

# SEACAR Continuous Water Quality Analysis: NE Region for Dissolved Oxygen Saturation

Last compiled on 24 June, 2022

## Contents

<b>Important Notes</b>	<b>1</b>
<b>Libraries and Settings</b>	<b>1</b>
<b>File Import</b>	<b>2</b>
<b>Data Filtering</b>	<b>2</b>
<b>Monitoring Location Statistics</b>	<b>4</b>
<b>Seasonal Kendall Tau Analysis</b>	<b>5</b>
<b>Appendix I: Dataset Summary Box Plots</b>	<b>10</b>
<b>Appendix II: Excluded Monitoring Locations</b>	<b>16</b>
<b>Appendix III: Monitoring Location Trendlines</b>	<b>19</b>
<b>Appendix IV: Monitoring Location Summary Box Plots</b>	<b>30</b>

## Important Notes

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

[https://github.com/FloridaSEACAR/SEACAR\\_Panzik](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 and Settings

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(plyr)
library(dplyr)
library(lubridate)
library(ggplot2)
library(ggpubr)
library(scales)
library(EnvStats)
library(tidyr)
library(kableExtra)

windowsFonts(`Segoe UI` = windowsFont('Segoe UI'))
options(scipen=999)
opts_chunk$set(warning=FALSE, message=FALSE, dpi=200)

```

## File Import

Imports file that is determined in the WC\_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,]

  #temporarily removing FKNMS Temp. data because I think it might be causing R to run out of memory.
  # data <- data[data$ManagedAreaName != "Florida Keys National Marine Sanctuary"]
} else{
  data <- data[data$ResultValue>=0,]
}

data <- data %>%
  group_by(ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
           SampleDate) %>%
  dplyr::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$YearMonth <- format(data$SampleDate, format = "%m-%Y")
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())

Mon_Summ <- data %>%
  group_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,
           ProgramLocationID) %>%
  dplyr::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_Summ <- as.data.table(Mon_Summ[order(Mon_Summ$MonitoringID), ])

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

data$Use_In_Analysis <- ifelse(data$Include==TRUE &
                                    data$SufficientData==TRUE, TRUE, FALSE)
setDT(data)
data[, `:=` (relyear = Year - min(Year), relyear_dd = DecDate - min(DecDate)), by = "ManagedAreaName"]

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

```

Mon_YM_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
  group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
           Year, Month) %>%
  dplyr::summarize(ParameterName=parameter,
                   RelativeDepth=unique(RelativeDepth),
                   EarliestSampleDate=min(SampleDate),
                   LastSampleDate=max(SampleDate), N=length(ResultValue),
                   Min=min(ResultValue), Max=max(ResultValue),
                   Median=median(ResultValue), Mean=mean(ResultValue),
                   StandardDeviation=sd(ResultValue))
Mon_YM_Stats <- as.data.table(Mon_YM_Stats[order(Mon_YM_Stats$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$Use_In_Analysis==TRUE, ] %>%
  group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
           Year) %>%
  dplyr::summarize(ParameterName=parameter,
                   RelativeDepth=unique(RelativeDepth),
                   EarliestSampleDate=min(SampleDate),
                   LastSampleDate=max(SampleDate), N=length(ResultValue),
                   Min=min(ResultValue), Max=max(ResultValue),
                   Median=median(ResultValue), Mean=mean(ResultValue),
                   StandardDeviation=sd(ResultValue))
Mon_Y_Stats <- as.data.table(Mon_Y_Stats[order(Mon_Y_Stats$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$Use_In_Analysis==TRUE, ] %>%
  group_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,
           Month) %>%
  dplyr::summarize(ParameterName=parameter,
                   RelativeDepth=unique(RelativeDepth),
                   EarliestSampleDate=min(SampleDate),
                   LastSampleDate=max(SampleDate), N=length(ResultValue),
                   Min=min(ResultValue), Max=max(ResultValue),
                   Median=median(ResultValue), Mean=mean(ResultValue),
                   StandardDeviation=sd(ResultValue))
Mon_M_Stats <- as.data.table(Mon_M_Stats[order(Mon_M_Stats$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 `KTStats` for easier use while plotting.

```

tauSeasonal <- function(data, independent, stats.median, stats.minYear,
                         stats.maxYear, seasondata = Mon_M_Stats[Mon_M_Stats$ProgramLocationID==unique(d
setDT(data)
tau <- NULL
tryCatch({ken <- kendallSeasonalTrendTest(
  y=data$ResultValue,
  season=data$Month,
  year=data$relyear,
  independent.obs=independent)

tau <- ken$estimate[1]
z <- ken$statistic[2]
p_z <- ken$p.value[2]
chi_sq <- ken$statistic[1]
p_chi_sq <- ken$p.value[1]
slope <- ken$estimate[2]
intercept <- ken$estimate[3]
trend <- trend_calculator(slope, stats.median, p_z)

seasonresults <- as.data.table(ken$seasonal.estimates)
rm(ken)
}, warning = function(w) {
  print(w)
}, error = function(e) {
  print(e)
}, finally = {
  if (!exists("tau")) {
    tau <- NA
  }
  if (!exists("z")) {
    z <- NA
  }
}

```

```

    }
    if (!exists("p_z")) {
      p_z <- NA
    }
    if (!exists("chi_sq")) {
      chi_sq <- NA
    }
    if (!exists("p_chi_sq")) {
      p_chi_sq <- NA
    }
    if (!exists("slope")) {
      slope <- NA
    }
    if (!exists("intercept")) {
      intercept <- NA
    }
    if (!exists("trend")) {
      trend <- NA
    }
  })
KT <- data.table(MonitoringID = unique(data$MonitoringID),
                 season = "All",
                 stats.median = stats.median,
                 independent = independent,
                 tau = tau,
                 z = z,
                 p_z = p_z,
                 chi_sq = chi_sq,
                 p_chi_sq = p_chi_sq,
                 slope = slope,
                 intercept = intercept,
                 trend = trend)

seasonresults[, `:=` (MonitoringID = unique(data$MonitoringID),
                      season = sort(unique(data$Month)),
                      stats.median = as.numeric(NA),
                      independent = independent,
                      z = as.numeric(NA),
                      p_z = as.numeric(NA),
                      chi_sq = as.numeric(NA),
                      p_chi_sq = as.numeric(NA),
                      trend = as.integer(NA))]

for(s in as.integer(unique(seasonresults$season))){
  seasondat_s <- data[Month == s, ]
  if(!is.na(unique(seasondat_s$Month))){
    trend_s <- trend_calculator(seasonresults[season == s, slope], seasondata[Month == s, Median], p
    ken_s <- kendallTrendTest(ResultValue ~ relyear, data = seasondat_s)
    seasonresults[season == s, `:=` (stats.median = unique(seasondata[Month == s, Median]),
                                      z = ken_s$statistic,
                                      p_z = ken_s$p.value,
                                      chi_sq = NA,
                                      p_chi_sq = NA,

```

```

                trend = trend_s)]
} else{
  next
}
}

seasonresults[, season := as.character(season)]

KT <- rbind(KT, seasonresults)
KT[, season := factor(season, levels = c("All", seq(1:12)), ordered = TRUE)]
return(KT)
}

runStats <- function(data, Mon_M_Stats) {
  data$Index <- as.Date(data$SampleDate) # , "%Y-%m-%d")
  data$resultValue <- as.numeric(data$resultValue)
  # Calculate basic stats
  stats.median <- median(data$resultValue, na.rm=TRUE)
  stats.minYear <- min(data$relyear, na.rm=TRUE)
  stats.maxYear <- max(data$relyear, na.rm=TRUE)
  # Calculate Kendall Tau and Slope stats,
  # then update appropriate columns and table
  seasondata <- Mon_M_Stats[Mon_M_Stats$ProgramLocationID==unique(data$ProgramLocationID[data$MonitoringID]),]
  KT <- tauSeasonal(data, TRUE, stats.median,
                     stats.minYear, stats.maxYear, seasondata)
  #if (is.null(KT[8])) {
  if (is.na(KT$season == "All", trend)) {
    KT <- tauSeasonal(data, FALSE, stats.median,
                      stats.minYear, stats.maxYear, seasondata)
  }
  if (is.null(KT$Stats)==TRUE) {
    KT$Stats <- KT
  } else{
    KT$Stats <- rbind(KT$Stats, KT)
  }
  return(KT$Stats)
}

trend_calculator <- function(slope, median_value, p) {
  trend <-
    if (p < .05 & abs(slope) > abs(median_value) / 10.) {
      if (slope > 0) {
        2
      }
      else {
        -2
      }
    }
    else if (p < .05 & abs(slope) < abs(median_value) / 10.) {
      if (slope > 0) {
        1
      }
      else {
        -1
      }
    }
}

```

```

    }
    else
      0
  return(trend)
}
KT.Stats <- NULL
# Loop that goes through each managed area.
# List of managed areas stored in MA_Years$ManagedAreaName
c_names <- c("MonitoringID", "Season", "Median", "Independent",
            "tau", "z", "p_z", "chi_sq", "p_chi_sq", "SennSlope", "SennIntercept", "Trend")
if(n==0){
  KT.Stats <- data.frame(matrix(ncol=length(c_names),
                                 nrow=nrow(Mon_Summ)))
  colnames(KT.Stats) <- c_names
  #KT.Stats[, c("MonitoringID")] <- Mon_Summ[, c("MonitoringID")]
} else{
  for (i in 1:n) {
    x <- nrow(data[data$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], ], Mon_M_Stats)
    }
  }
  KT.Stats <- as.data.frame(KT.Stats)

  if(dim(KT.Stats)[2]==1){
    KT.Stats <- as.data.frame(t(KT.Stats))
  }
  colnames(KT.Stats) <- c_names
  rownames(KT.Stats) <- seq(1:nrow(KT.Stats))
  KT.Stats$tau <- round(as.numeric(KT.Stats$tau), digits=4)
  KT.Stats$z <- round(as.numeric(KT.Stats$z), digits=4)
  KT.Stats$p_z <- round(as.numeric(KT.Stats$p_z), digits=4)
  KT.Stats$chi_sq <- round(as.numeric(KT.Stats$chi_sq), digits=4)
  KT.Stats$p_chi_sq <- round(as.numeric(KT.Stats$p_chi_sq), digits=4)
  KT.Stats$SennSlope <- as.numeric(KT.Stats$SennSlope)
  KT.Stats$SennIntercept <- as.numeric(KT.Stats$SennIntercept)
  KT.Stats$Trend <- as.integer(KT.Stats$Trend)
}

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

KT.Stats <- as.data.table(KT.Stats[order(KT.Stats$MonitoringID), ])
KT.Stats2 <- copy(KT.Stats)
KT.Stats[, `:=` (Region = region, Units = unit)]
KT.Stats_all <- rbind(KT.Stats_all, KT.Stats)

KT.Stats2$MonitoringID <- NULL
fwrite(KT.Stats2, paste0(out_dir, "/", param_name, "_", region,
                        "_KendallTau_Stats.txt"), sep="|")
rm(KT.Stats2)

```

```
#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.title.x = element_text(margin = margin(t = 5, r = 0,
                                                    b = 10, l = 0)),
        axis.title.y = element_text(margin = margin(t = 0, r = 10,
                                                    b = 0, l = 0)),
        axis.text=element_text(size=10),
        #axis.text.x=element_text(face="bold", angle = 60, hjust = 1),
        axis.text.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=SampleDate, y=ResultValue, group=Year)) +
  geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
               outlier.size=3, outlier.color="#333333",
               outlier.fill="#cccccc", outlier.alpha=0.75) +
  labs(subtitle="Autoscale", x="Year",
       y=paste0("Values (", unit, ")")) +
  plot_theme
```

```

p2 <- ggplot(data=data[data$Include==TRUE, ],
              aes(x=SampleDate, y=ResultValue, group=Year)) +
  geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
               outlier.size=3, outlier.color="#333333",
               outlier.fill="#cccccc", outlier.alpha=0.75) +
  labs(subtitle="Scaled to 4x Standard Deviation", x="Year",
       y=paste0("Values (", unit, ")")) +
  ylim(0, y_scale) +
  plot_theme

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

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))) +

```

```

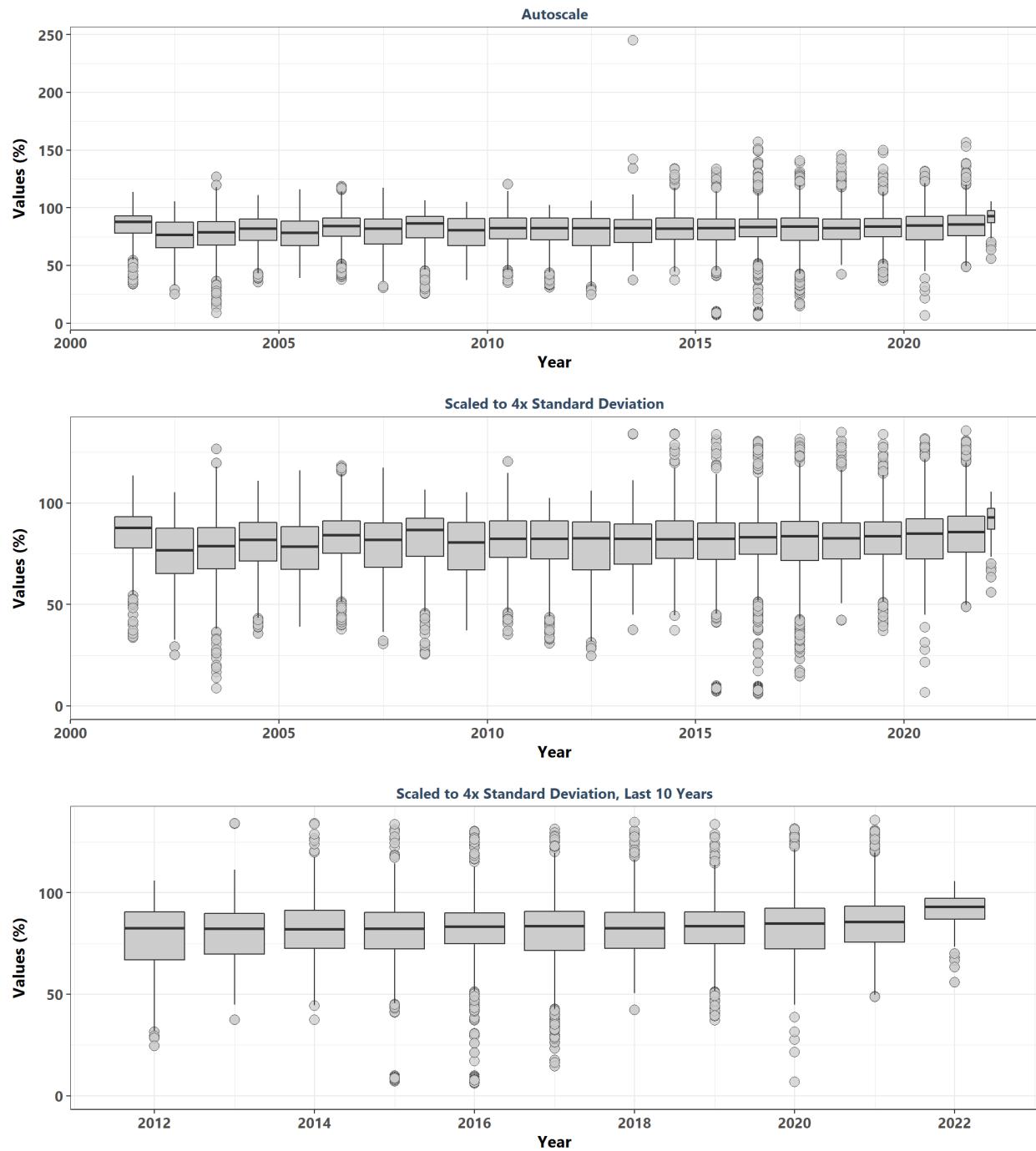
geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
             outlier.color="#333333", outlier.alpha=0.75) +
  labs(subtitle="Scaled to 5x Standard Deviation, Last 10 Years",
       x="Month", y=paste0("Values (", unit, ")")) +
  ylim(0, y_scale) +
  scale_x_continuous(limits=c(0, 13), 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())

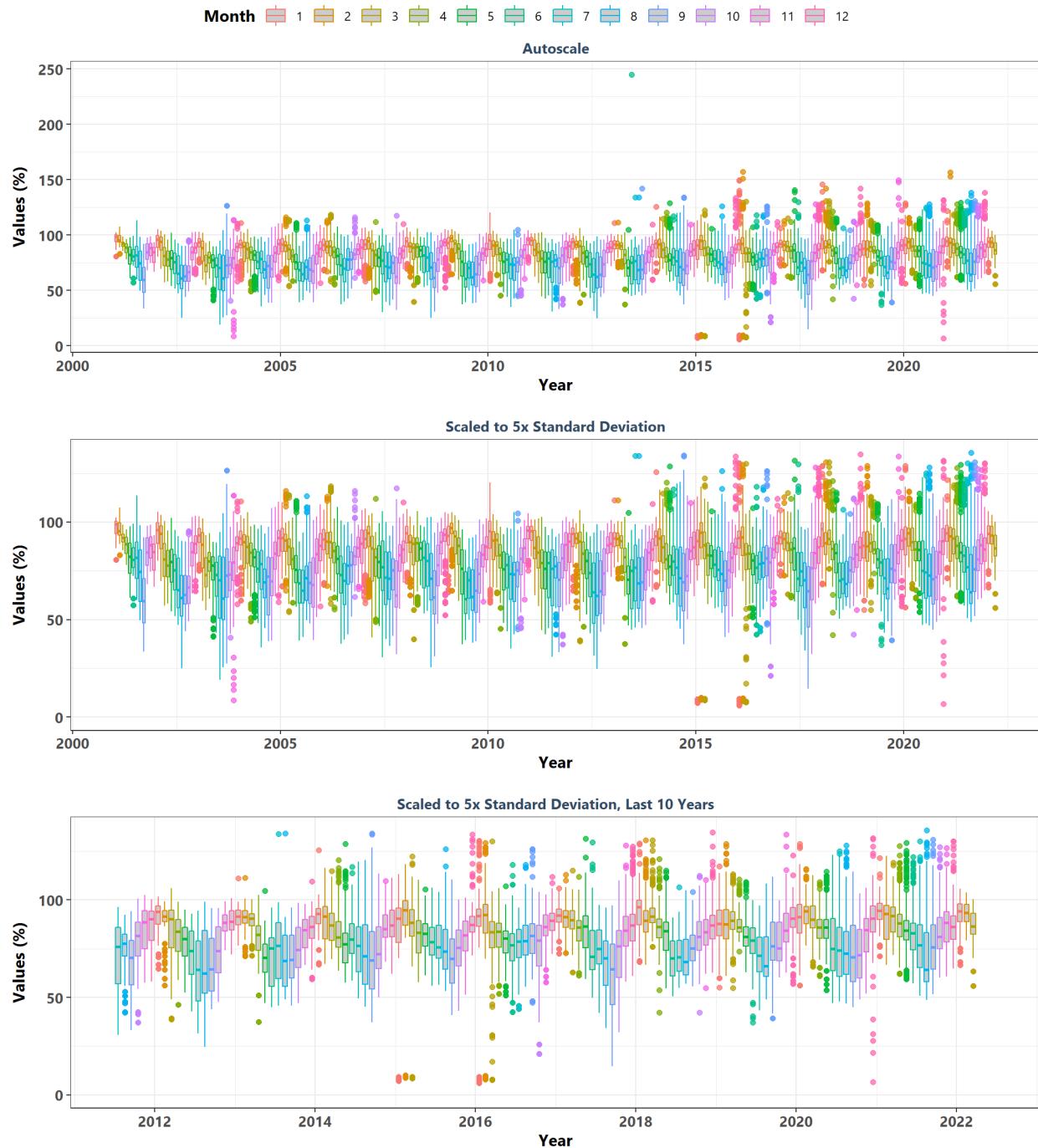
Mset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))

```

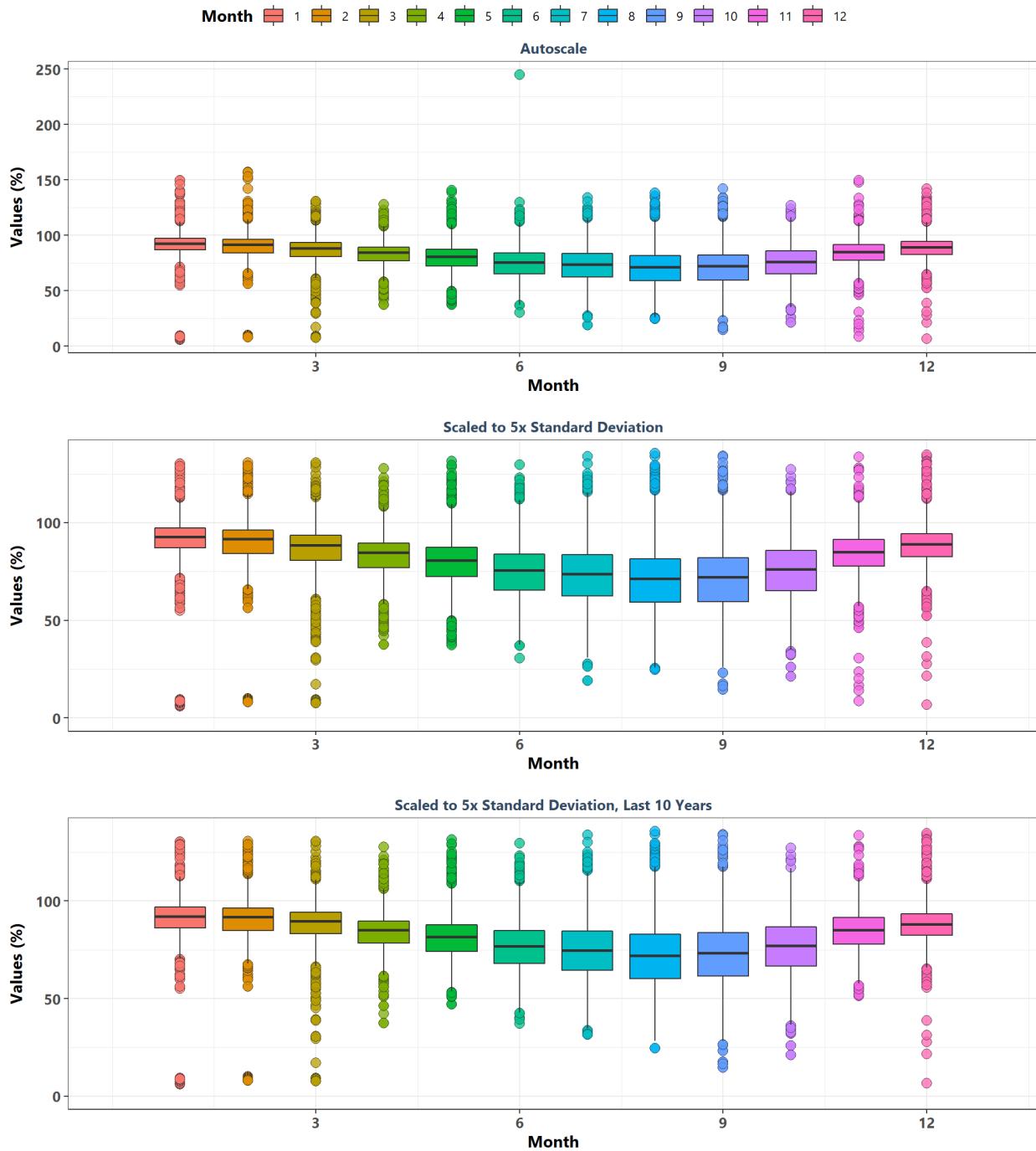
**Summary Box Plots for Entire Data**  
By Year



**Summary Box Plots for Entire Data**  
By Year & Month



### Summary Box Plots for Entire Data By Month



## Appendix II: Excluded Monitoring Locations

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

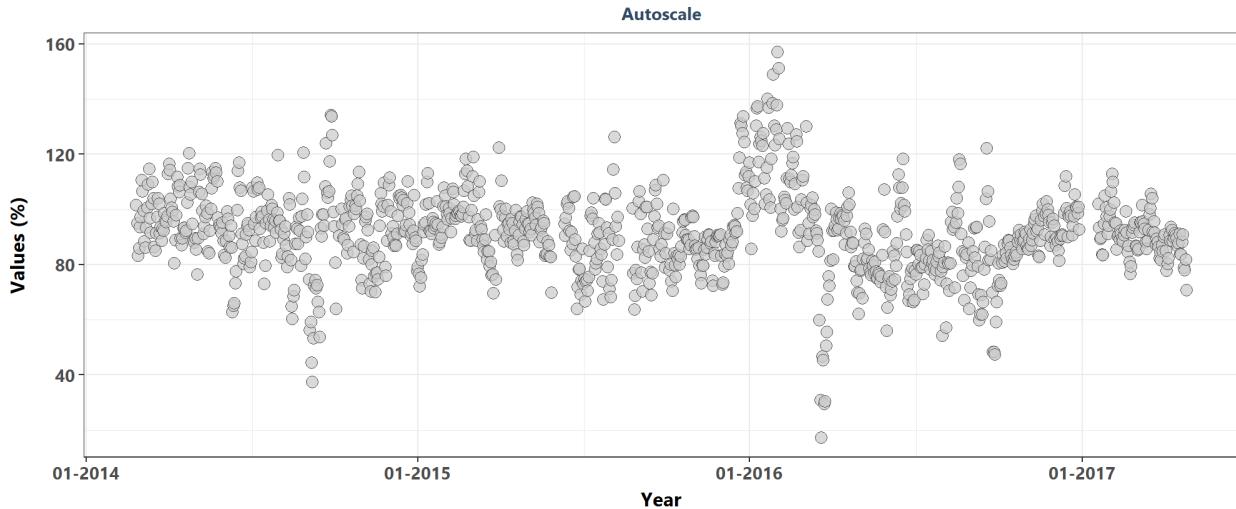
```

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$MonitoringID[i]])
    Mon_name <- paste0(unique(data$ProgramID[
      data$MonitoringID==Mon_Exclude$MonitoringID[i]]), " | ",
      unique(data$ProgramName[
        data$MonitoringID==Mon_Exclude$MonitoringID[i]]), "\n",
      unique(data$ProgramLocationID[
        data$MonitoringID==Mon_Exclude$MonitoringID[i]])))
  }
}

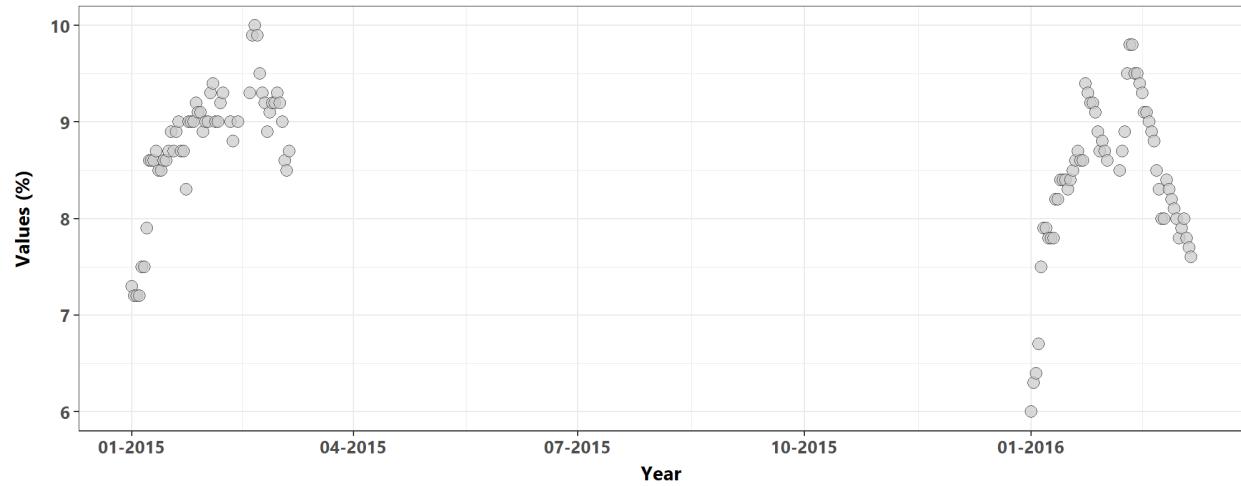
```

**Banana River Aquatic Preserve  
5061 | St. Johns River Water Management District Continuous Water Quality Programs  
CMMerritt (4 Unique Years)**



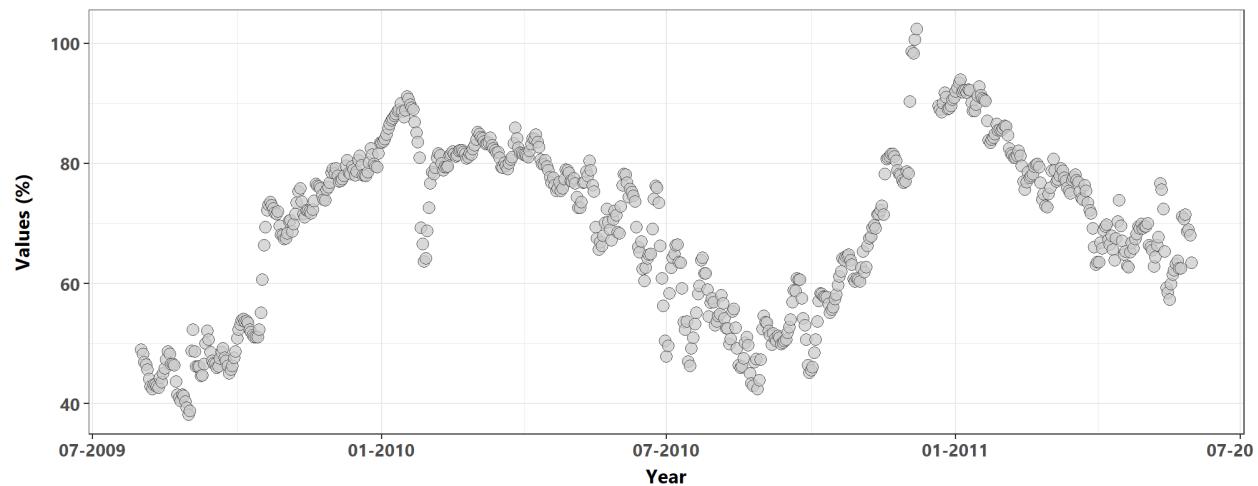
**Nassau River-St. Johns River Marshes Aquatic Preserve**  
7 | National Water Information System  
**02231291 (2 Unique Years)**

Autoscale



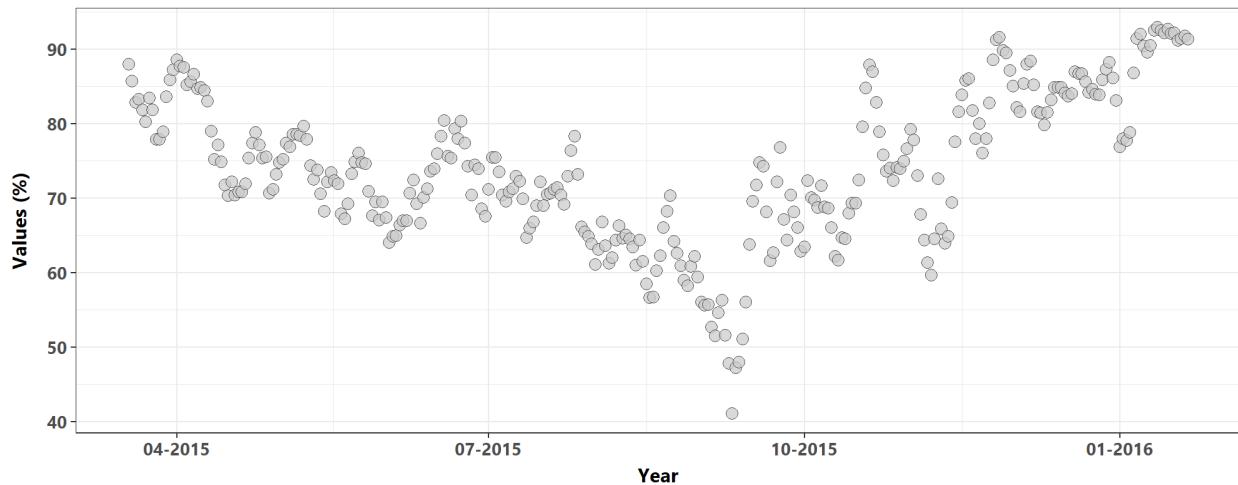
**Nassau River-St. Johns River Marshes Aquatic Preserve**  
5006 | Northeast Aquatic Preserves Continuous Water Quality Monitoring  
**NENR (3 Unique Years)**

Autoscale



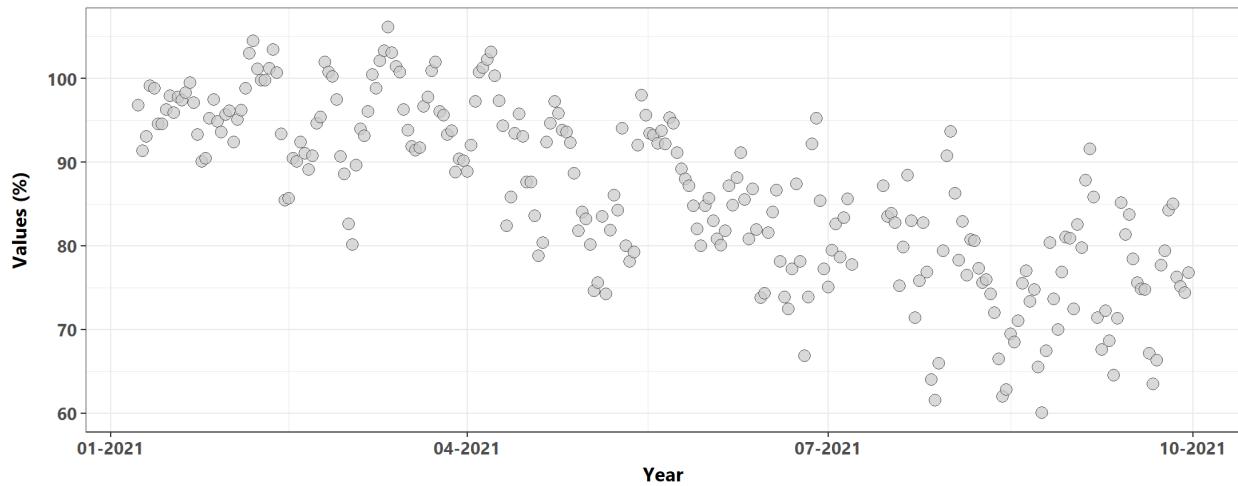
**Nassau River-St. Johns River Marshes Aquatic Preserve  
5061 | St. Johns River Water Management District Continuous Water Quality Programs  
NCBARD16 (2 Unique Years)**

Autoscale



**Tomoka Marsh Aquatic Preserve  
10003 | Tomoka Marsh Aquatic Preserve Continuous Water Quality Monitoring  
TMGR (1 Unique Years)**

Autoscale



### 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],]
    plot_data$Season <- factor(plot_data$Month, levels = c("All", seq(1, 12)), ordered = TRUE)
    year_lower <- min(plot_data$relyear)
    year_upper <- max(plot_data$relyear)
    # relyear_dd_lower <- min(plot_data$relyear_dd)
    # relyear_dd_upper <- max(plot_data$relyear_dd)
    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]]
    z <- KT.Stats$z[KT.Stats$MonitoringID==Mon_IDs[i]]
    p_z <- KT.Stats$p_z[KT.Stats$MonitoringID==Mon_IDs[i]]
    chi_sq <- KT.Stats$chi_sq[KT.Stats$MonitoringID==Mon_IDs[i]]
    p_chi_sq <- KT.Stats$p_chi_sq[KT.Stats$MonitoringID==Mon_IDs[i]]

    MA_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID==Mon_IDs[i]]
    Mon_name <- paste0(KT.Stats$ProgramID[KT.Stats$MonitoringID==Mon_IDs[i]],
                       " | ", KT.Stats$ProgramName[KT.Stats$MonitoringID==Mon_IDs[i]], "\n",
                       KT.Stats$ProgramLocationID[KT.Stats$MonitoringID==Mon_IDs[i]])

    xbrks <- seq(round_any(min(plot_data$relyear_dd), 5, floor), round_any(max(plot_data$relyear_dd),
                           by = (round_any(max(plot_data$relyear_dd), 5, ceiling) - round_any(min(plot_data$relyear_dd), 5, floor))))
    xlabs <- seq(max(plot_data$Year) - round_any(max(plot_data$relyear_dd), 5, ceiling),
                  max(plot_data$Year),
                  by = (max(plot_data$Year) - (max(plot_data$Year) - round_any(max(plot_data$relyear_dd), 5, ceiling)) / 5))

    # x1 <- min(unique(data$relyear_dd[data$Year == KT.Stats$ManagedAreaName==MA_Include[i] & Season == "All", SennSlope]))
    # y1 <- min(unique(data$relyear_dd[data$Year == KT.Stats$ManagedAreaName==MA_Include[i] & Season == "All", SennSlope]))
    # x_end1 <- max(unique(data$relyear_dd[data$Year == KT.Stats$ManagedAreaName==MA_Include[i] & Season == "All", SennSlope]))
    # y_end1 <- max(unique(data$relyear_dd[data$Year == KT.Stats$ManagedAreaName==MA_Include[i] & Season == "All", SennSlope]))
    # x1 <- relyear_dd_lower
    # y1 <- relyear_dd_lower * KT.Stats$ManagedAreaName==MA_Include[i] & Season == "All", SennSlope]
  }
}

```

```

# x_end1 <- relyear_dd_upper
# y_end1 <- relyear_dd_upper * KT.Stats[ManagedAreaName==MA_Include[i] & Season == "All", SennSlo

KT.Stats[, season := Season]
KT.Stats[MonitoringID == Mon_IDs[i] & season != "All", `:=` (N_Data = nrow(plot_data[Season == se
KT.Stats[MonitoringID == Mon_IDs[i] & season == "All", `:=` (relyear_dd_lower = min(plot_data[Mon
KT.Stats[, season := NULL]

p1 <- ggplot(data=plot_data,
              aes(x=relyear_dd, y=ResultValue)) +
  geom_point(shape=21, size=3, color="#333333", fill="#cccccc",
             alpha=0.75) +
  #geom_abline(data = KT.Stats[ManagedAreaName==MA_Include[i] & Season == "All", ], aes(slope=Se
  #                           color="#000099", size=1.2, alpha=0.7) +
  geom_segment(data = KT.Stats[MonitoringID == Mon_IDs[i] & Season == "All", ], aes(x = relyear_
                                         color="#000099", size=1.2, alpha=0.7) +
  labs(subtitle="Autoscale",
       x="Year", y=paste0("Values (", unit, ")")) +
  plot_theme +
  scale_x_continuous(breaks = xbrks,
                     labels = xlabs)

p2 <- ggplot(data=plot_data,
              aes(x=relyear_dd, 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) +
  geom_segment(data = KT.Stats[MonitoringID == Mon_IDs[i] & Season == "All", ], aes(x = relyear_
                                         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 +
  scale_x_continuous(breaks = xbrks,
                     labels = xlabs)

splot <- ggplot(plot_data, aes(x = relyear_dd, y = ResultValue)) +
  geom_point(shape = 21, size = 1.5, color="#333333", fill="#cccccc", alpha=0.75) +
  geom_segment(data = KT.Stats[MonitoringID == Mon_IDs[i] & Season != "All", ], aes(x = relyear_
                                         color="#000099", size=1.2, alpha=0.7) +
  #ylim(min_RV-0.1*y_scale, y_scale) +

```

```

scale_x_continuous(breaks = xbrks,
                   labels = xlabs) +
  labs(y = paste0("Values (", unit, ")"), x = "Year", subtitle = "Results for Individual Seas",
       facet_wrap(~Season, ncol = 3) +
       plot_theme

KTset <- ggarrange(p1, p2, splot, ncol=1, heights=c(1, 1, 1.5))

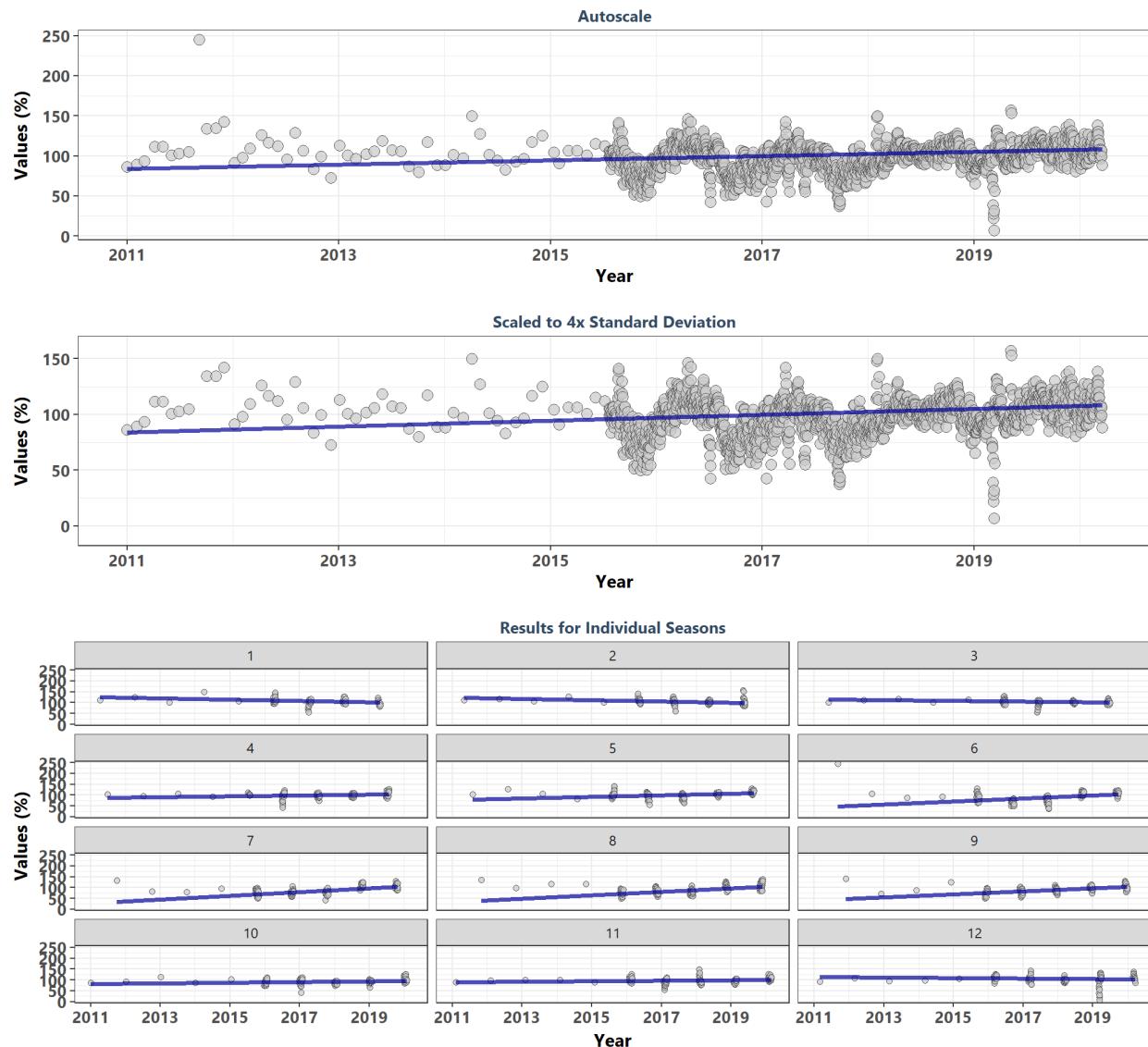
p0 <- ggplot() + labs(title=paste0(MA_name, "\n", Mon_name)) +
  plot_theme + theme(panel.border=element_blank(),
                     panel.grid.major=element_blank(),
                     panel.grid.minor=element_blank(),
                     axis.line=element_blank())

KTStats[MonitoringID==Mon_IDs[i], `:=` (N = N_Data,
                                         Median = round(Median, 2),
                                         Slope = round(SennSlope, 4),
                                         Int. = round(SennIntercept, 4),
                                         z = round(z, 1),
                                         chi_sq = round(chi_sq, 1))]

print(ggarrange(p0, KTset, ncol=1, heights=c(0.1, 1.25)))
cat('\n')
print(KTStats[KTStats$MonitoringID==Mon_IDs[i], ] %>%
  select(Season, N, Median, tau, Slope, Int., z, p_z, chi_sq, p_chi_sq, Trend) %>%
  kable(format="latex") %>%
  row_spec(0, bold=TRUE) %>%
  kable_styling(latex_options = "HOLD_position",
                font_size = 7) %>%
  add_footnote(
    "p < 0.00005 appear as 0 due to rounding"))
cat('\n')
rm(plot_data)
rm(KTset, leg)
}
}

```

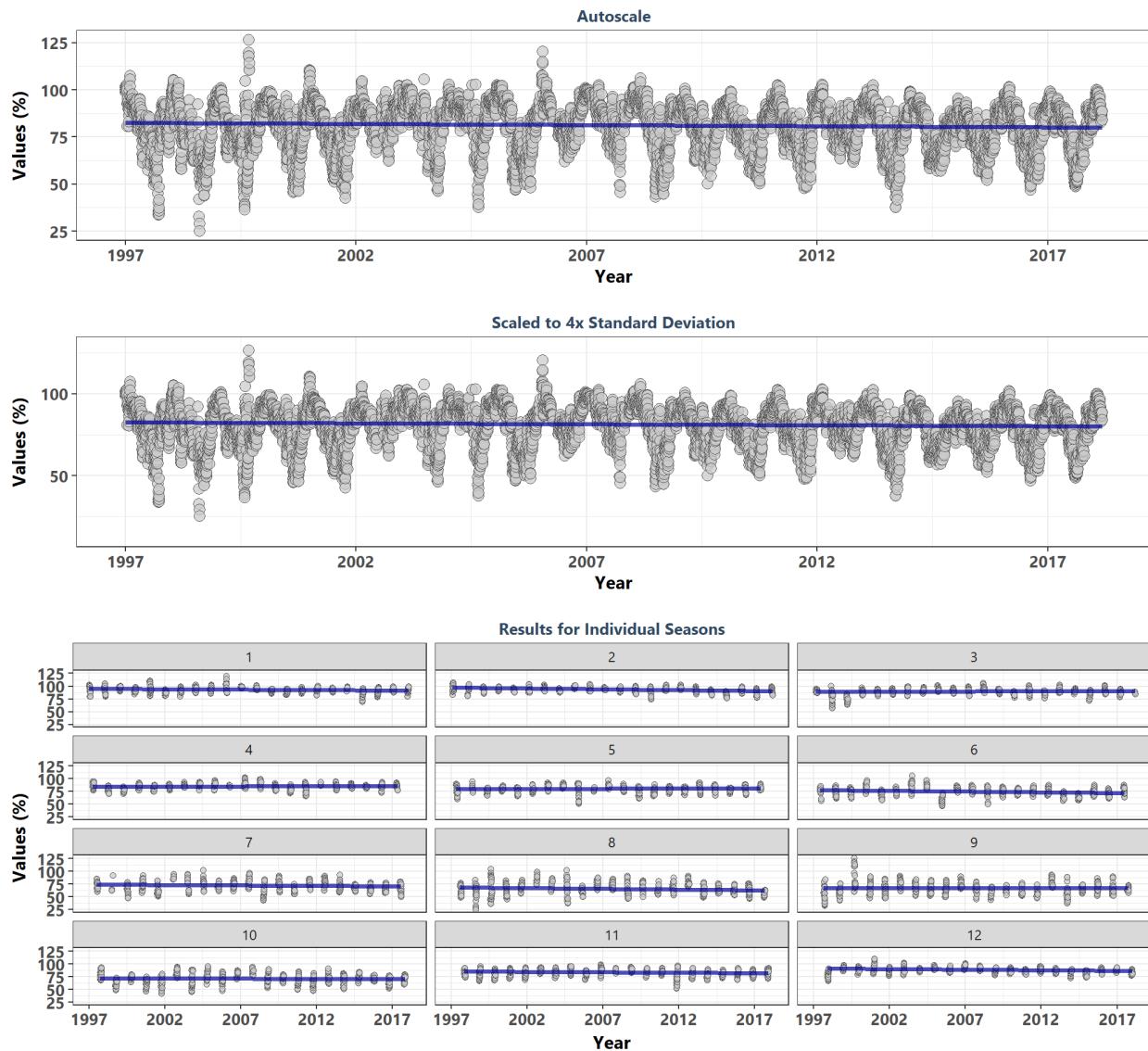
**Banana River Aquatic Preserve**  
**5061 | St. Johns River Water Management District Continuous Water Quality Programs**  
**IRLB04**



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	1755	98.82	0.1459	2.6435	83.9063	10.3	0.0000	237	0	1
1	129	104.66	-0.2488	-3.1599	126.7792	-4.3	0.0000	NA	NA	-1
2	118	103.64	-0.2148	-2.7728	123.0534	-3.5	0.0004	NA	NA	-1
3	129	104.13	-0.1669	-1.5692	115.1130	-2.9	0.0039	NA	NA	-1
4	130	101.30	0.1891	2.2326	85.6684	3.3	0.0011	NA	NA	1
5	158	103.04	0.2840	3.7185	77.0055	5.4	0.0000	NA	NA	1
6	154	92.70	0.2932	6.9893	43.7755	5.5	0.0000	NA	NA	1
7	159	87.61	0.4215	8.7251	26.5311	8.0	0.0000	NA	NA	1
8	159	89.04	0.3982	8.1532	31.9648	7.6	0.0000	NA	NA	1
9	154	89.61	0.4422	7.0516	40.2451	8.3	0.0000	NA	NA	1
10	160	91.73	0.1219	1.3691	82.1442	2.3	0.0197	NA	NA	1
11	155	97.07	0.0950	1.1820	88.7986	1.8	0.0739	NA	NA	1
12	150	105.63	-0.0813	-1.2981	114.7185	-1.5	0.1322	NA	NA	-1

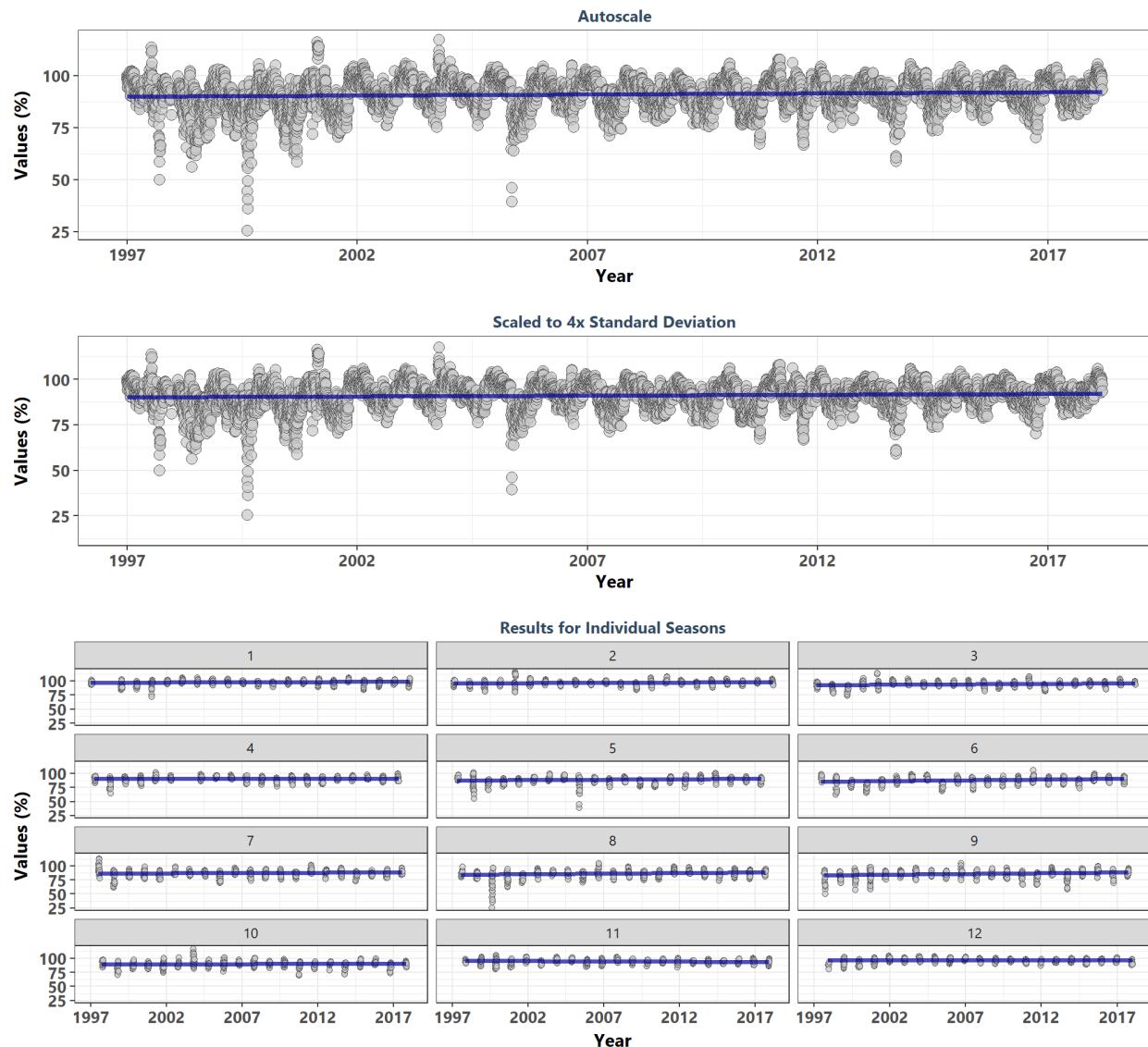
<sup>a</sup> p < 0.00005 appear as 0 due to rounding

**Guana River Marsh Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**



<sup>a</sup> p < 0.00005 appear as 0 due to rounding

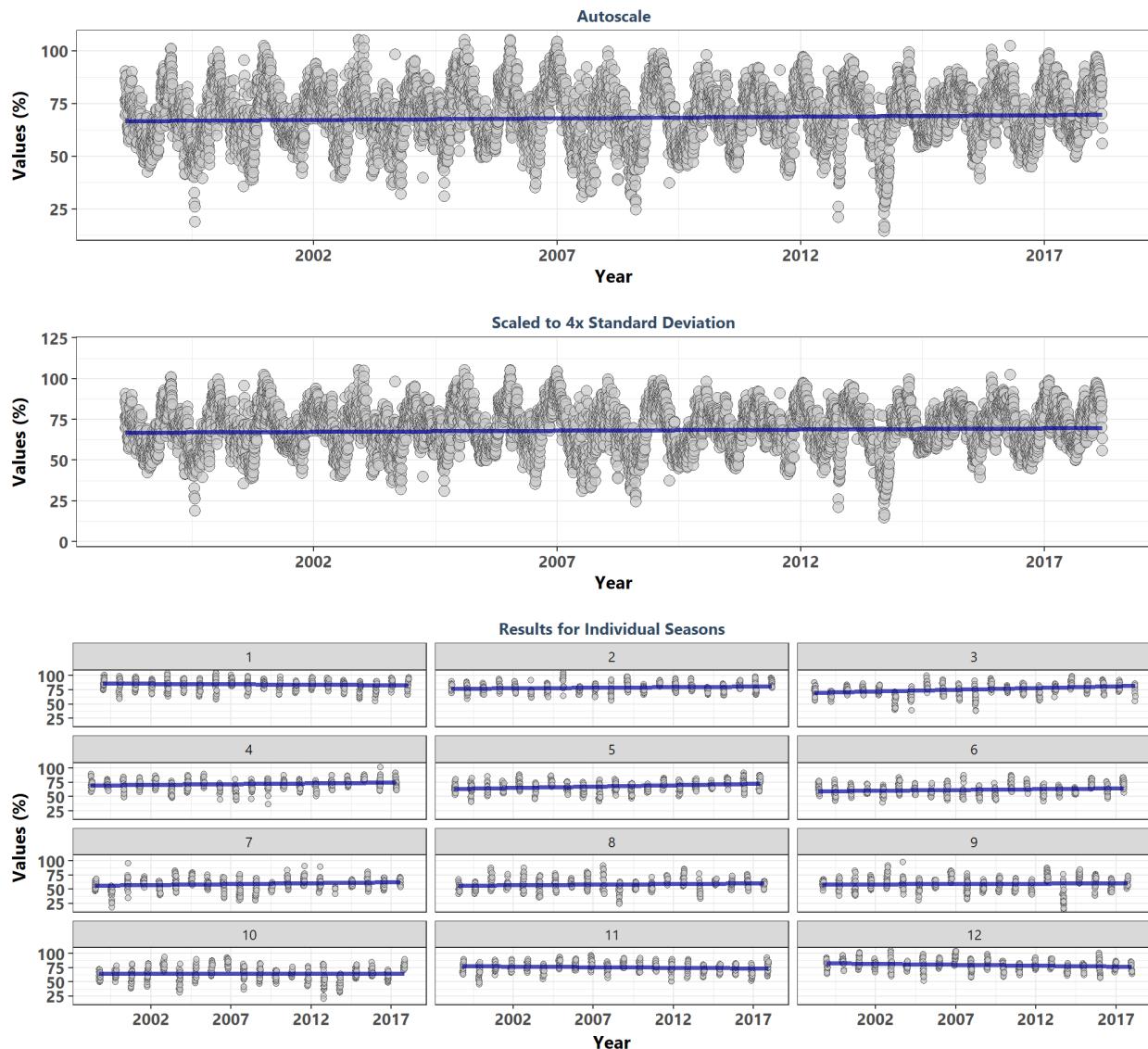
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmfmwq**



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	7252	91.45	0.0817	0.1000	90.0541	10.5	0.0000	75	0	1
1	611	97.94	0.1117	0.1006	96.8326	4.1	0.0000	NA	NA	1
2	559	97.30	0.0968	0.0928	96.1884	3.4	0.0006	NA	NA	1
3	621	94.61	0.1289	0.1503	92.9523	4.8	0.0000	NA	NA	1
4	593	90.76	0.0338	0.0360	90.3983	1.2	0.2172	NA	NA	1
5	615	88.89	0.1092	0.1482	87.4053	4.1	0.0000	NA	NA	1
6	597	88.00	0.1566	0.2542	85.4604	5.7	0.0000	NA	NA	1
7	616	87.26	0.0725	0.1076	86.1814	2.7	0.0070	NA	NA	1
8	641	86.25	0.1542	0.2250	83.9985	5.8	0.0000	NA	NA	1
9	593	86.08	0.1431	0.2331	83.5165	5.2	0.0000	NA	NA	1
10	630	90.21	0.0380	0.0498	89.7099	1.4	0.1531	NA	NA	1
11	589	94.82	-0.0822	-0.0809	95.6292	-3.0	0.0028	NA	NA	-1
12	587	96.32	0.0088	0.0084	96.2377	0.3	0.7507	NA	NA	1

<sup>a</sup> p < 0.00005 appear as 0 due to rounding

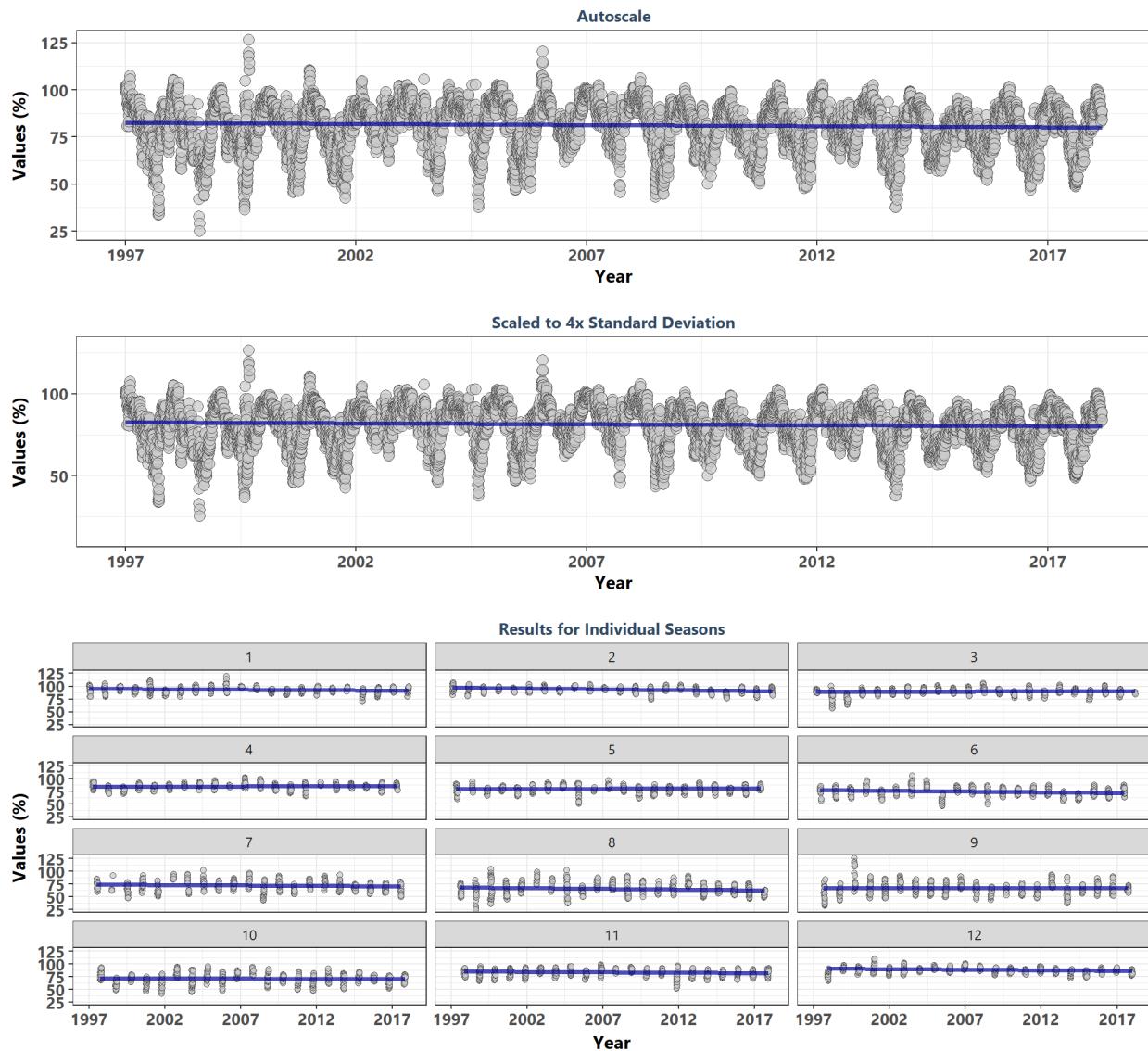
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	7006	70.69	0.0602	0.1493	66.6869	7.5	0.0000	198.7	0	1
1	604	84.52	-0.0658	-0.1457	86.1217	-2.4	0.0155	NA	NA	-1
2	528	79.39	0.0867	0.1931	77.0718	3.0	0.0029	NA	NA	1
3	615	76.12	0.2582	0.5992	69.5339	9.6	0.0000	NA	NA	1
4	583	72.02	0.1252	0.2785	68.9601	4.5	0.0000	NA	NA	1
5	588	68.15	0.1850	0.4493	63.2092	6.7	0.0000	NA	NA	1
6	564	61.79	0.1071	0.2679	58.8413	3.8	0.0001	NA	NA	1
7	575	59.06	0.1249	0.3047	56.0130	4.5	0.0000	NA	NA	1
8	570	58.64	0.0878	0.2009	56.6289	3.1	0.0017	NA	NA	1
9	577	59.38	0.0470	0.1464	57.9147	1.7	0.0910	NA	NA	1
10	614	64.47	0.0017	0.0056	64.4136	0.1	0.9511	NA	NA	1
11	582	76.03	-0.0854	-0.1994	78.0198	-3.1	0.0020	NA	NA	-1
12	606	79.87	-0.1411	-0.3265	83.1345	-5.2	0.0000	NA	NA	-1

<sup>a</sup> p < 0.00005 appear as 0 due to rounding

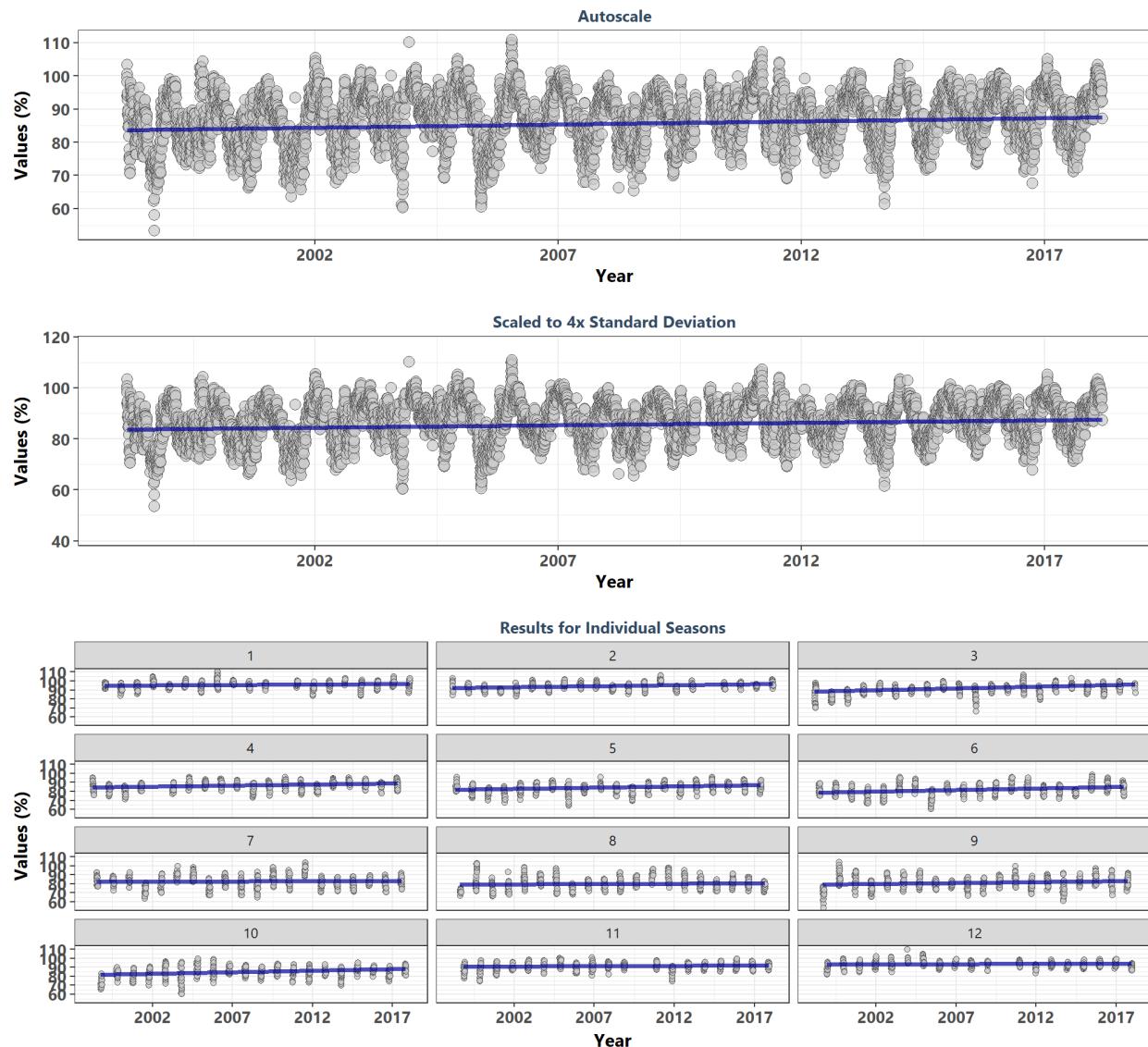
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	7024	82.60	-0.0680	-0.1189	82.5378	-8.6	0.0000	110.3	0	-1
1	649	93.14	-0.1294	-0.1734	94.8708	-4.9	0.0000	NA	NA	-1
2	573	93.61	-0.2252	-0.2771	96.6542	-8.1	0.0000	NA	NA	-1
3	615	90.06	0.0378	0.0494	89.5652	1.4	0.1601	NA	NA	1
4	578	85.53	0.0472	0.0635	84.8995	1.7	0.0895	NA	NA	1
5	545	80.60	0.0201	0.0357	80.1762	0.7	0.4828	NA	NA	1
6	549	74.30	-0.1244	-0.2736	77.3058	-4.4	0.0000	NA	NA	-1
7	543	72.05	-0.0621	-0.1572	73.7761	-2.2	0.0302	NA	NA	-1
8	582	65.90	-0.0963	-0.2908	68.8081	-3.5	0.0005	NA	NA	-1
9	559	67.08	0.0013	0.0037	67.0454	0.0	0.9642	NA	NA	1
10	634	71.65	-0.0152	-0.0392	72.0390	-0.6	0.5661	NA	NA	-1
11	597	83.99	-0.0927	-0.1562	85.5563	-3.4	0.0007	NA	NA	-1
12	600	88.46	-0.1724	-0.2281	90.8498	-6.3	0.0000	NA	NA	-1

<sup>a</sup> p < 0.00005 appear as 0 due to rounding

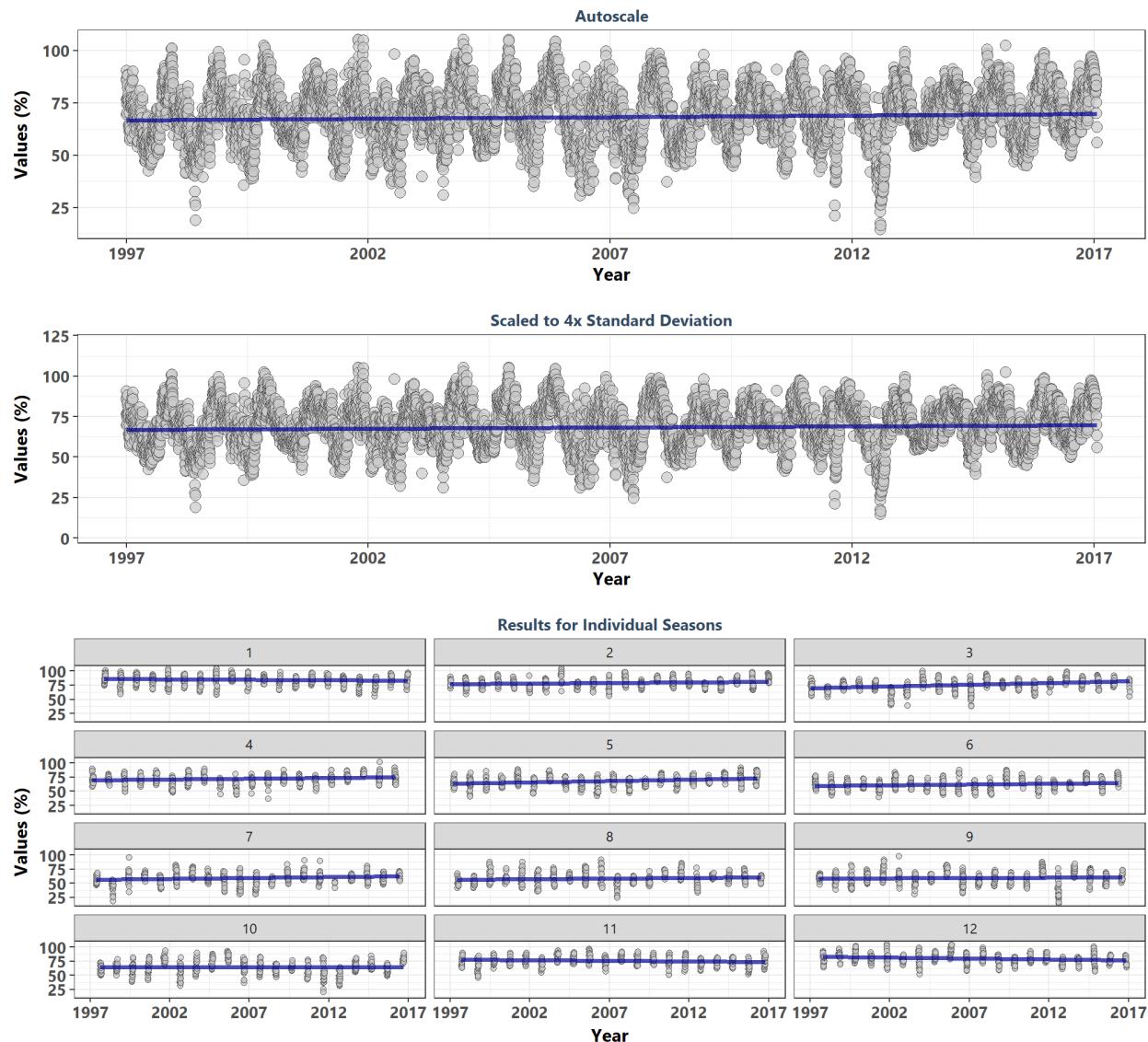
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmsswq**



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	6855	87.81	0.1440	0.1963	83.3877	18.0	0.0000	115.2	0	1
1	564	95.68	0.1099	0.1086	94.4845	3.9	0.0001	NA	NA	1
2	526	94.40	0.2260	0.2246	92.1536	7.8	0.0000	NA	NA	1
3	618	91.71	0.2925	0.3884	87.8218	10.9	0.0000	NA	NA	1
4	546	86.95	0.1794	0.2260	84.4625	6.3	0.0000	NA	NA	1
5	580	84.12	0.2038	0.2736	81.3877	7.4	0.0000	NA	NA	1
6	580	82.09	0.1978	0.3229	78.5391	7.1	0.0000	NA	NA	1
7	584	82.79	0.0237	0.0434	82.3128	0.9	0.3909	NA	NA	1
8	574	80.10	0.0389	0.0733	79.3697	1.4	0.1632	NA	NA	1
9	591	81.08	0.1222	0.2230	78.6301	4.4	0.0000	NA	NA	1
10	612	84.89	0.2116	0.3422	81.1229	7.8	0.0000	NA	NA	1
11	534	91.36	0.0772	0.0861	90.5029	2.7	0.0076	NA	NA	1
12	546	93.54	0.0278	0.0267	93.2707	1.0	0.3302	NA	NA	1

<sup>a</sup> p < 0.00005 appear as 0 due to rounding

**Pellicer Creek Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	7006	70.69	0.0602	0.1493	66.8289	7.5	0.0000	198.7	0	1
1	604	84.52	-0.0658	-0.1457	85.9760	-2.4	0.0155	NA	NA	-1
2	528	79.39	0.0867	0.1931	77.2649	3.0	0.0029	NA	NA	1
3	615	76.12	0.2582	0.5992	70.1331	9.6	0.0000	NA	NA	1
4	583	72.02	0.1252	0.2785	69.2386	4.5	0.0000	NA	NA	1
5	588	68.15	0.1850	0.4493	63.6585	6.7	0.0000	NA	NA	1
6	564	61.79	0.1071	0.2679	59.1092	3.8	0.0001	NA	NA	1
7	575	59.06	0.1249	0.3047	56.3178	4.5	0.0000	NA	NA	1
8	570	58.64	0.0878	0.2009	56.8298	3.1	0.0017	NA	NA	1
9	577	59.38	0.0470	0.1464	58.0611	1.7	0.0910	NA	NA	1
10	614	64.47	0.0017	0.0056	64.4192	0.1	0.9511	NA	NA	1
11	582	76.03	-0.0854	-0.1994	77.8204	-3.1	0.0020	NA	NA	-1
12	606	79.87	-0.1411	-0.3265	82.8080	-5.2	0.0000	NA	NA	-1

<sup>a</sup> p < 0.00005 appear as 0 due to rounding

## 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

```
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$MonitoringID==Mon_IDs[i]]  
    Mon_name <- paste0(KT.Stats$ProgramID[KT.Stats$MonitoringID==Mon_IDs[i]],  
                       " | ", KT.Stats$ProgramName[KT.Stats$MonitoringID==Mon_IDs[i]], "\n",  
                       KT.Stats$ProgramLocationID[KT.Stats$MonitoringID==Mon_IDs[i]])  
  
    ##Year plots  
    p1 <- ggplot(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

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(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(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(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())

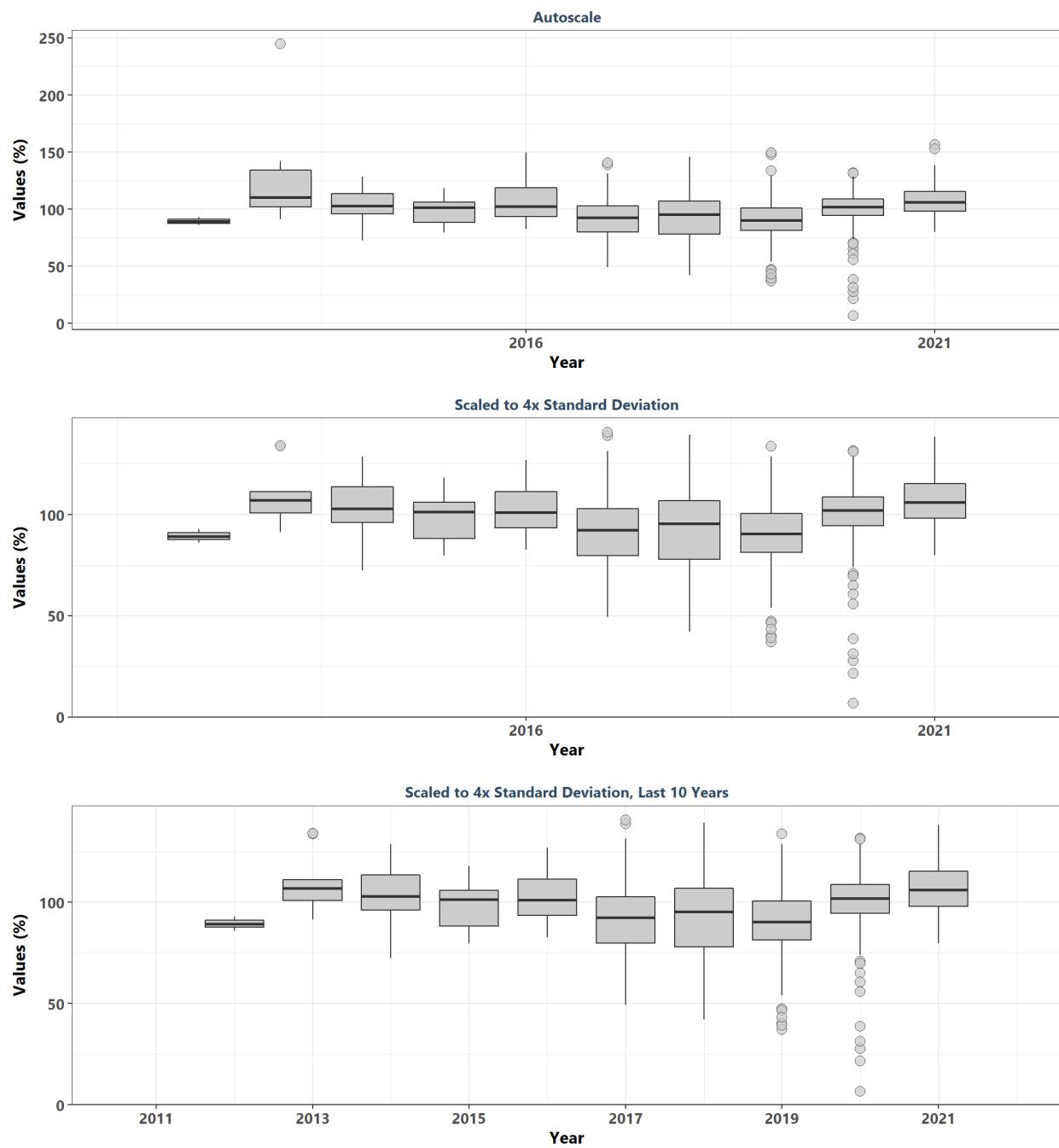
print(ggarrange(p0, Yset, ncol=1, heights=c(0.1, 1)))
print(ggarrange(p00, YMset, ncol=1, heights=c(0.1, 1)))
print(ggarrange(p000, Mset, ncol=1, heights=c(0.1, 1)))

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

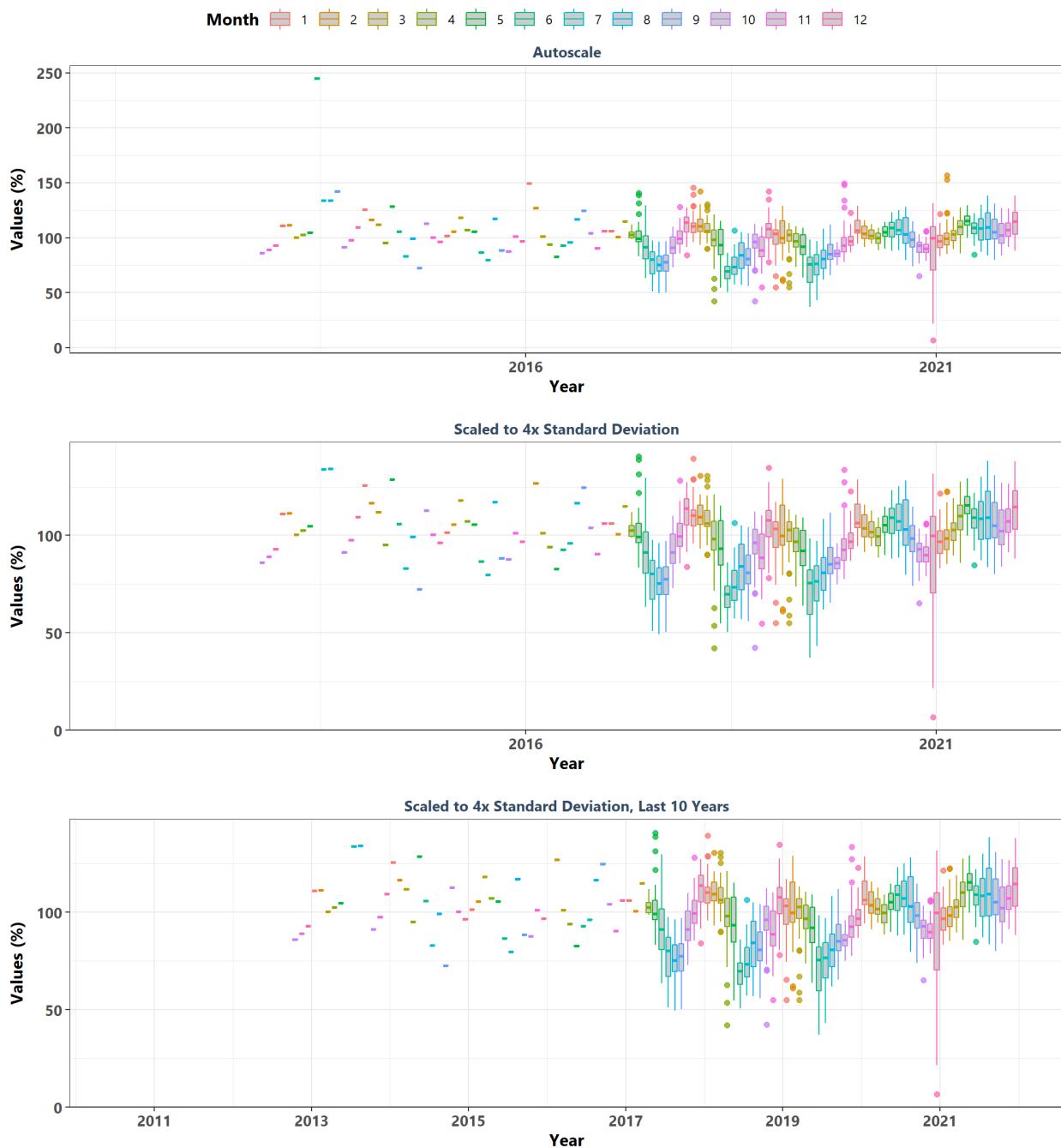
```

```
}
```

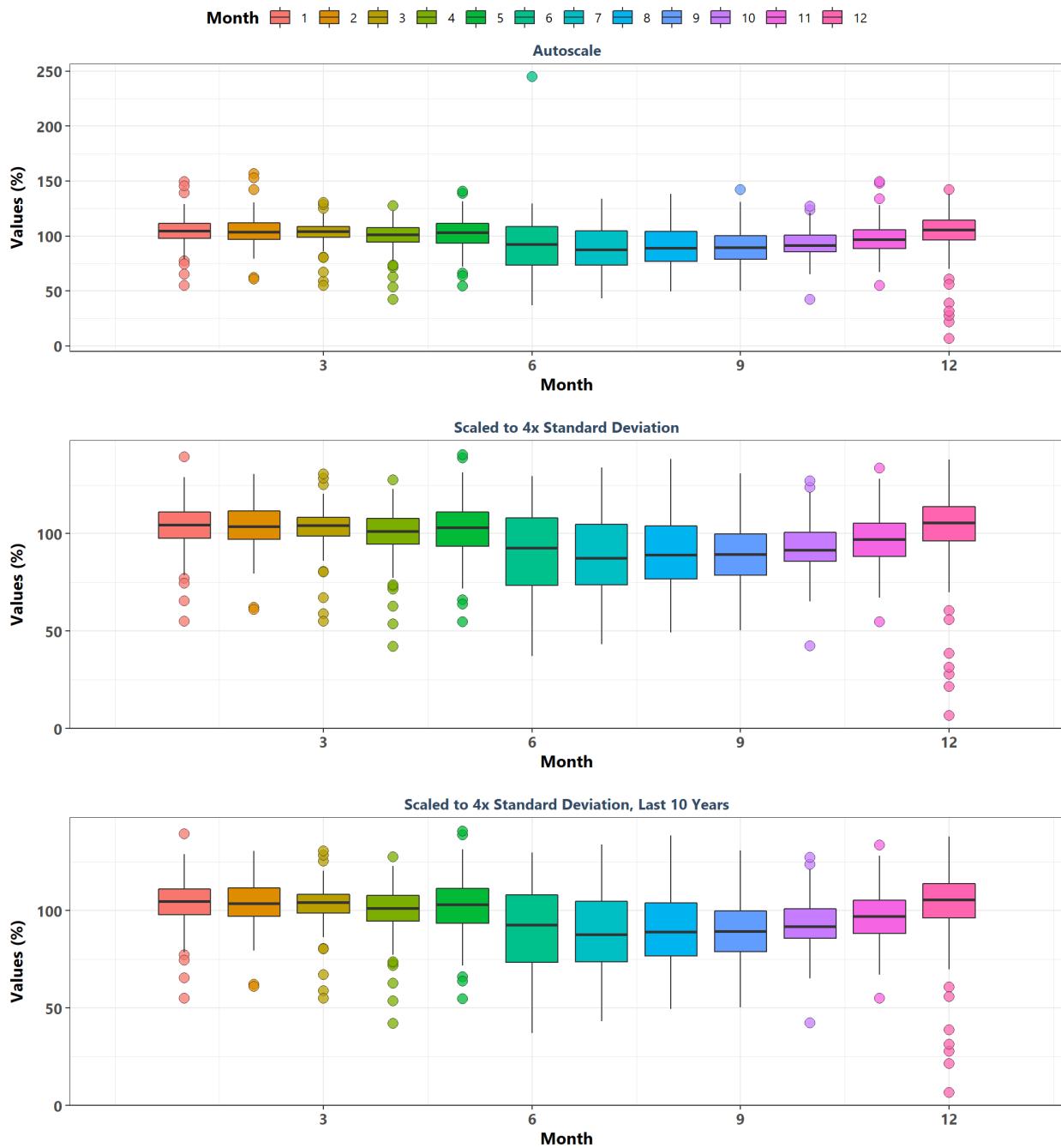
**Banana River Aquatic Preserve**  
**5061 | St. Johns River Water Management District Continuous Water Quality Programs**  
**IRLB04**  
By Year



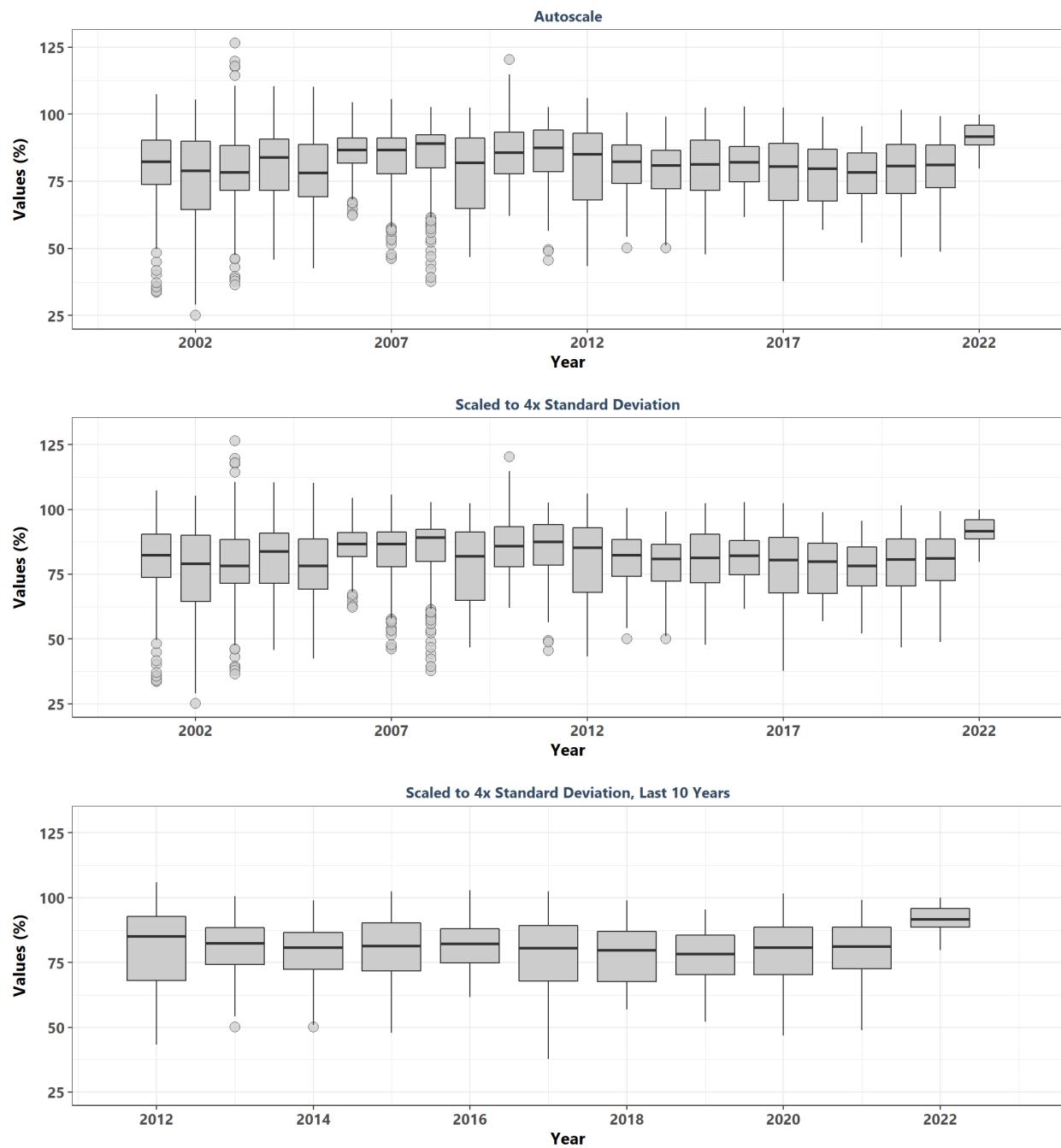
**Banana River Aquatic Preserve**  
**5061 | St. Johns River Water Management District Continuous Water Quality Programs**  
**IRLB04**  
**By Year & Month**



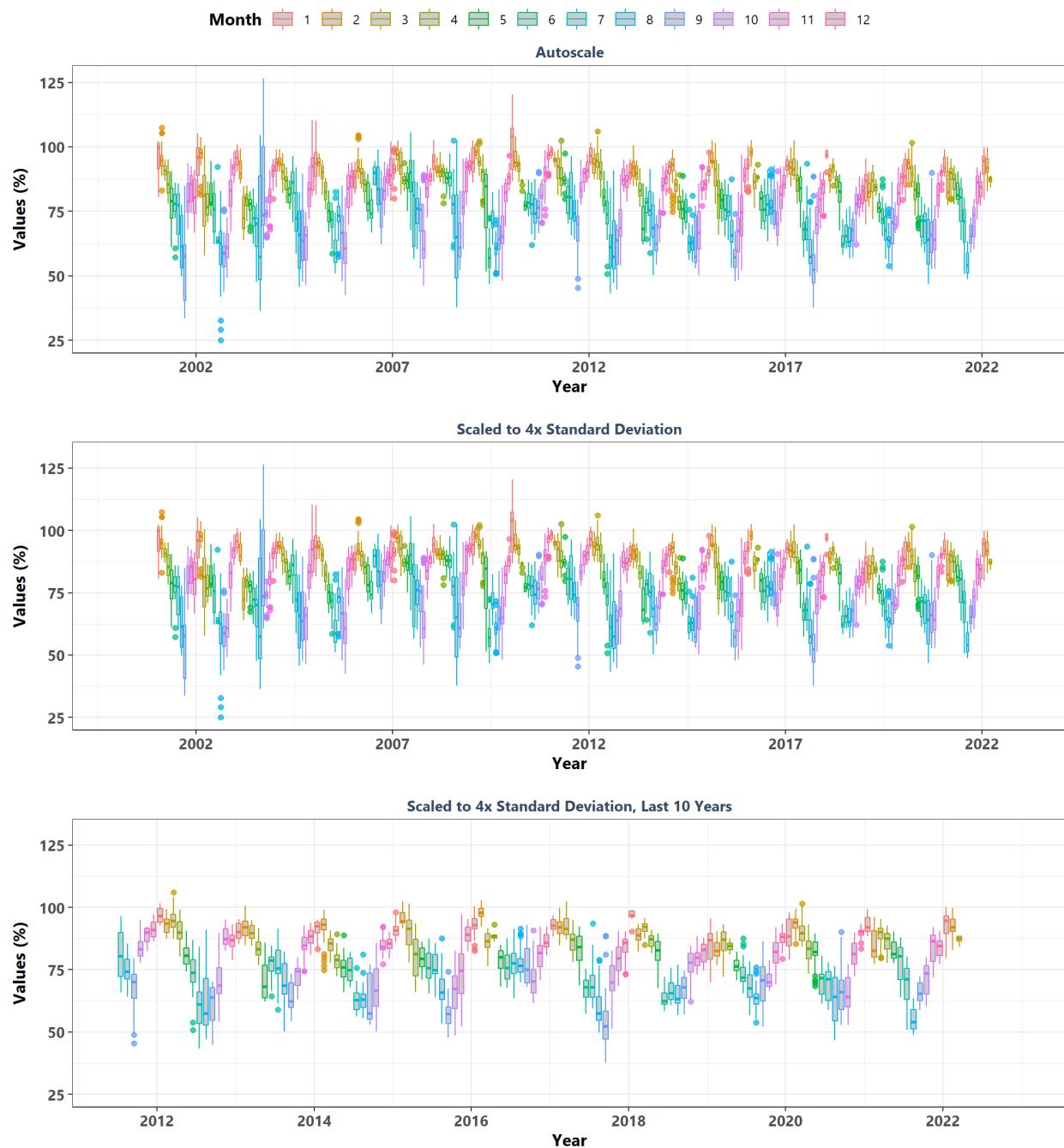
**Banana River Aquatic Preserve**  
**5061 | St. Johns River Water Management District Continuous Water Quality Programs**  
**IRLB04**  
**By Month**



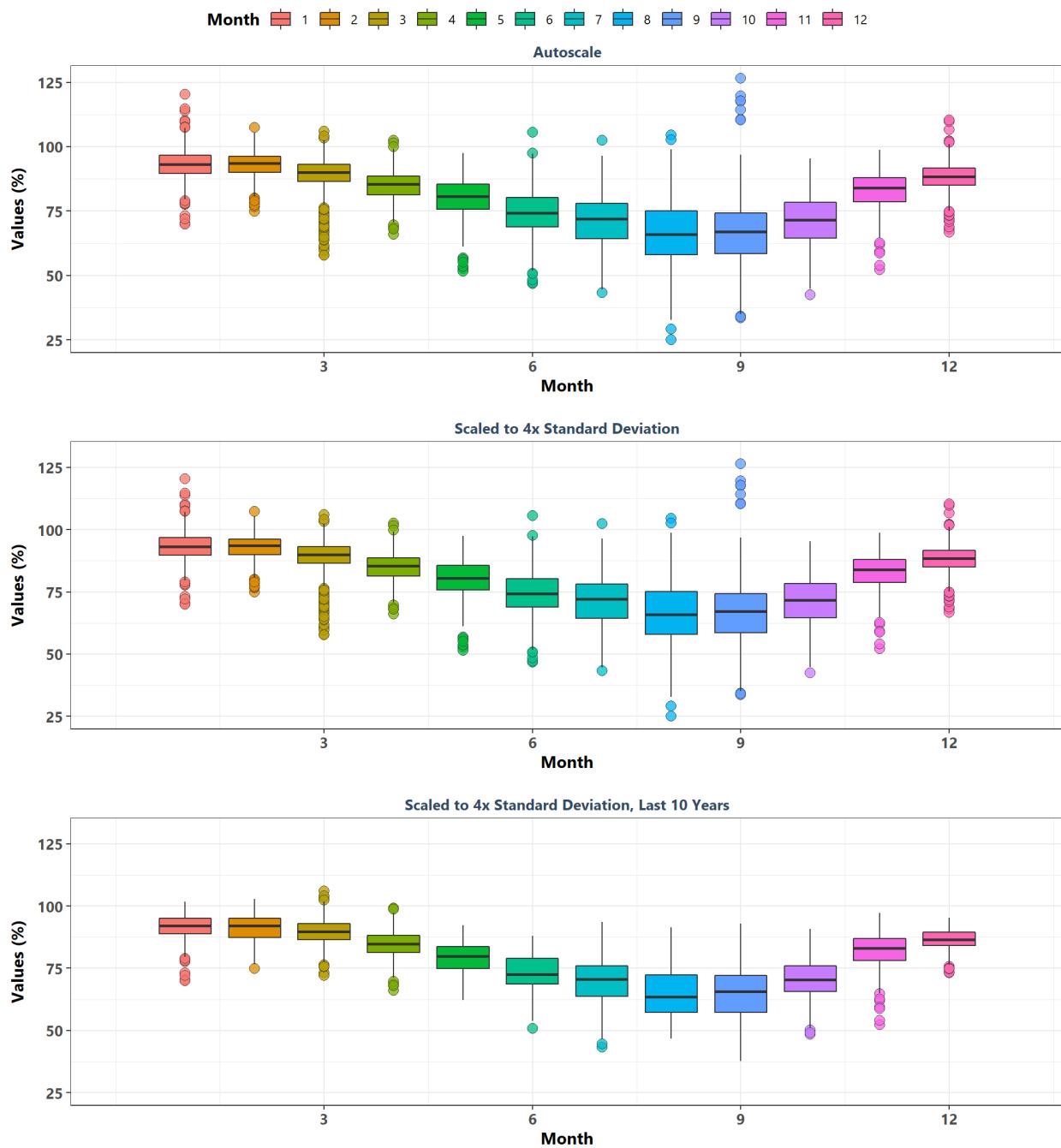
**Guana River Marsh Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**  
**By Year**



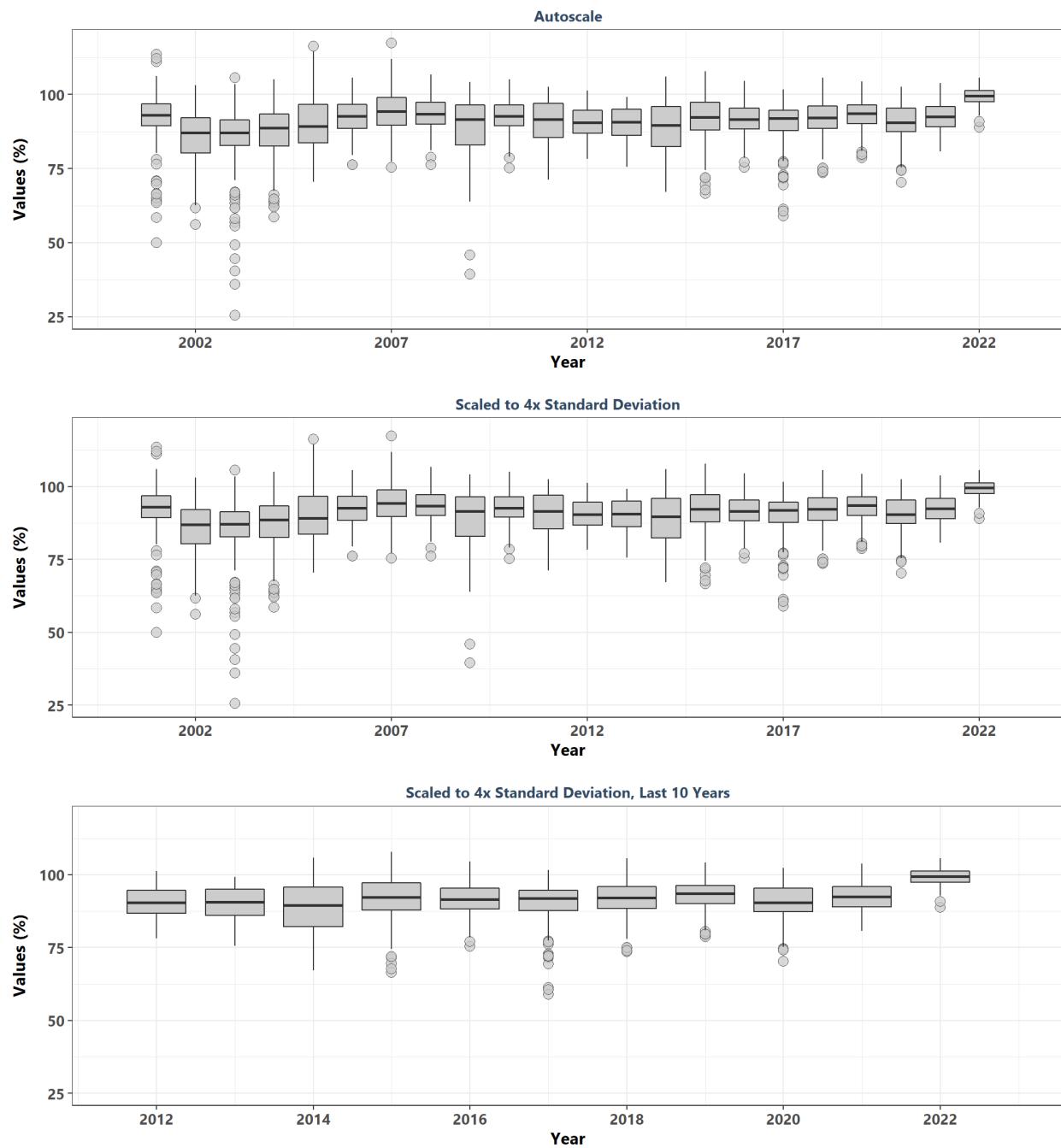
**Guana River Marsh Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**  
**By Year & Month**



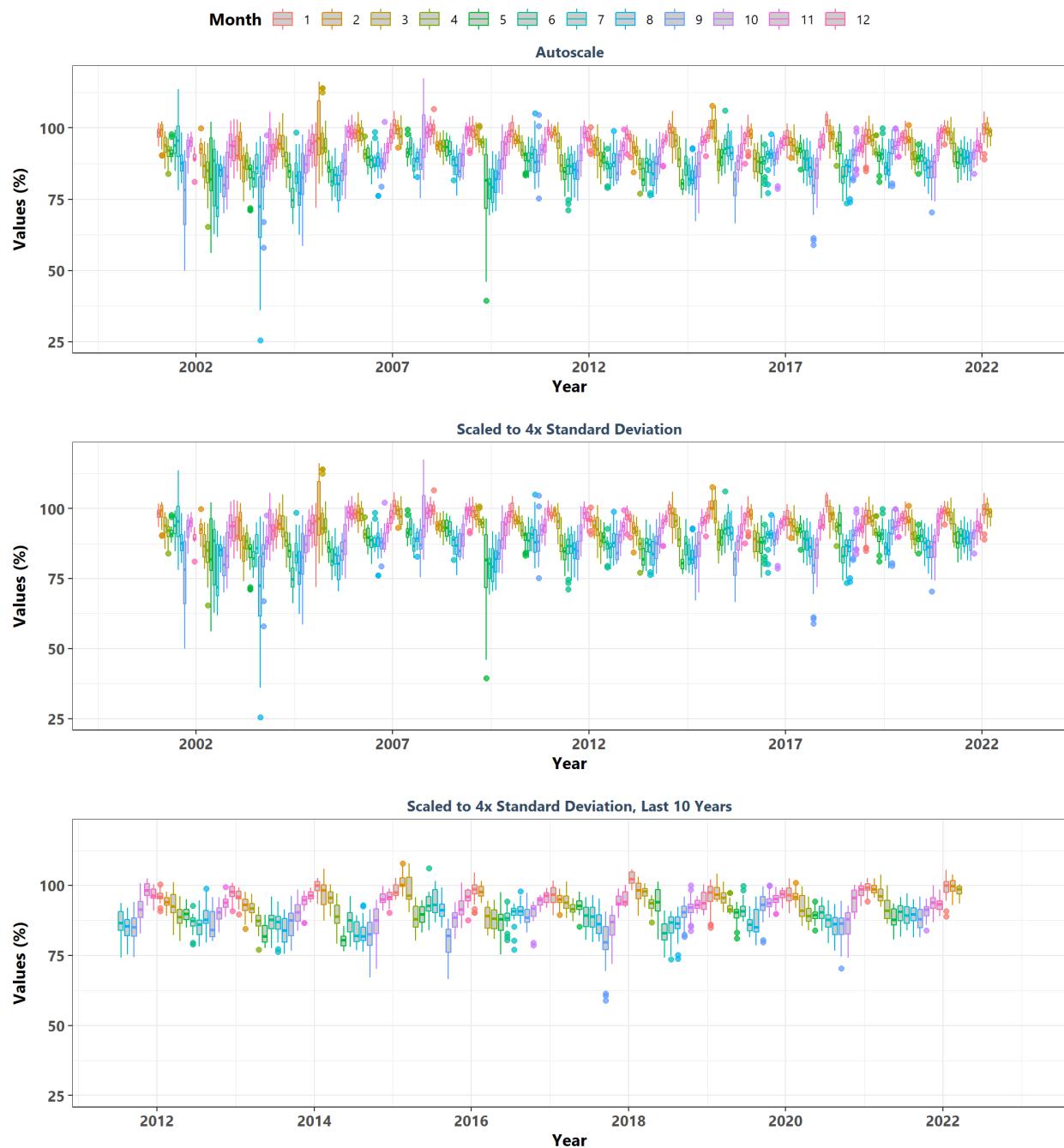
**Guana River Marsh Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**  
**By Month**



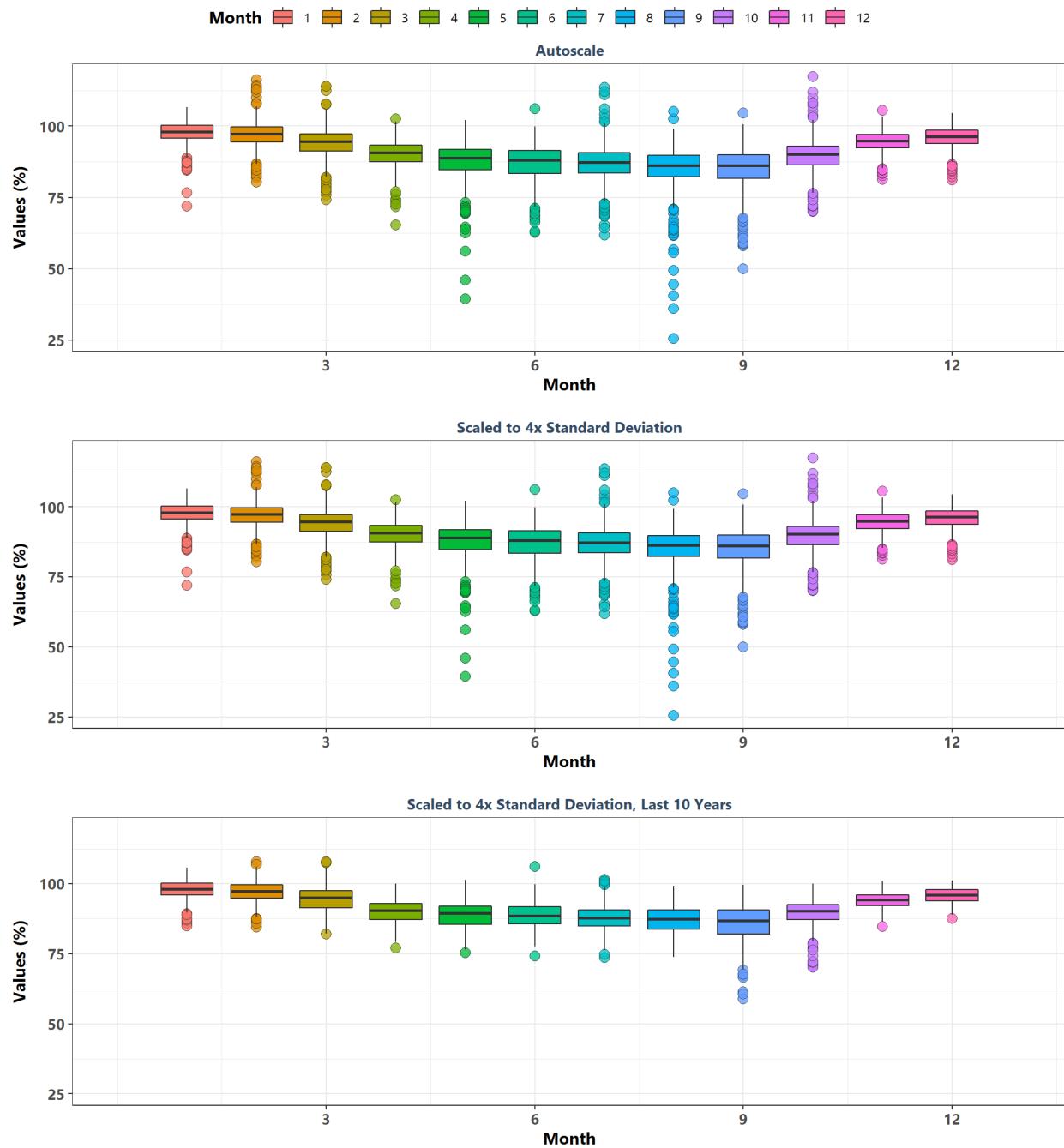
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmfmwq**  
**By Year**



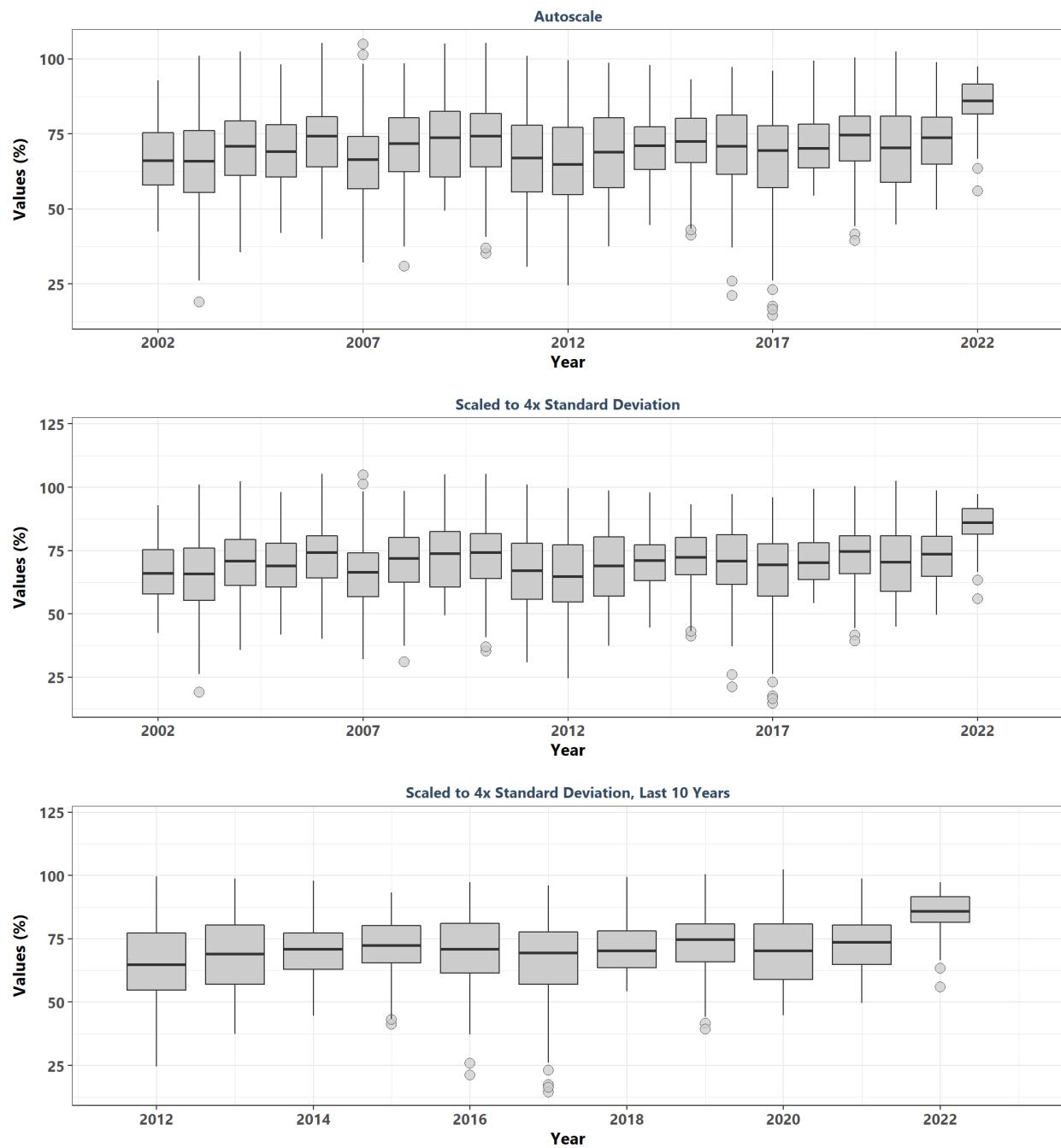
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmfmwq**  
**By Year & Month**



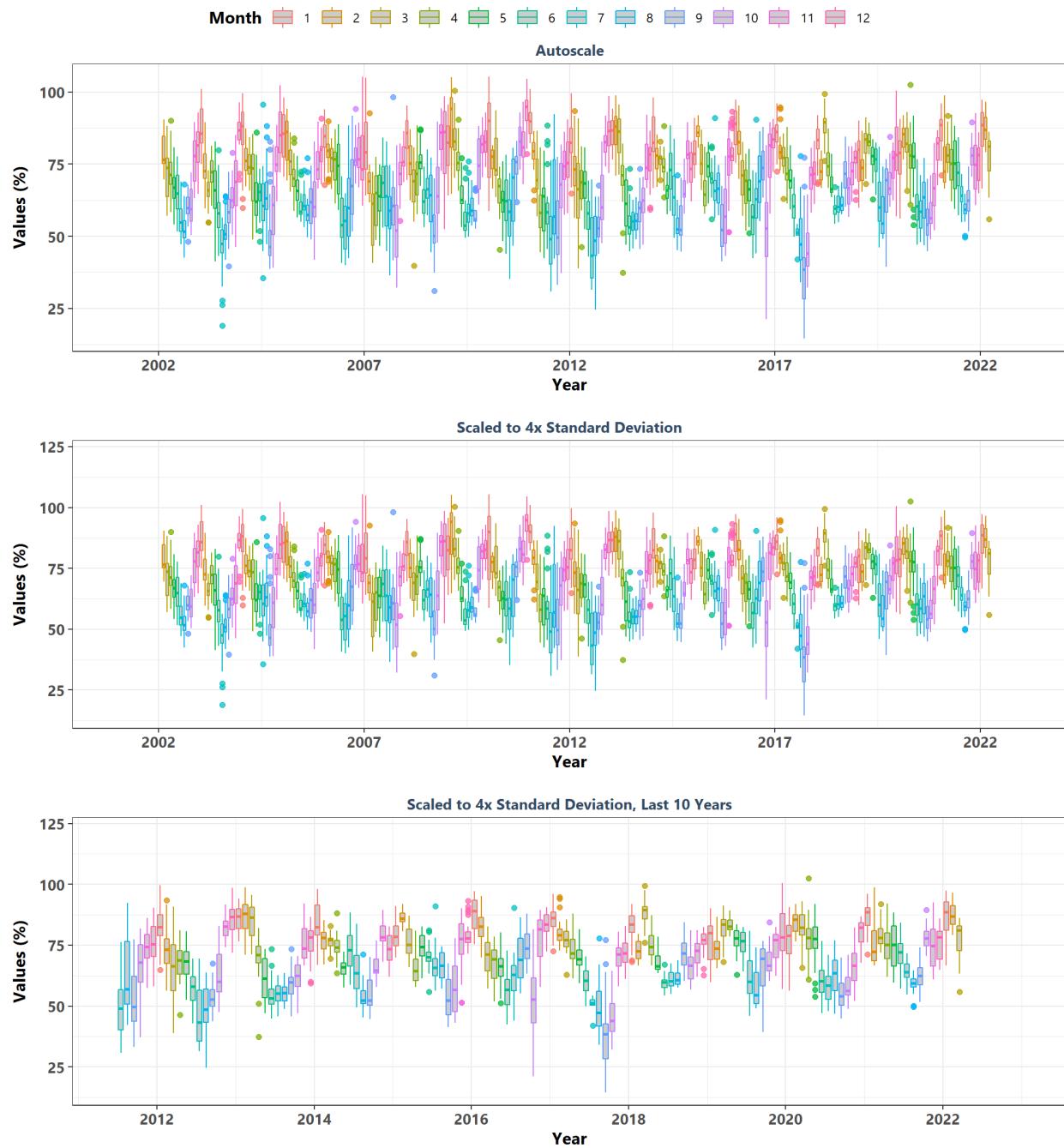
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmfmwq**  
**By Month**



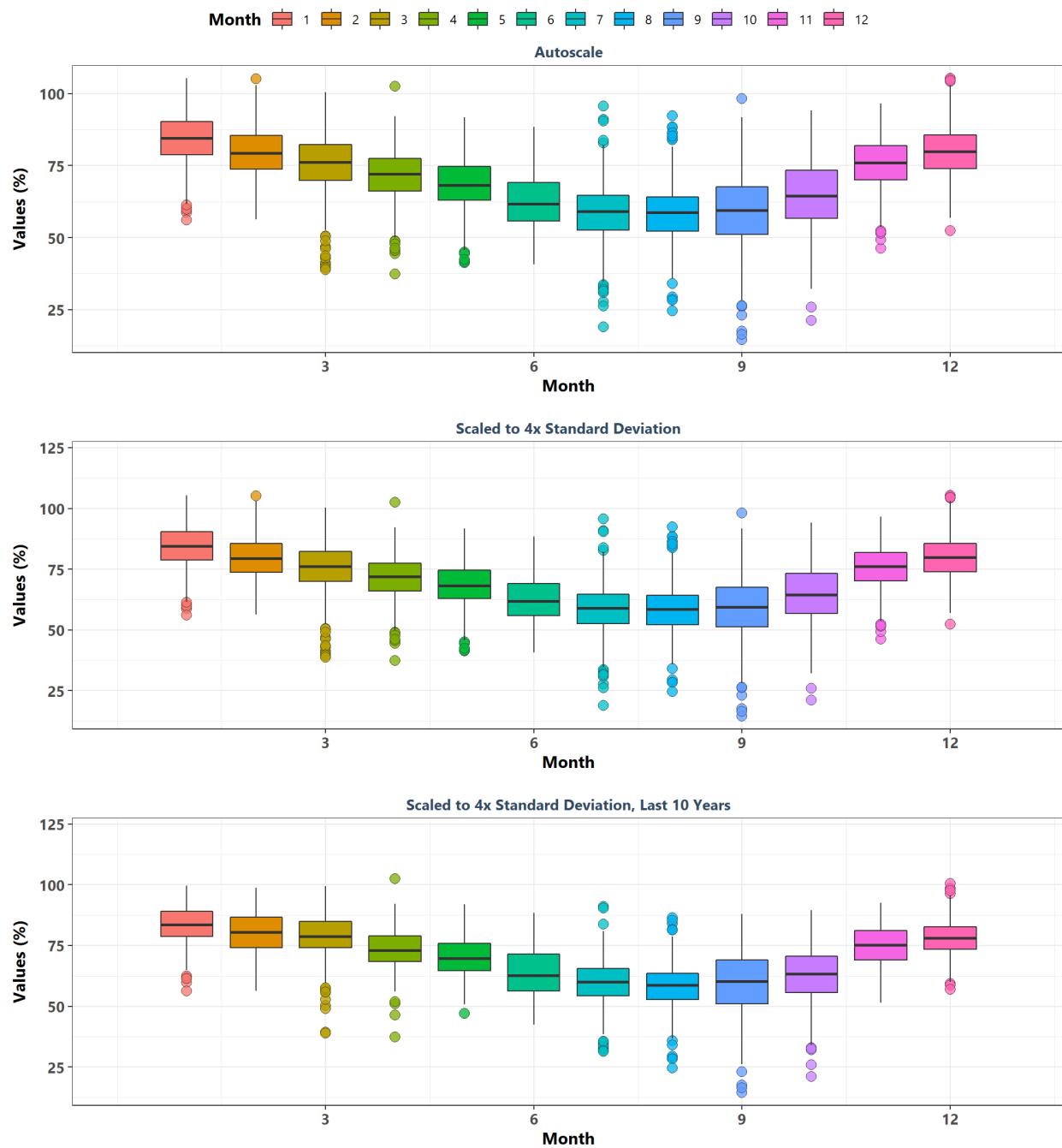
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**  
**By Year**



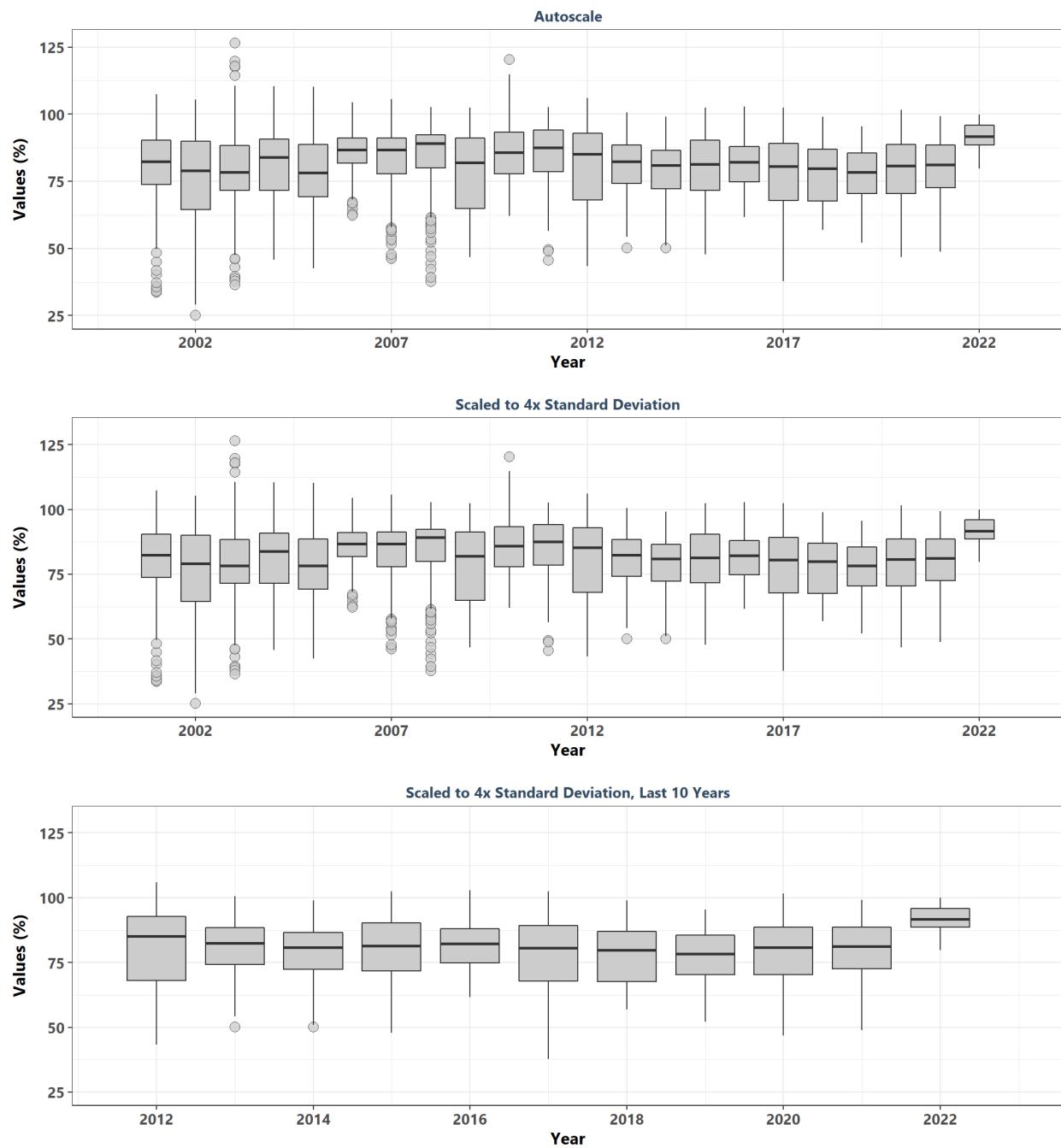
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**  
**By Year & Month**



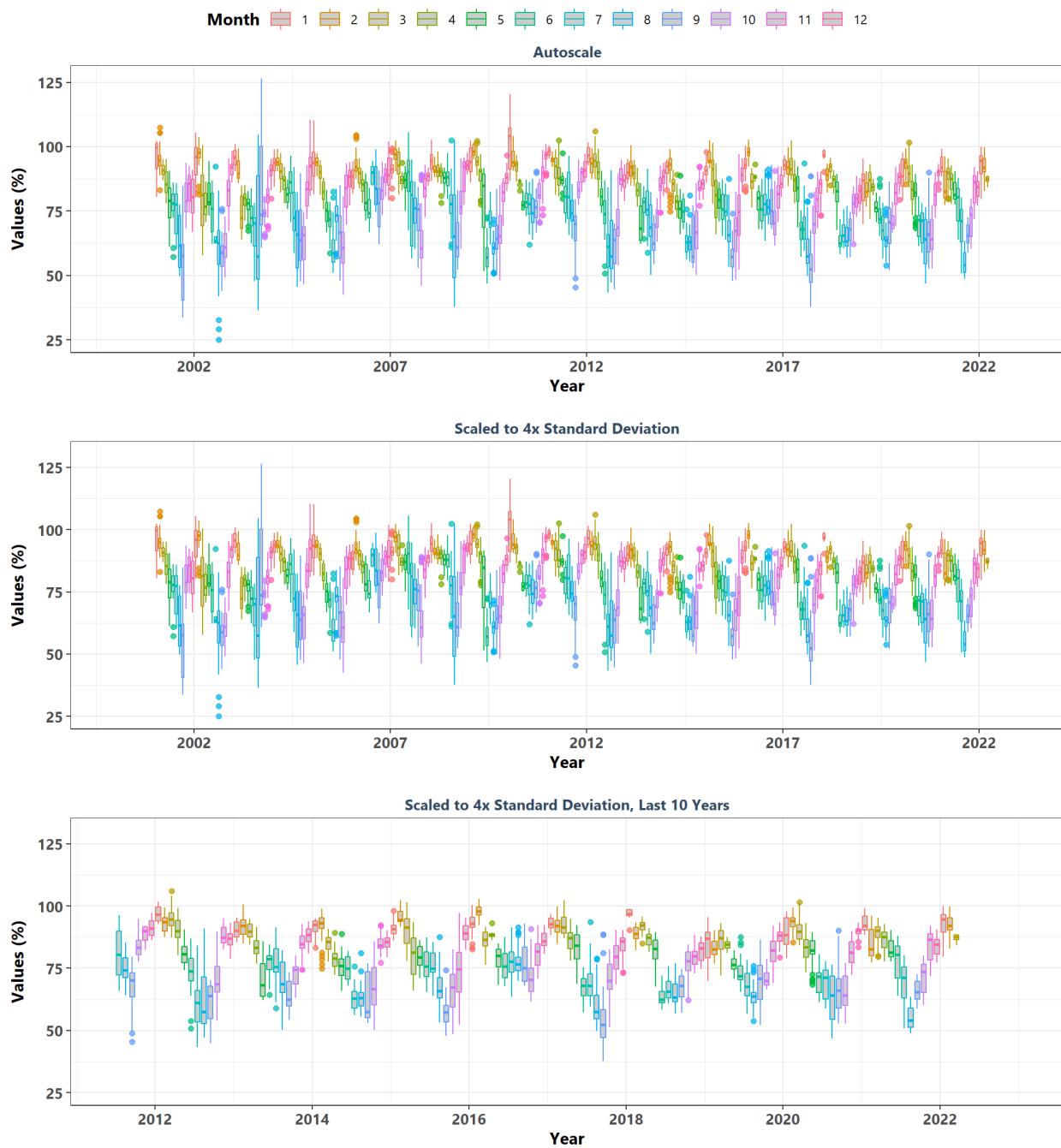
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**  
**By Month**



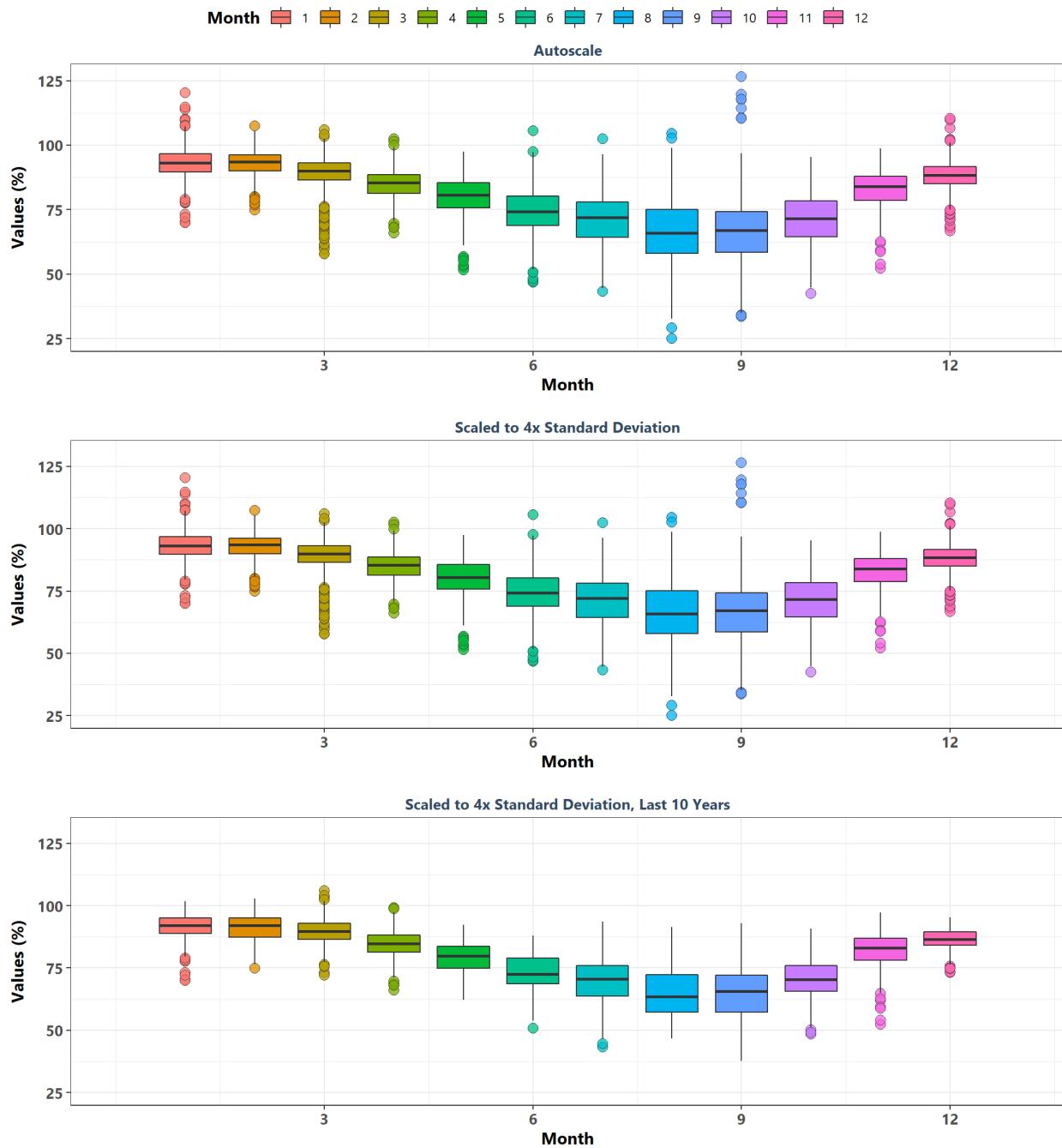
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**  
**By Year**



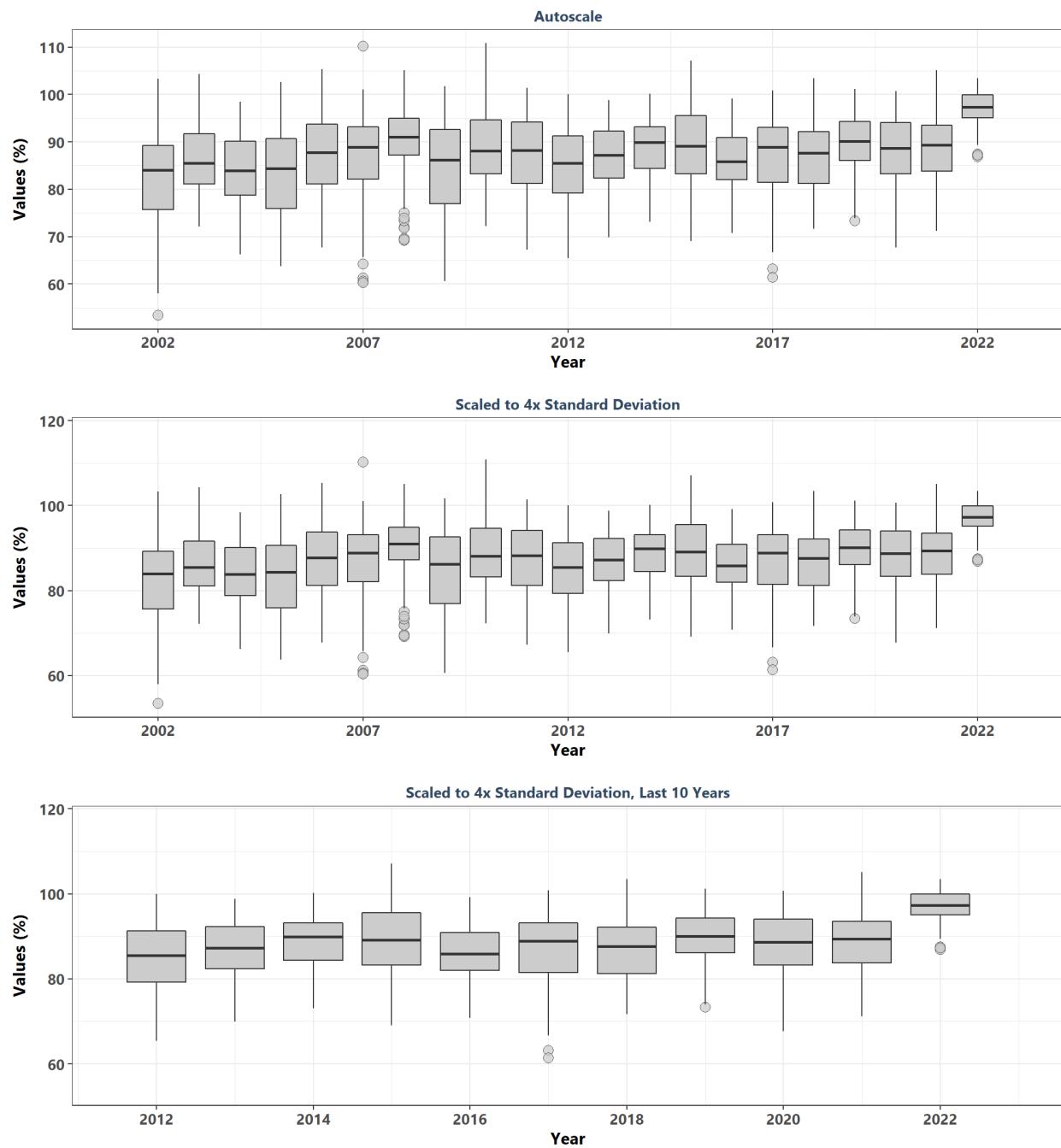
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**  
By Year & Month



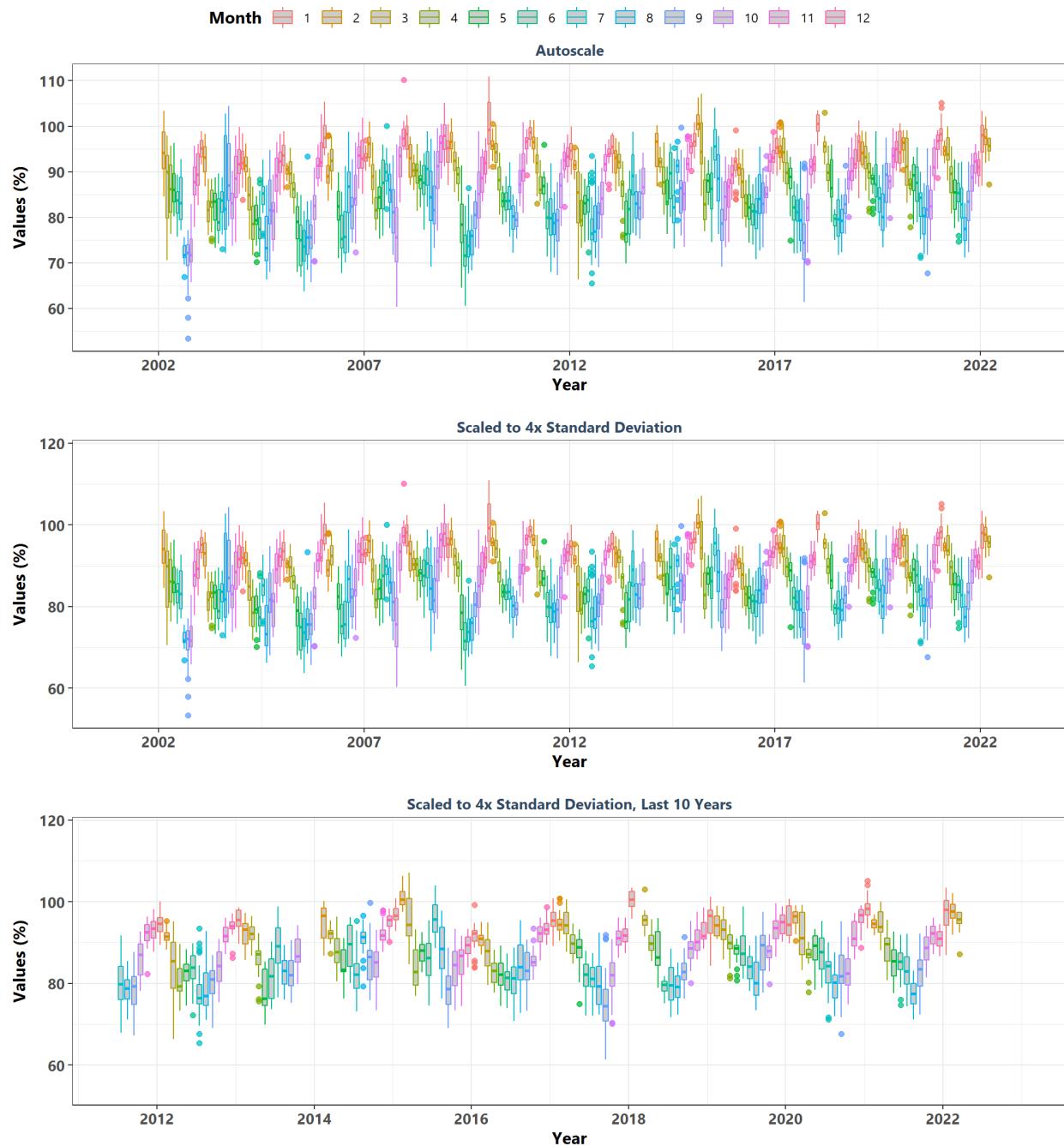
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpiwq**  
By Month



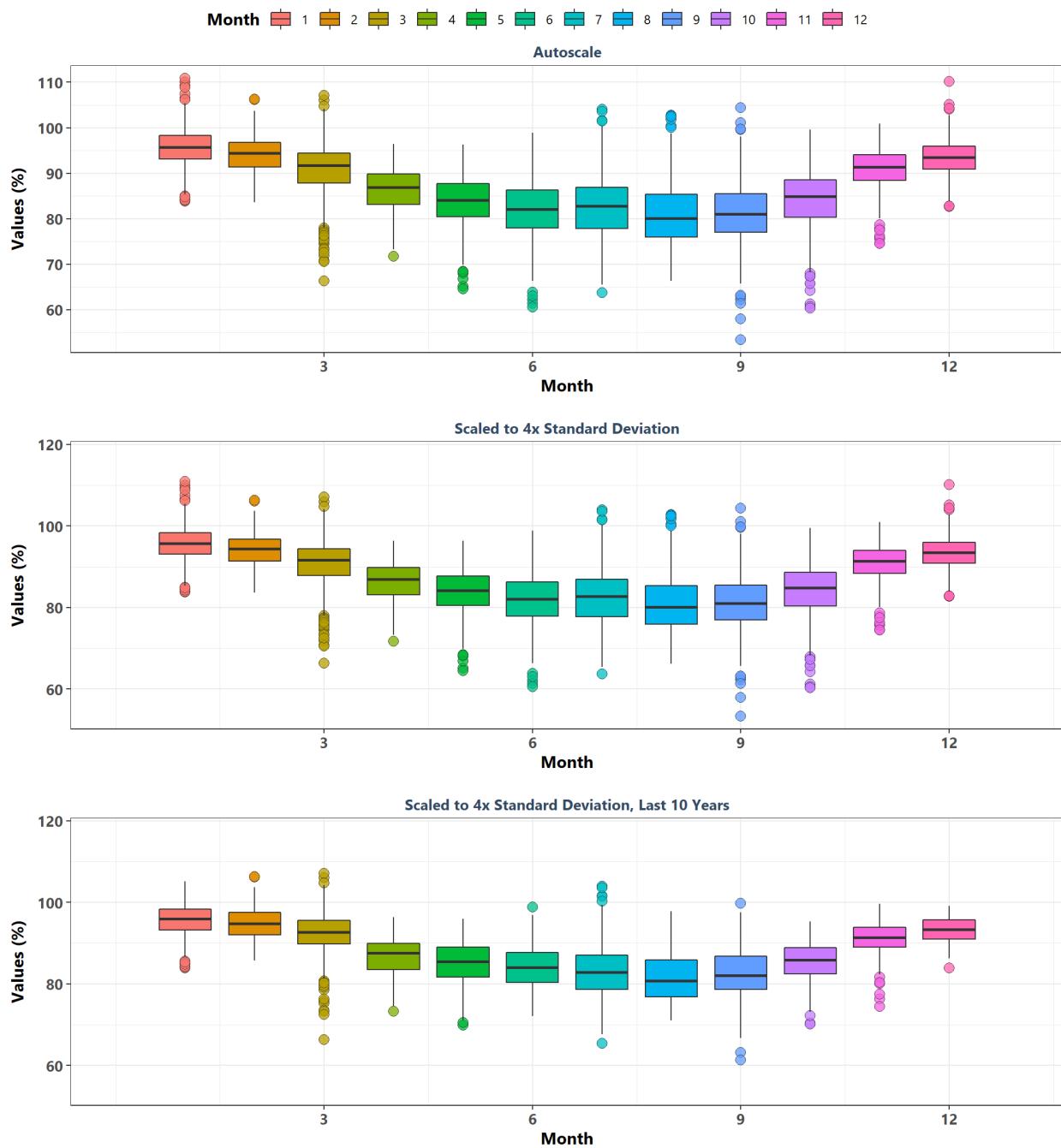
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmsswq**  
**By Year**



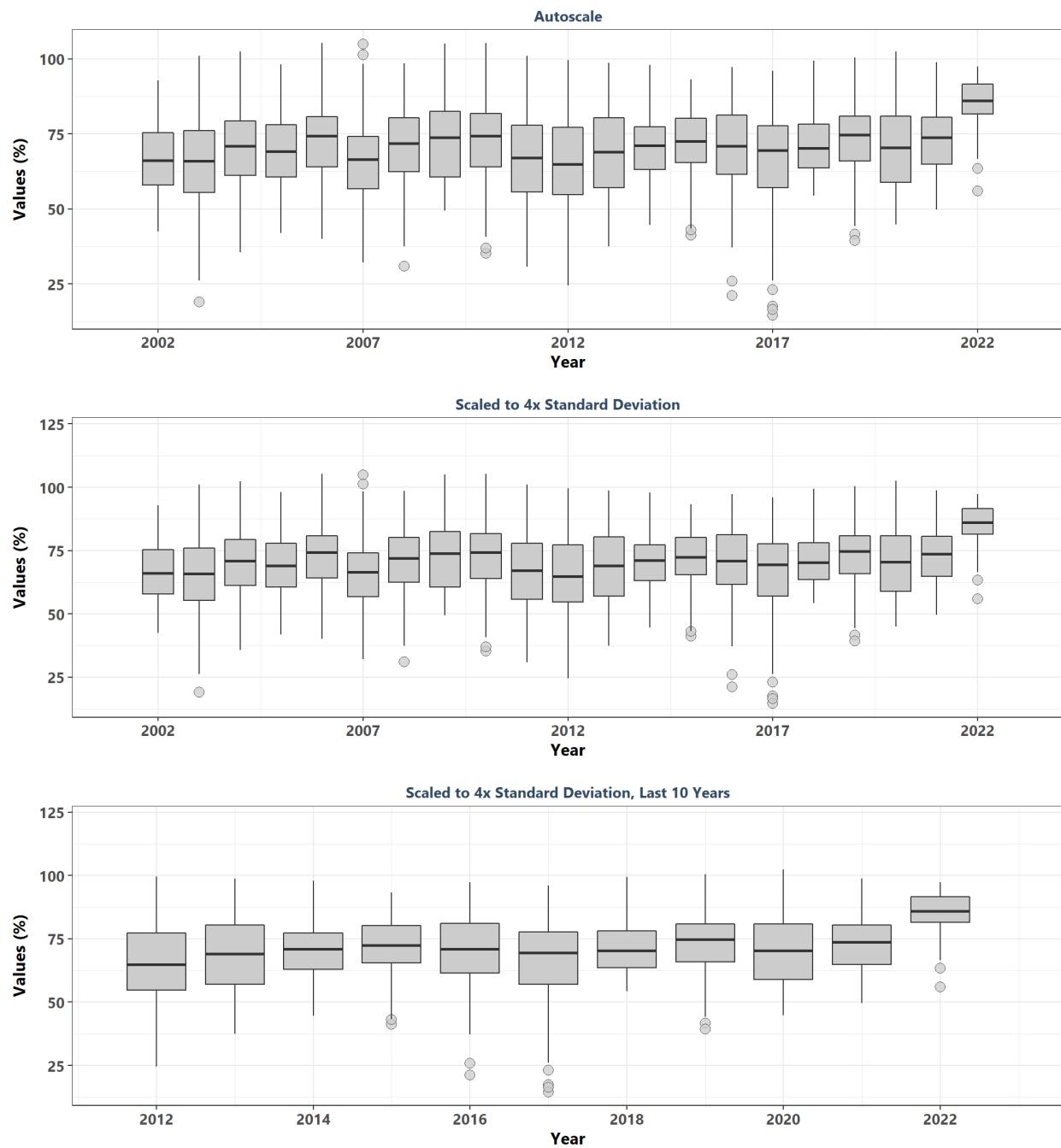
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmsswq**  
**By Year & Month**



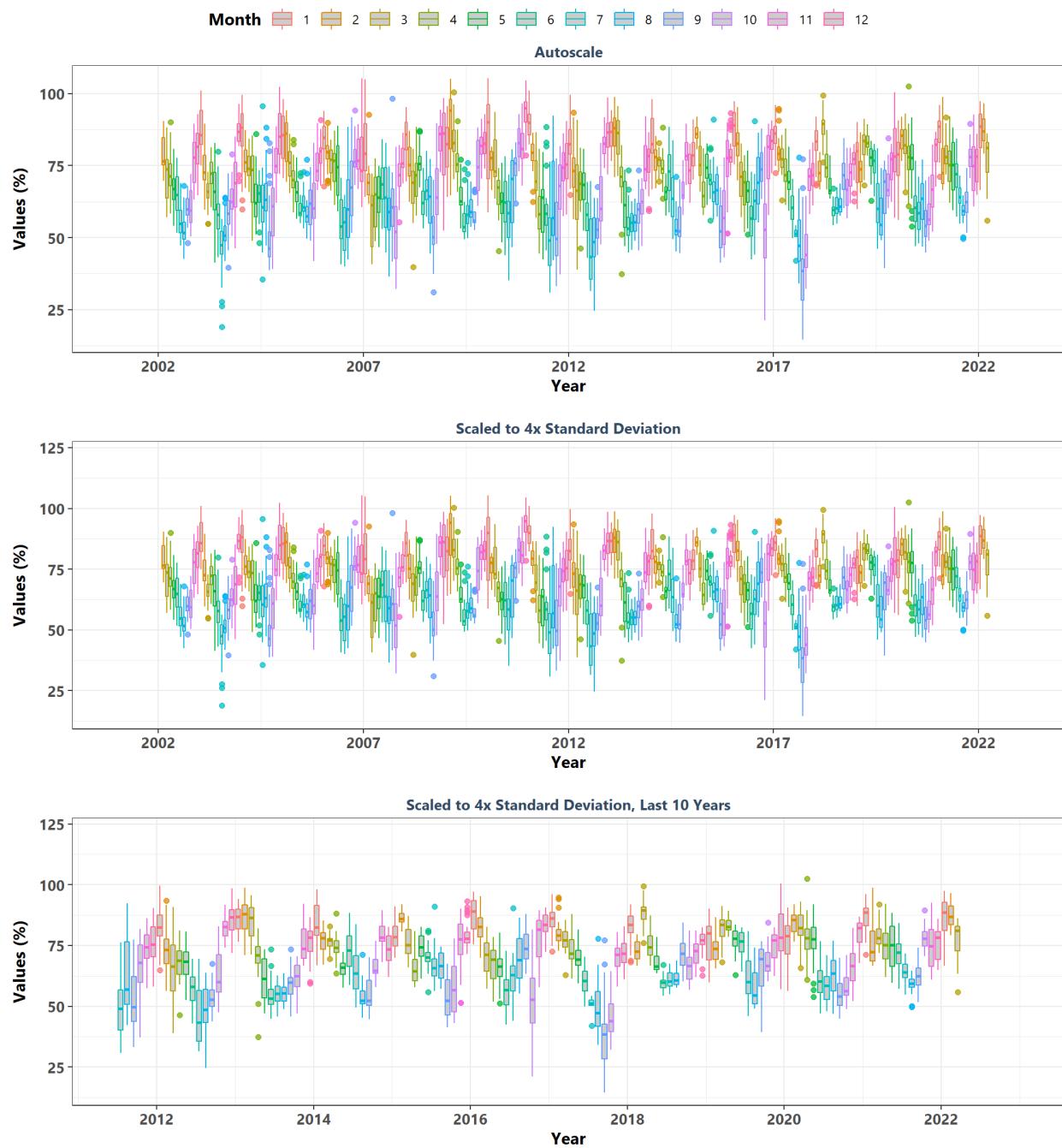
**Guana Tolomato Matanzas National Estuarine Research Reserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmsswq**  
**By Month**



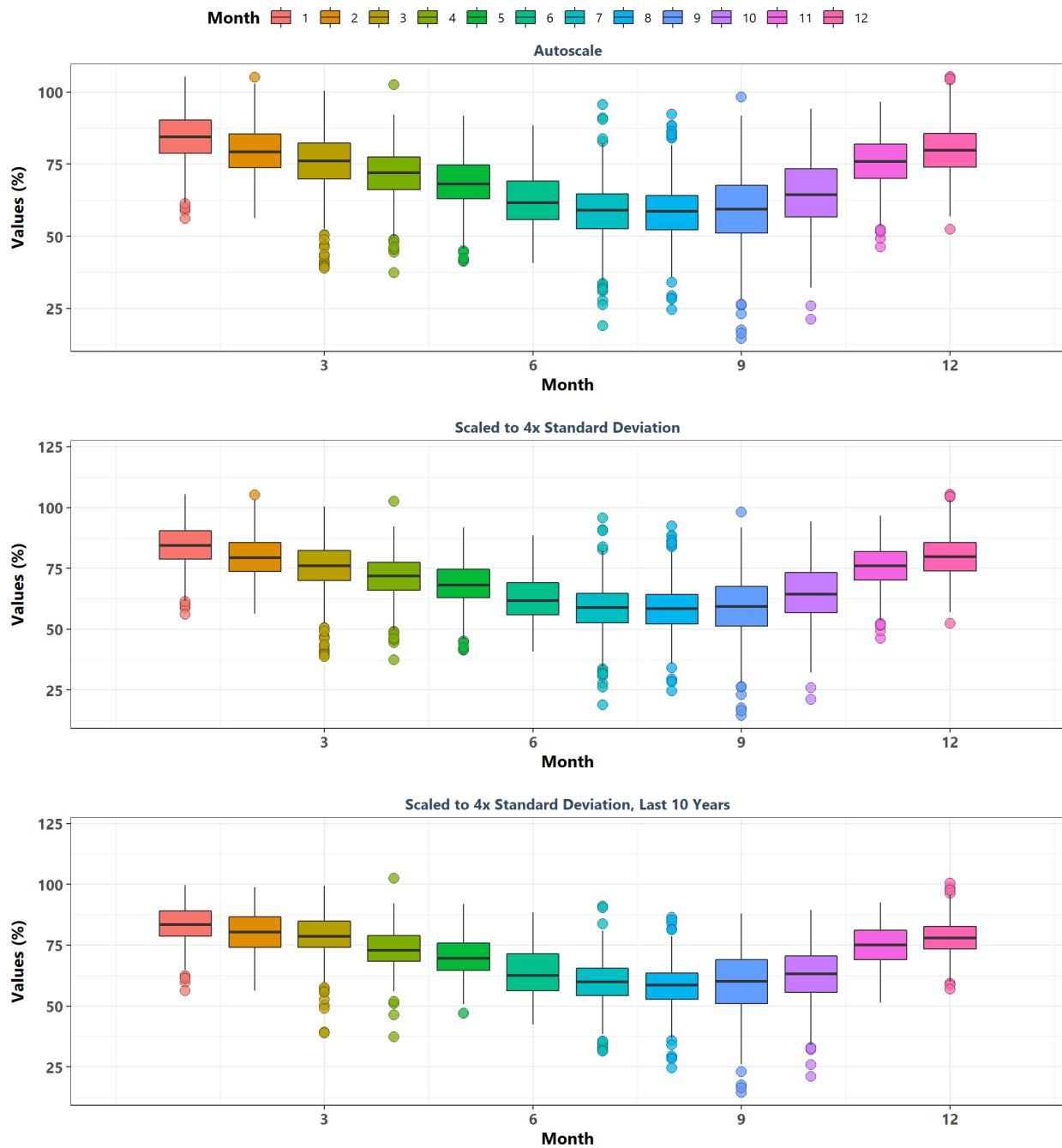
**Pellicer Creek Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**  
By Year



**Pellicer Creek Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**  
**By Year & Month**



**Pellicer Creek Aquatic Preserve**  
**4054 | Guana Tolomato Matanzas National Estuarine Research Reserve System-Wide Monitoring Program**  
**gtmpcwq**  
By Month



```
rm(list = setdiff(ls(), c("param_name", "all_regions", "file_list", "KT.Stats_all", "MA_All", "APP_Plot"))
```