_IDs[i]])
      mn_RV <- mean(data$ResultValue[data$Use_In_Analysis == TRUE &
                                    data$MonitoringID == Mon_IDs[i] &
                                    data$ResultValue <
                                    quantile(data$ResultValue, 0.98)])
      sd_RV <- sd(data$ResultValue[data$Use_In_Analysis == TRUE &
                                  data$MonitoringID == Mon_IDs[i] &
                                  data$ResultValue <
                                  quantile(data$ResultValue, 0.98)])
      x_scale <- ifelse(year_upper - year_lower > 30, 10, 5)
      y_scale <- mn_RV + 4 * sd_RV
      MA_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID == Mon_IDs[i]]
      Mon_name <- paste(KT.Stats$ProgramID[KT.Stats$MonitoringID == Mon_IDs[i]],
                        KT.Stats$ProgramName[KT.Stats$MonitoringID == Mon_IDs[i]],
                        KT.Stats$ProgramLocationID[KT.Stats$MonitoringID == Mon_IDs[i]],
                        sep = " | ")

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

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

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

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

p0 <- ggplot() + labs(title = paste0("Summary Box Plots for ",
                                     MA_name, "\n", Mon_name),
                      subtitle = "By Year") +
  plot_theme + theme(panel.border = element_blank(),
                     panel.grid.major = element_blank(),
                     panel.grid.minor = element_blank(),
                     axis.line = element_blank())


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

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

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

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_name, "\n", Mon_name),
                       subtitle = "By Year & Month") + plot_theme +
   theme(panel.border = element_blank(),
         panel.grid.major = element_blank(),
         panel.grid.minor = element_blank(), axis.line = element_blank())

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

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

p9 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                            data$MonitoringID == Mon_IDs[i] &
```

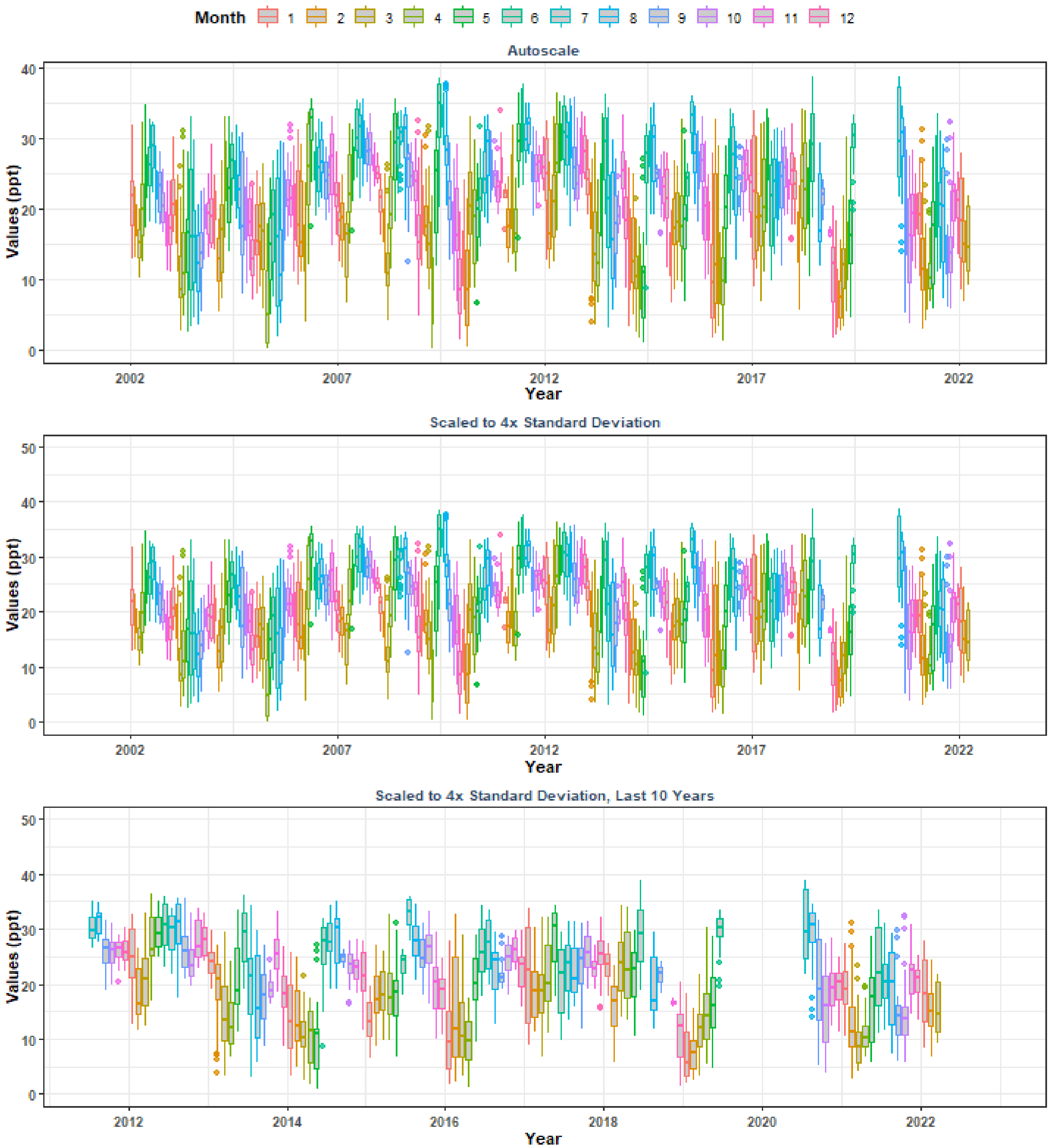
```
                                  data$Year >= year_upper - 10, ],
                   aes(x = Month, y = ResultValue,
                       group = Month, fill = as.factor(Month))) +
         geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
                      outlier.color="#333333",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

      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_name, "\n", Mon_name),
                              subtitle = "By Month") + plot_theme +
         theme(panel.border = element_blank(),
               panel.grid.major = element_blank(),
               panel.grid.minor = element_blank(), axis.line = element_blank())

      print(ggarrange(p0, Yset, ncol = 1, heights = c(0.1, 1)))
      print(ggarrange(p00, YMset, ncol = 1, heights = c(0.1, 1)))
      print(ggarrange(p000, Mset, ncol = 1, heights = c(0.1, 1)))

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

Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apadbwq
By Year

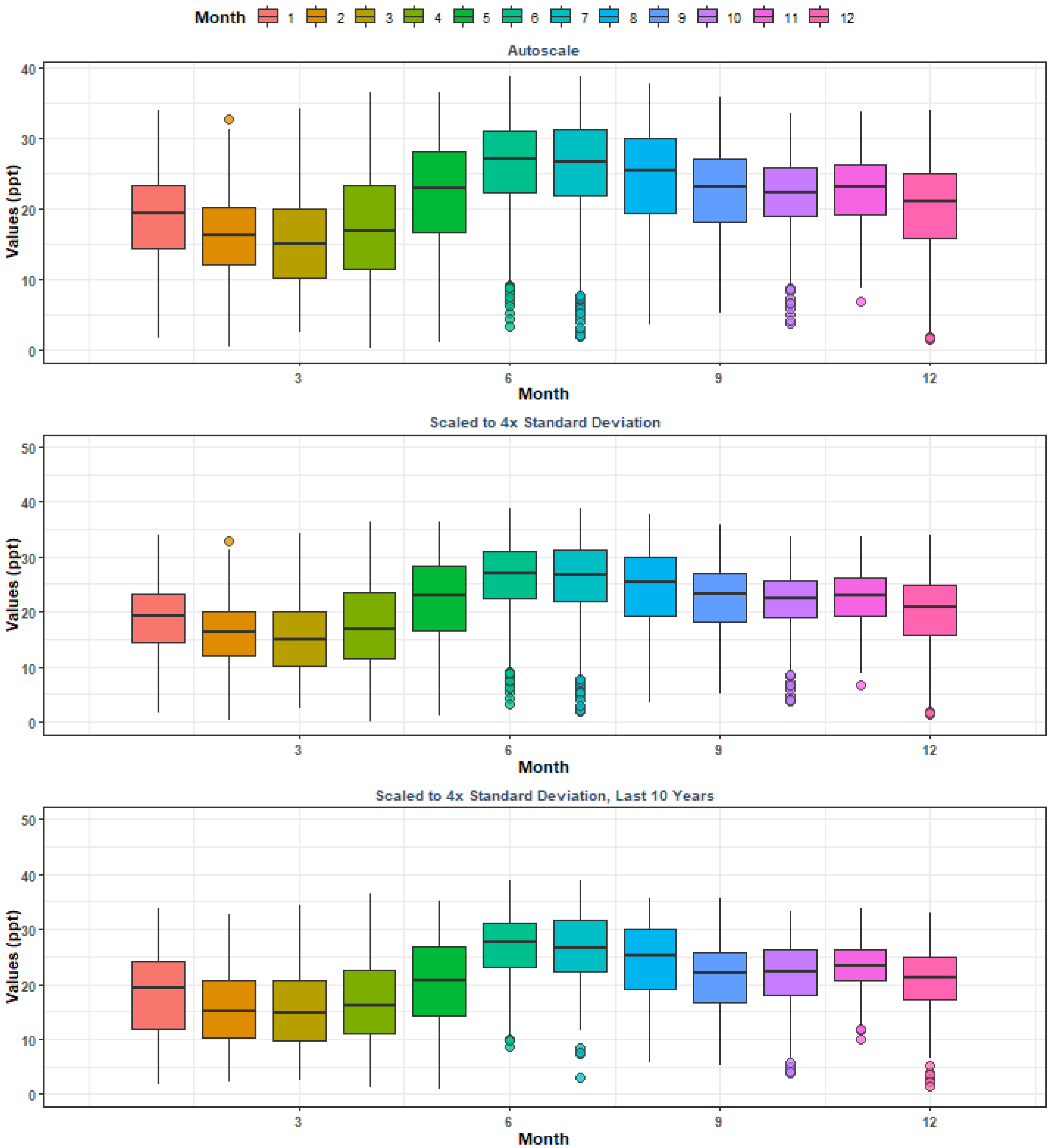


Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apadbwq

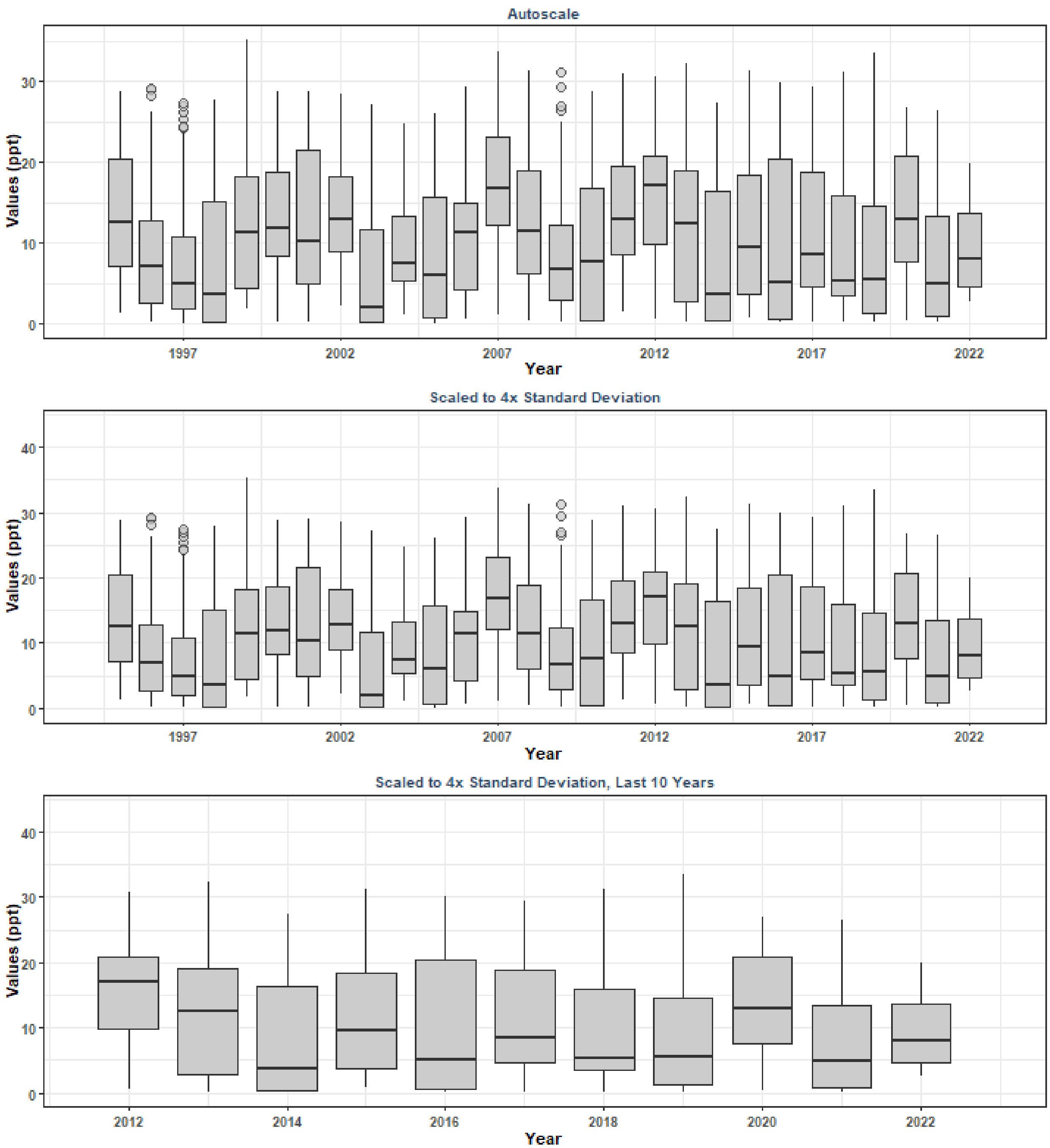
By Year & Month



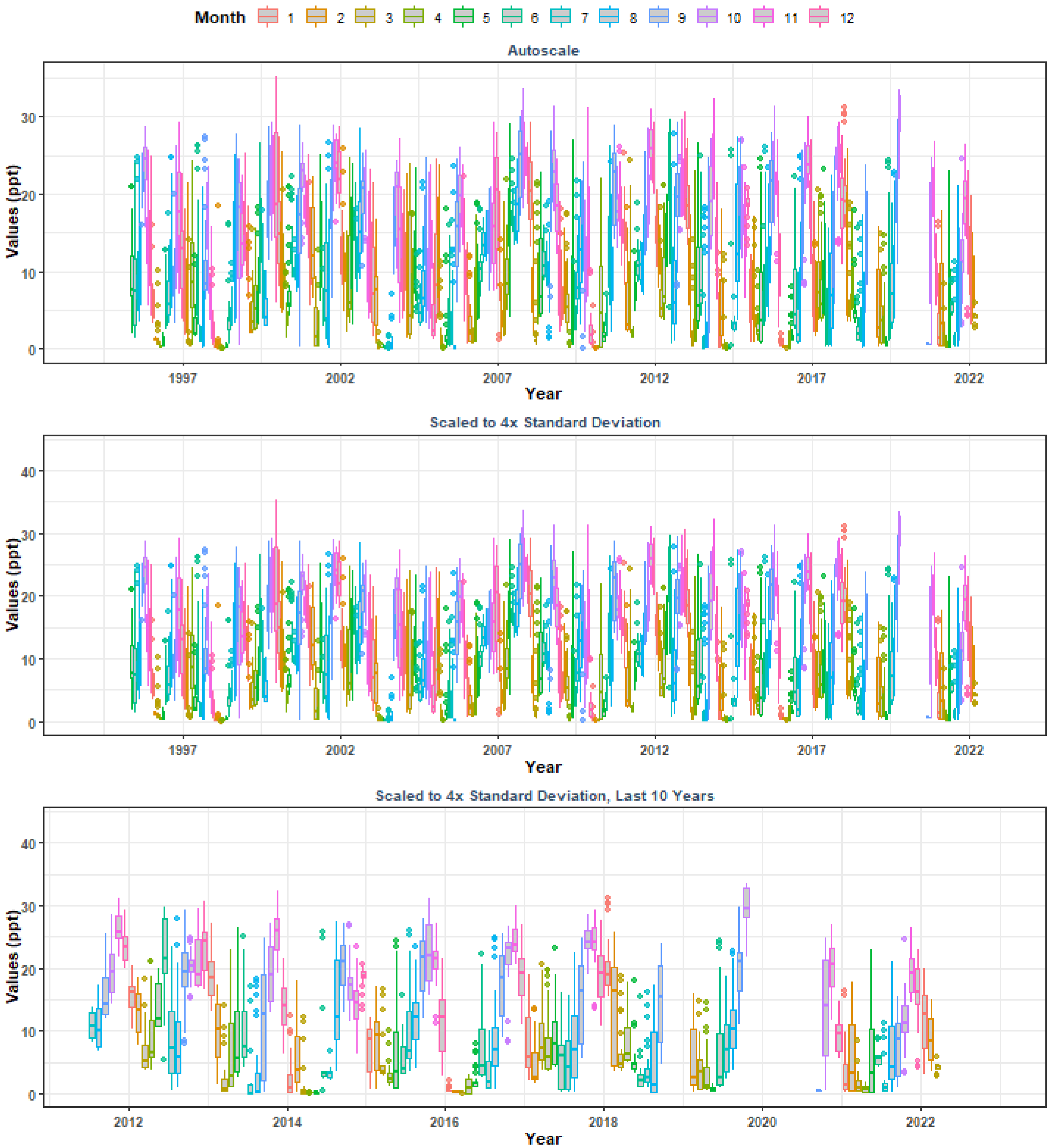
Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apadbwq
By Month



Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq
By Year

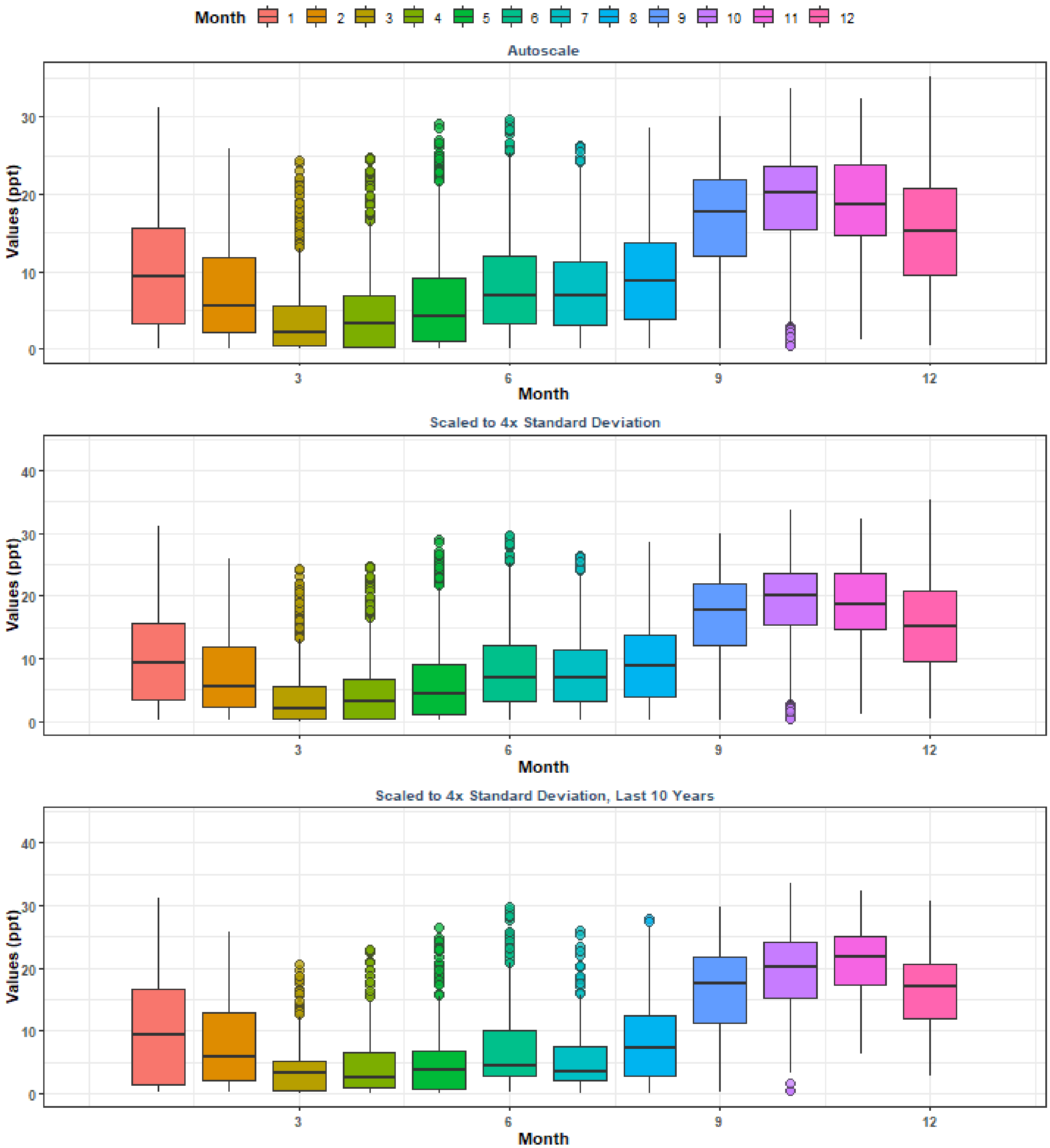


Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq
By Year & Month

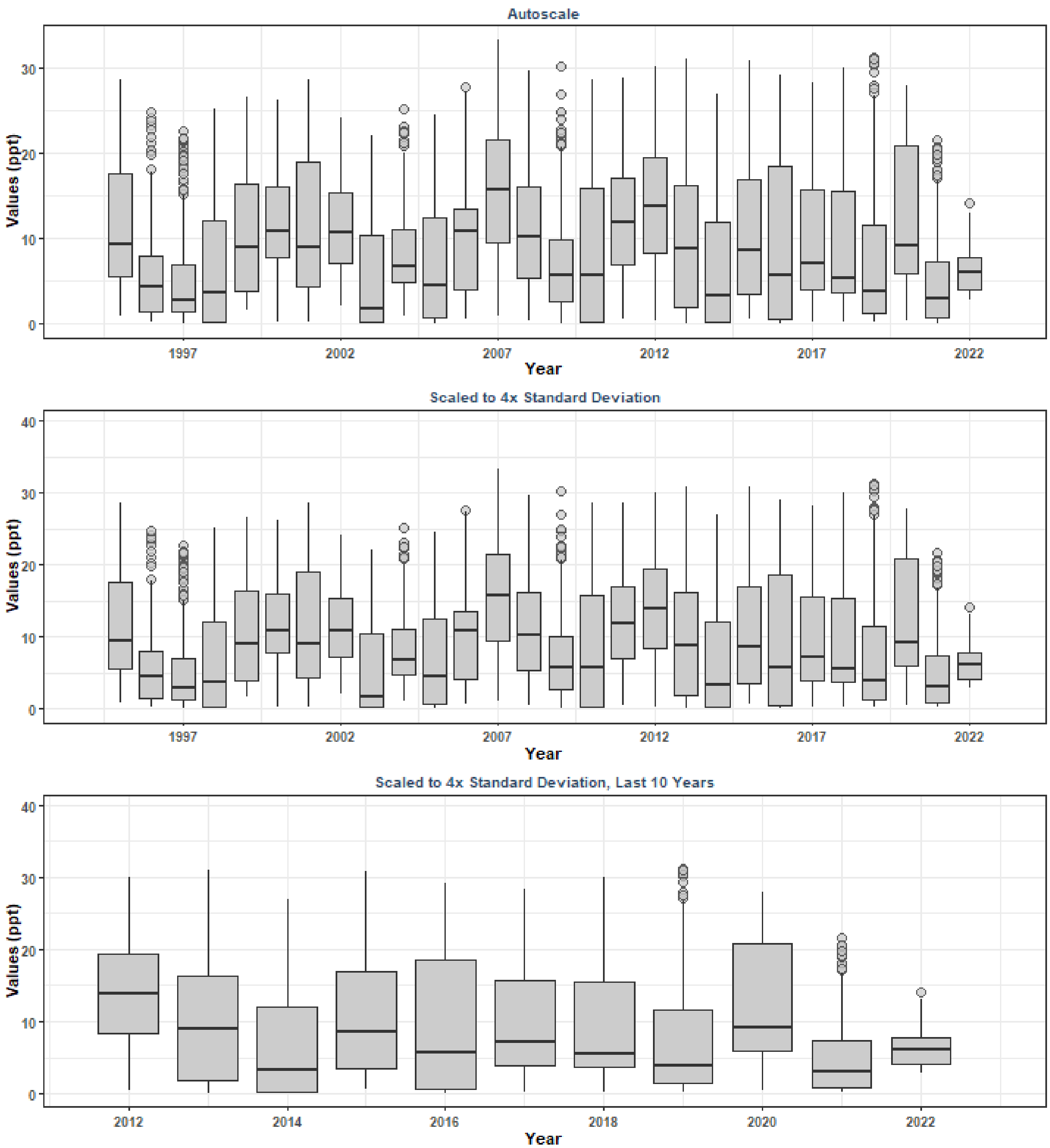


Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq

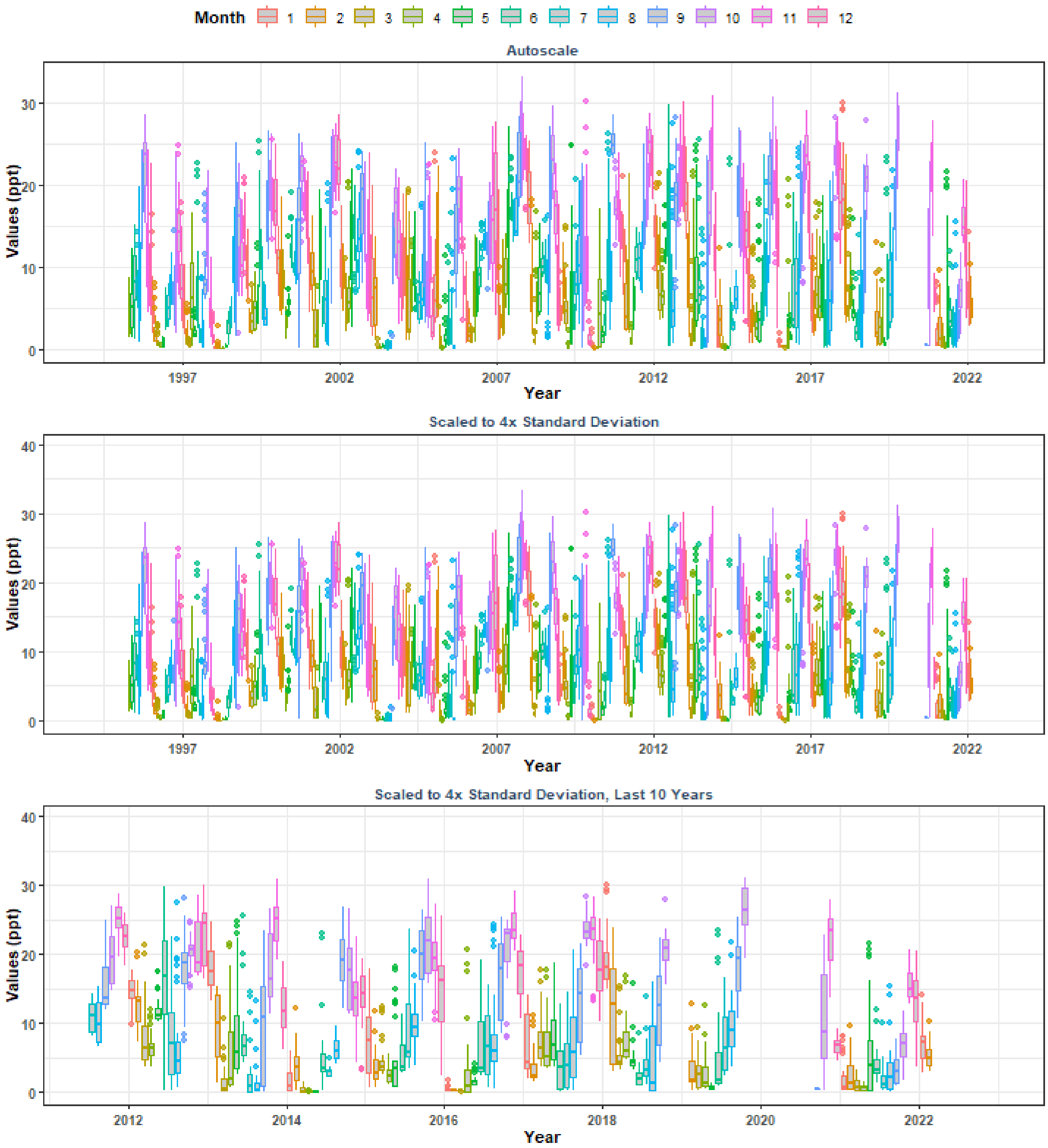
By Month



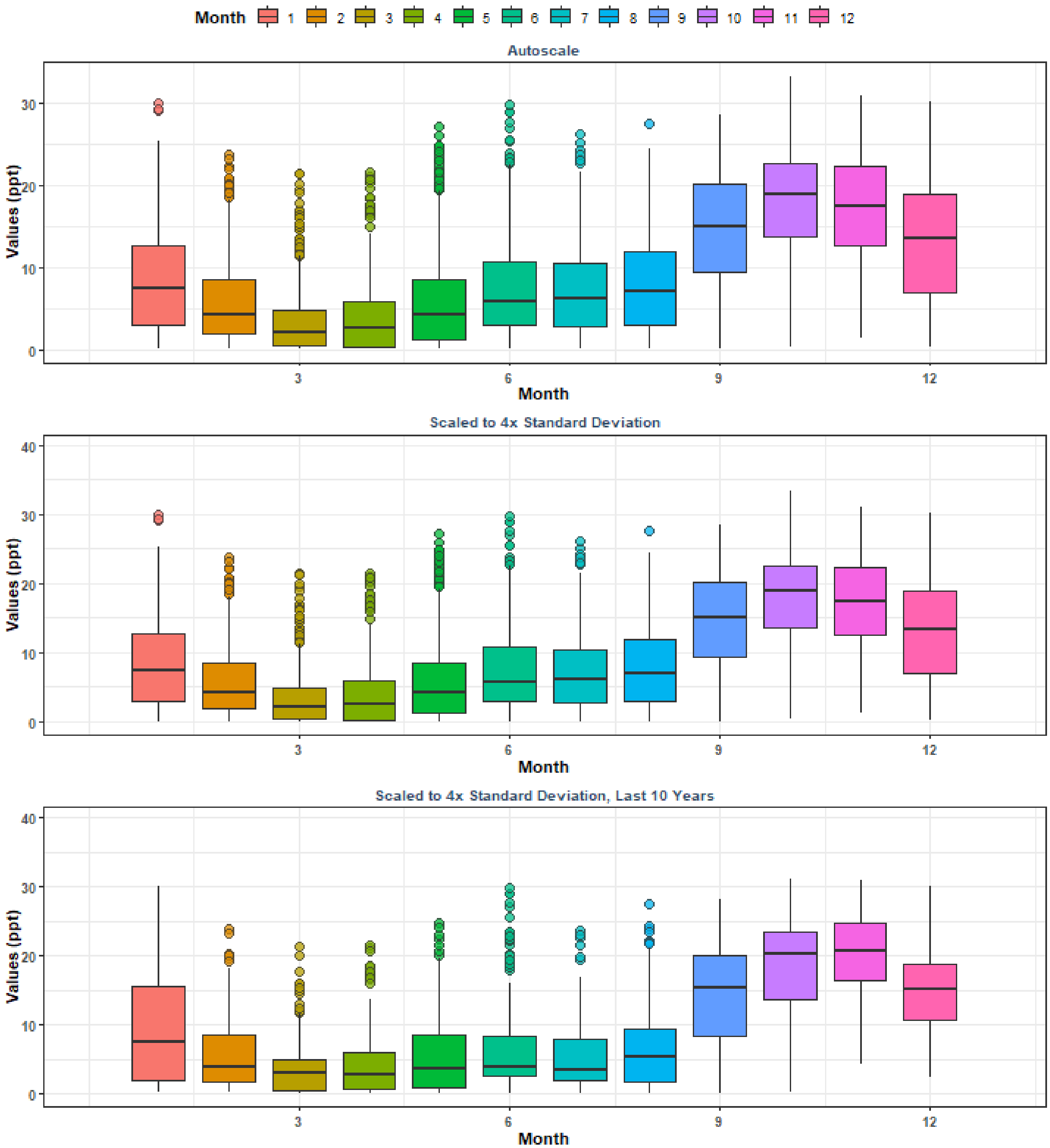
Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
By Year



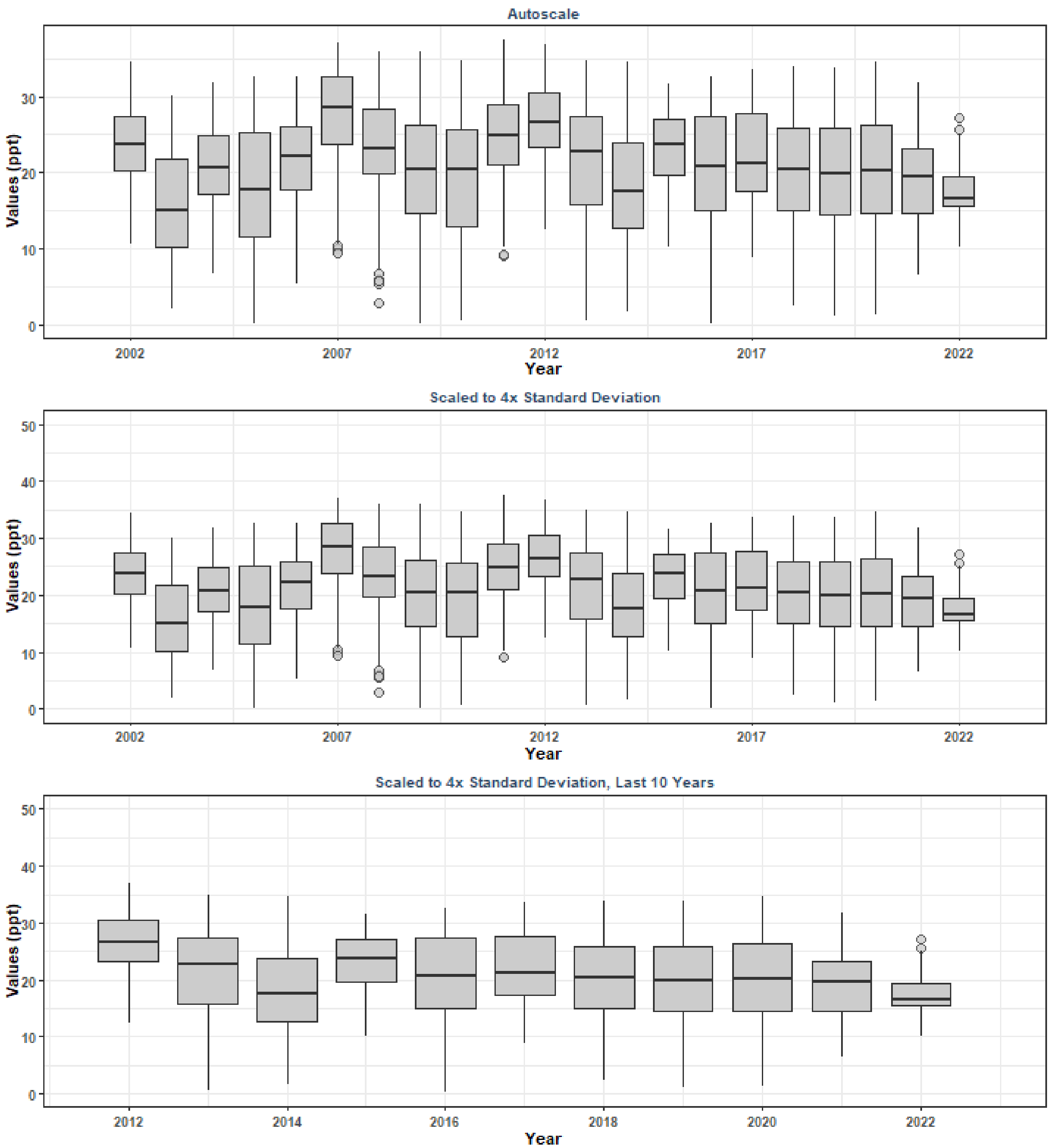
Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
By Year & Month



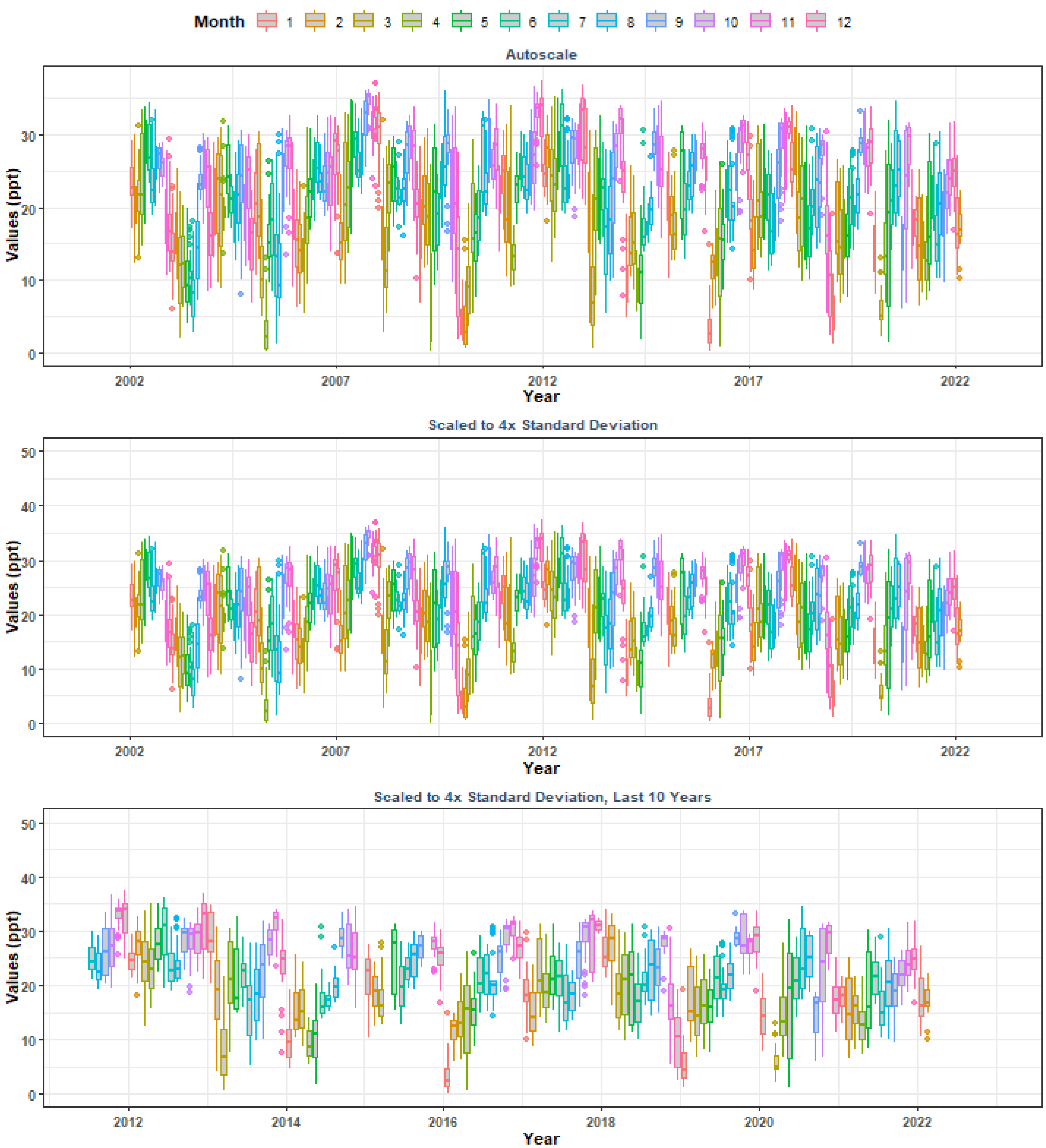
Summary Box Plots for Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
By Month



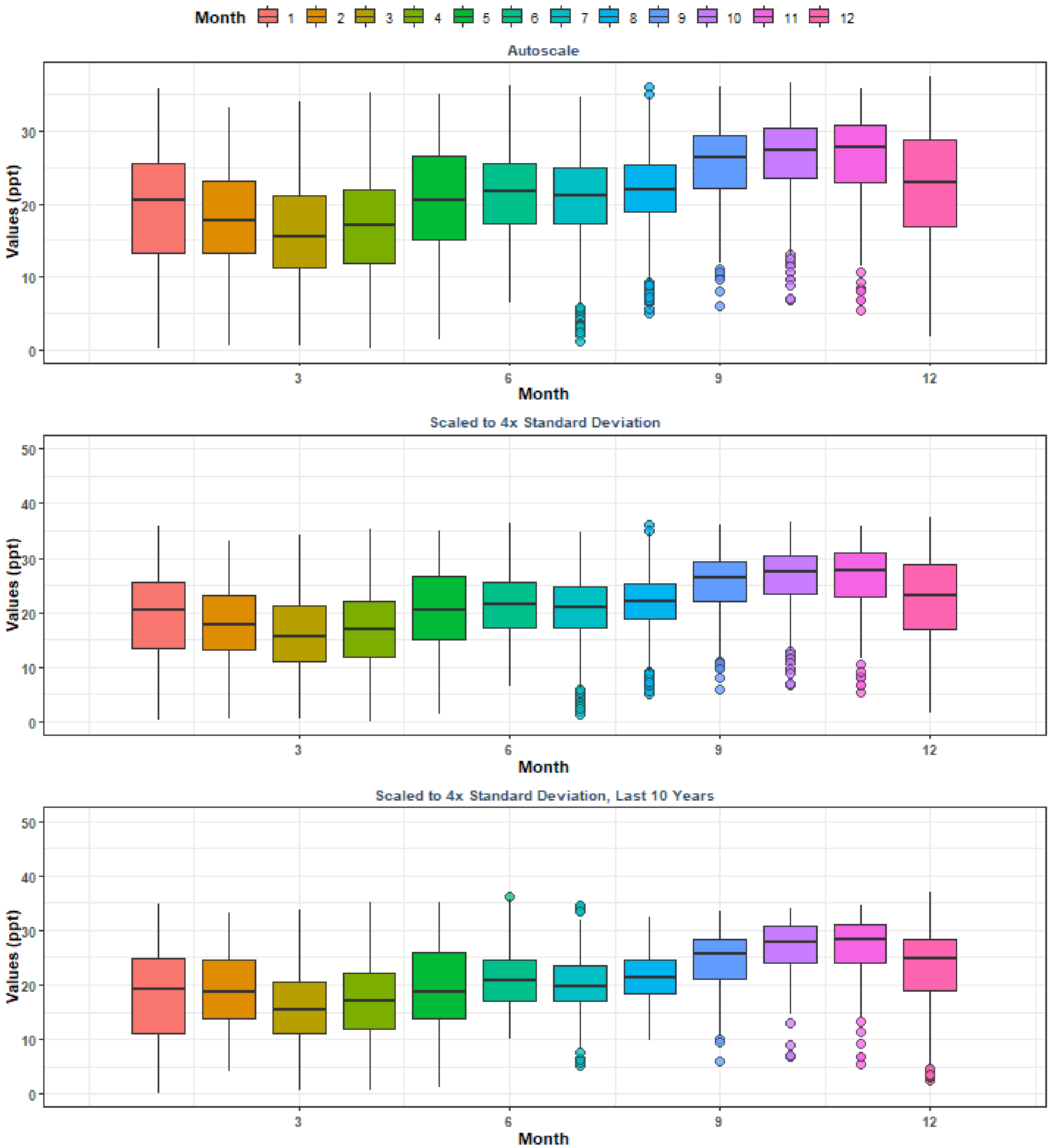
Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apacpwq
By Year



Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apacpwq
By Year & Month

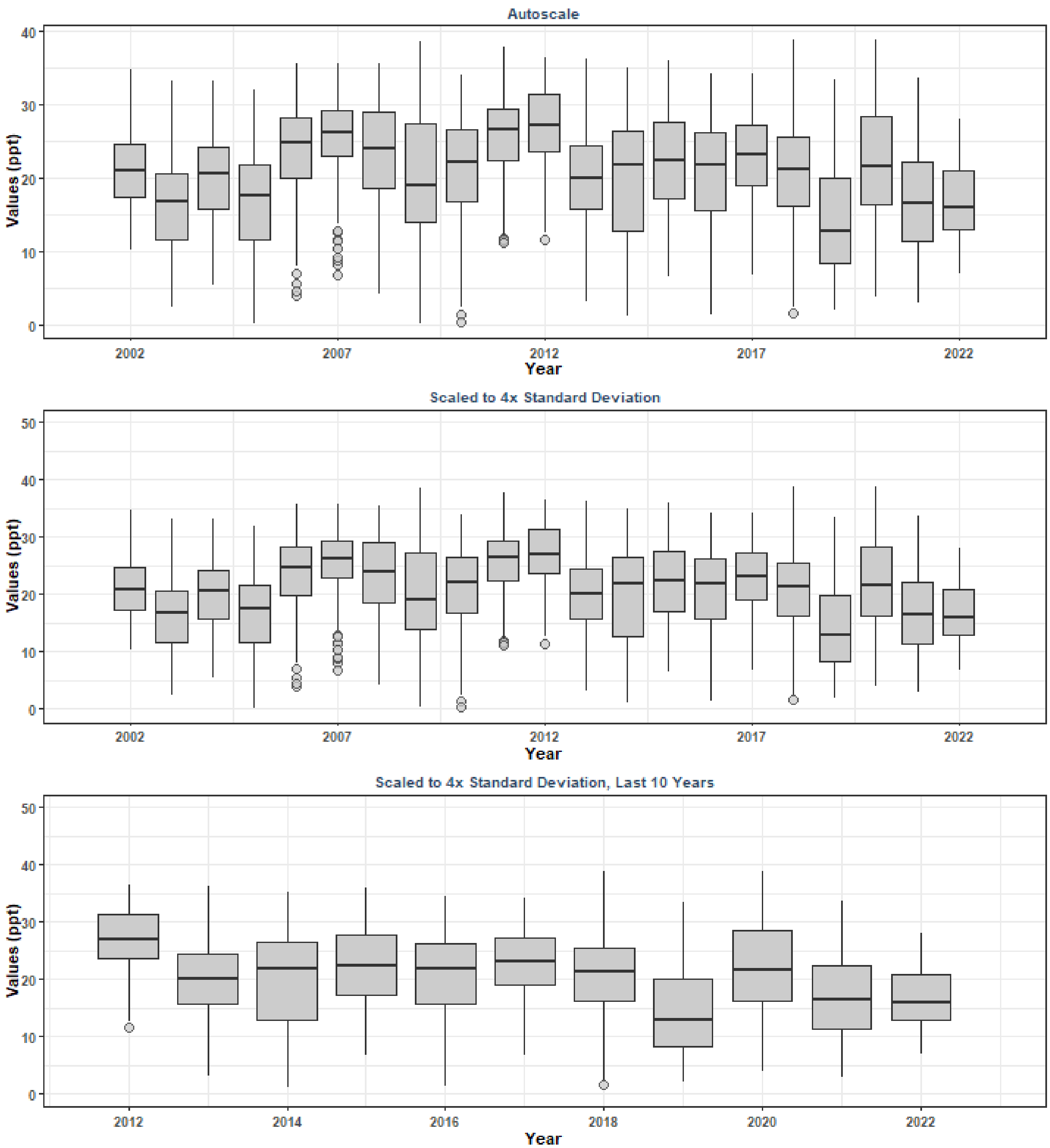


Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apacpwq
By Month

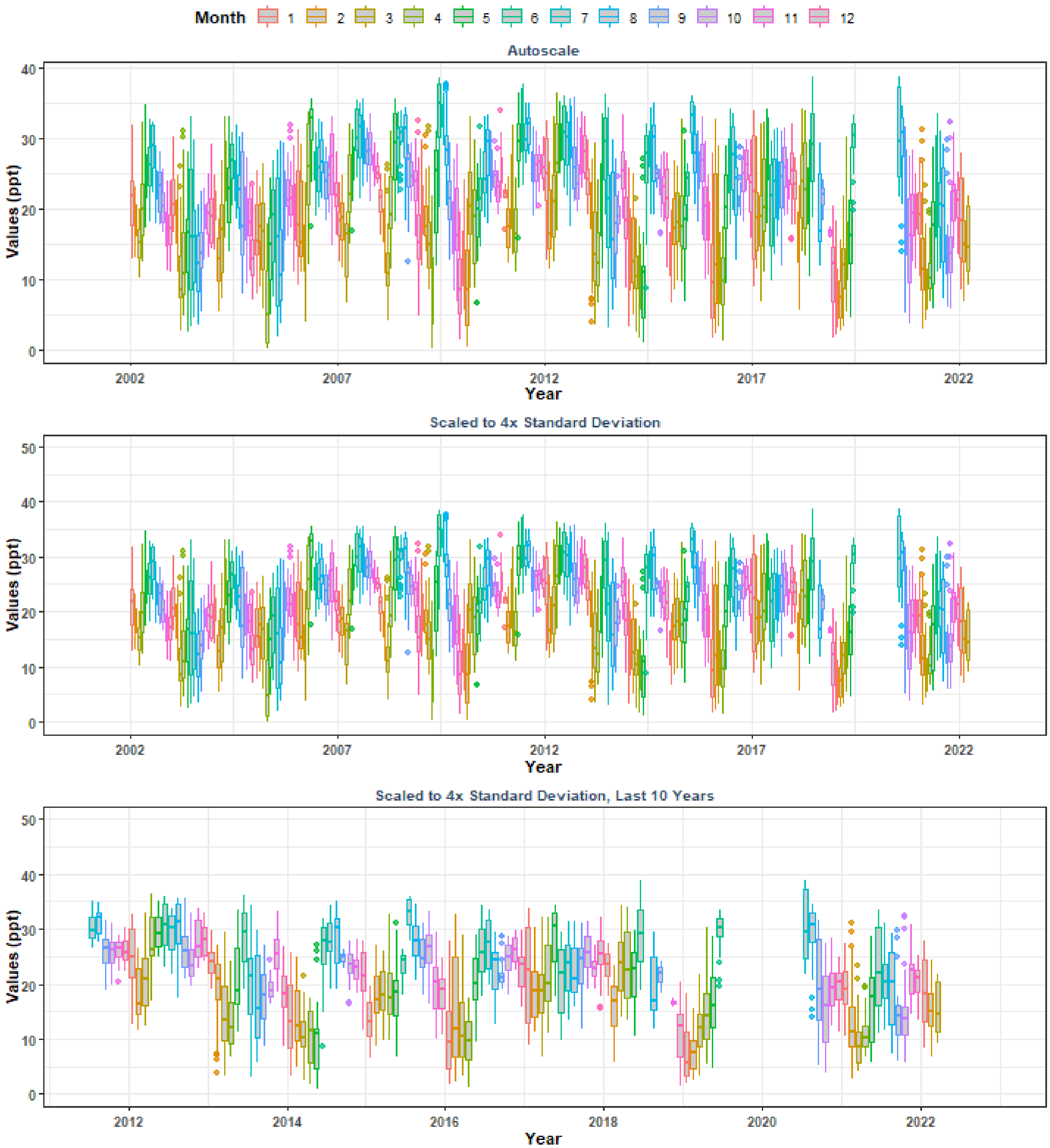


Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apadbwq

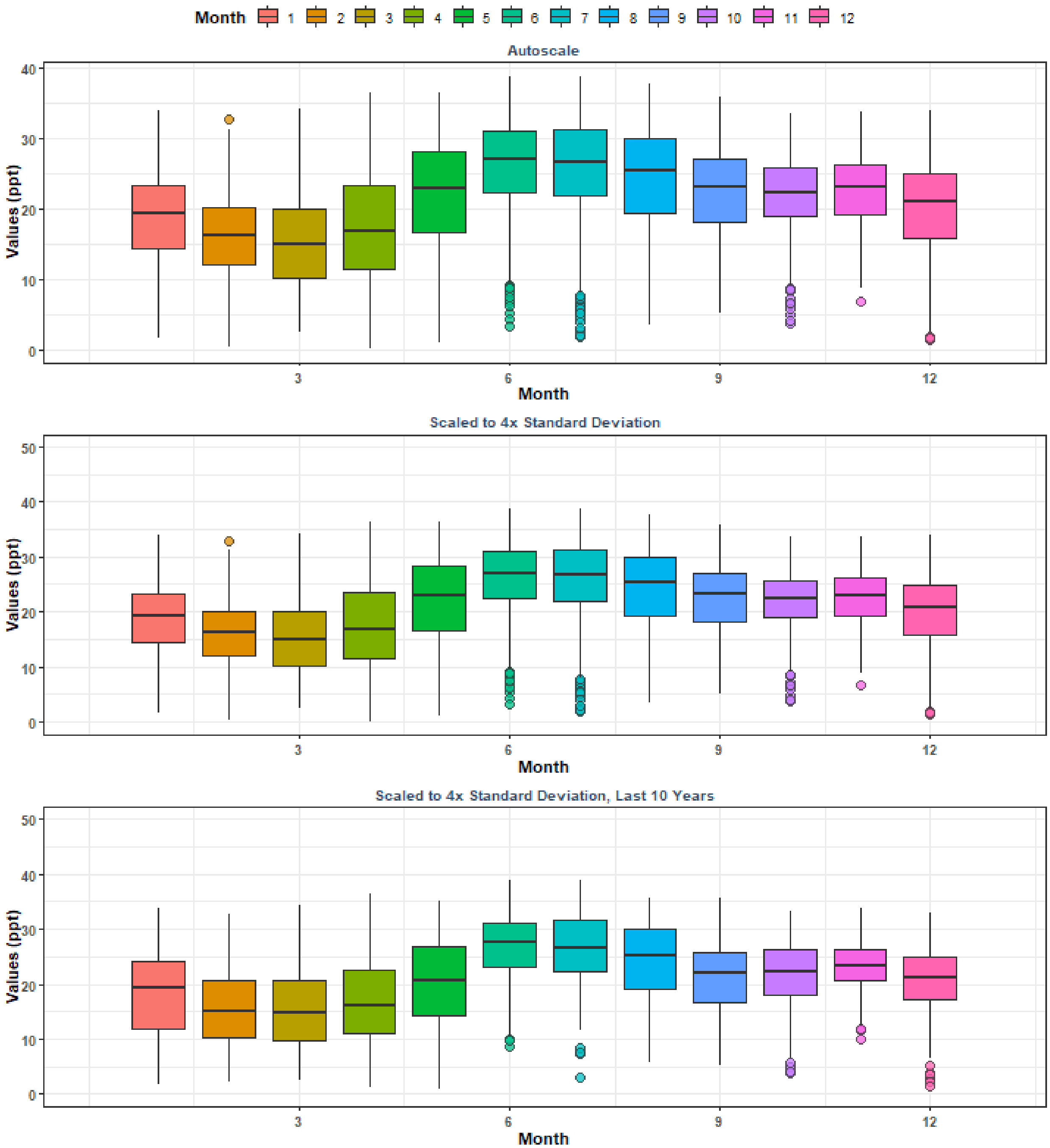
By Year



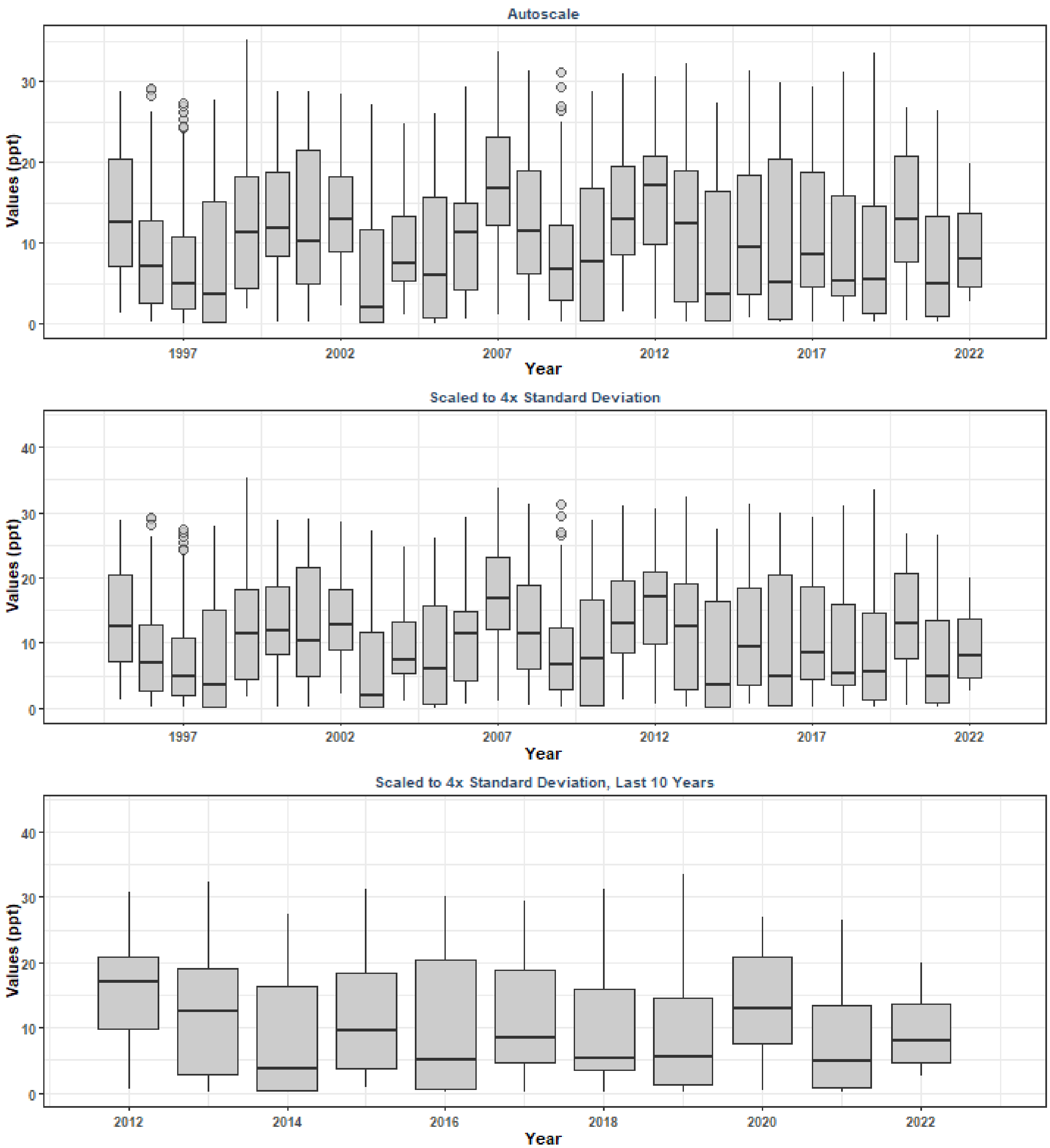
Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apadbwq
By Year & Month

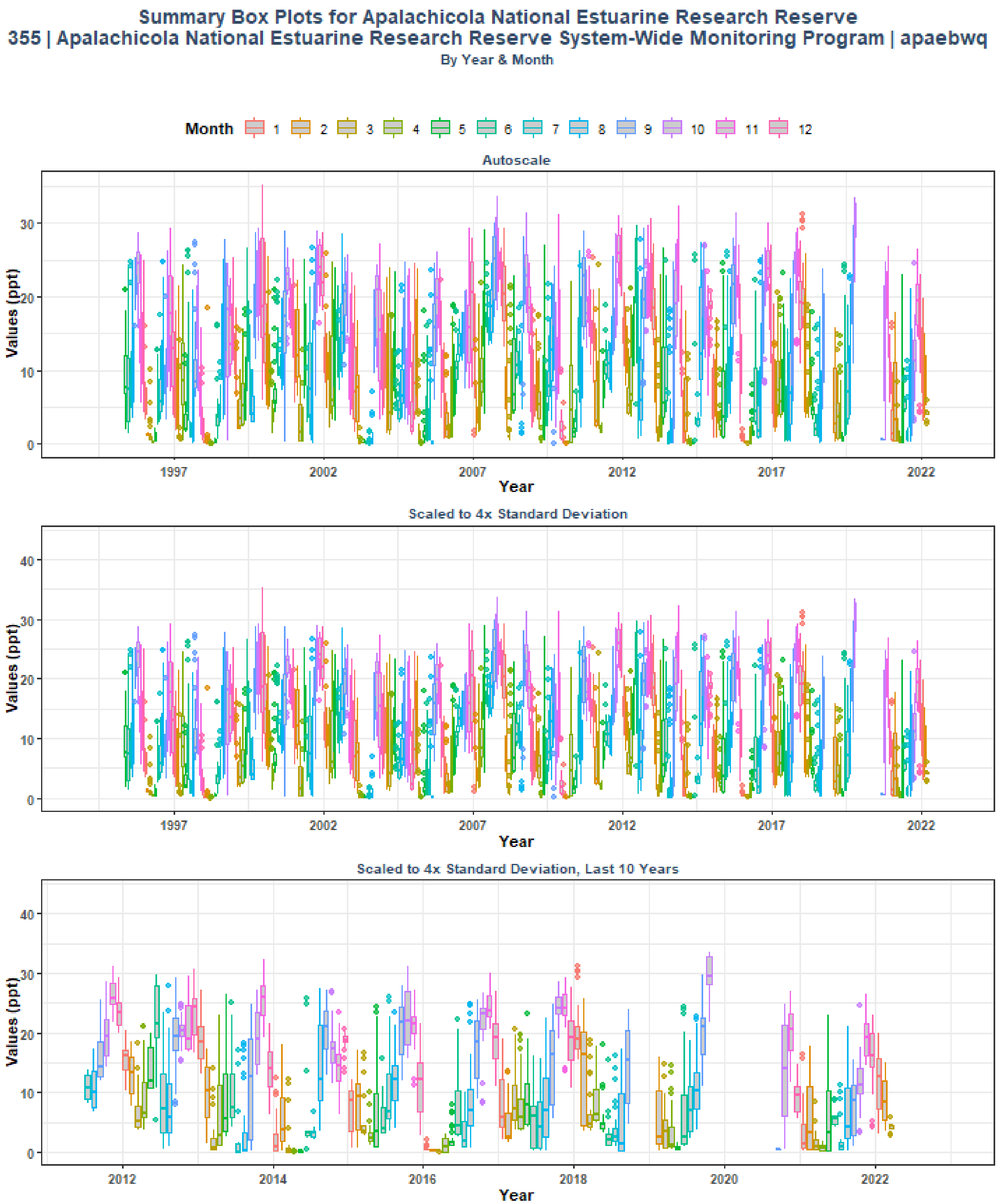


Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apadbwq
By Month

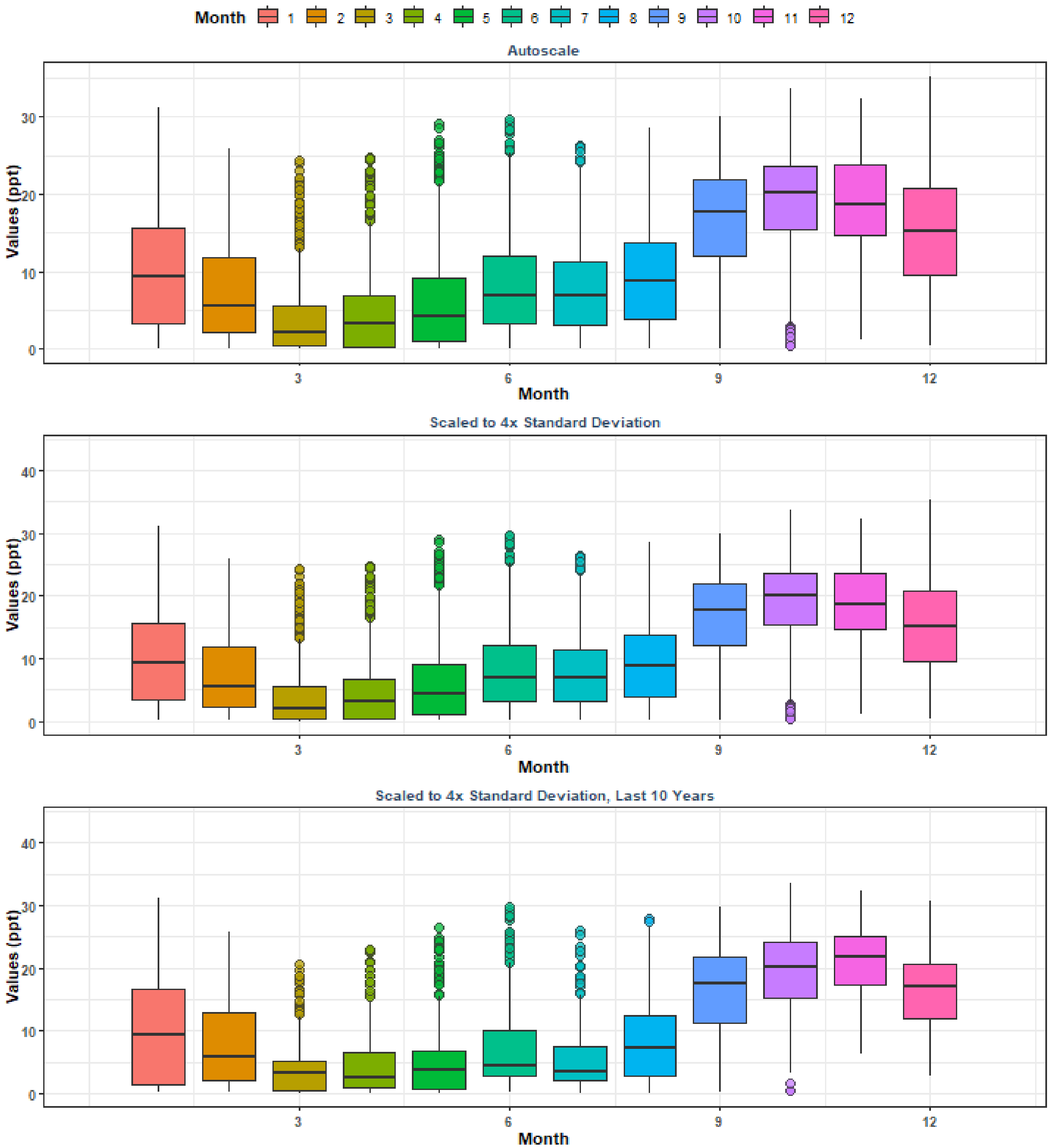


Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq
By Year



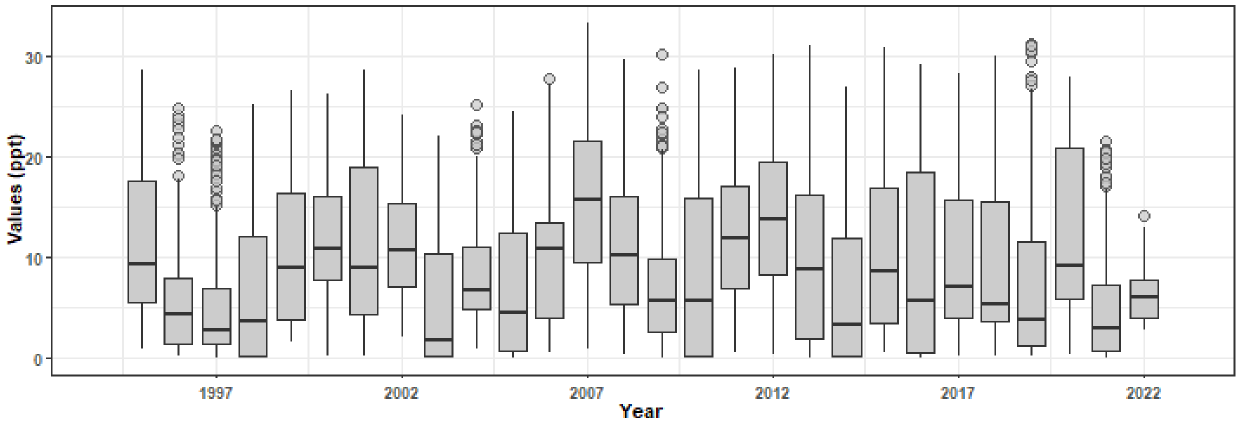

Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (ppt)
Year
Scaled to 4x Standard Deviation
Values (ppt)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (ppt)
Year

Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaebwq
By Month

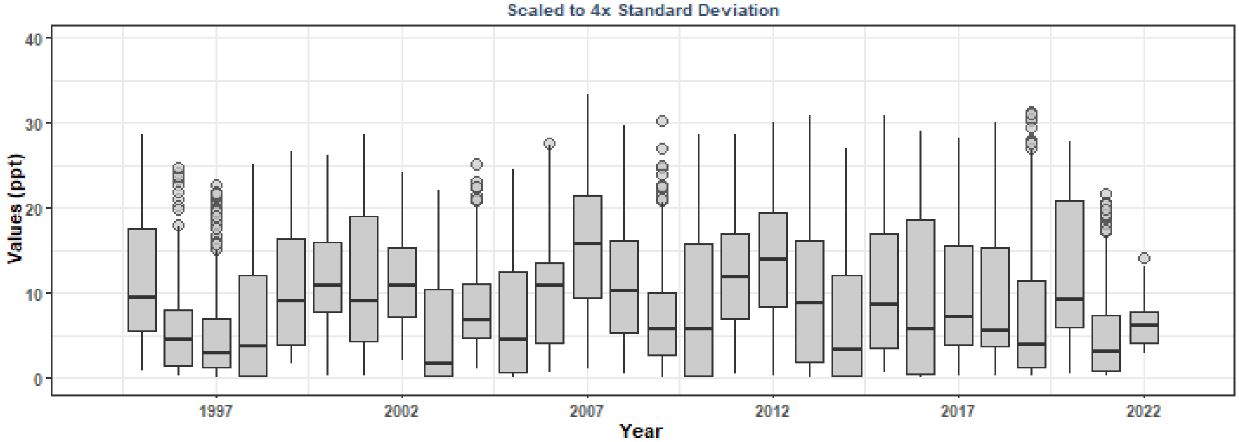


Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
By Year

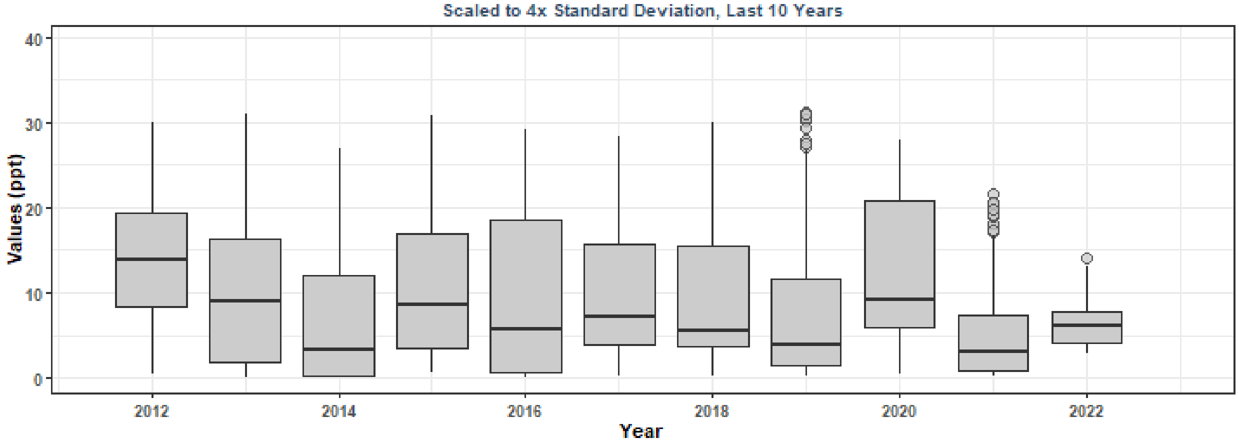
Autoscale



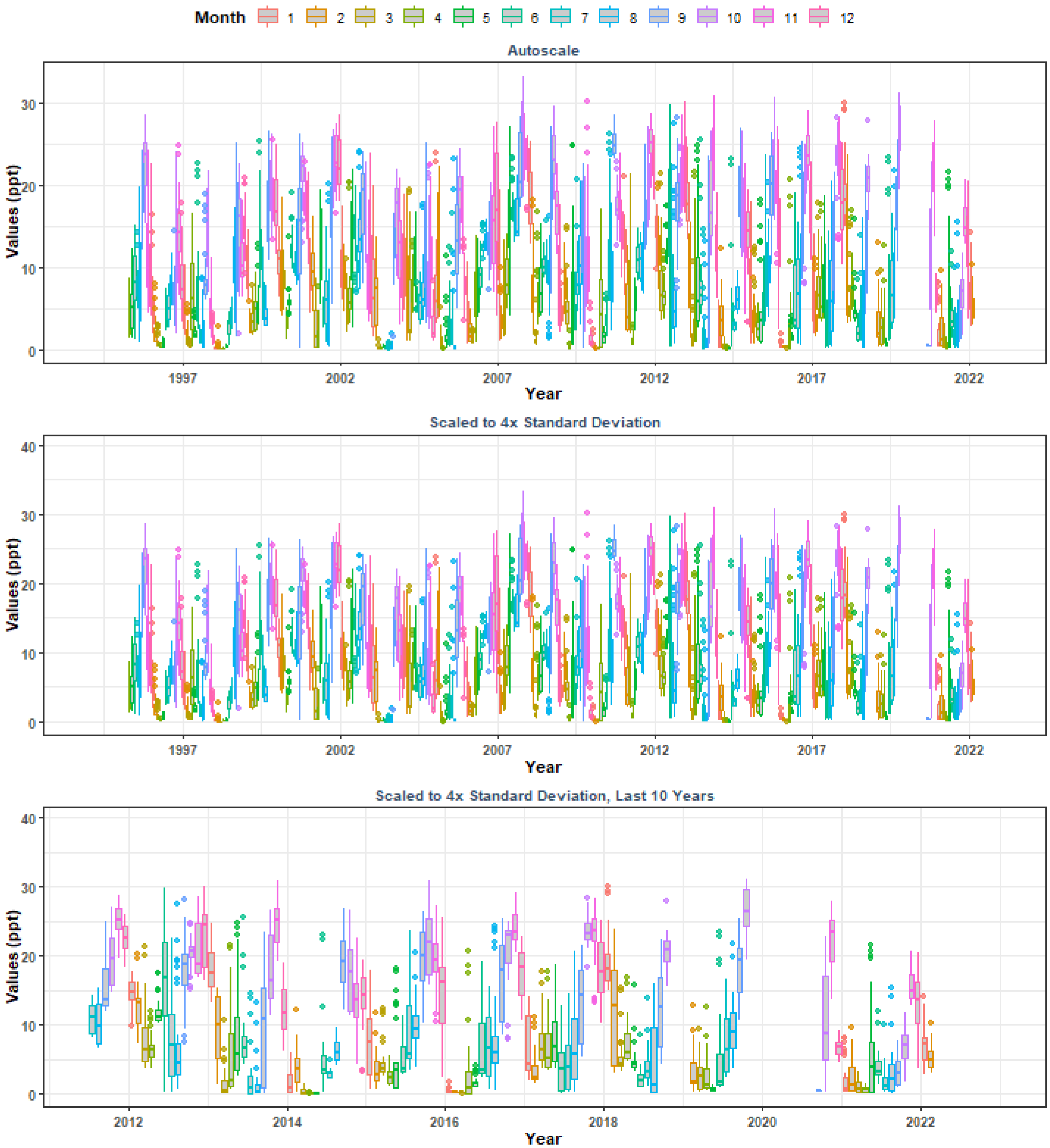
Scaled to 4x Standard Deviation



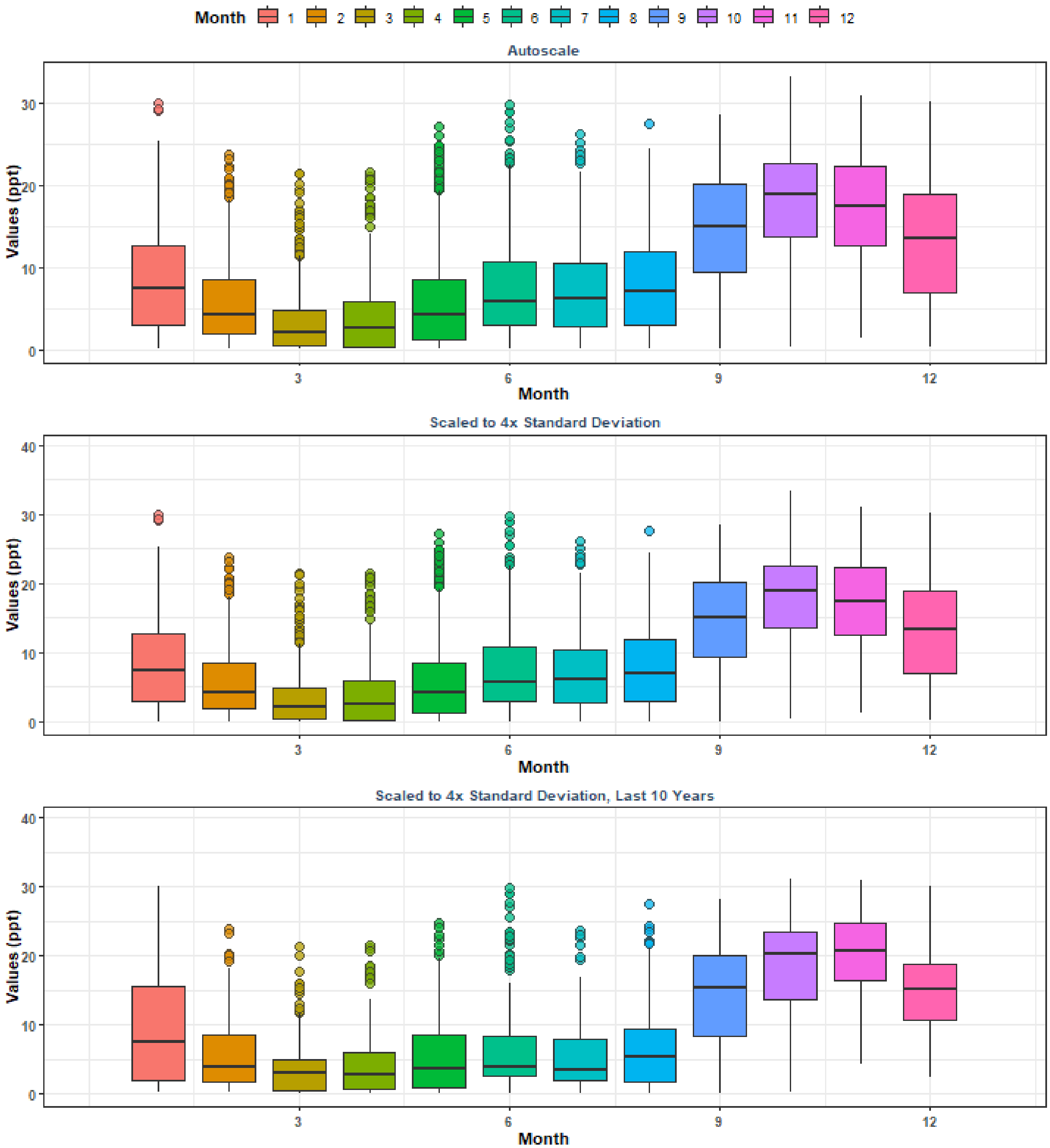
Scaled to 4x Standard Deviation, Last 10 Years



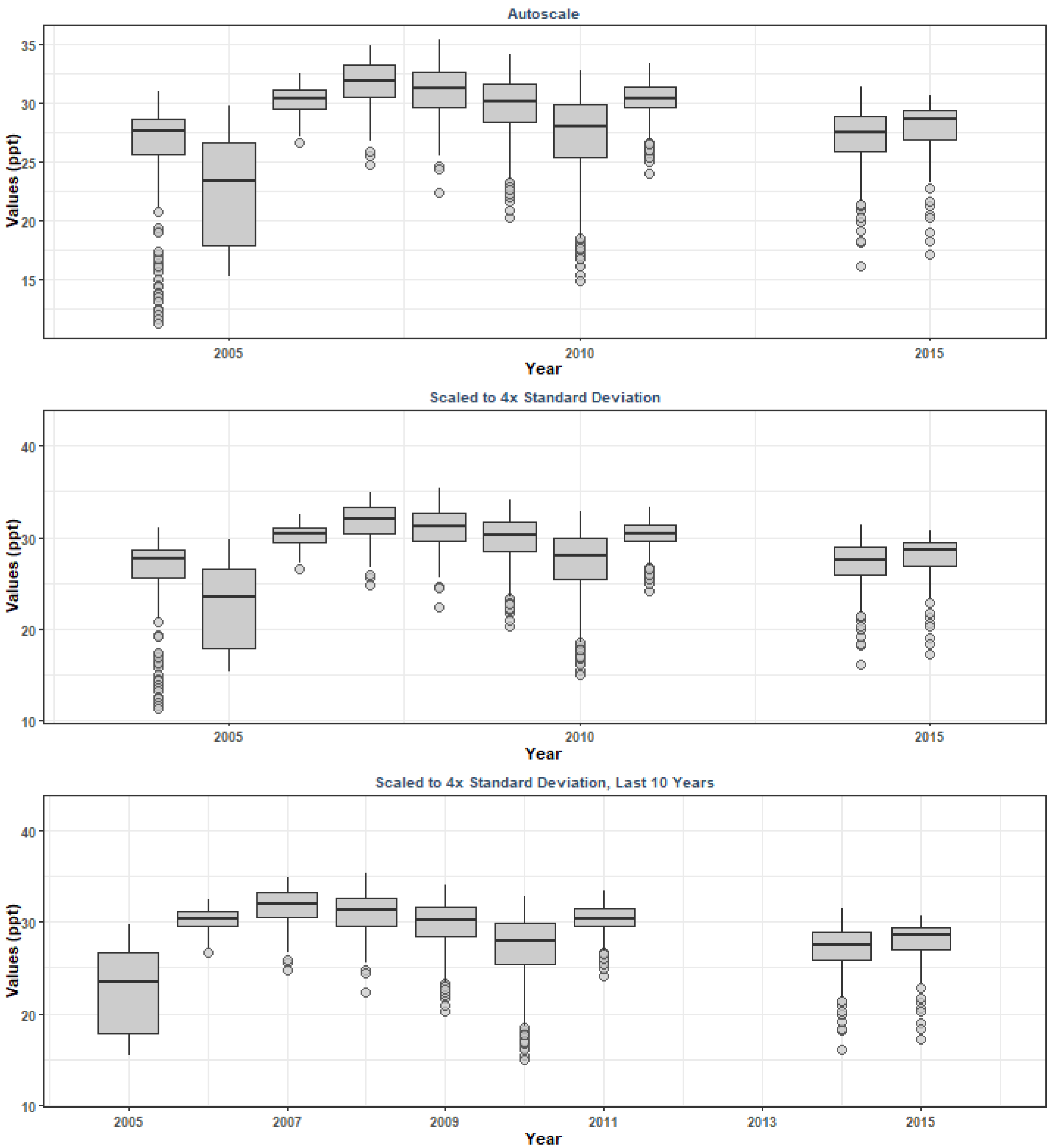
Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
By Year & Month



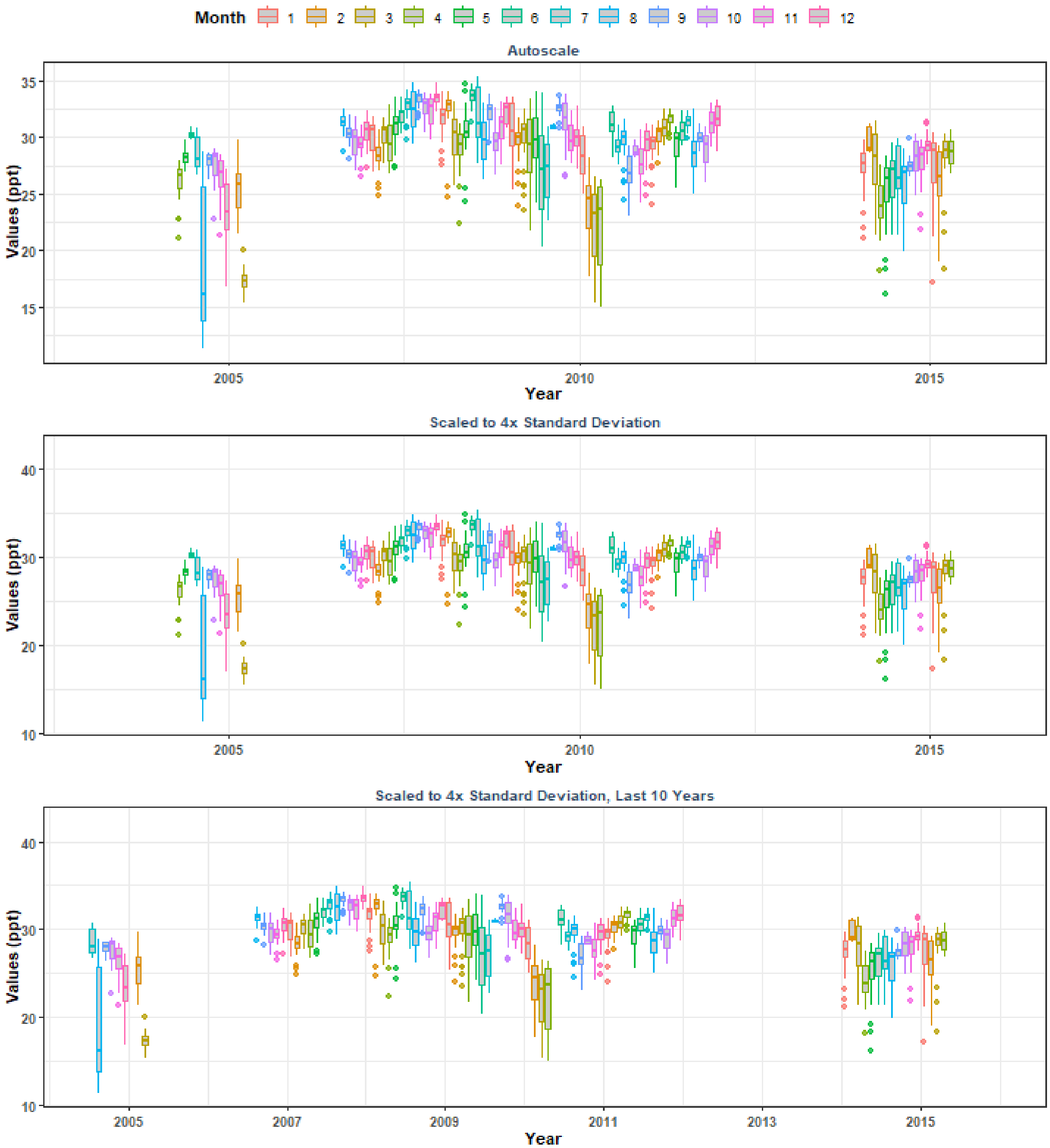
Summary Box Plots for Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program | apaeswq
By Month



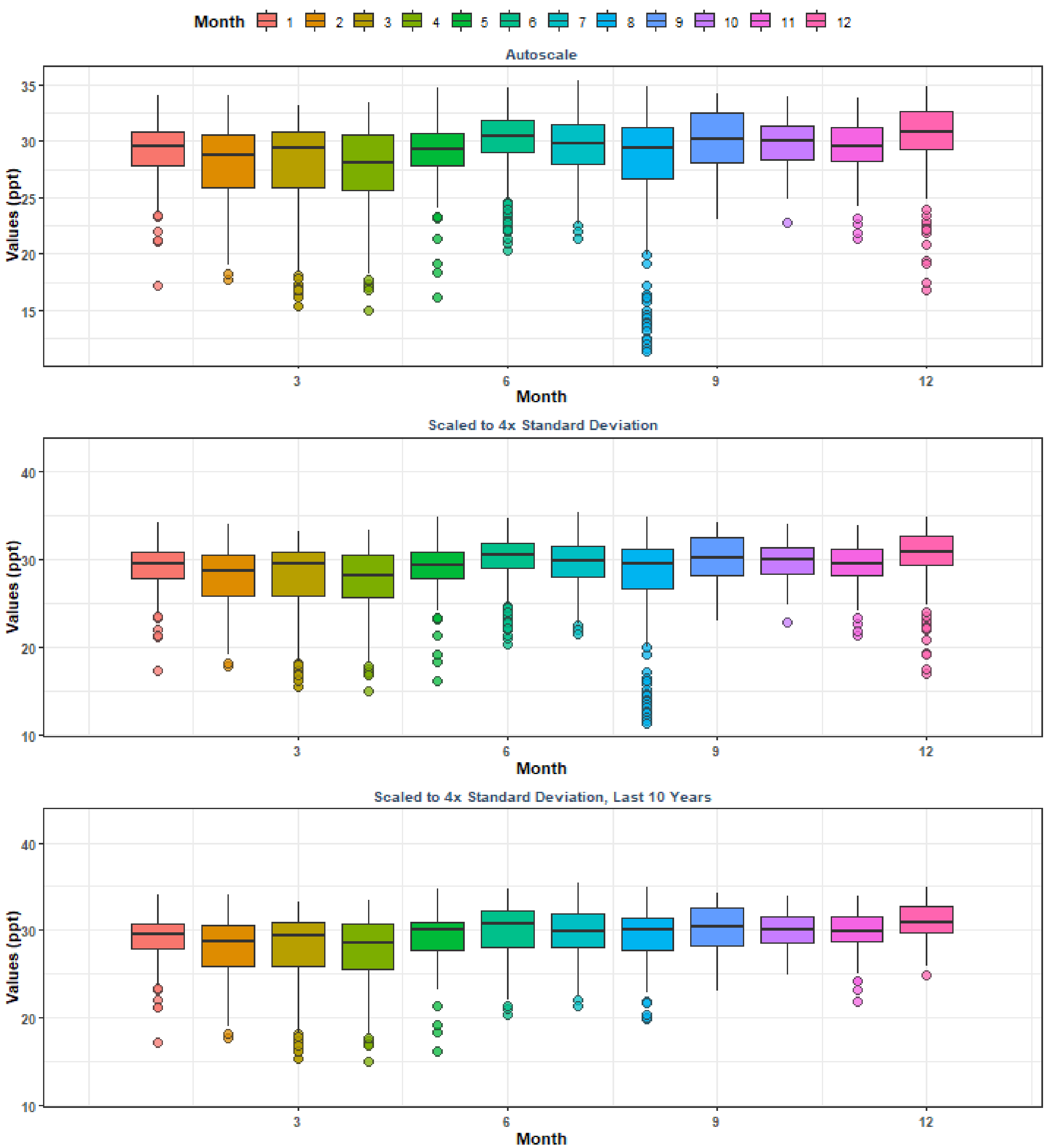
Summary Box Plots for Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSSK
By Year



Summary Box Plots for Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSSK
By Year & Month

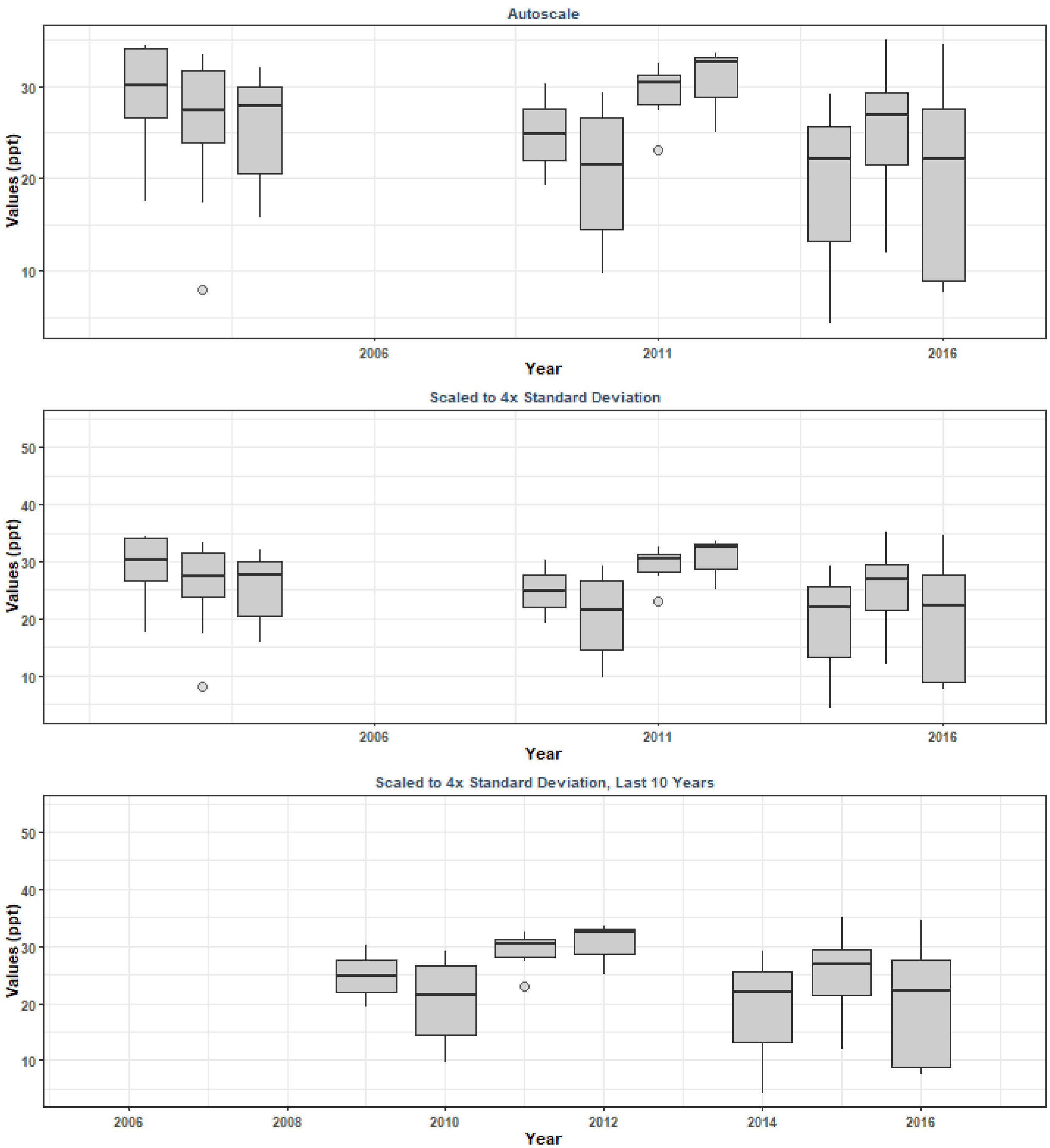


Summary Box Plots for Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSSK
By Month



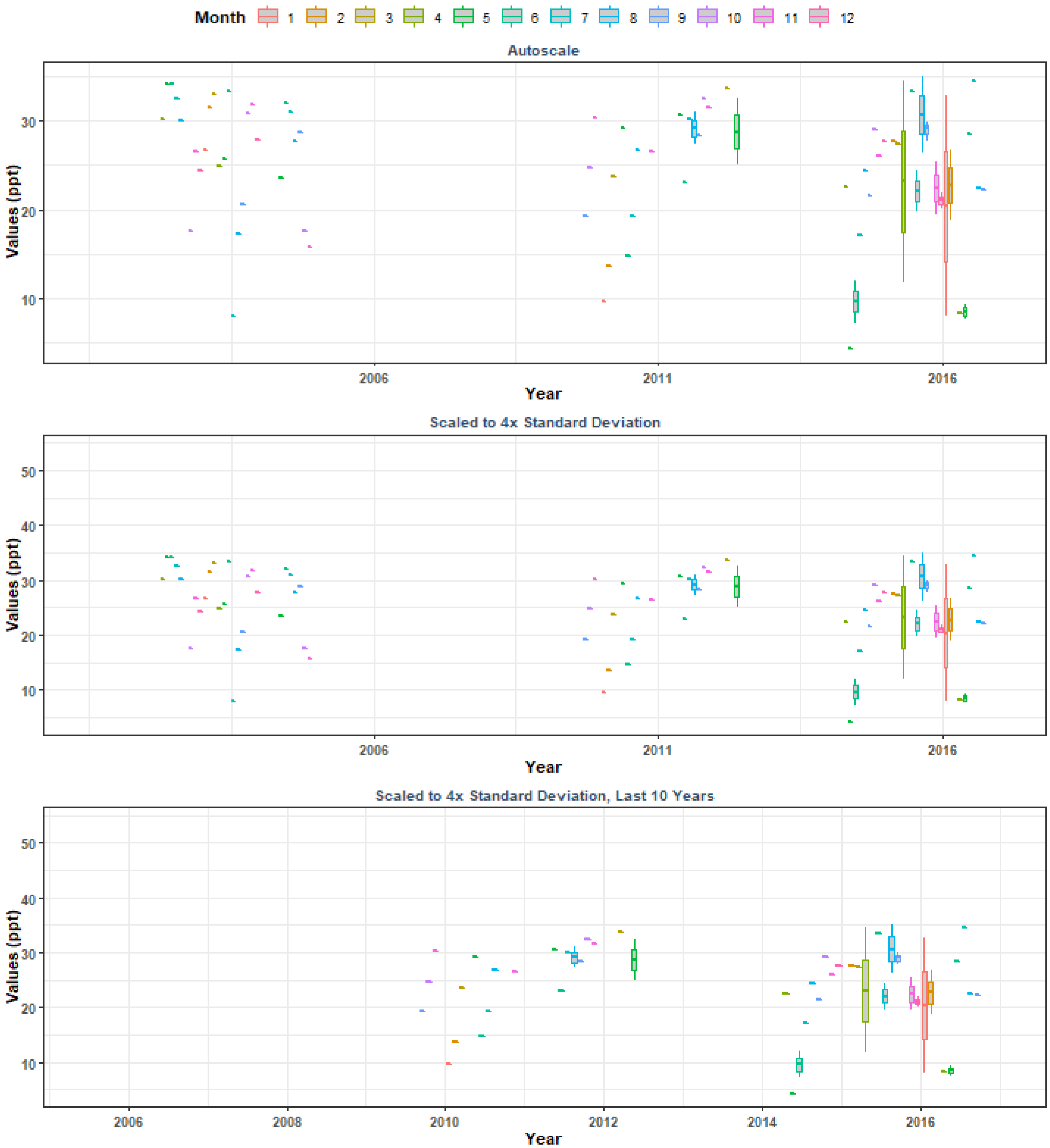
Summary Box Plots for Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program | P09

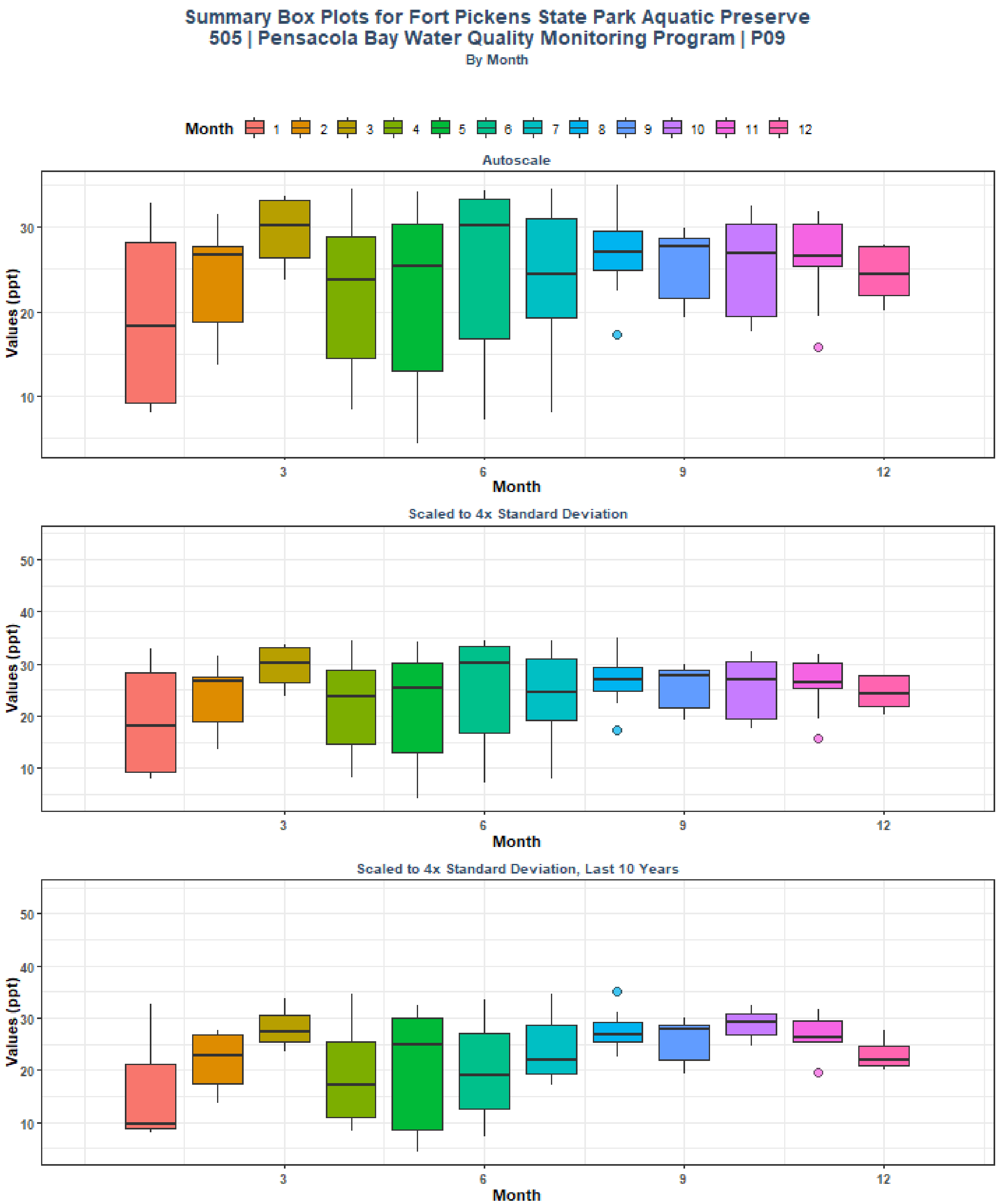
By Year



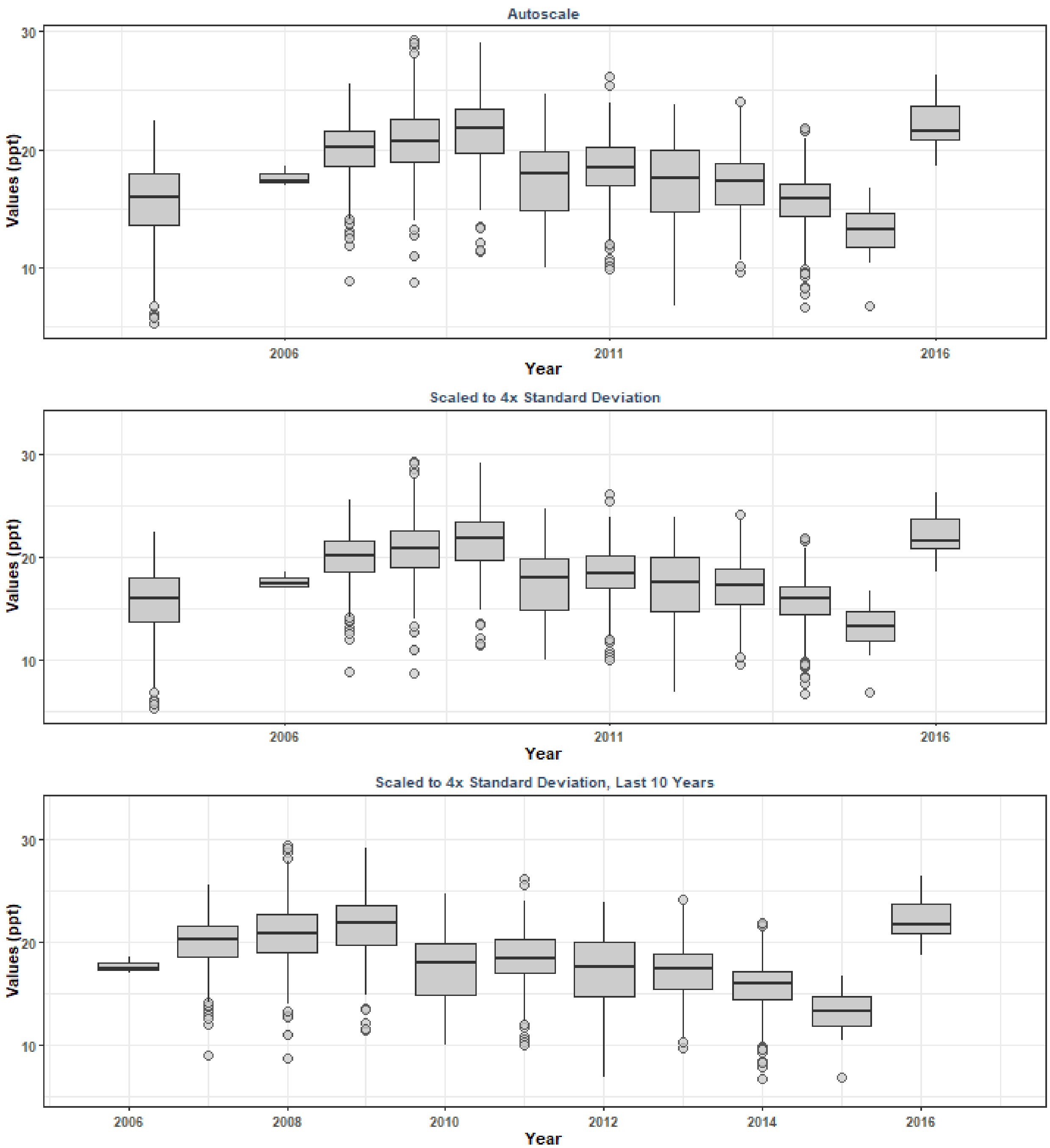
Summary Box Plots for Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program | P09

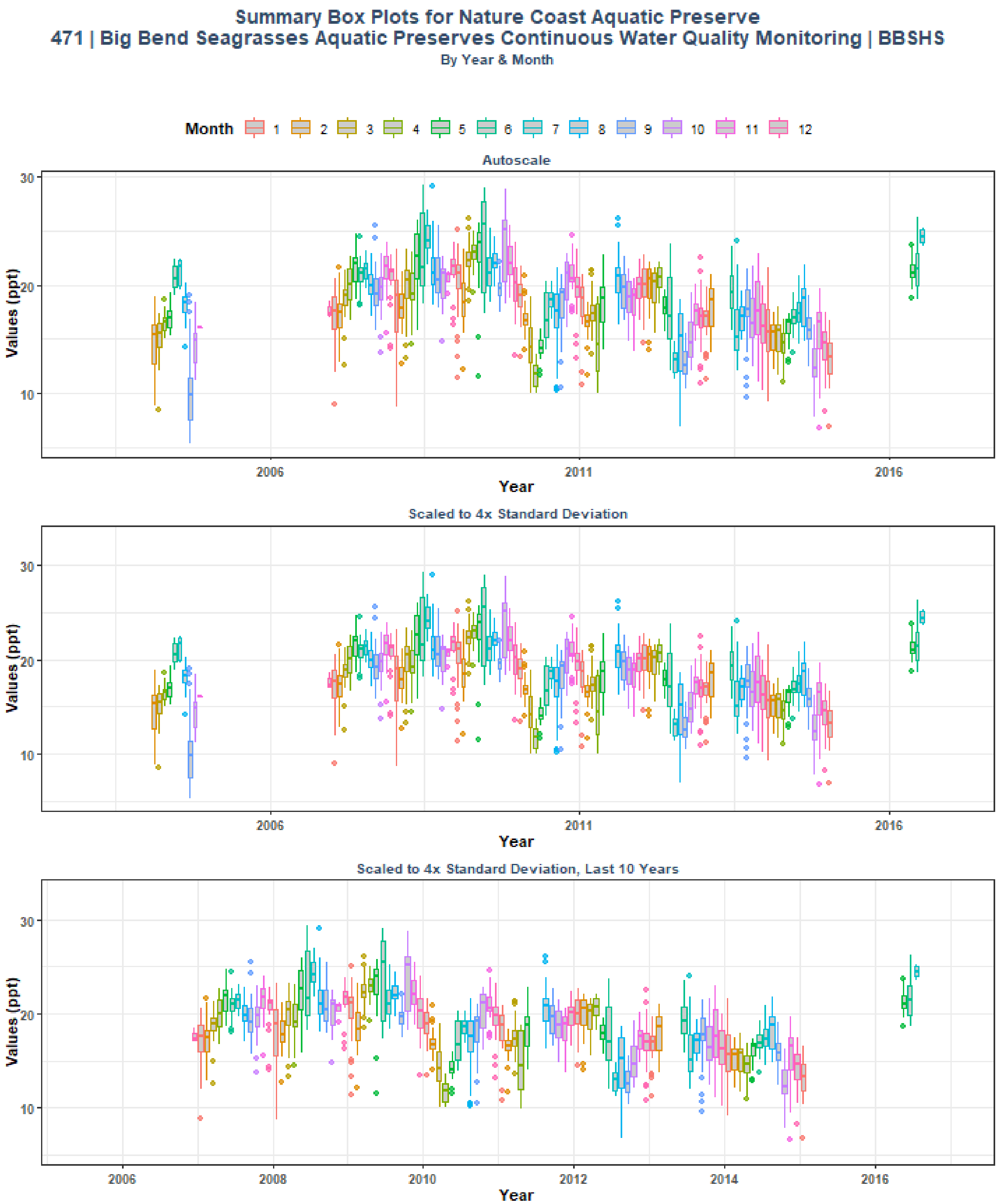
By Year & Month




Summary Box Plots for Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program | P09
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (ppt)
10
20
30
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (ppt)
10
20
30
40
50
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (ppt)
10
20
30
40
50
3
6
9
12
Month

Summary Box Plots for Nature Coast Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSHS
By Year




Summary Box Plots for Nature Coast Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSHS
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (ppt)
30
20
10
2006
2011
2016
Year
Scaled to 4x Standard Deviation
Values (ppt)
30
20
10
2006
2011
2016
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (ppt)
30
20
10
2006
2008
2010
2012
2014
2016
Year

Summary Box Plots for Nature Coast Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring | BBSHS
By Month