SEACAR Continuous Water Quality Analysis: NW Region for Dissolved Oxygen

Last compiled on 24 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 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 `KT.Stats` for easier use while plotting.

```
tauSeasonal <- function(data, independent, stats.median, stats.minYear,
                        stats.maxYear, seasondata = Mon_M_Stats[Mon_M_Stats$ProgramLocationID==unique(da
   setDT(data)
   tau <- NULL
   tryCatch({ken <- kendallSeasonalTrendTest(
      y=data$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$ProgramLocationID[data$Monitori
  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.

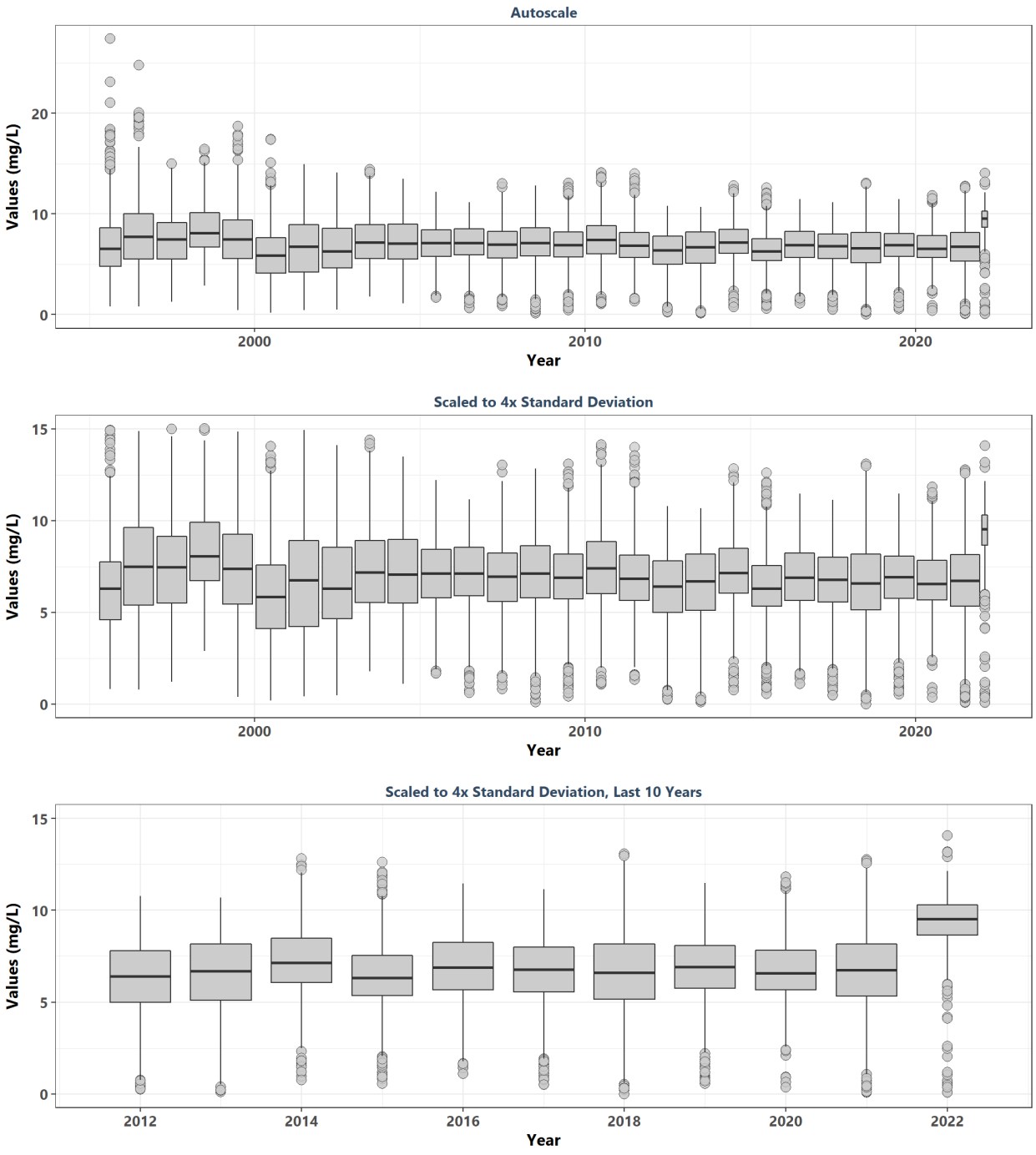
```
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Year",
        y=paste0("Values (", unit,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

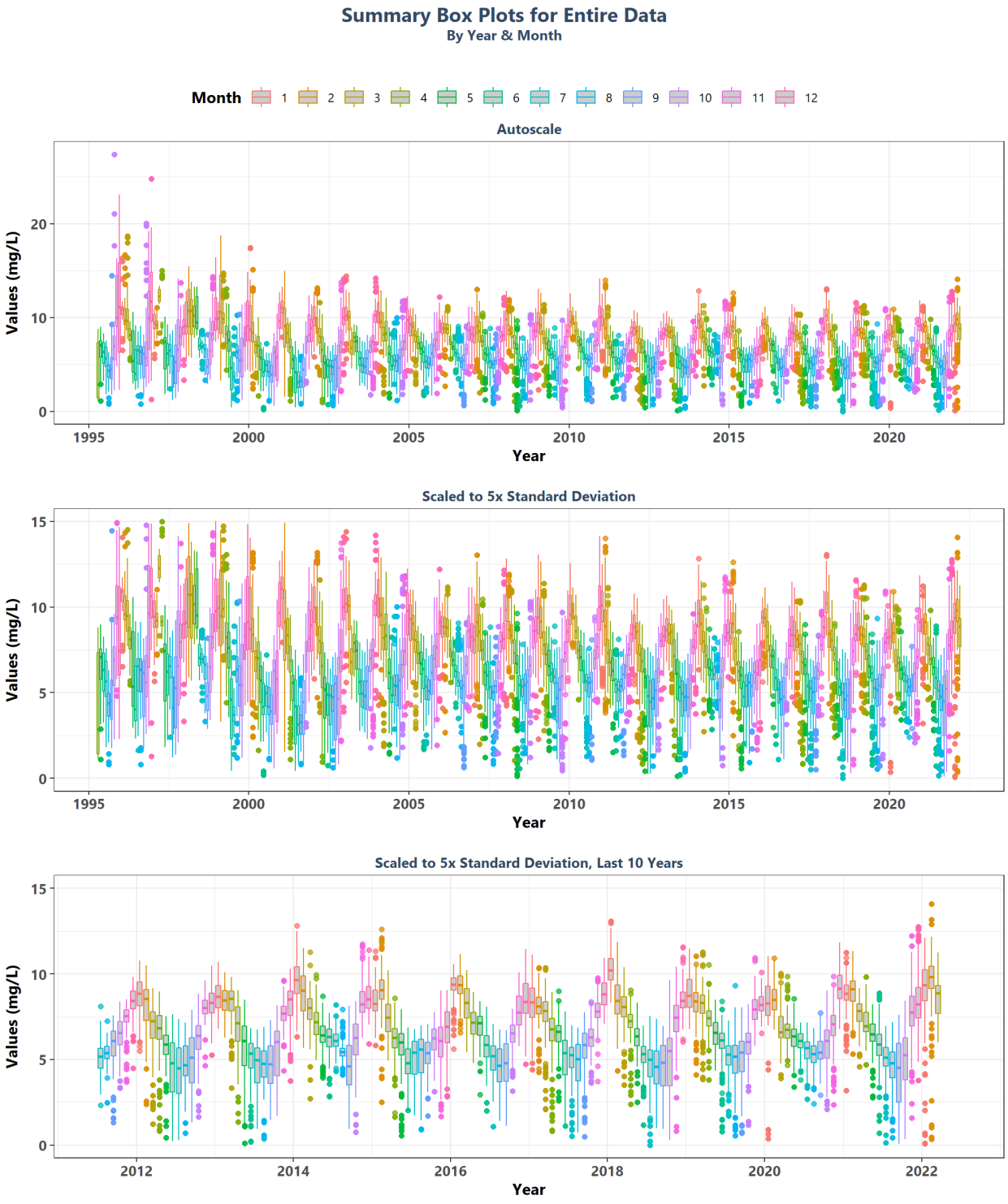
```
p3 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Scaled to 5x Standard Deviation, Last 10 Years",
        x="Year", y=paste0("Values (", unit, ")")) +
   ylim(0, y_scale) +
   scale_x_continuous(limits=c(max(data$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))
```

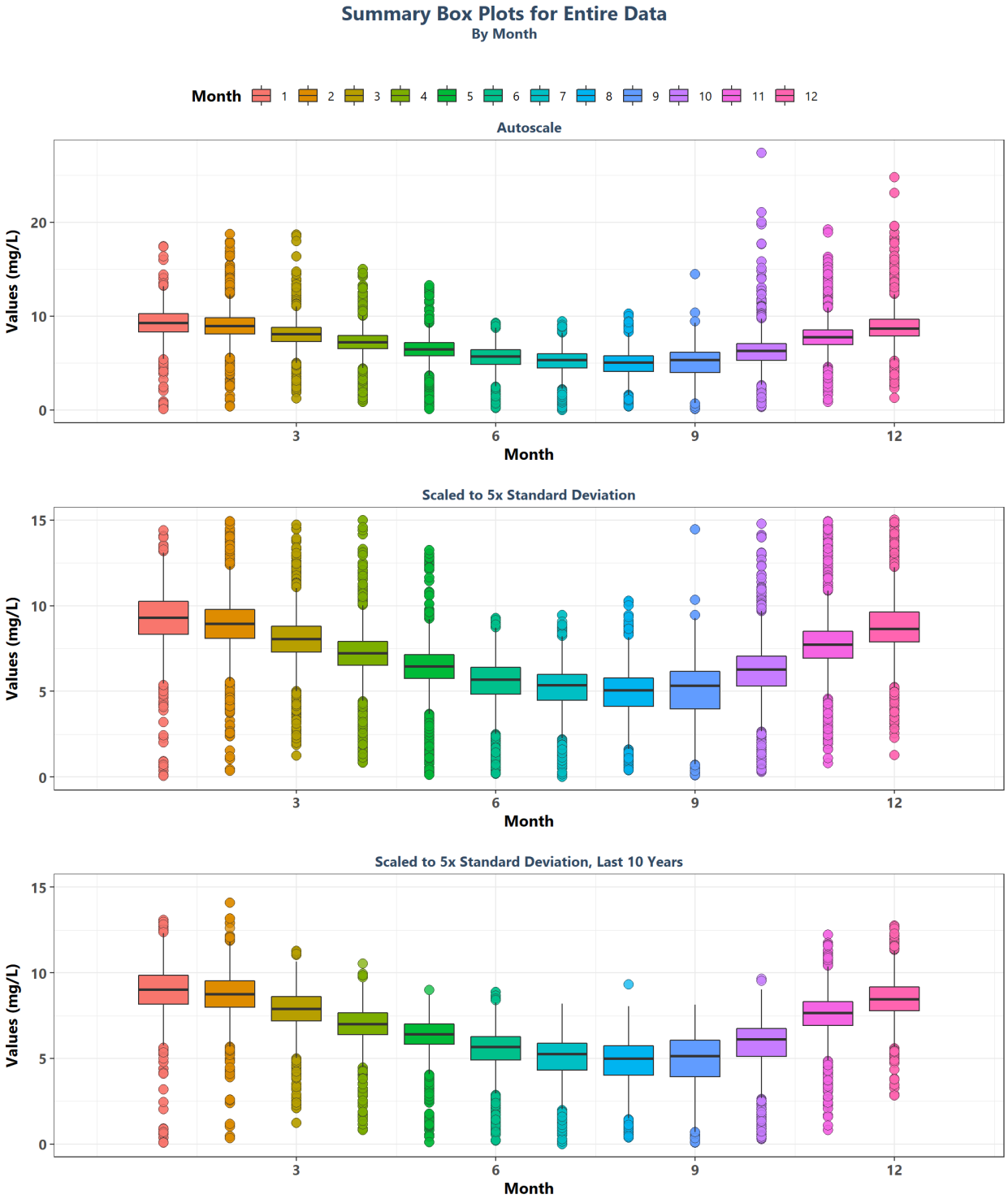
The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

```
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
Autoscale
Values (mg/L)
0
10
20
2000
2010
2020
Year
Scaled to 4x Standard Deviation
Values (mg/L)
0
5
10
15
2000
2010
2020
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0
5
10
15
2012
2014
2016
2018
2020
2022
Year

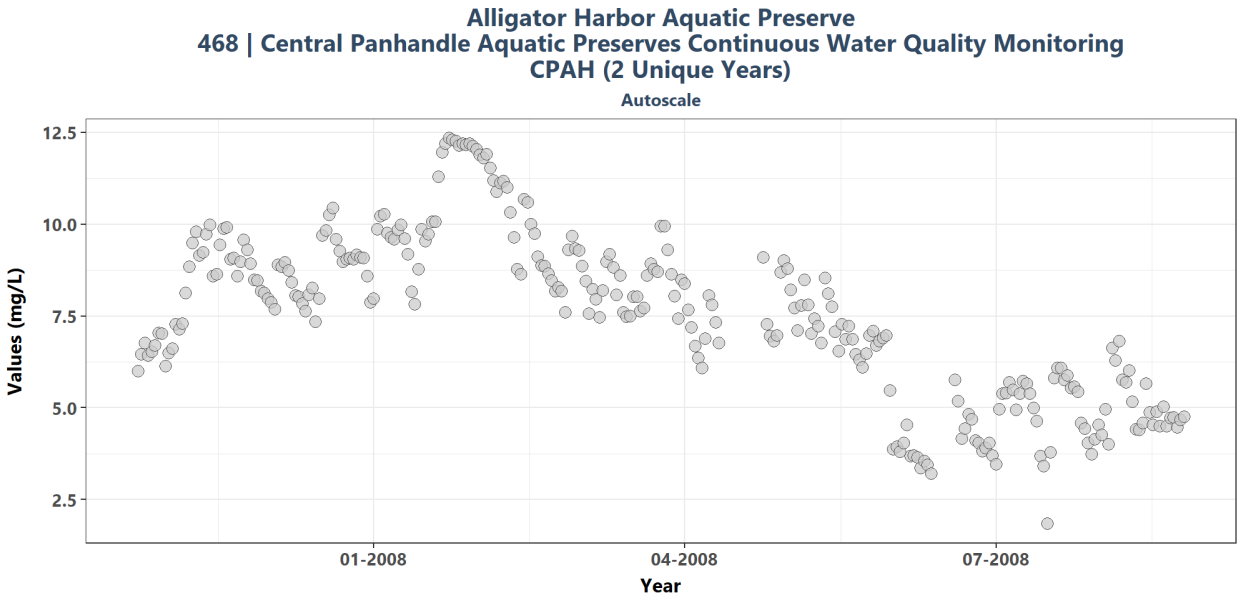

Summary Box Plots for Entire Data
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 5x Standard Deviation
Values (mg/L)
Year
Scaled to 5x Standard Deviation, Last 10 Years
Values (mg/L)
Year

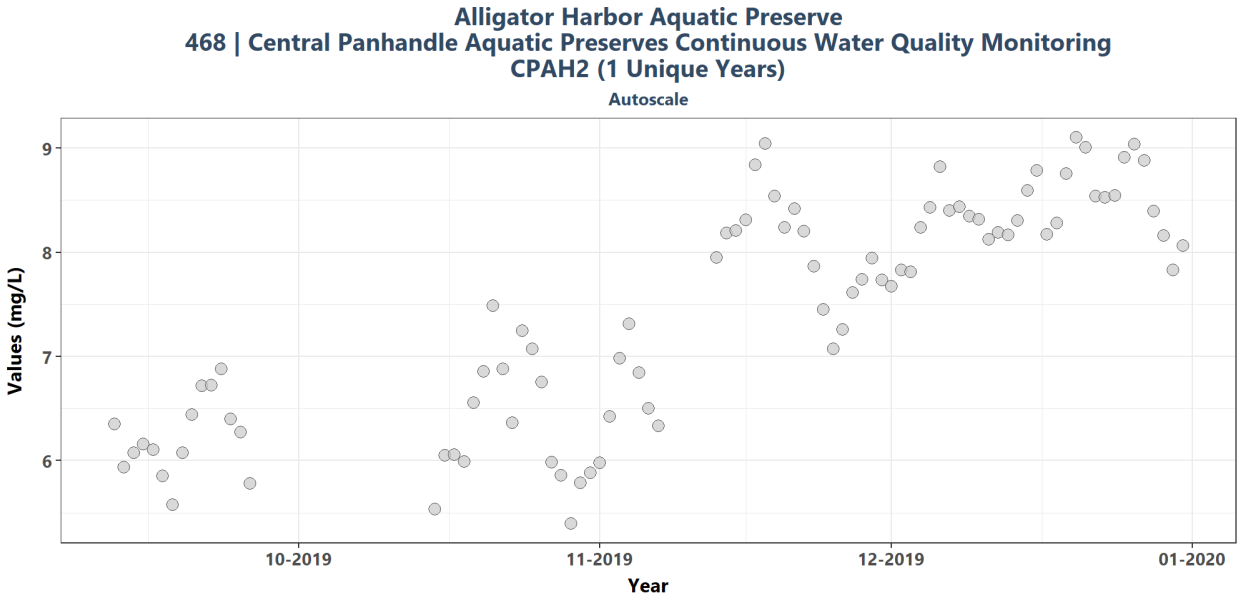


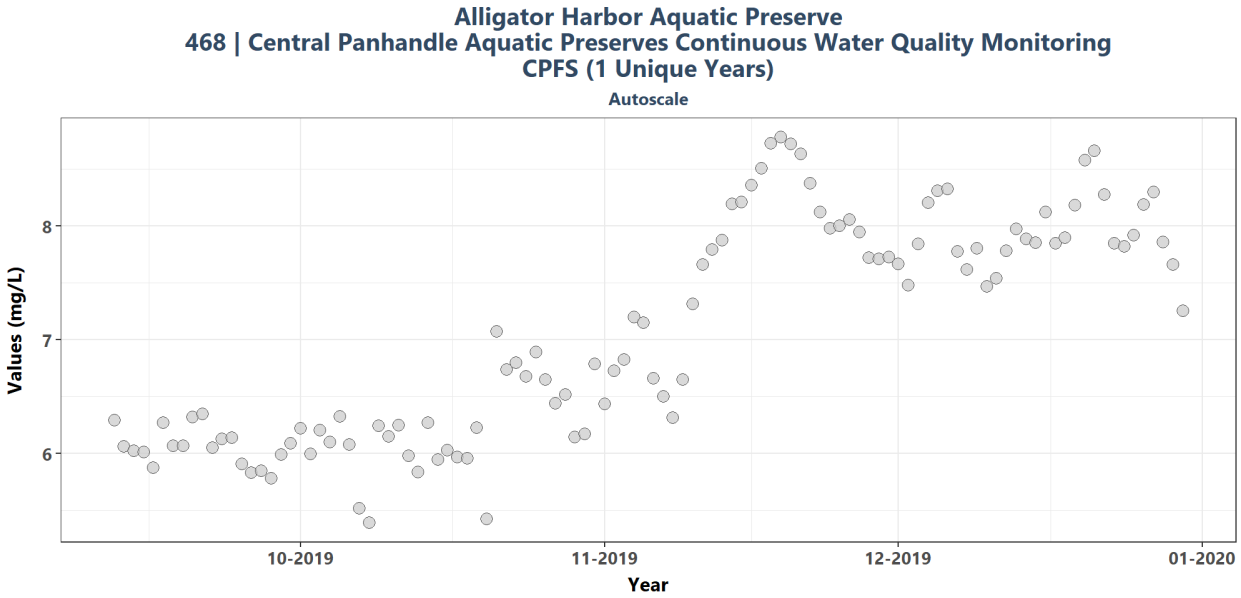
Appendix II: Excluded Monitoring Locations

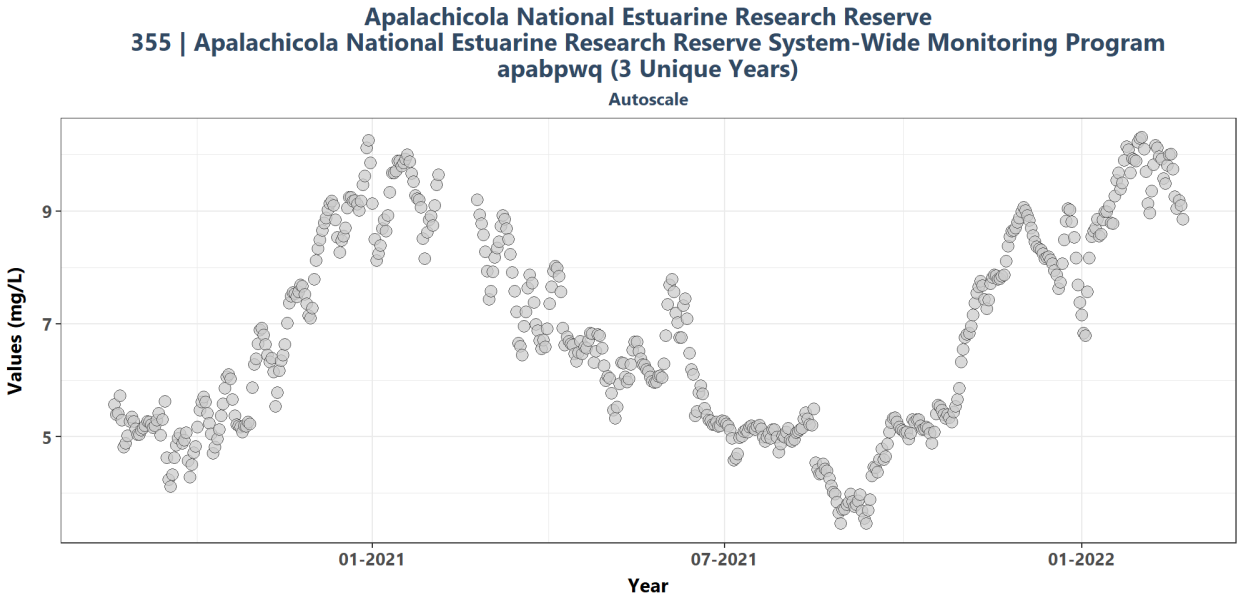
Scatter plots of data values are created for monitoring locations that have fewer than 5 separate years of data entries.

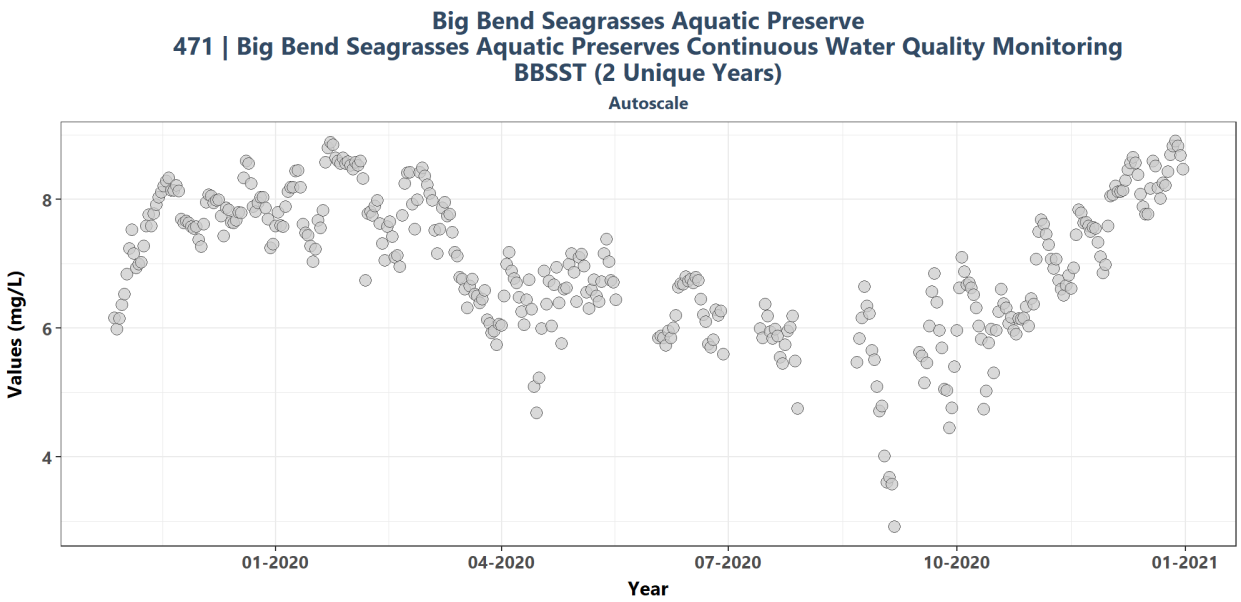
```
Mon_Exclude <- Mon_Summ[Mon_Summ$N_Years<5 & Mon_Summ$N_Years>0,]
Mon_Exclude <- Mon_Exclude[order(Mon_Exclude$MonitoringID),]
z=nrow(Mon_Exclude)

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]]))

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

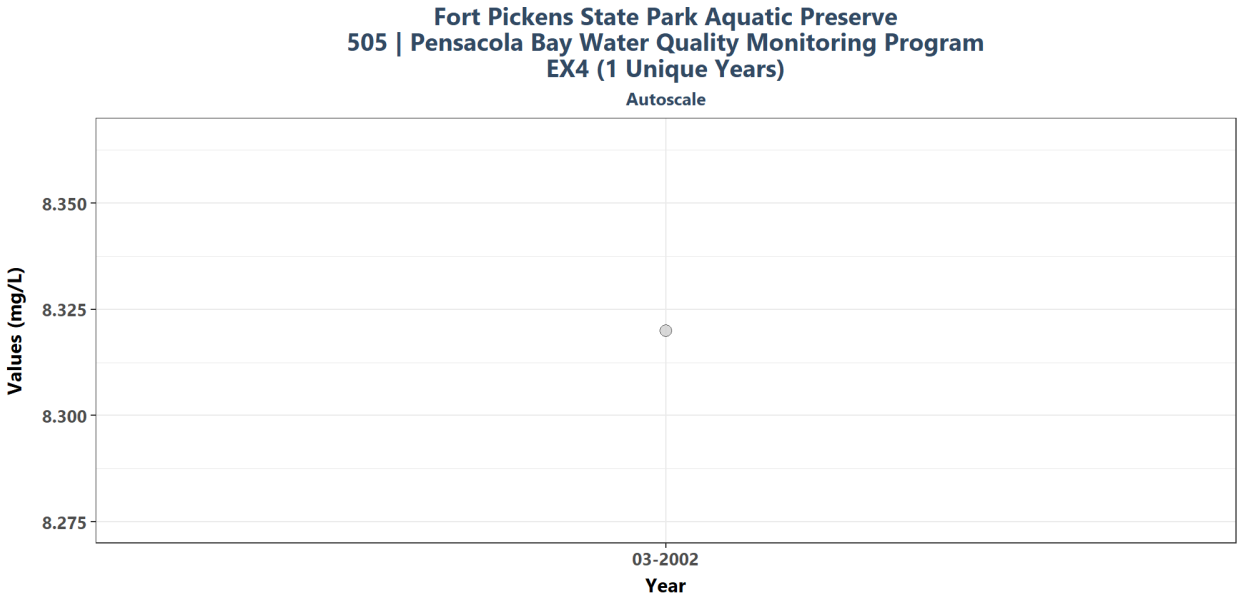


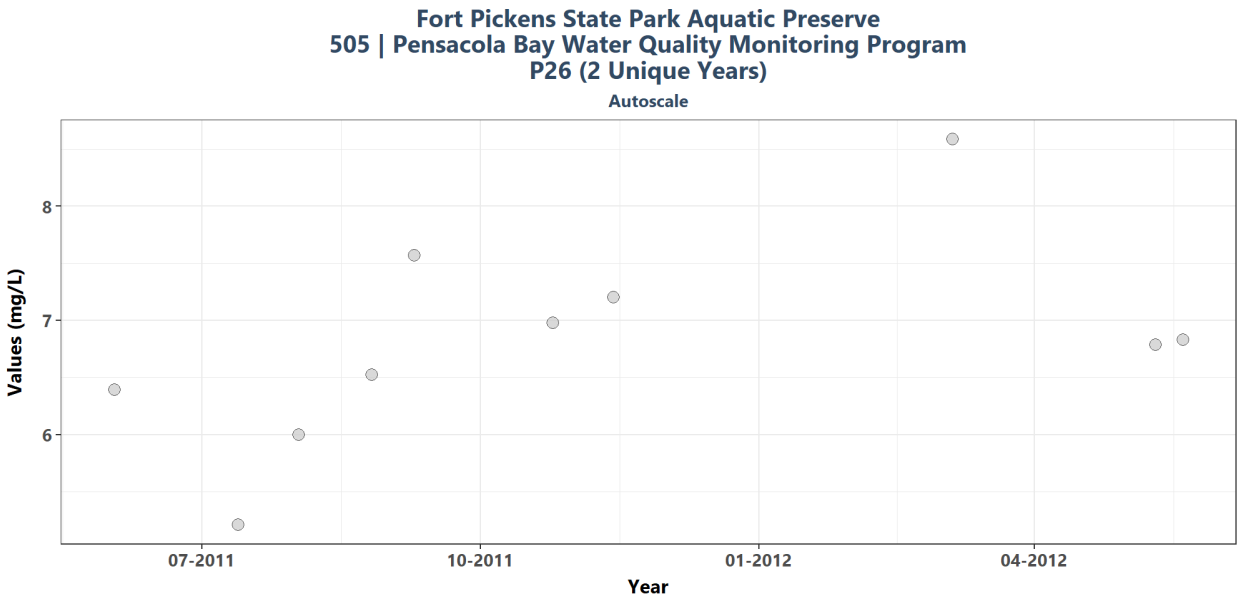

Alligator Harbor Aquatic Preserve
468 | Central Panhandle Aquatic Preserves Continuous Water Quality Monitoring
CPAH2 (1 Unique Years)
Autoscale
Values (mg/L)
9
8
7
6
10-2019
11-2019
12-2019
01-2020
Year

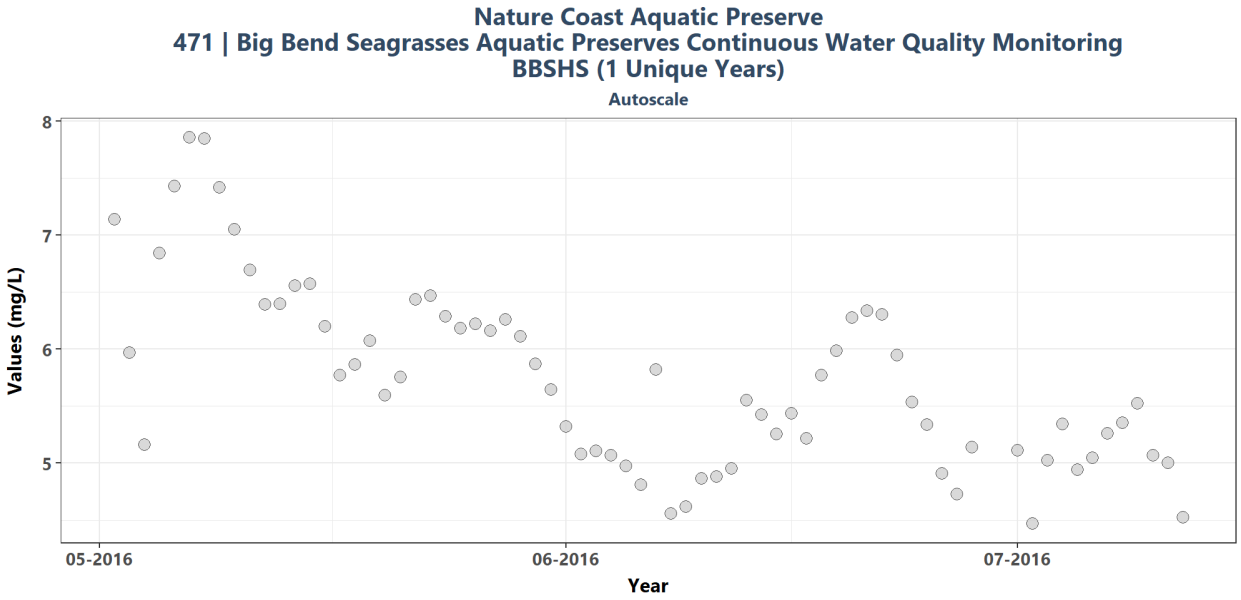

Alligator Harbor Aquatic Preserve
468 | Central Panhandle Aquatic Preserves Continuous Water Quality Monitoring
CPFS (1 Unique Years)
Autoscale
Values (mg/L)
8
7
6
10-2019
11-2019
12-2019
01-2020
Year

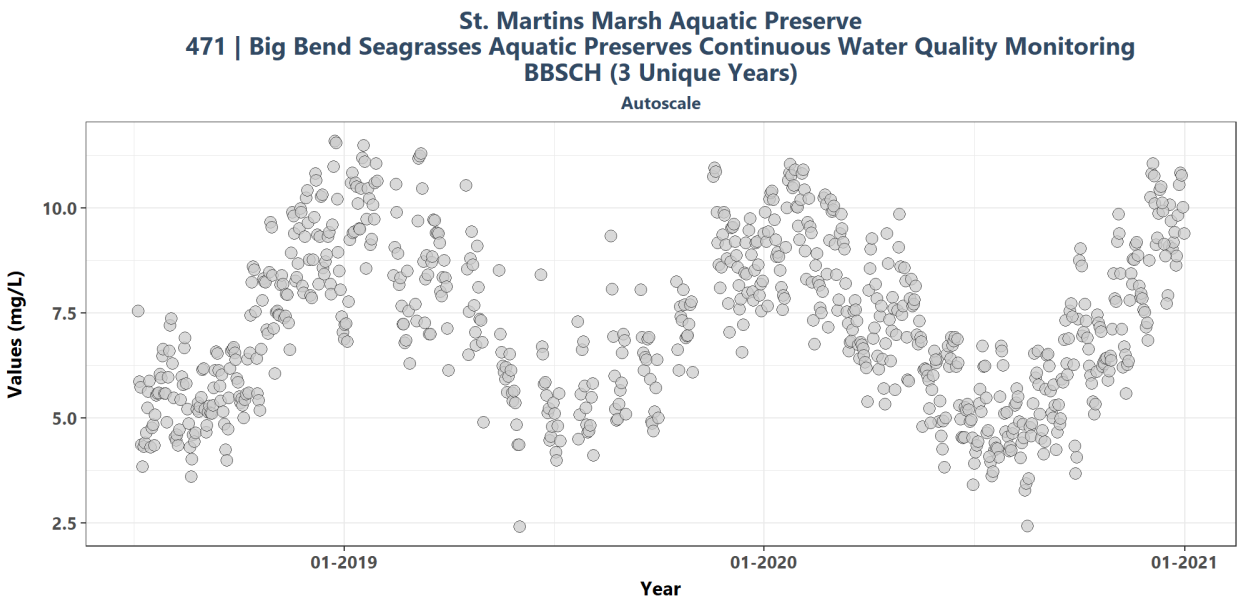

Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apabpwq (3 Unique Years)
Autoscale
Values (mg/L)
9
7
5
01-2021
07-2021
01-2022
Year


Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring
BBSST (2 Unique Years)
Autoscale
Values (mg/L)
8
6
4
01-2020
04-2020
07-2020
10-2020
01-2021
Year


Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program
EX4 (1 Unique Years)
Autoscale
8.350
8.325
8.300
8.275
Values (mg/L)
03-2002
Year


Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program
P26 (2 Unique Years)
Autoscale
8
7
6
Values (mg/L)
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

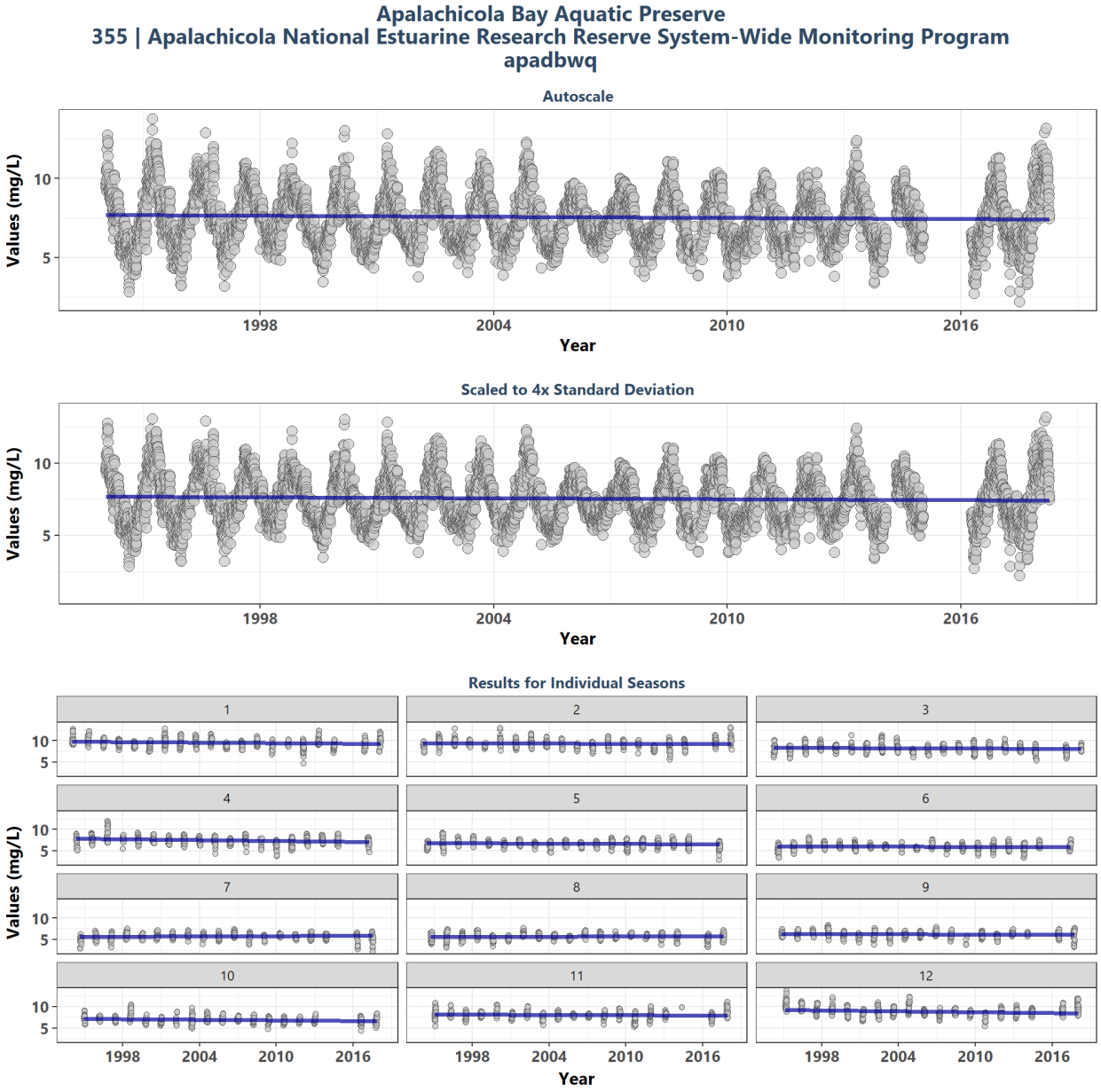
1. Use the data set that only has `Use_In_Analysis` of `TRUE` for the desired monitoring location
2. Determine the earliest and latest year of the data to create x-axis scale and intervals
3. Determine the minimum, mean, and standard deviation for the data to be used for y-axis scales
    - Excludes 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],]
      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$relyear_dd), 5, ceiling) - round_any(min(plot_data$rely

      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_d

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

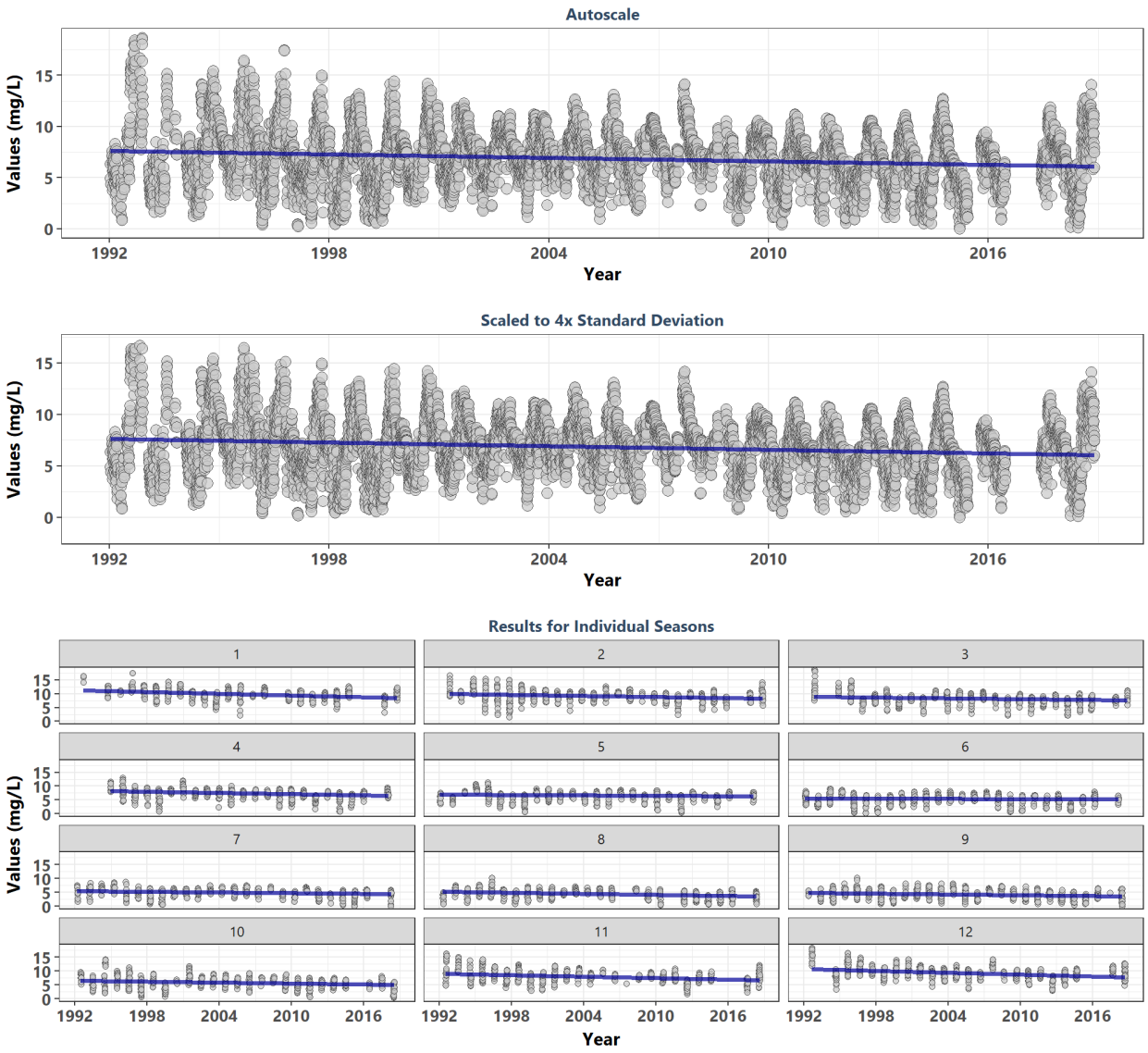
```
# x_end1 <- relyear_dd_upper
# y_end1 <- relyear_dd_upper * KT.Stats[ManagedAreaName==MA_Include[i] & Season == "All", SennSlo

KT.Stats[, season := Season]
KT.Stats[MonitoringID == Mon_IDs[i] & season != "All", `:=` (N_Data = nrow(plot_data[Season == sea
KT.Stats[MonitoringID == Mon_IDs[i] & season == "All", `:=` (relyear_dd_lower = min(plot_data[Moni
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_
                                                                                     y = relyear_
                                                                                     xend = relyea
                                                                                     yend = relyea


                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_
                                                                                     y = relyear_
                                                                                     xend = relyea
                                                                                     yend = relyea


                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 = relyea
                                                                                        y = relyea
                                                                                        xend = rel
                                                                                        yend = rel


                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())

      KT.Stats[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(KT.Stats[KT.Stats$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)
   }
}
```



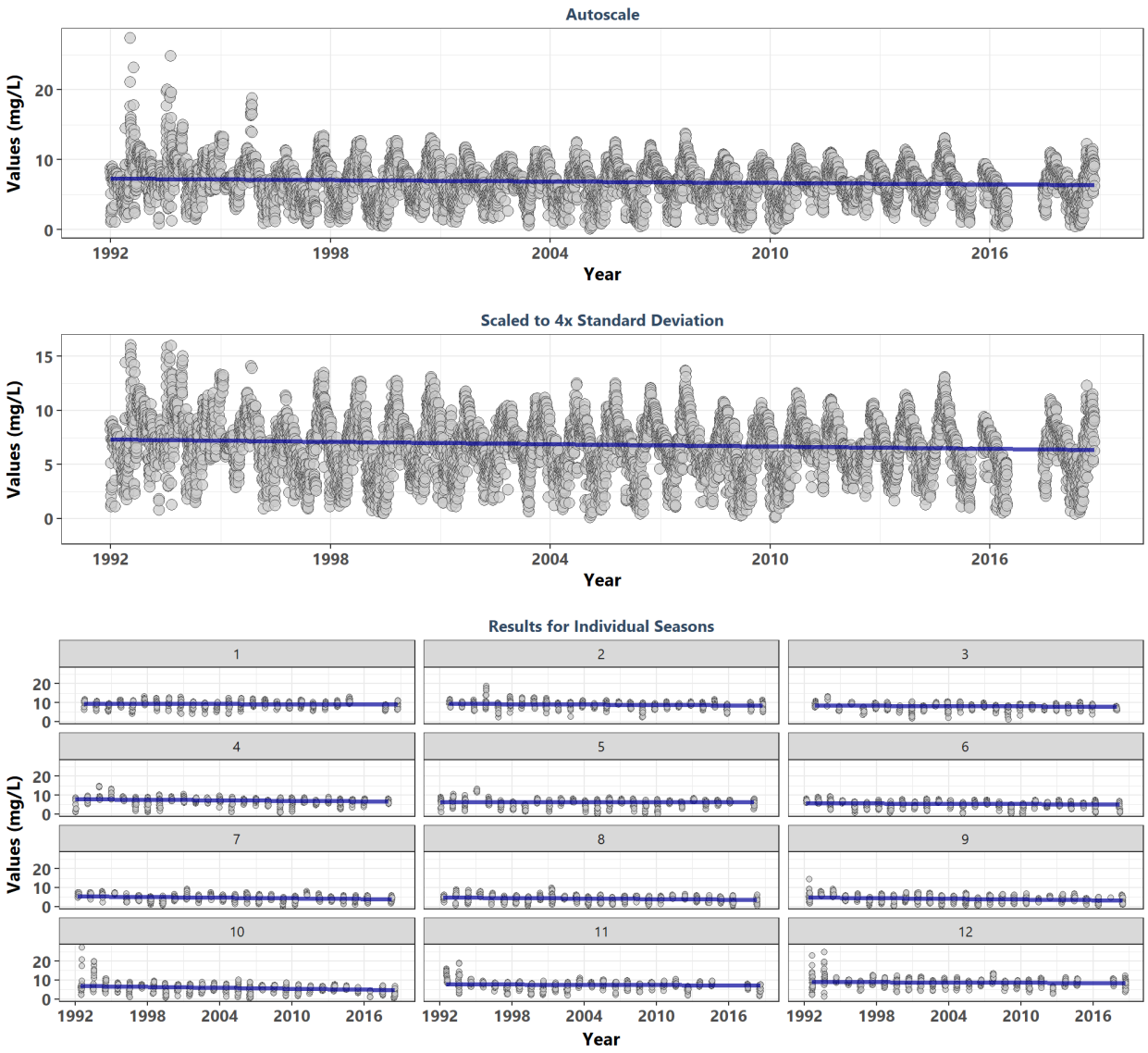
Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	6392	7.21	-0.0575	-0.0128	7.7667	-6.9	0.0000	64.8	0	-1
1	577	9.49	-0.0979	-0.0248	9.8866	-3.5	0.0004	NA	NA	-1
2	492	9.31	-0.0231	-0.0059	9.4038	-0.8	0.4423	NA	NA	-1
3	569	8.23	-0.0433	-0.0102	8.3996	-1.5	0.1218	NA	NA	-1
4	533	7.44	-0.1604	-0.0404	8.0871	-5.5	0.0000	NA	NA	-1
5	538	6.59	-0.0680	-0.0140	6.8243	-2.4	0.0181	NA	NA	-1
6	535	5.98	-0.0310	-0.0060	6.0788	-1.1	0.2837	NA	NA	-1
7	544	5.70	0.0598	0.0098	5.5405	2.1	0.0369	NA	NA	1
8	538	5.67	0.0345	0.0073	5.5562	1.2	0.2303	NA	NA	1
9	505	6.17	-0.0184	-0.0036	6.2276	-0.6	0.5366	NA	NA	-1
10	478	6.93	-0.1600	-0.0325	7.4464	-5.2	0.0000	NA	NA	-1
11	525	8.03	-0.0493	-0.0121	8.2241	-1.7	0.0905	NA	NA	-1
12	558	8.90	-0.1354	-0.0385	9.5120	-4.8	0.0000	NA	NA	-1

[a] $p < 0.00005$ appear as 0 due to rounding



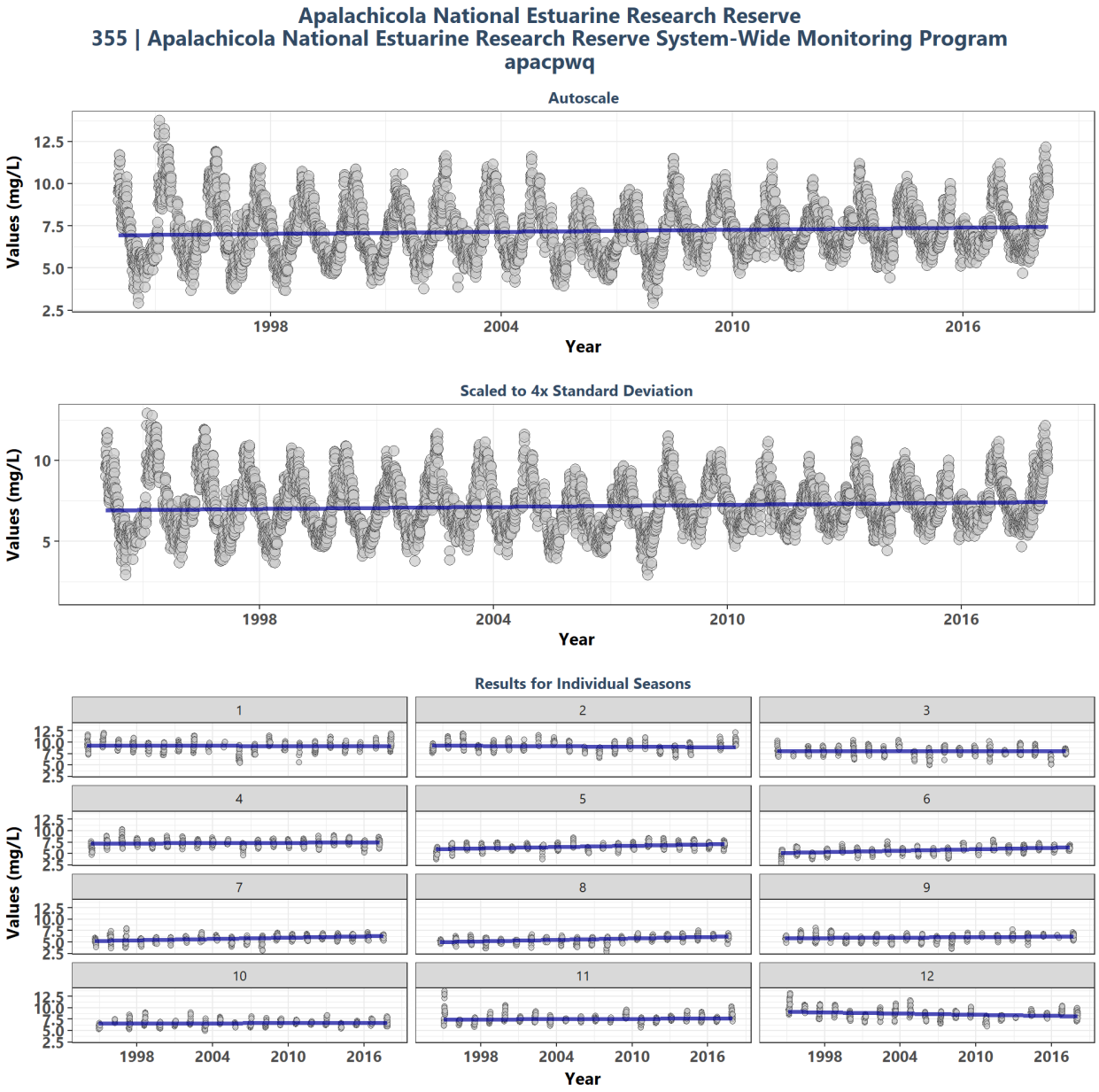
Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	7689	6.88	-0.1551	-0.0573	7.6262	-20.2	0.0000	82.3	0	-1
1	595	9.75	-0.2490	-0.0990	11.1385	-9.1	0.0000	NA	NA	-1
2	646	9.00	-0.1560	-0.0634	9.8828	-5.9	0.0000	NA	NA	-1
3	689	8.21	-0.1272	-0.0480	8.8870	-5.0	0.0000	NA	NA	-1
4	642	7.41	-0.2012	-0.0668	8.3399	-7.6	0.0000	NA	NA	-1
5	642	6.66	-0.0583	-0.0197	6.9126	-2.2	0.0271	NA	NA	-1
6	646	5.39	-0.0067	-0.0024	5.4174	-0.3	0.7982	NA	NA	-1
7	595	5.04	-0.1420	-0.0440	5.5261	-5.2	0.0000	NA	NA	-1
8	637	4.53	-0.2317	-0.0712	5.3824	-8.8	0.0000	NA	NA	-1
9	656	4.28	-0.1282	-0.0472	4.8968	-4.9	0.0000	NA	NA	-1
10	676	5.94	-0.1310	-0.0543	6.5889	-5.1	0.0000	NA	NA	-1
11	616	8.18	-0.2097	-0.0891	9.1635	-7.8	0.0000	NA	NA	-1
12	649	9.40	-0.2330	-0.1095	10.7149	-8.9	0.0000	NA	NA	-1

[a] $p < 0.00005$ appear as 0 due to rounding



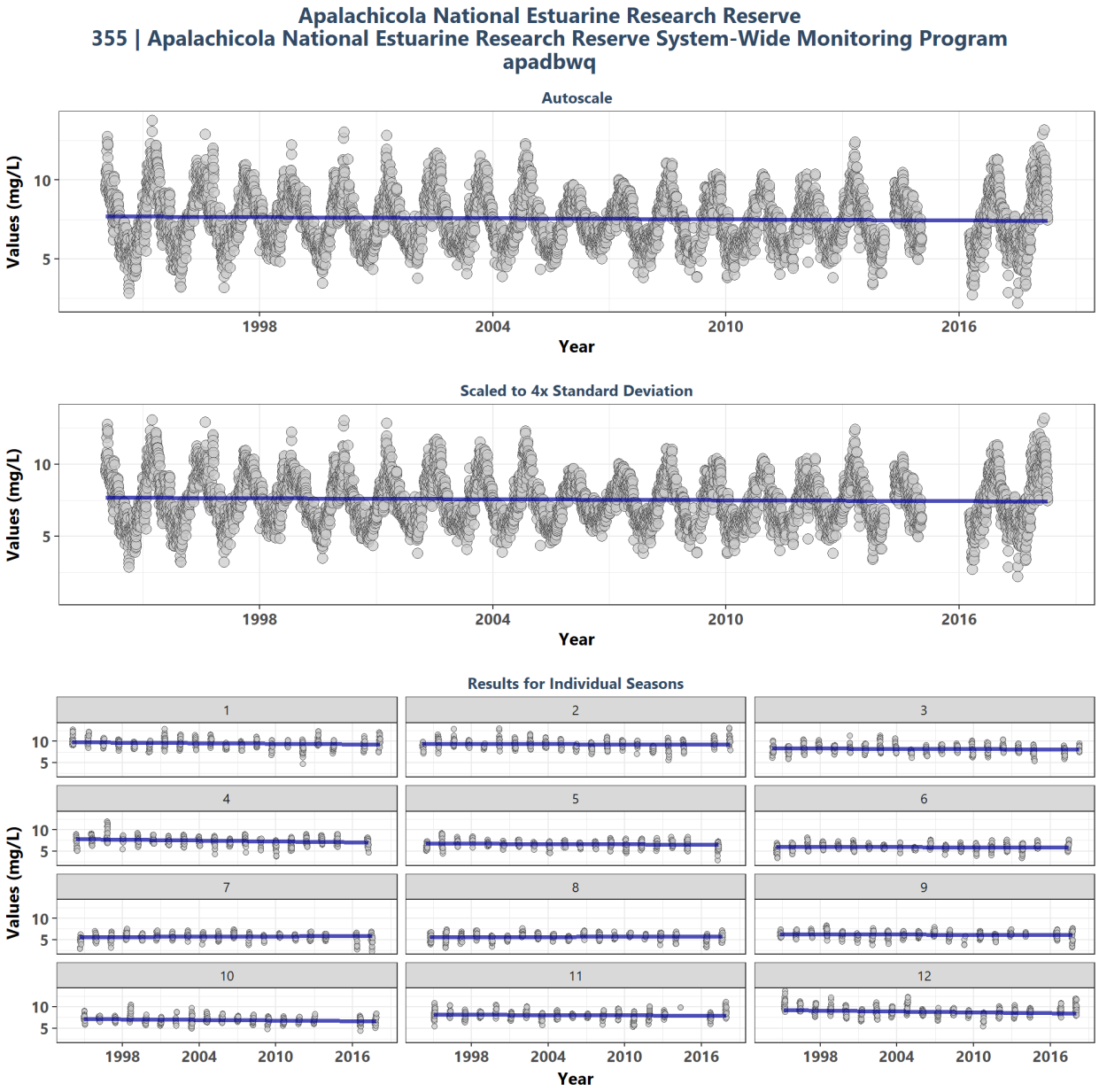
Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	8418	6.78	-0.1036	-0.0344	7.2893	-14.1	0.0000	62.7	0	-1
1	755	9.34	-0.0344	-0.0121	9.4949	-1.4	0.1569	NA	NA	-1
2	701	8.99	-0.1194	-0.0339	9.4255	-4.7	0.0000	NA	NA	-1
3	666	8.12	-0.0840	-0.0259	8.4781	-3.2	0.0012	NA	NA	-1
4	680	7.25	-0.1647	-0.0466	7.8988	-6.4	0.0000	NA	NA	-1
5	727	6.42	0.0080	0.0027	6.3891	0.3	0.7470	NA	NA	1
6	684	5.43	-0.0946	-0.0312	5.8661	-3.7	0.0002	NA	NA	-1
7	692	4.59	-0.1787	-0.0598	5.3631	-7.0	0.0000	NA	NA	-1
8	721	4.22	-0.1381	-0.0427	4.7292	-5.6	0.0000	NA	NA	-1
9	652	4.05	-0.1276	-0.0484	4.6776	-4.9	0.0000	NA	NA	-1
10	699	5.85	-0.1937	-0.0760	6.7667	-7.7	0.0000	NA	NA	-1
11	716	7.50	-0.0793	-0.0288	7.8118	-3.2	0.0015	NA	NA	-1
12	725	8.80	-0.0506	-0.0196	9.0392	-2.0	0.0411	NA	NA	-1

[a] $p < 0.00005$ appear as 0 due to rounding



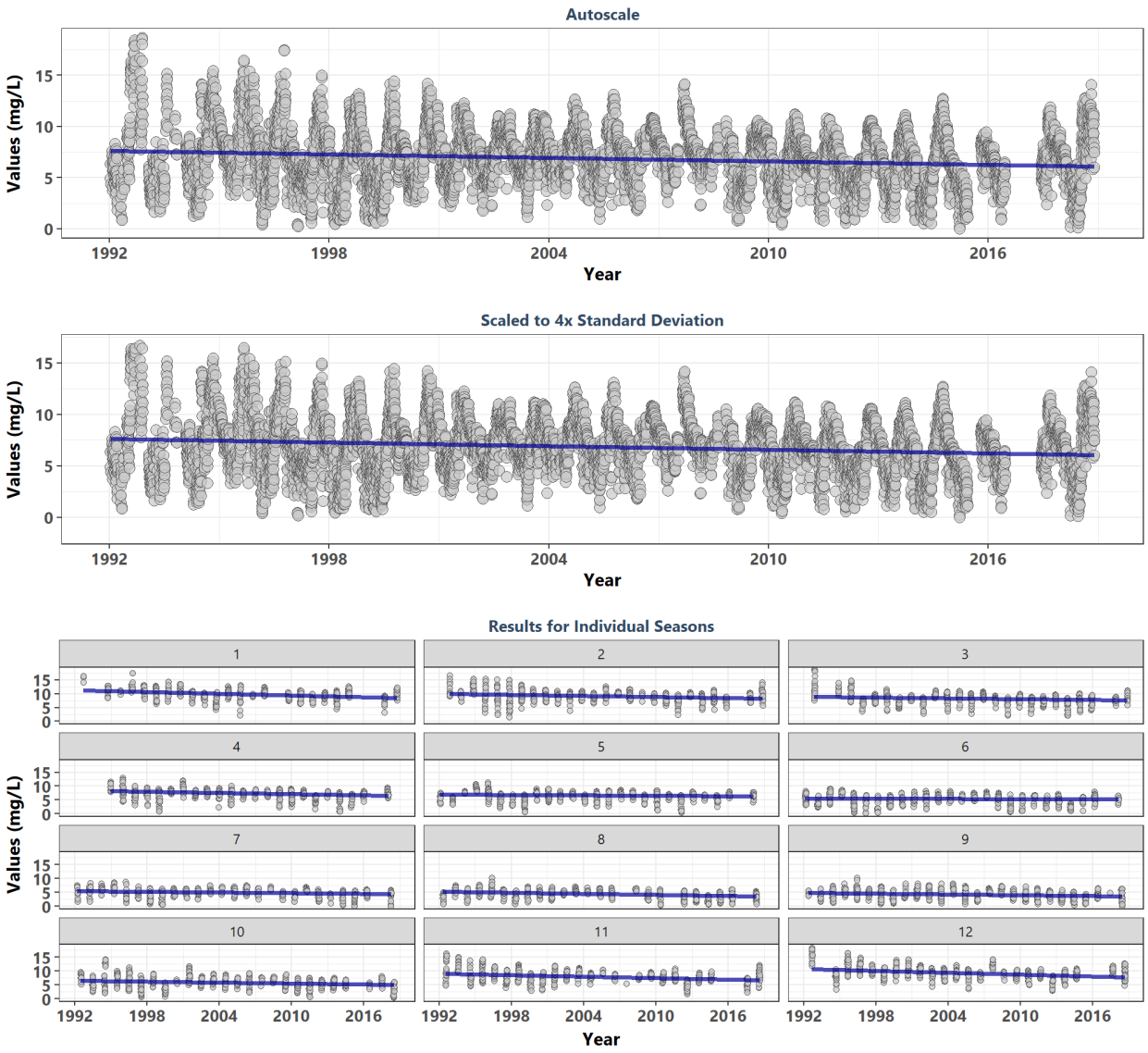
Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	6259	6.98	0.1185	0.0250	6.7630	14.5	0.0000	450.4	0	1
1	519	9.24	-0.0369	-0.0106	9.4045	-1.3	0.2086	NA	NA	-1
2	482	9.14	-0.0669	-0.0179	9.4284	-2.2	0.0280	NA	NA	-1
3	514	8.08	-0.0151	-0.0036	8.1385	-0.5	0.6084	NA	NA	-1
4	546	7.32	0.0788	0.0151	7.0609	2.8	0.0058	NA	NA	1
5	542	6.55	0.3085	0.0530	5.6500	10.8	0.0000	NA	NA	1
6	534	5.81	0.3624	0.0652	4.6992	12.5	0.0000	NA	NA	1
7	575	5.79	0.3043	0.0535	4.8803	10.9	0.0000	NA	NA	1
8	512	5.63	0.3930	0.0645	4.5356	13.3	0.0000	NA	NA	1
9	476	6.03	0.0975	0.0182	5.7554	3.2	0.0015	NA	NA	1
10	491	6.60	0.0532	0.0078	6.4651	1.8	0.0777	NA	NA	1
11	544	7.48	0.0753	0.0134	7.2667	2.6	0.0086	NA	NA	1
12	524	8.65	-0.1773	-0.0516	9.4763	-6.1	0.0000	NA	NA	-1

[a] $p < 0.00005$ appear as 0 due to rounding



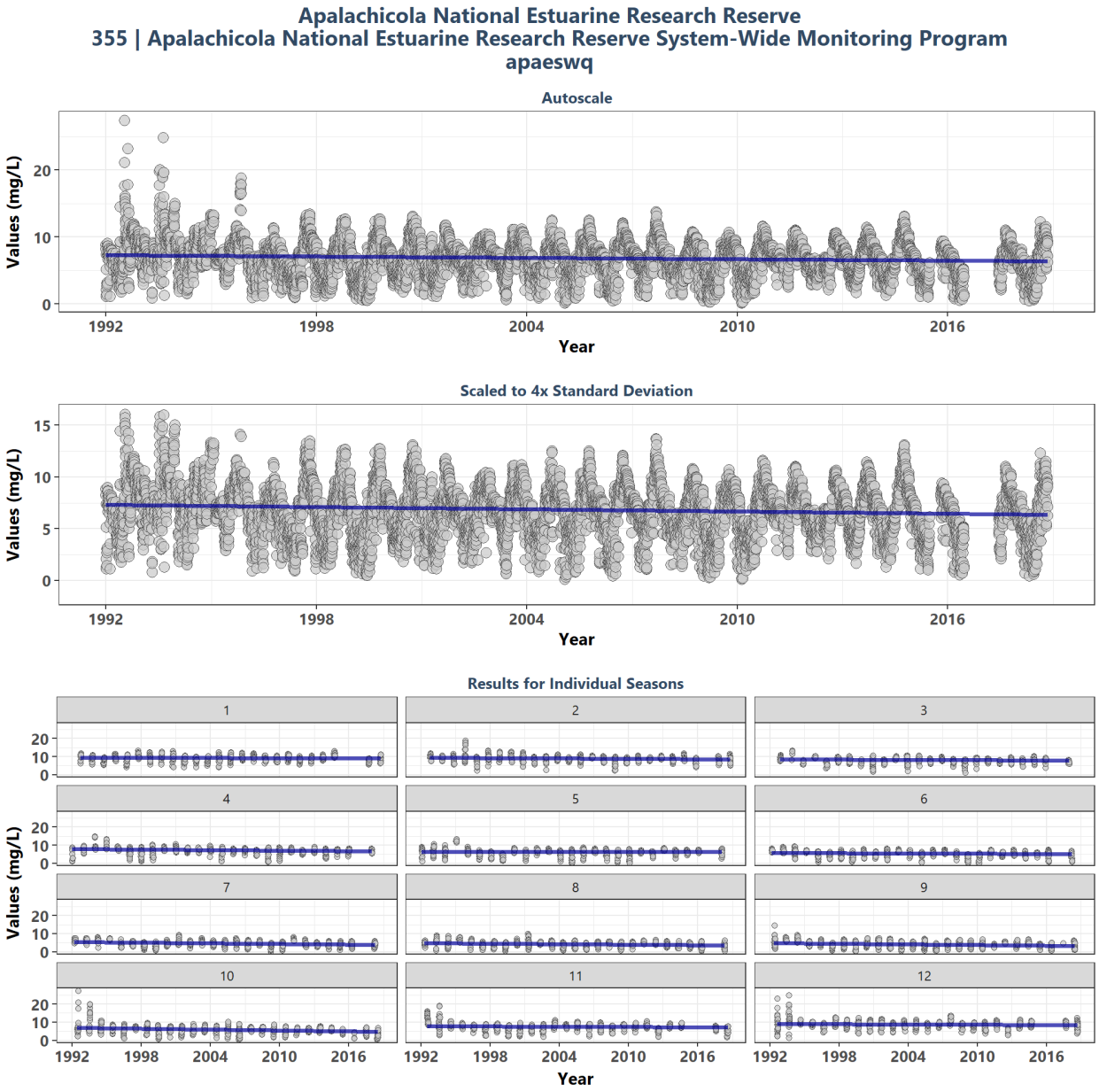
Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	6392	7.21	-0.0575	-0.0128	7.7667	-6.9	0.0000	64.8	0	-1
1	577	9.49	-0.0979	-0.0248	9.8866	-3.5	0.0004	NA	NA	-1
2	492	9.31	-0.0231	-0.0059	9.4038	-0.8	0.4423	NA	NA	-1
3	569	8.23	-0.0433	-0.0102	8.3996	-1.5	0.1218	NA	NA	-1
4	533	7.44	-0.1604	-0.0404	8.0871	-5.5	0.0000	NA	NA	-1
5	538	6.59	-0.0680	-0.0140	6.8243	-2.4	0.0181	NA	NA	-1
6	535	5.98	-0.0310	-0.0060	6.0788	-1.1	0.2837	NA	NA	-1
7	544	5.70	0.0598	0.0098	5.5405	2.1	0.0369	NA	NA	1
8	538	5.67	0.0345	0.0073	5.5562	1.2	0.2303	NA	NA	1
9	505	6.17	-0.0184	-0.0036	6.2276	-0.6	0.5366	NA	NA	-1
10	478	6.93	-0.1600	-0.0325	7.4464	-5.2	0.0000	NA	NA	-1
11	525	8.03	-0.0493	-0.0121	8.2241	-1.7	0.0905	NA	NA	-1
12	558	8.90	-0.1354	-0.0385	9.5120	-4.8	0.0000	NA	NA	-1

[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	7689	6.88	-0.1551	-0.0573	7.6262	-20.2	0.0000	82.3	0	-1
1	595	9.75	-0.2490	-0.0990	11.1385	-9.1	0.0000	NA	NA	-1
2	646	9.00	-0.1560	-0.0634	9.8828	-5.9	0.0000	NA	NA	-1
3	689	8.21	-0.1272	-0.0480	8.8870	-5.0	0.0000	NA	NA	-1
4	642	7.41	-0.2012	-0.0668	8.3399	-7.6	0.0000	NA	NA	-1
5	642	6.66	-0.0583	-0.0197	6.9126	-2.2	0.0271	NA	NA	-1
6	646	5.39	-0.0067	-0.0024	5.4174	-0.3	0.7982	NA	NA	-1
7	595	5.04	-0.1420	-0.0440	5.5261	-5.2	0.0000	NA	NA	-1
8	637	4.53	-0.2317	-0.0712	5.3824	-8.8	0.0000	NA	NA	-1
9	656	4.28	-0.1282	-0.0472	4.8968	-4.9	0.0000	NA	NA	-1
10	676	5.94	-0.1310	-0.0543	6.5889	-5.1	0.0000	NA	NA	-1
11	616	8.18	-0.2097	-0.0891	9.1635	-7.8	0.0000	NA	NA	-1
12	649	9.40	-0.2330	-0.1095	10.7149	-8.9	0.0000	NA	NA	-1

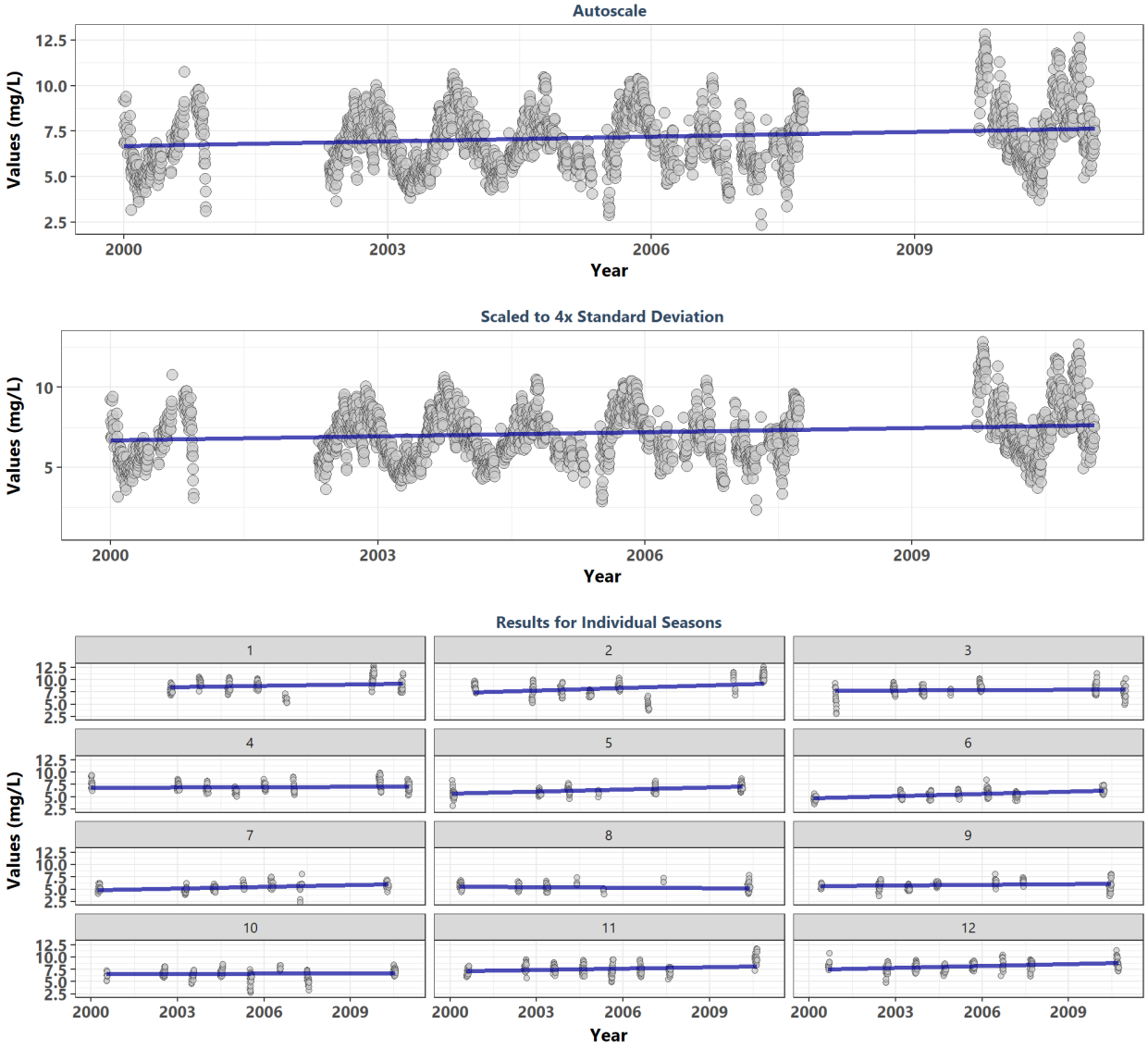
[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	8418	6.78	-0.1036	-0.0344	7.2893	-14.1	0.0000	62.7	0	-1
1	755	9.34	-0.0344	-0.0121	9.4949	-1.4	0.1569	NA	NA	-1
2	701	8.99	-0.1194	-0.0339	9.4255	-4.7	0.0000	NA	NA	-1
3	666	8.12	-0.0840	-0.0259	8.4781	-3.2	0.0012	NA	NA	-1
4	680	7.25	-0.1647	-0.0466	7.8988	-6.4	0.0000	NA	NA	-1
5	727	6.42	0.0080	0.0027	6.3891	0.3	0.7470	NA	NA	1
6	684	5.43	-0.0946	-0.0312	5.8661	-3.7	0.0002	NA	NA	-1
7	692	4.59	-0.1787	-0.0598	5.3631	-7.0	0.0000	NA	NA	-1
8	721	4.22	-0.1381	-0.0427	4.7292	-5.6	0.0000	NA	NA	-1
9	652	4.05	-0.1276	-0.0484	4.6776	-4.9	0.0000	NA	NA	-1
10	699	5.85	-0.1937	-0.0760	6.7667	-7.7	0.0000	NA	NA	-1
11	716	7.50	-0.0793	-0.0288	7.8118	-3.2	0.0015	NA	NA	-1
12	725	8.80	-0.0506	-0.0196	9.0392	-2.0	0.0411	NA	NA	-1

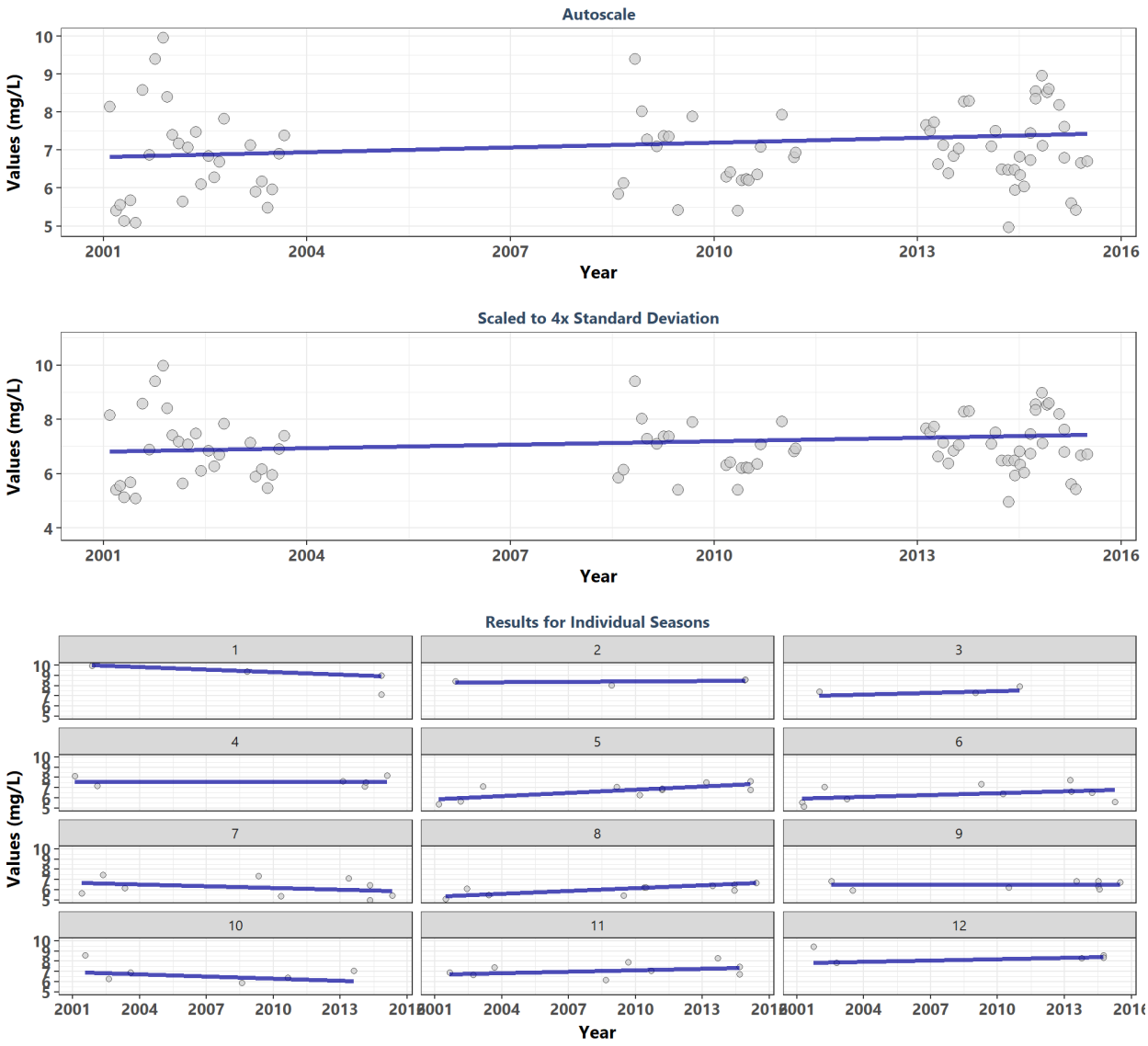
[a] $p < 0.00005$ appear as 0 due to rounding

Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring
BBSSK



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	2240	6.84	0.1534	0.0855	6.6838	10.9	0.0000	117.6	0	1
1	203	8.80	0.0782	0.0752	8.3470	1.7	0.0941	NA	NA	1
2	197	8.20	0.1482	0.1692	7.3507	3.1	0.0018	NA	NA	1
3	175	7.94	0.0361	0.0204	7.8179	0.7	0.4729	NA	NA	1
4	207	6.96	0.0396	0.0212	6.8340	0.9	0.3931	NA	NA	1
5	157	6.21	0.3964	0.1377	5.6597	7.5	0.0000	NA	NA	1
6	198	5.46	0.4217	0.1553	4.6846	8.9	0.0000	NA	NA	1
7	167	5.31	0.3472	0.1150	4.8547	6.8	0.0000	NA	NA	1
8	129	5.44	-0.1301	-0.0398	5.5606	-2.2	0.0251	NA	NA	-1
9	161	5.83	0.1046	0.0498	5.6301	2.0	0.0458	NA	NA	1
10	196	6.60	0.0295	0.0171	6.5336	0.6	0.5351	NA	NA	1
11	238	7.59	0.1271	0.0912	7.1370	2.9	0.0033	NA	NA	1
12	212	8.08	0.2091	0.1165	7.5012	4.6	0.0000	NA	NA	1

[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	84	6.86	0.1440	0.0427	6.8183	2.0	0.0417	15.8	0.1472	1
1	4	9.19	-0.8333	-0.0802	10.0692	-1.4	0.1486	NA	NA	-1
2	4	8.46	0.5000	0.0157	8.2913	0.7	0.4701	NA	NA	1
3	3	7.41	0.3333	0.0575	6.9467	0.0	1.0000	NA	NA	1
4	6	7.59	0.0000	-0.0014	7.6034	0.0	1.0000	NA	NA	-1
5	10	6.87	0.4222	0.1052	5.8682	1.6	0.1046	NA	NA	1
6	10	6.45	0.2444	0.0602	5.9408	0.9	0.3672	NA	NA	1
7	9	6.17	-0.3056	-0.0575	6.6837	-1.0	0.2945	NA	NA	-1
8	10	6.15	0.6444	0.0915	5.3277	2.5	0.0116	NA	NA	1
9	8	6.52	-0.0357	-0.0005	6.5269	0.0	1.0000	NA	NA	-1
10	6	6.63	-0.0667	-0.0712	6.9502	0.0	1.0000	NA	NA	-1
11	9	7.08	0.2500	0.0487	6.6900	0.8	0.4017	NA	NA	1
12	5	8.35	0.1000	0.0426	7.8372	0.0	1.0000	NA	NA	1

[a] $p < 0.00005$ 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

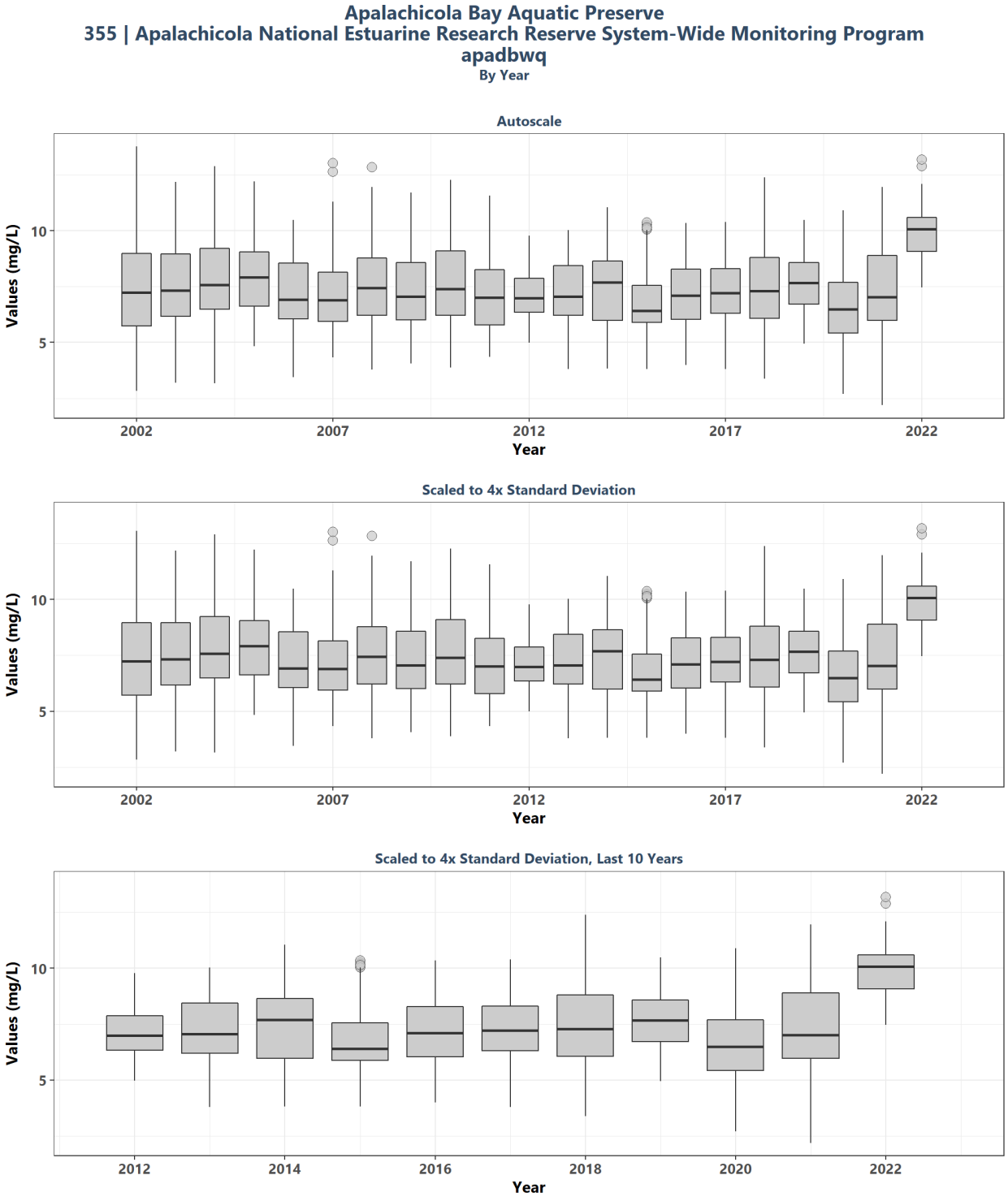
```
if(n==0){
   print("There are no monitoring locations that qualify.")
} else {
   for (i in 1:n) {
      year_lower <- min(data$Year[data$Use_In_Analysis==TRUE &
                                  data$MonitoringID==Mon_IDs[i]])
      year_upper <- max(data$Year[data$Use_In_Analysis==TRUE &
     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

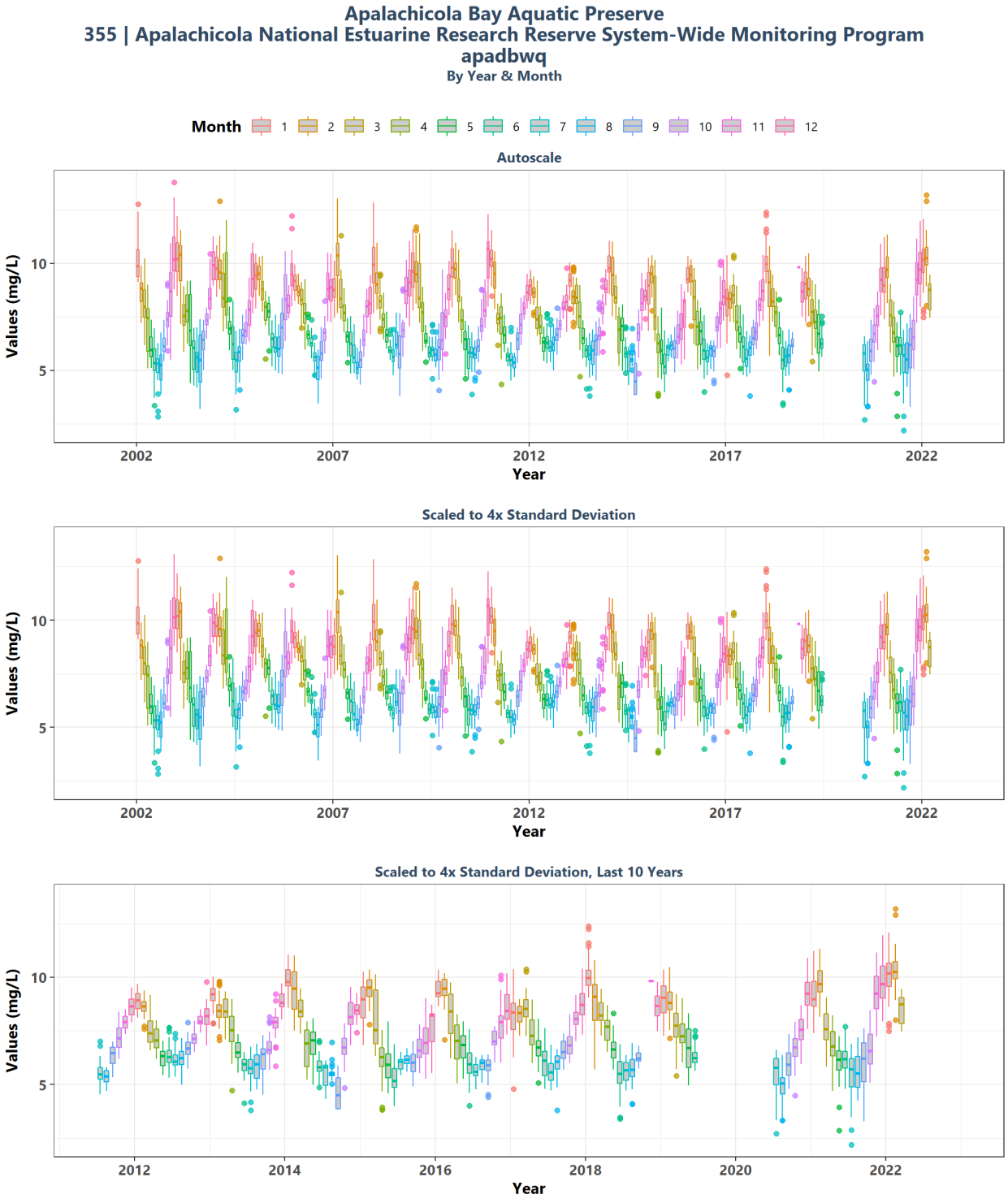
```
            aes(x=Year, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Autoscale",
        x="Year", y=paste0("Values (", 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,
```

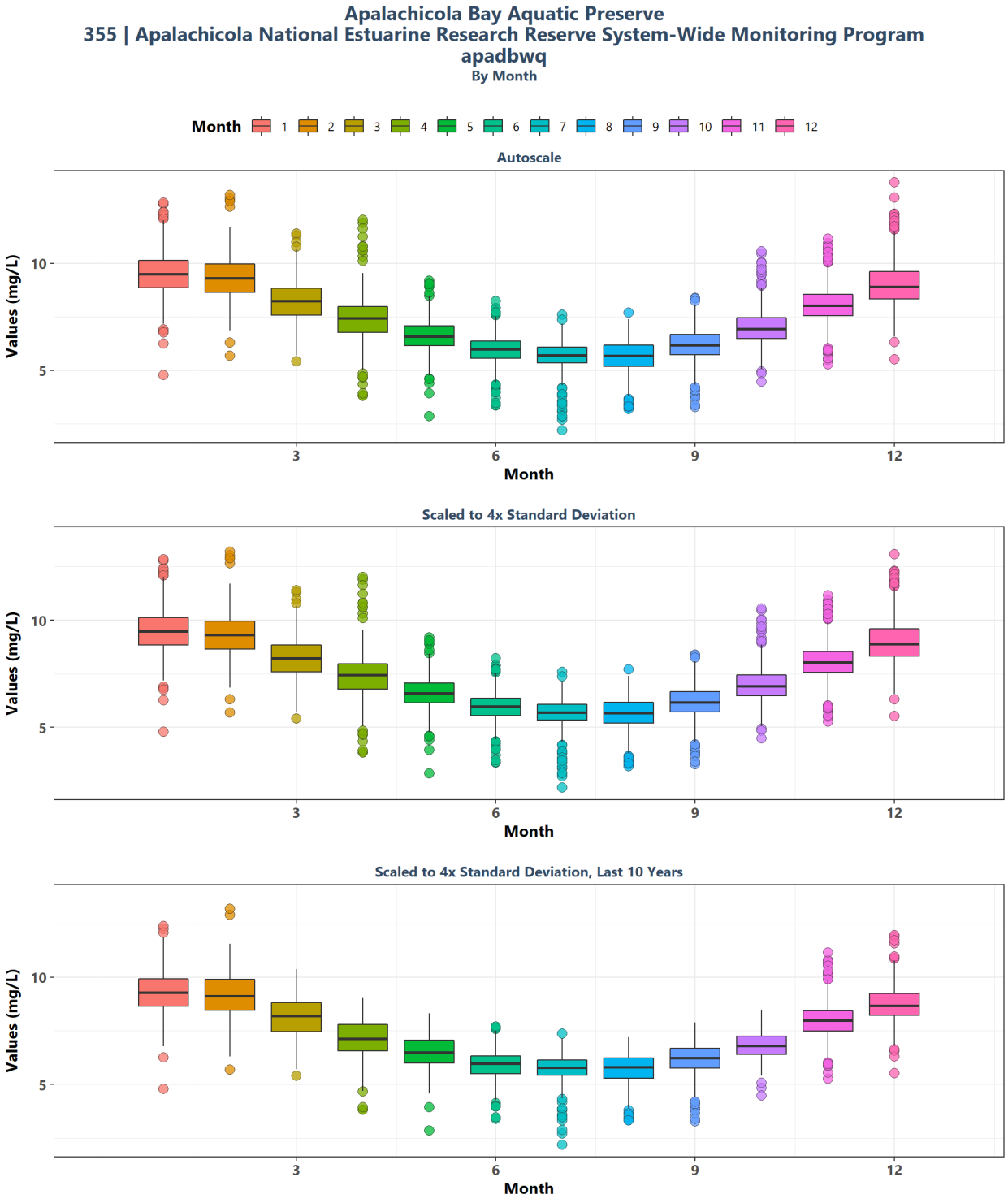
```
                    group=YearMonth, color=as.factor(Month))) +
    geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
    labs(subtitle="Autoscale",
         x="Year", y=paste0("Values (", unit, ")"), color="Month") +
    scale_x_continuous(limits=c(year_lower - 1, year_upper + 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)
```

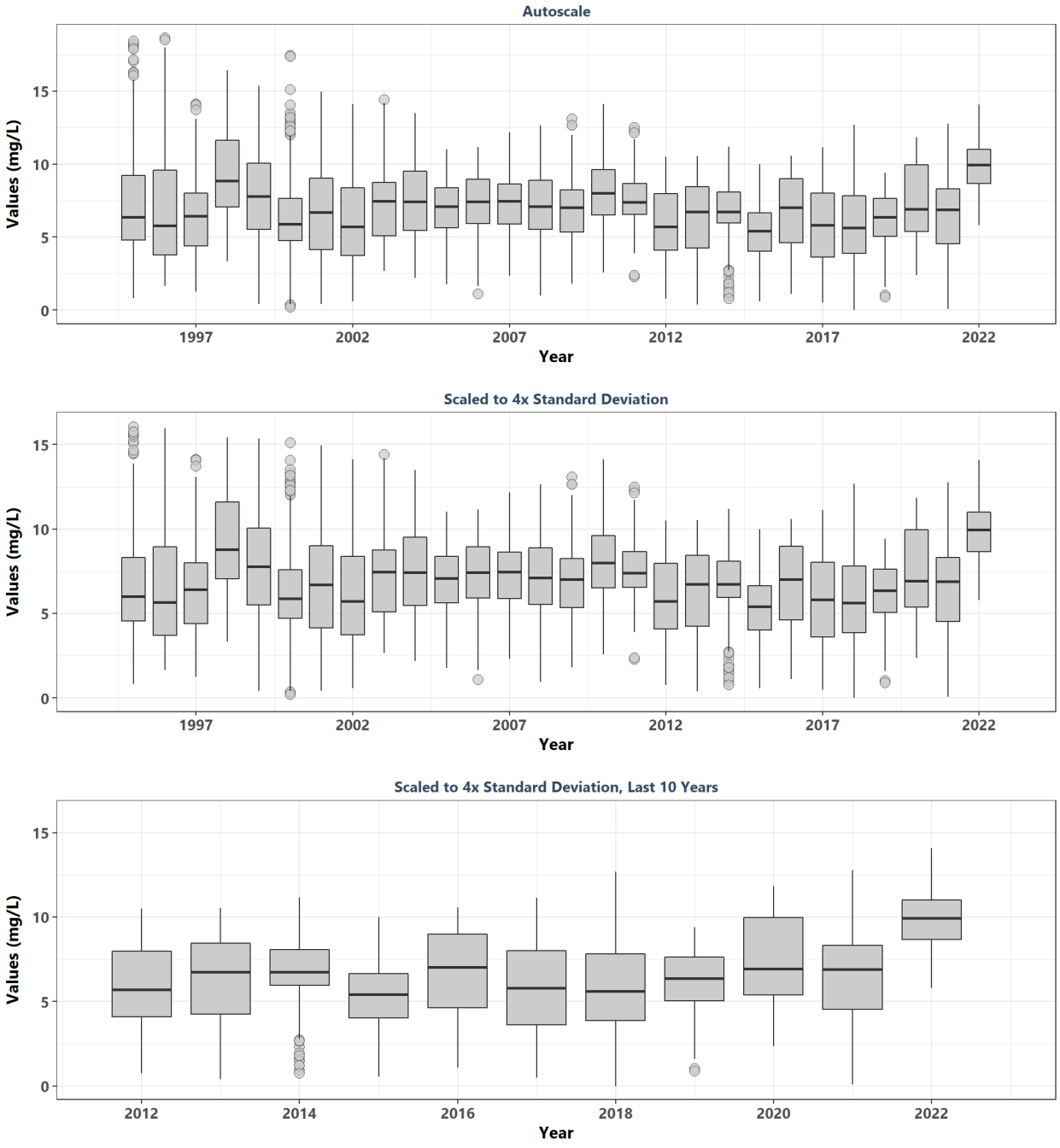
```
    }
}
```

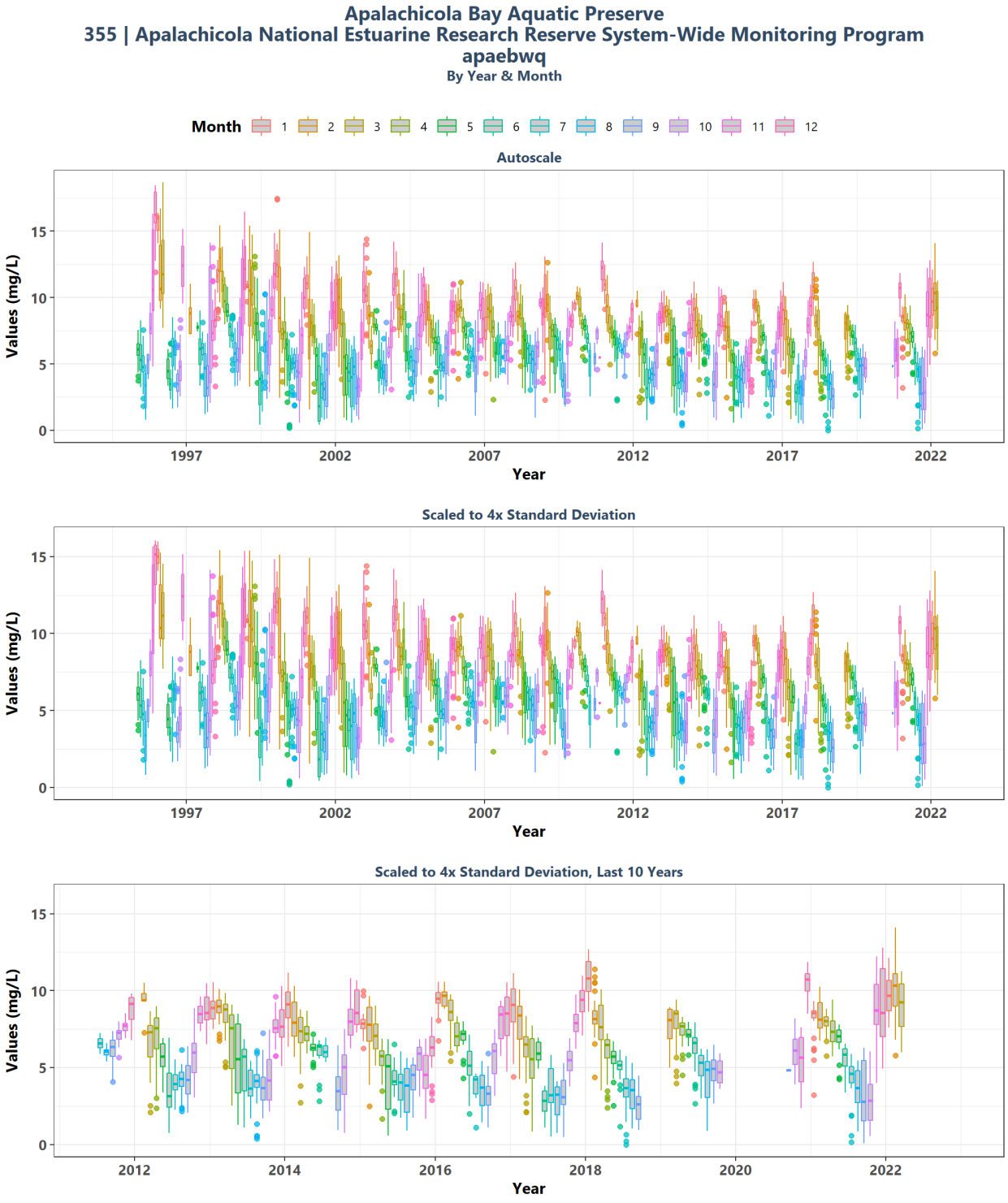

Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apadbwq
By Year
Autoscale
Values (mg/L)
10
5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (mg/L)
10
5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
10
5
2012
2014
2016
2018
2020
2022
Year

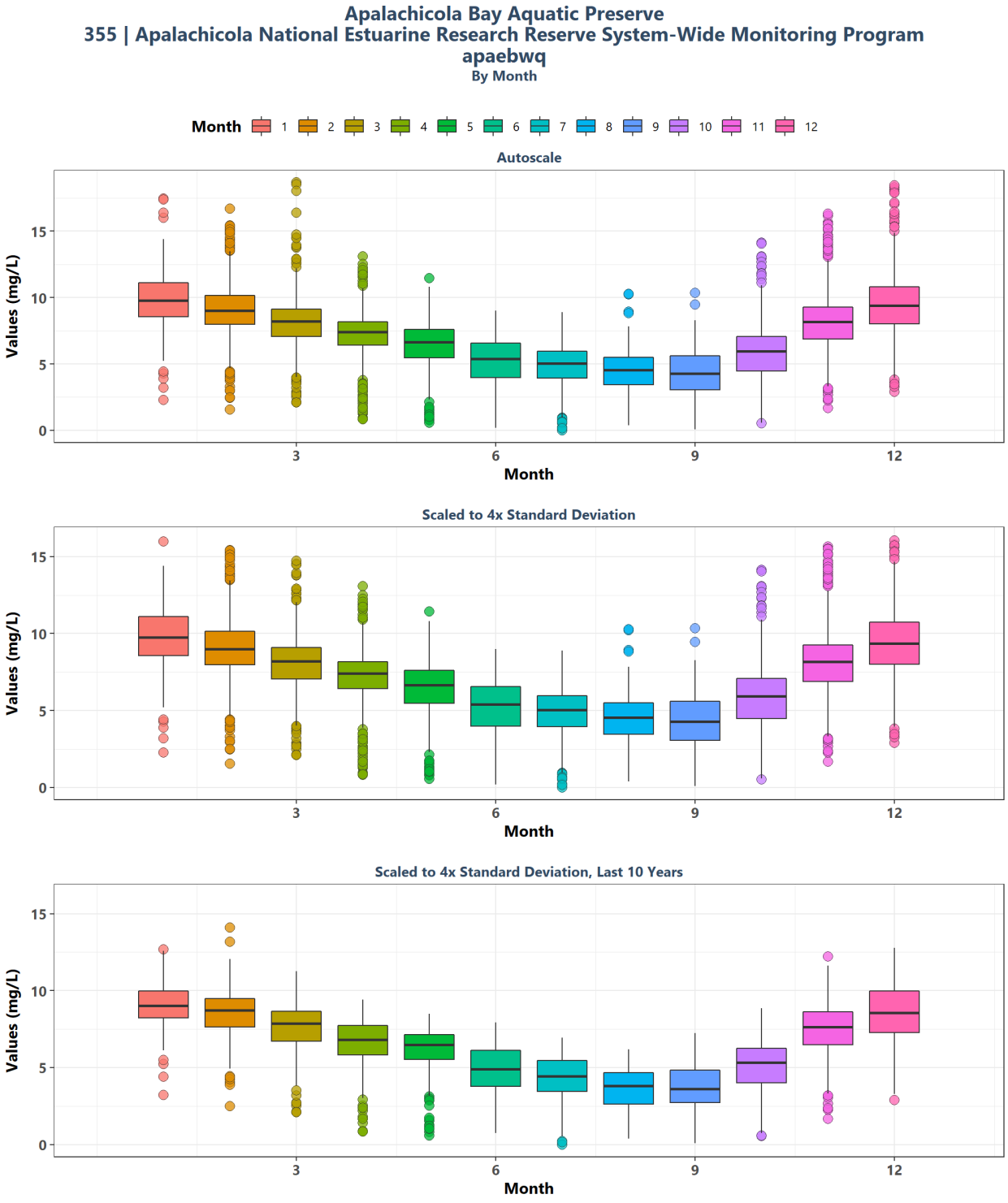

Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apadbwq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
10
5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (mg/L)
10
5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
10
5
2012
2014
2016
2018
2020
2022
Year


Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apadbwq
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
10
5
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
10
5
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
10
5
3
6
9
12
Month

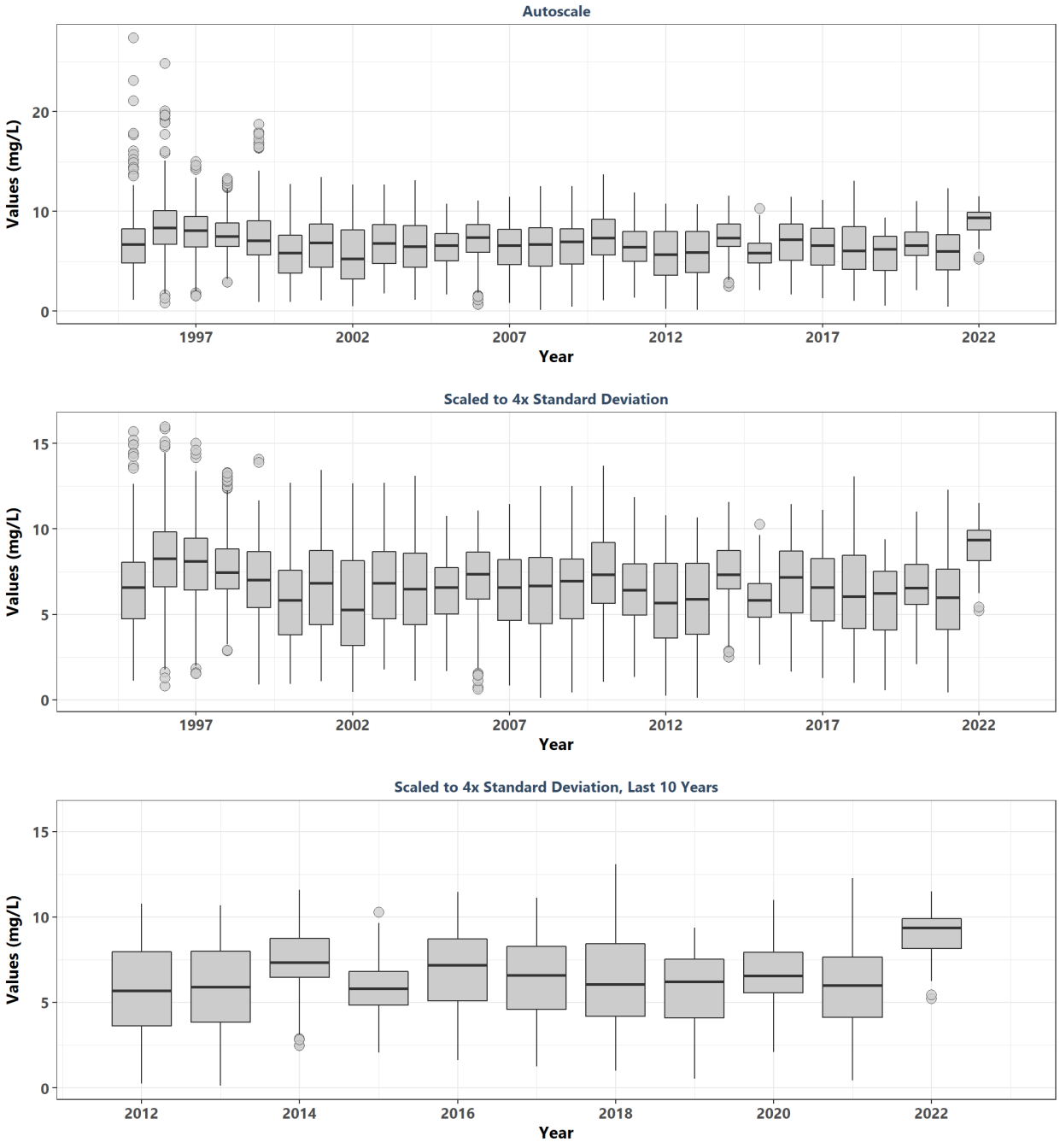
Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaebwq
By Year

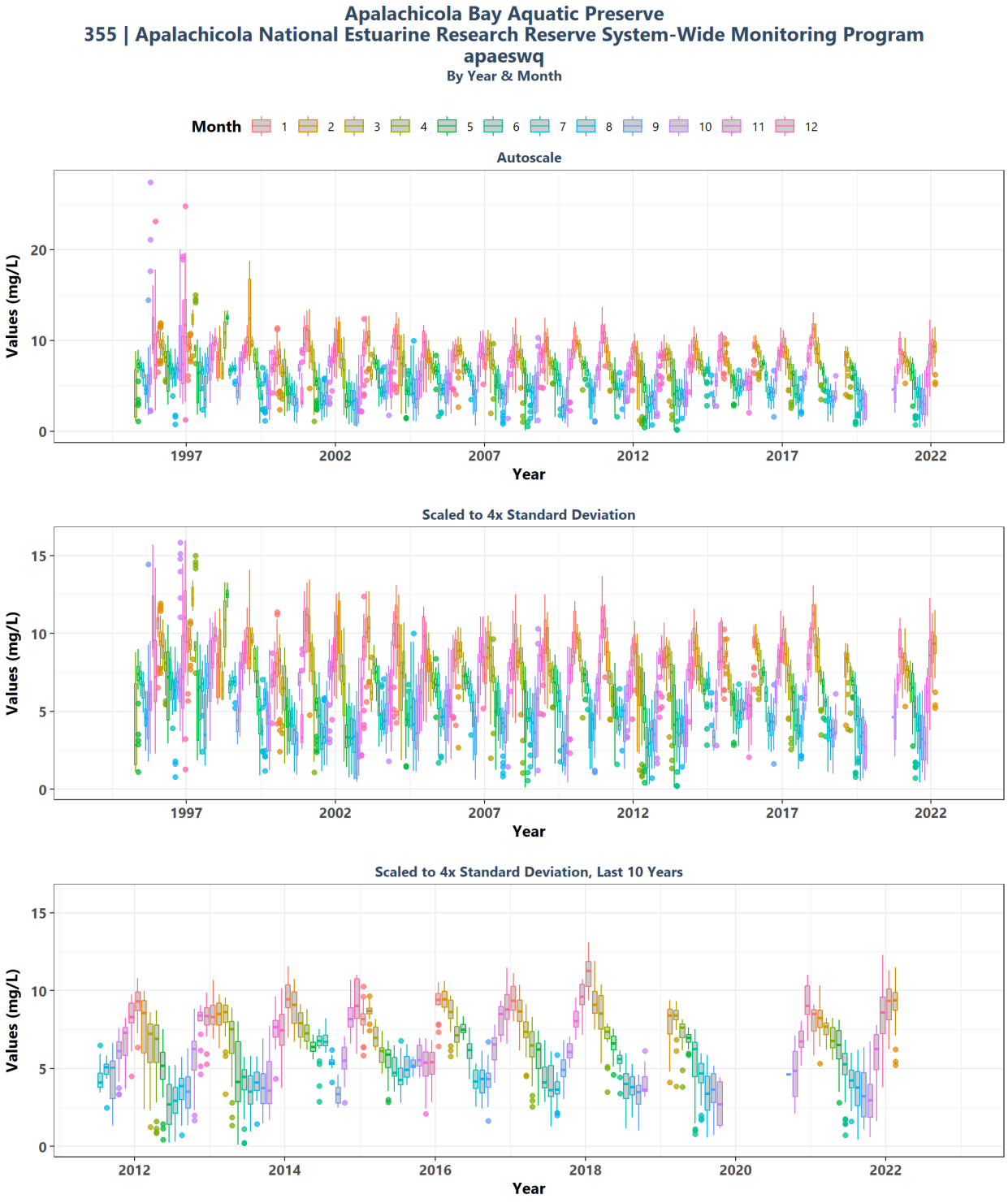



Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaebwq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year

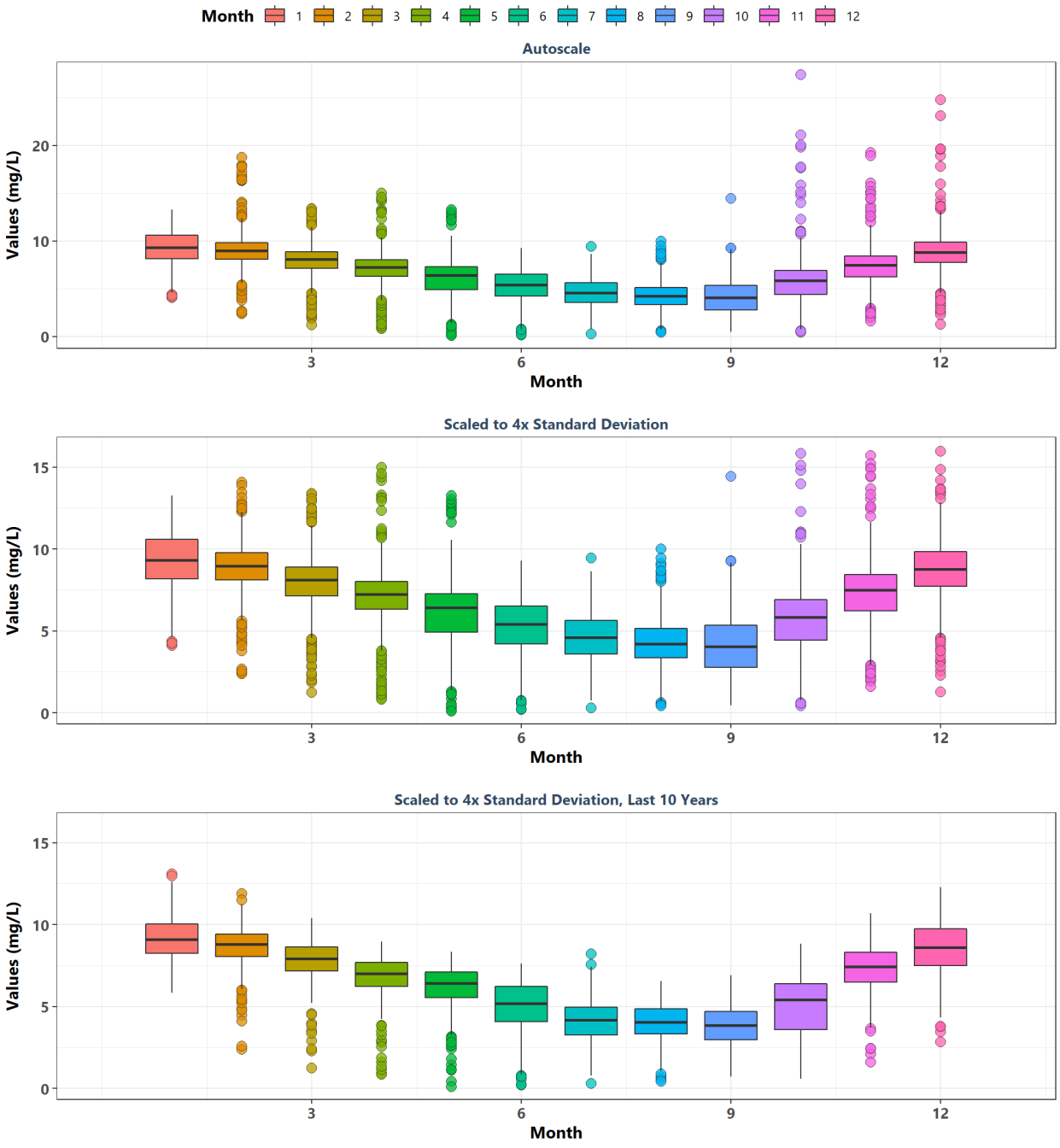

Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaebwq
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Month
Scaled to 4x Standard Deviation
Values (mg/L)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Month

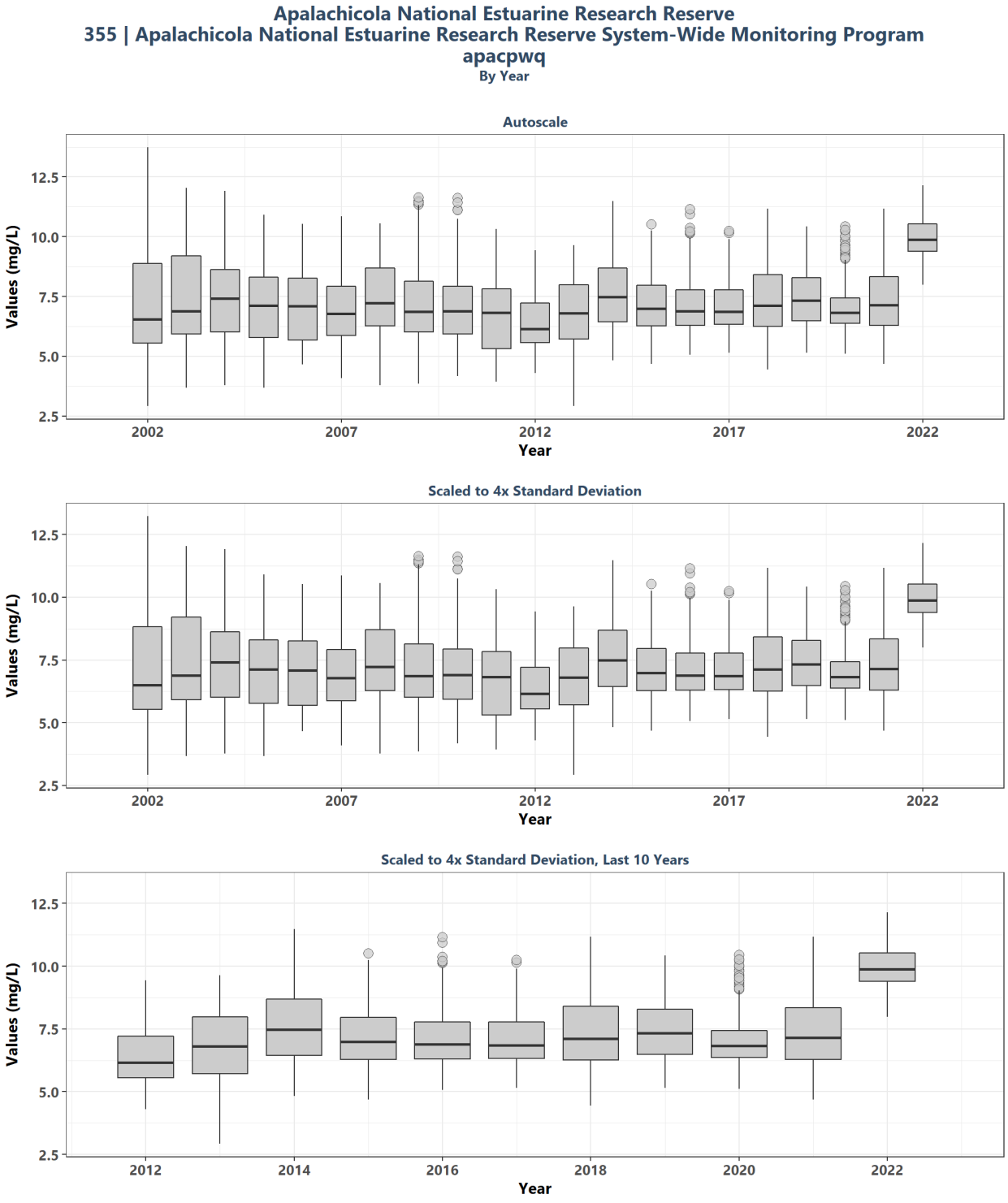
Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaeswq
By Year

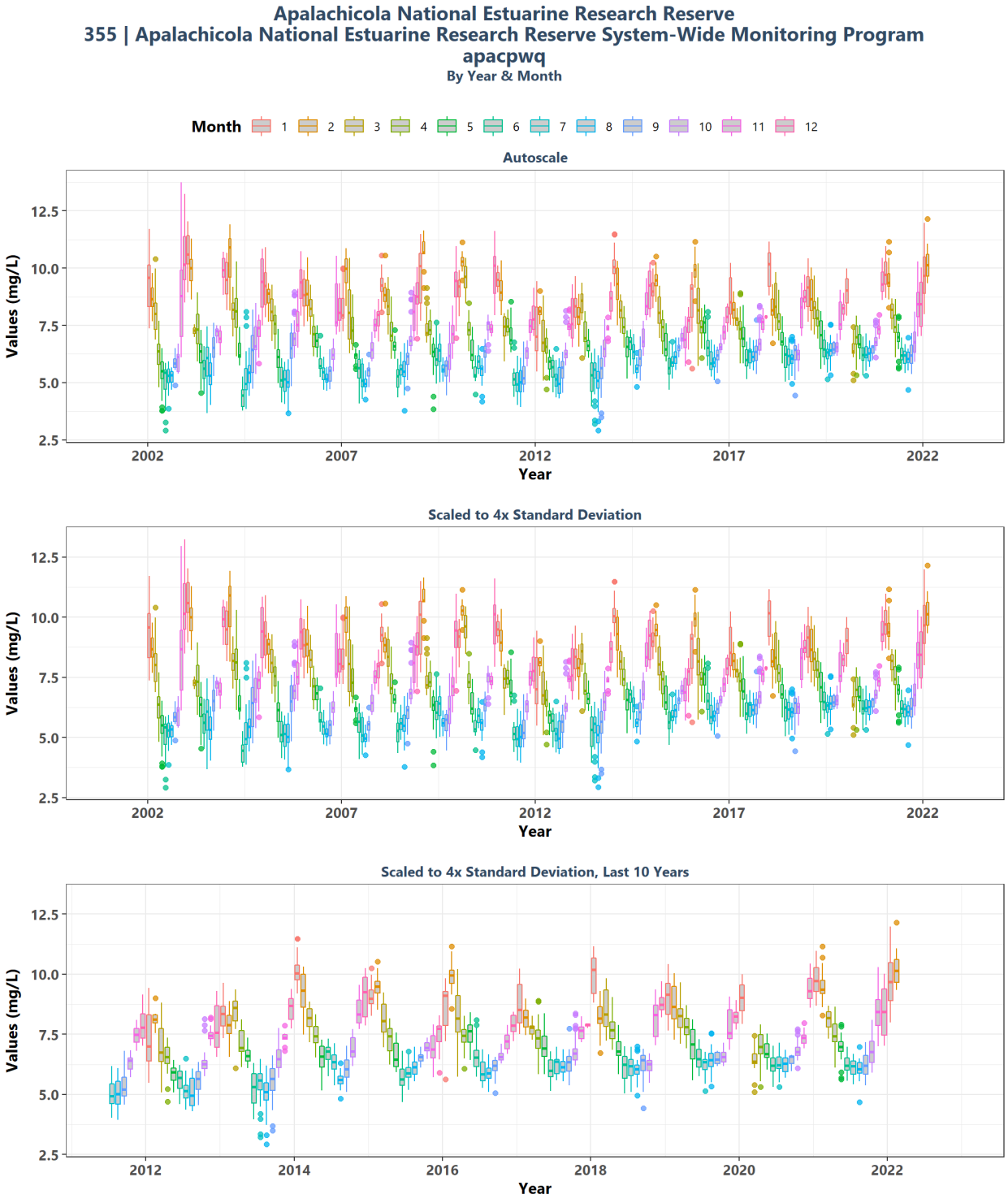


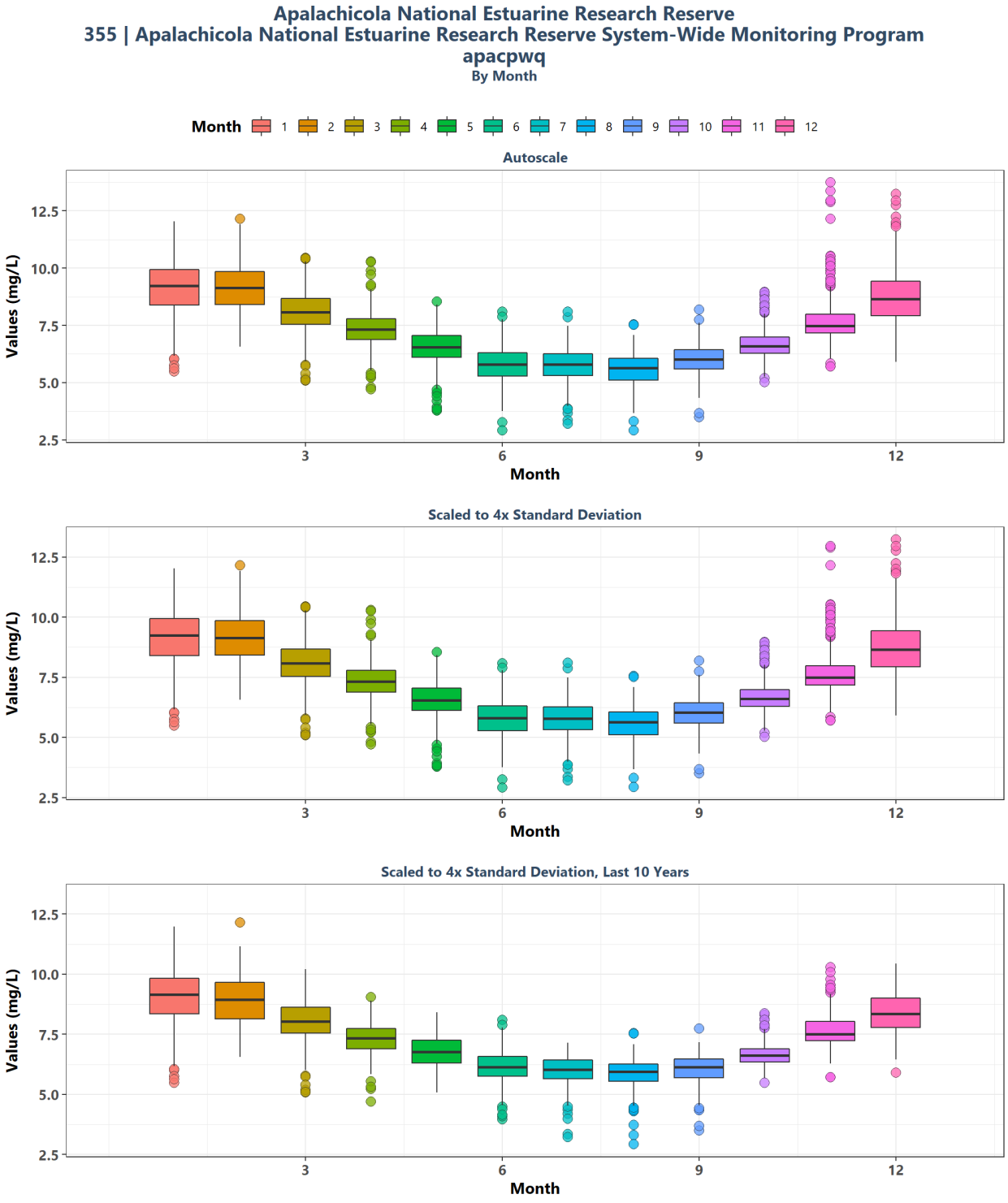

Apalachicola Bay Aquatic Preserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaeswq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year

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

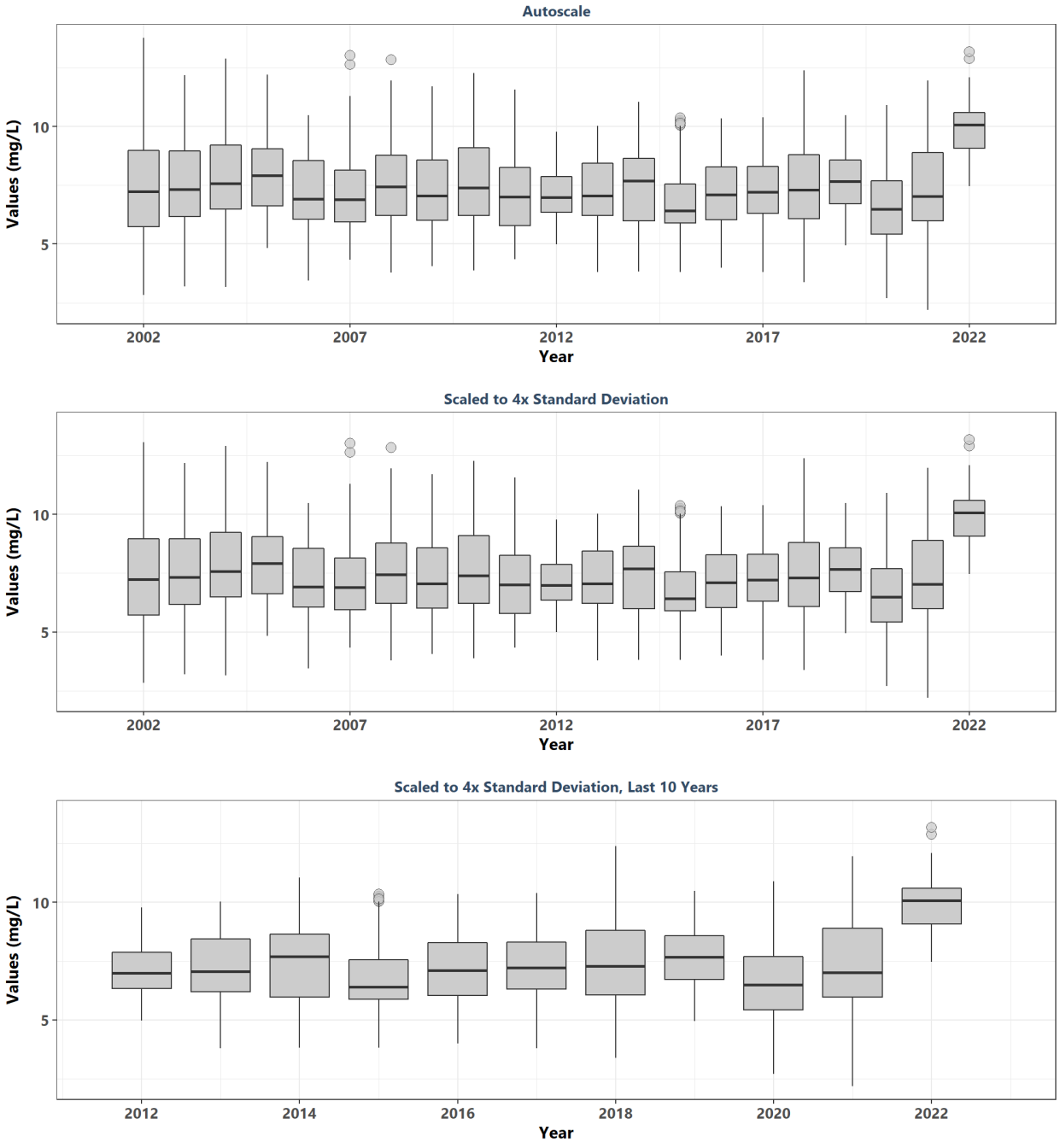


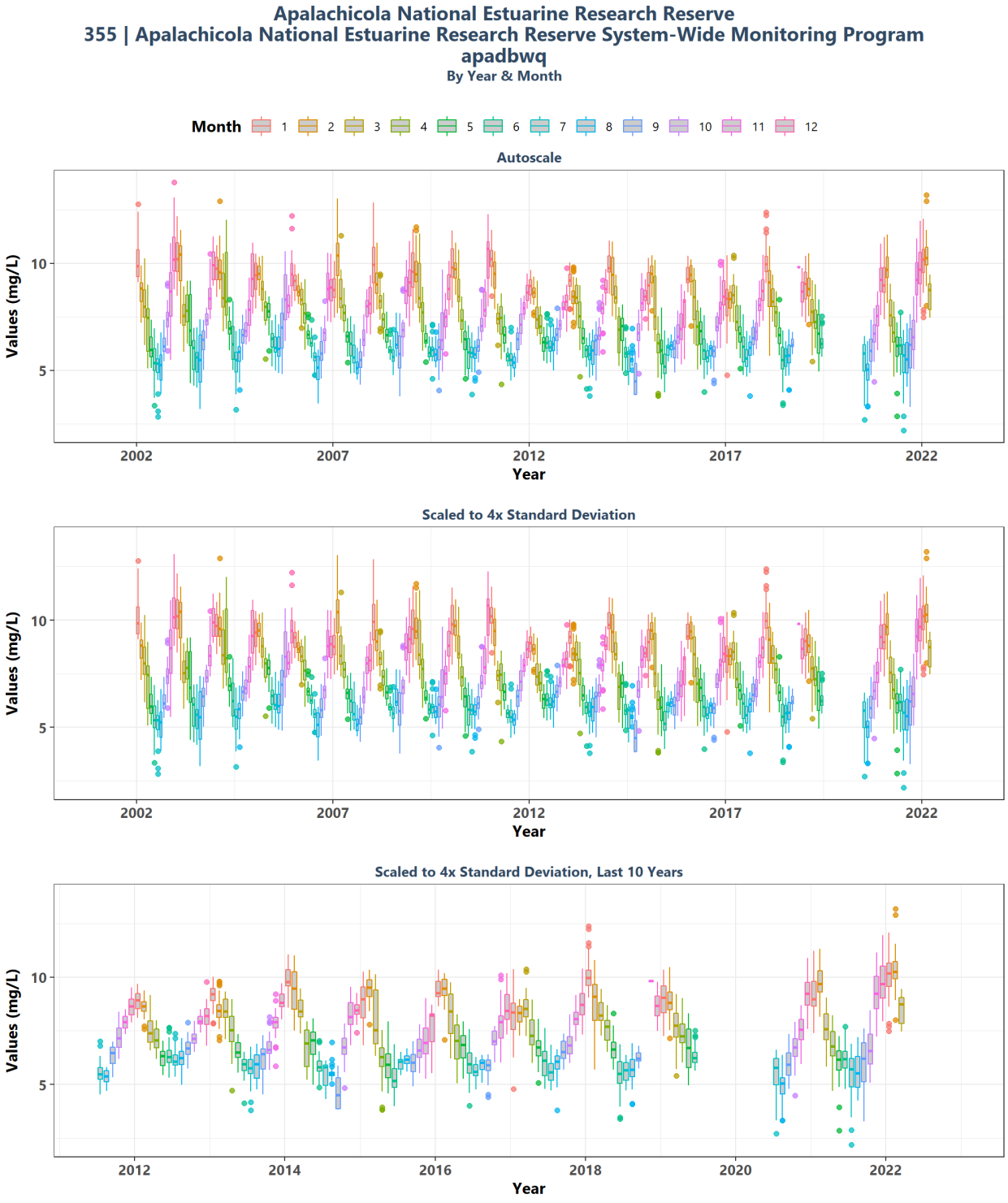

Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apacpwq
By Year
Autoscale
Values (mg/L)
12.5
10.0
7.5
5.0
2.5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (mg/L)
12.5
10.0
7.5
5.0
2.5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
12.5
10.0
7.5
5.0
2.5
2012
2014
2016
2018
2020
2022
Year

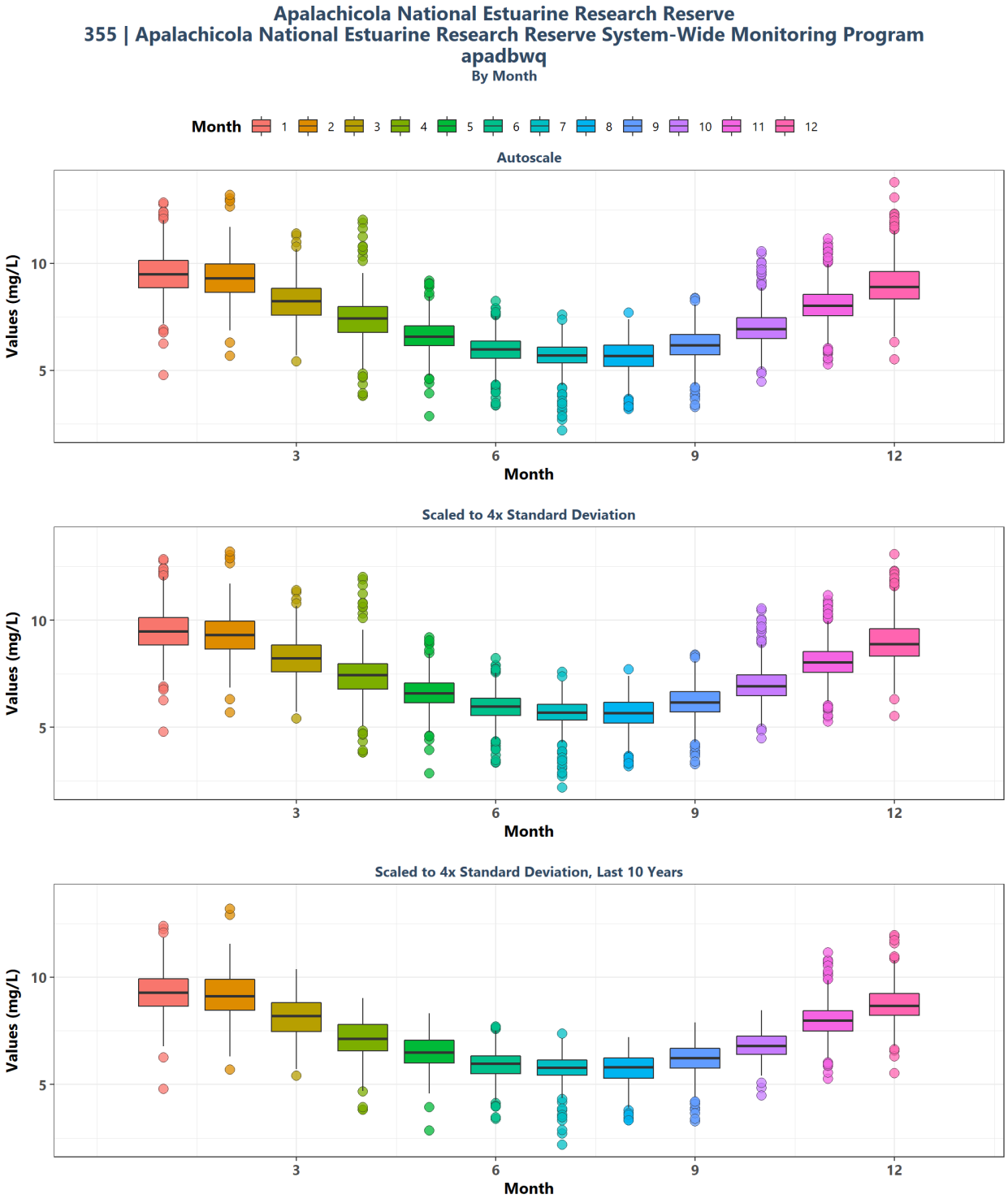

Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apacpwq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation,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


Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apacpwq
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
12.5
10.0
7.5
5.0
2.5
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
12.5
10.0
7.5
5.0
2.5
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
12.5
10.0
7.5
5.0
2.5
3
6
9
12
Month

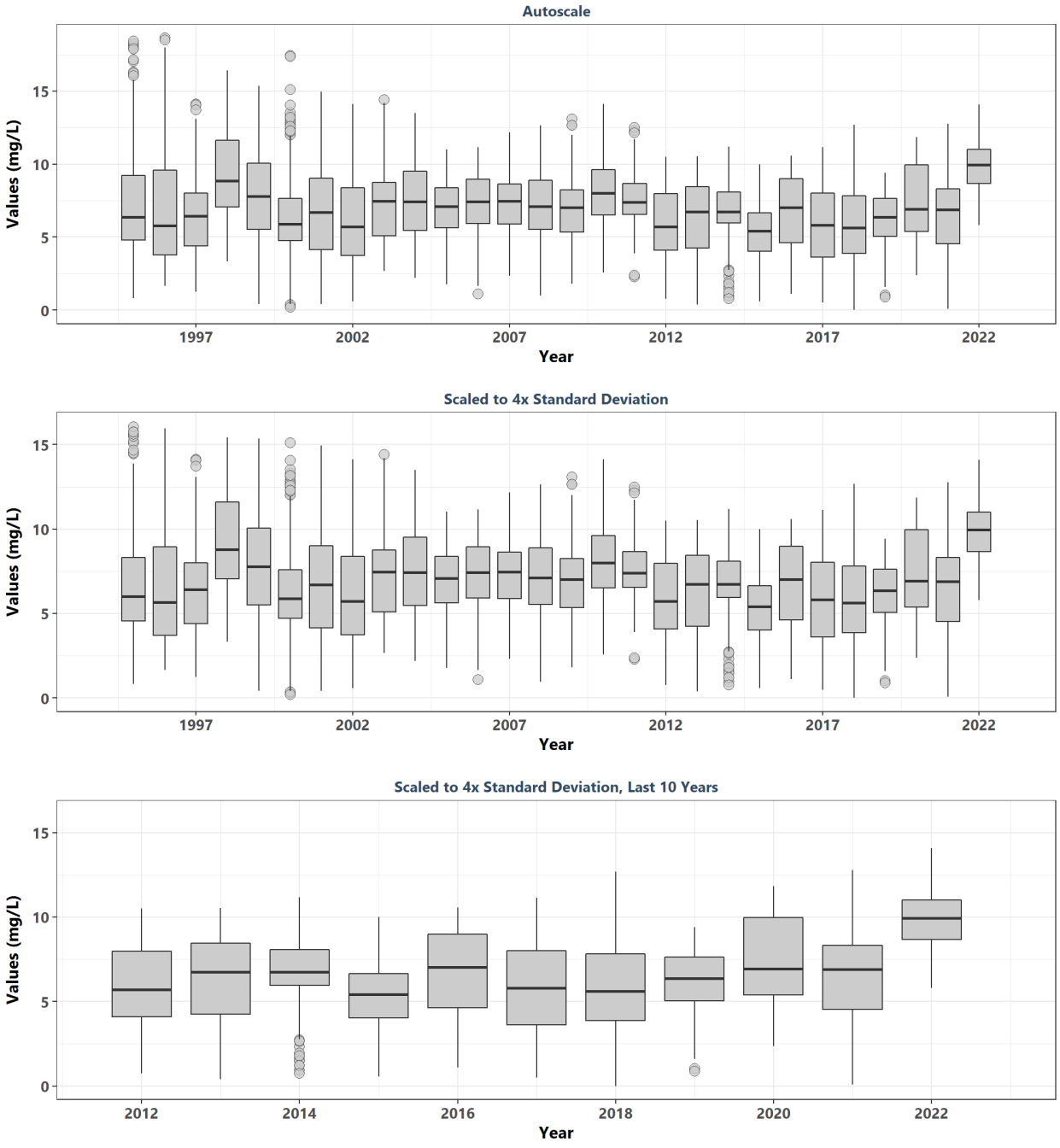
Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apadbwq
By Year

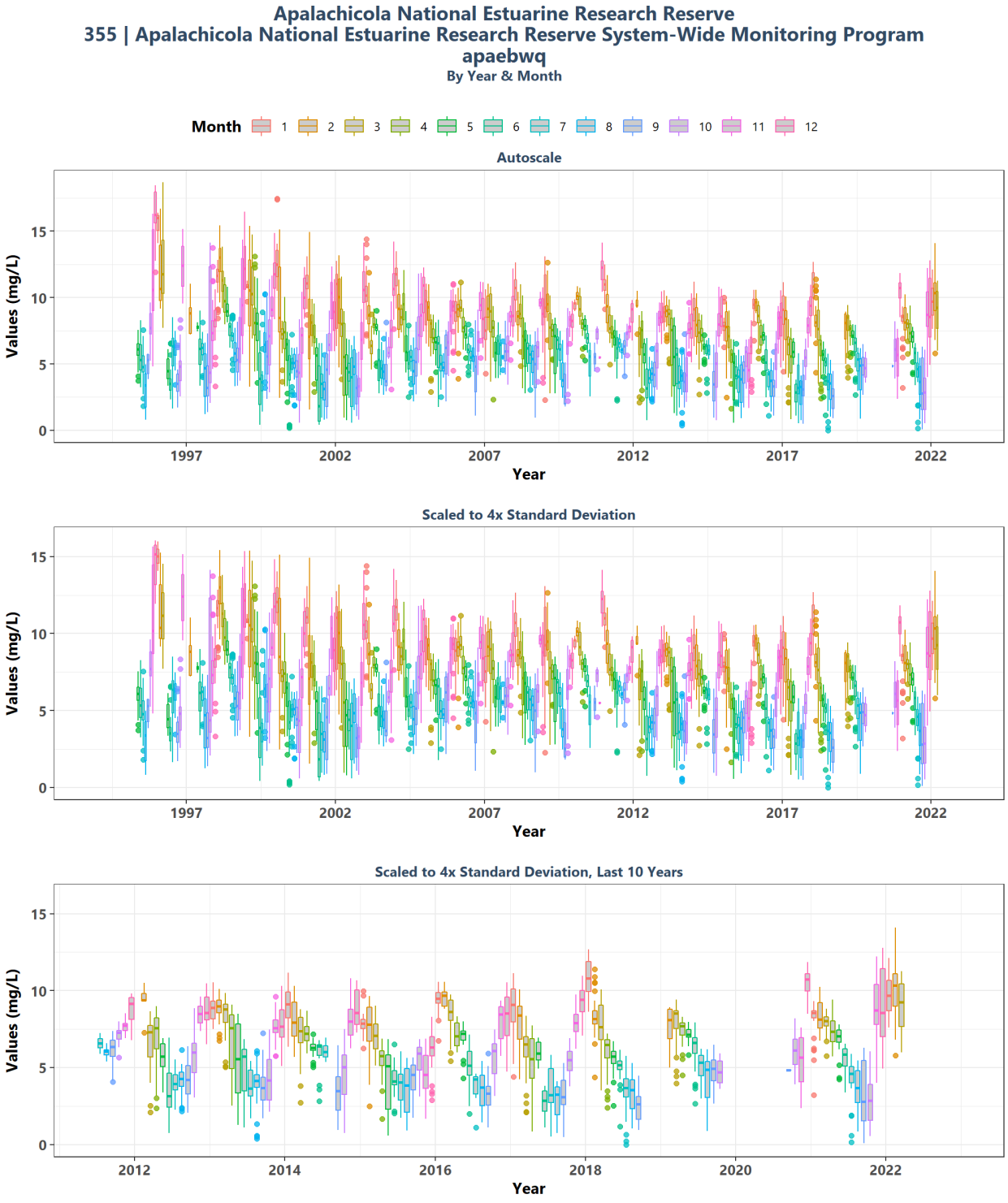



Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apadbwq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
10
5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (mg/L)
10
5
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
10
5
2012
2014
2016
2018
2020
2022
Year

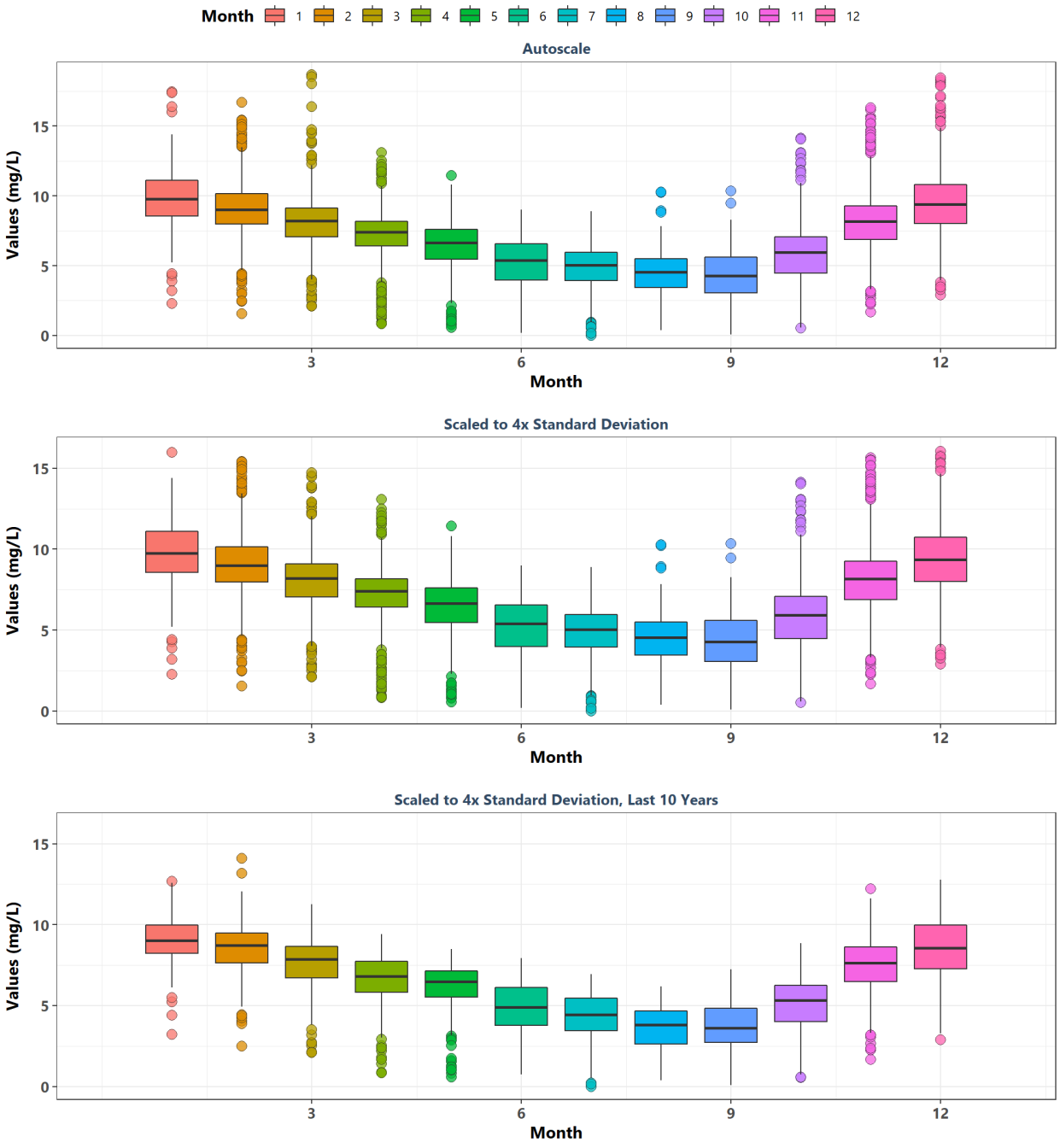

Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apadbwq
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
10
5
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
10
5
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
10
5
3
6
9
12
Month

Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaebwq
By Year

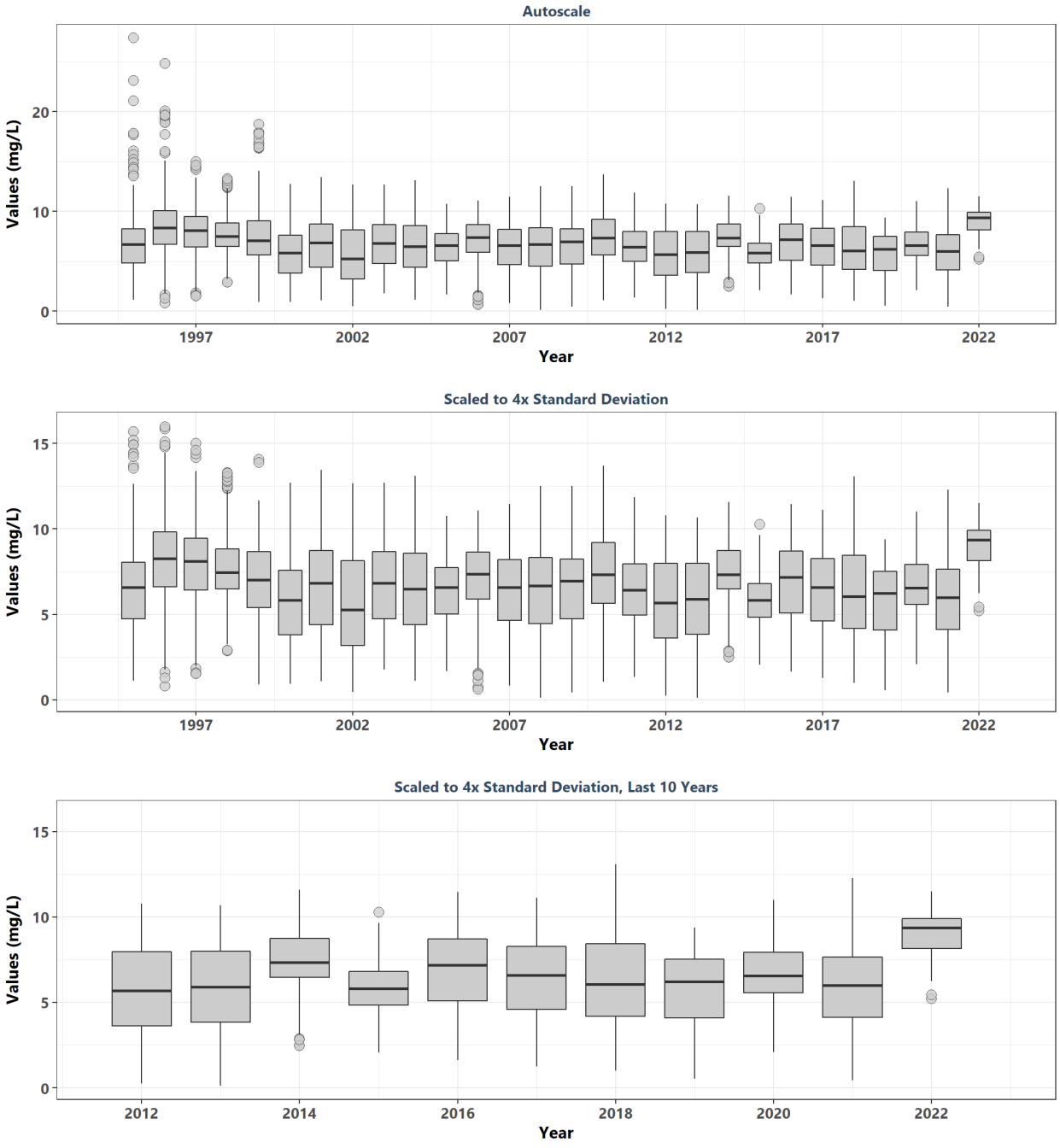


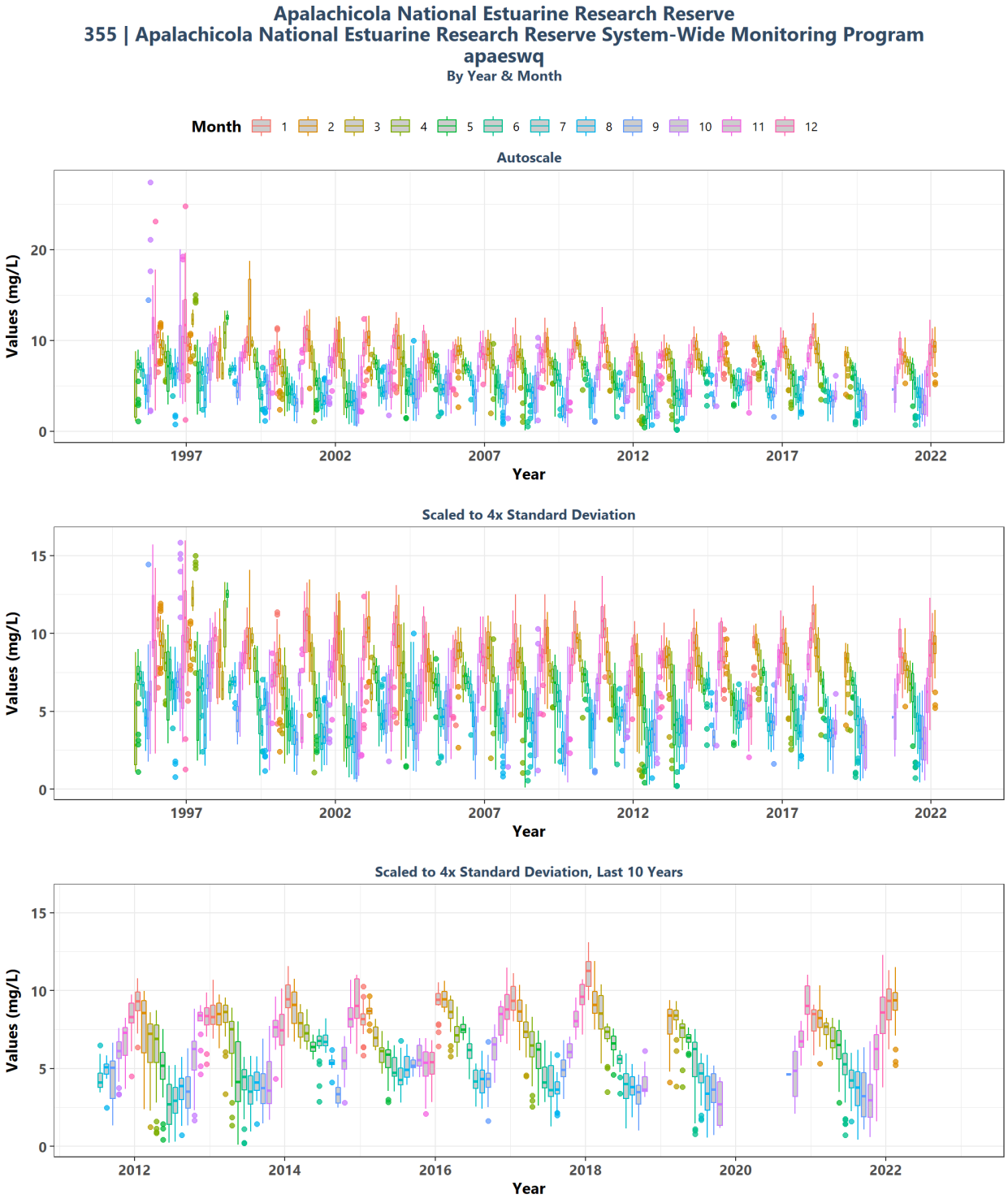

Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaebwq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation,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

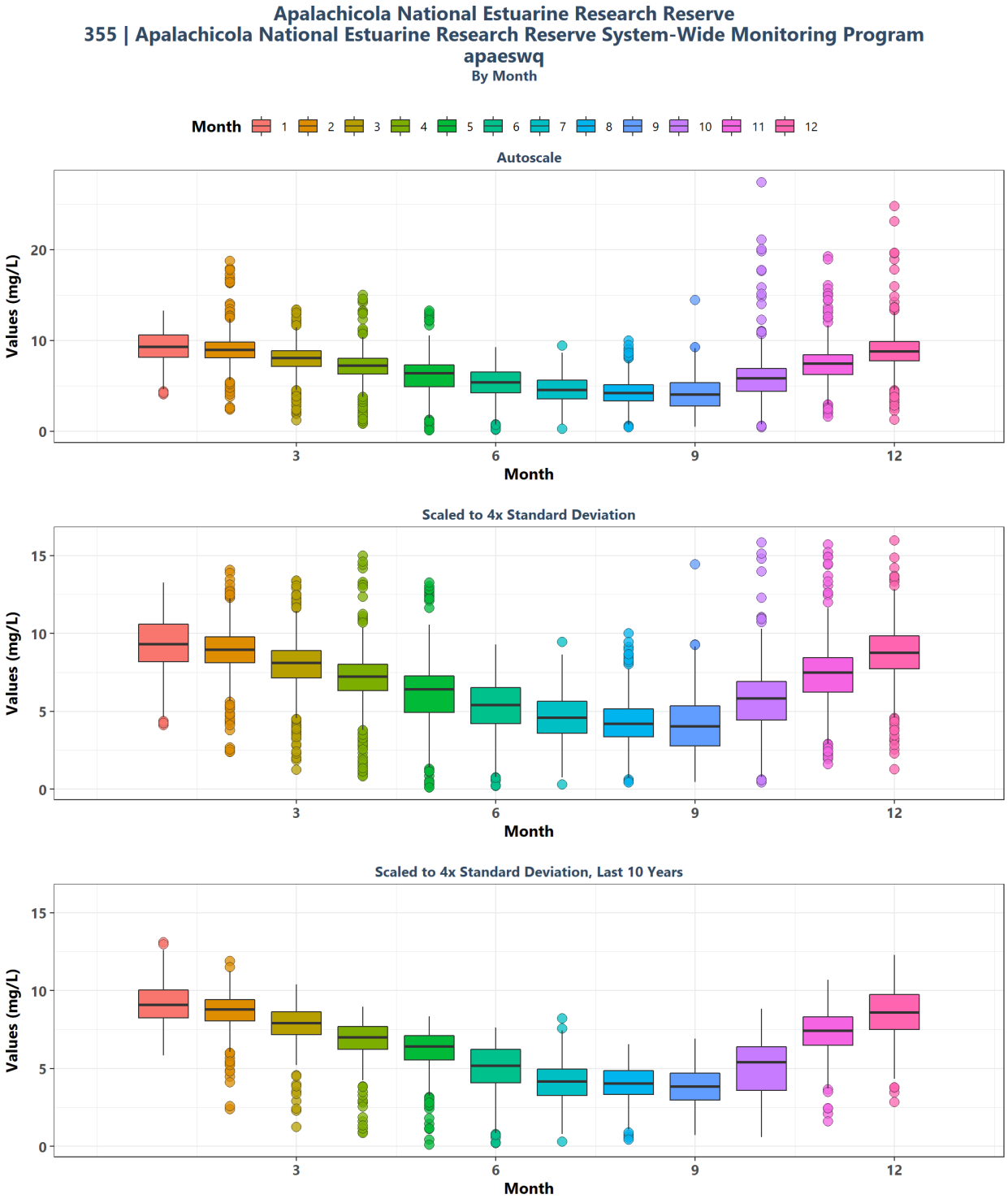
Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaebwq
By Month



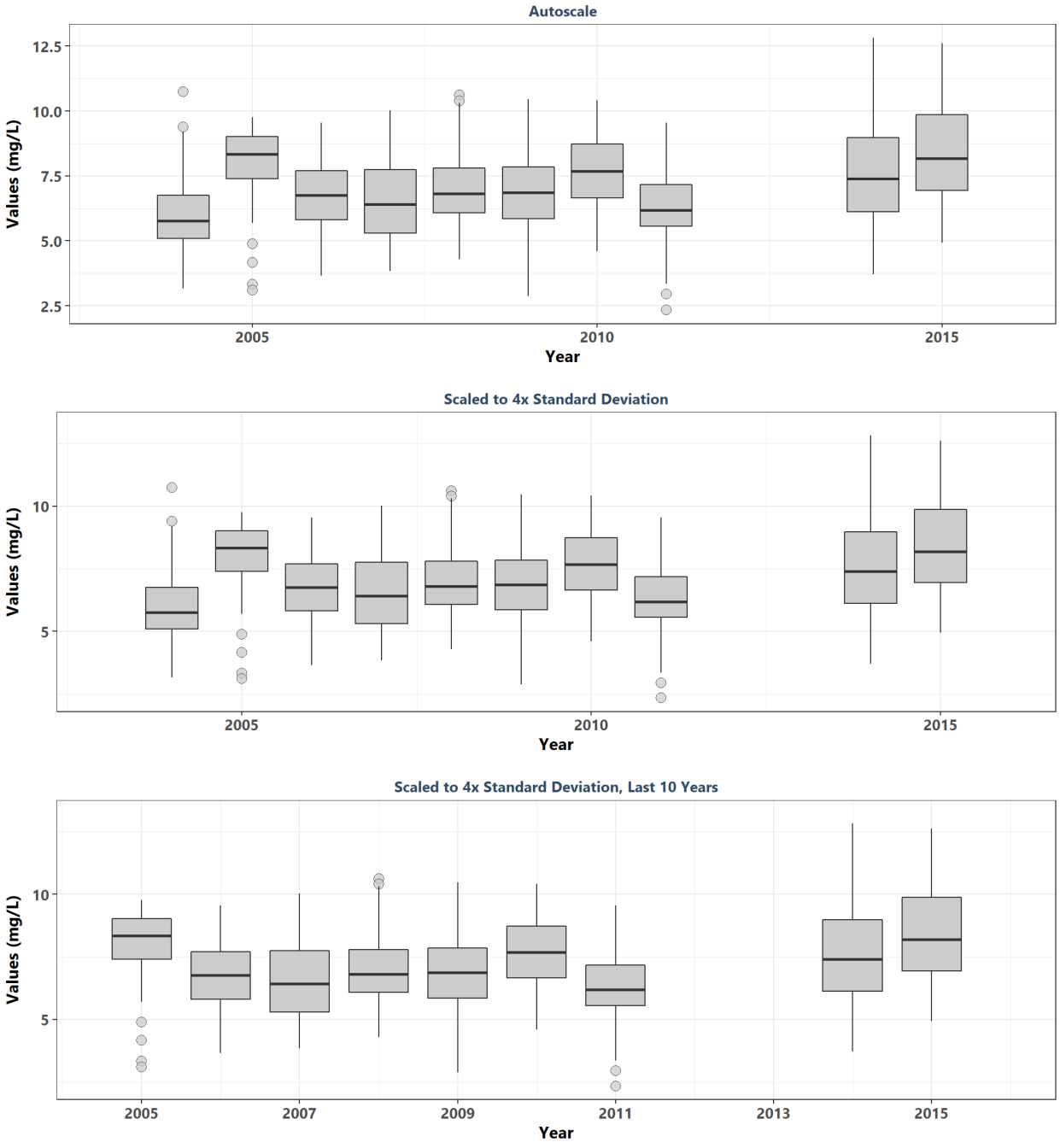
Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaeswq
By Year

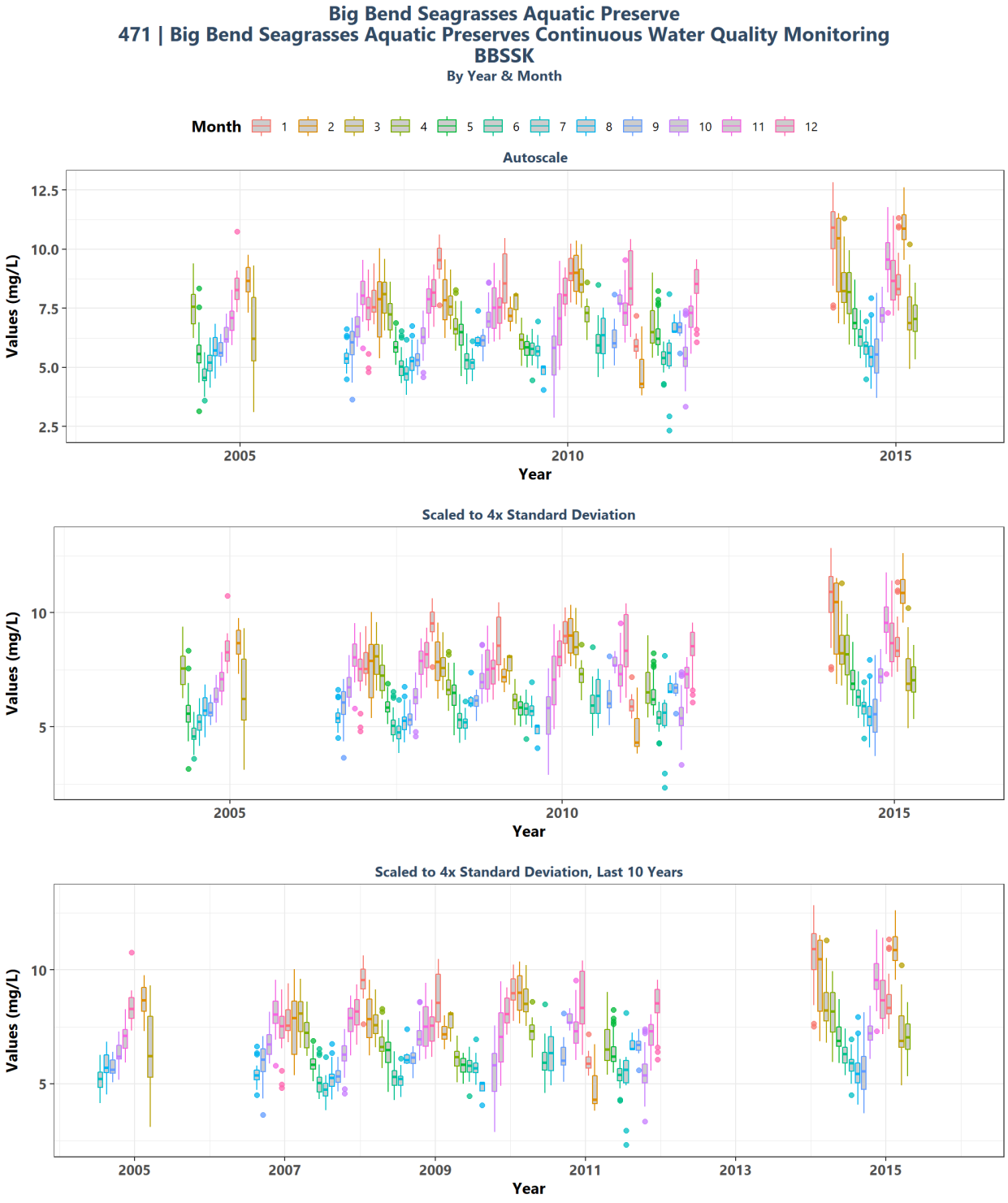



Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaeswq
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
Scaled to 4x Standard Deviation
Values (mg/L)
Year
Scaled to 4x Standard Deviation,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

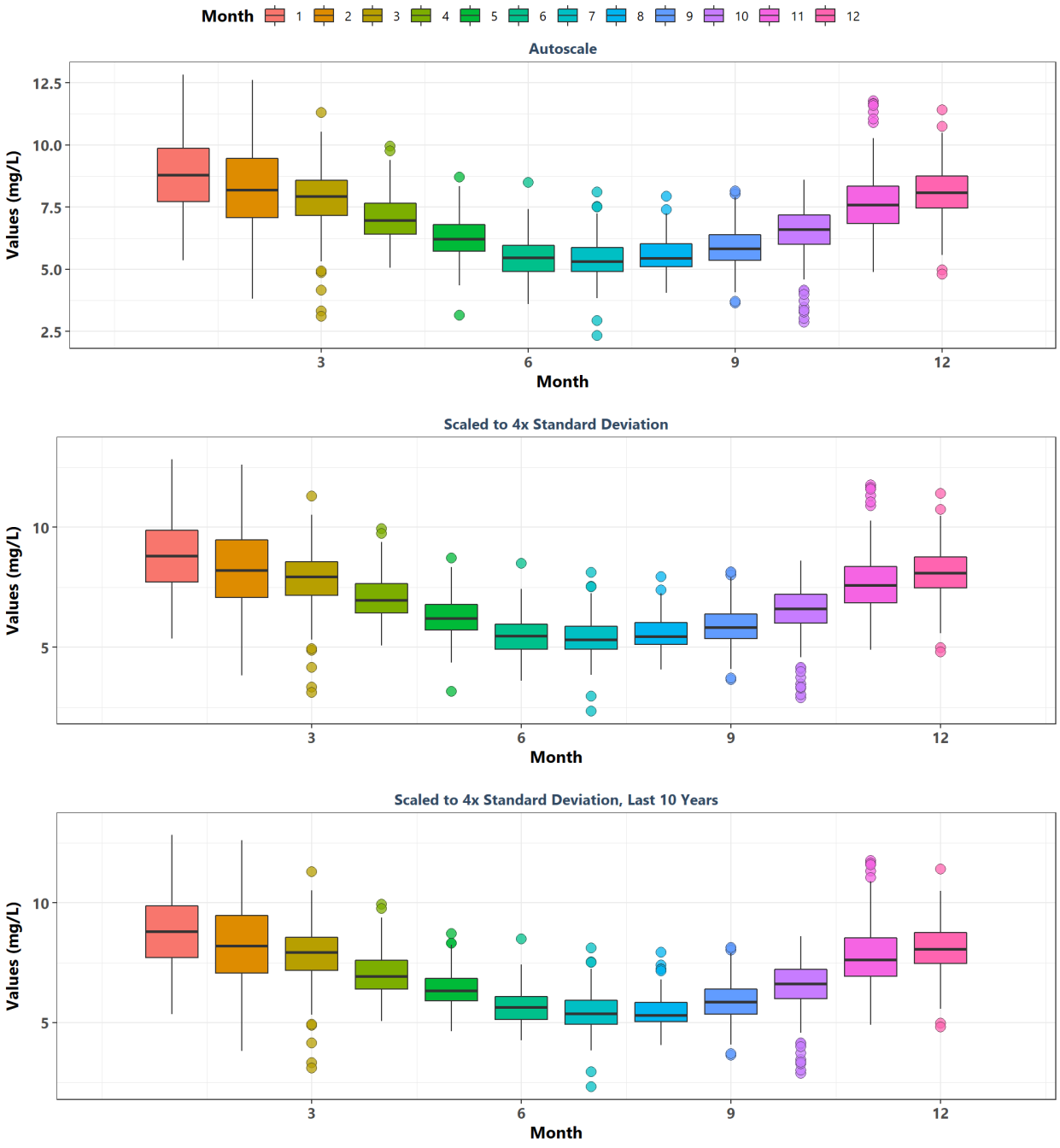

Apalachicola National Estuarine Research Reserve
355 | Apalachicola National Estuarine Research Reserve System-Wide Monitoring Program
apaeswq
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
0
10
20
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (mg/L)
0
5
10
15
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
0
5
10
15
3
6
9
12
Month

Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring
BBSSK
By Year

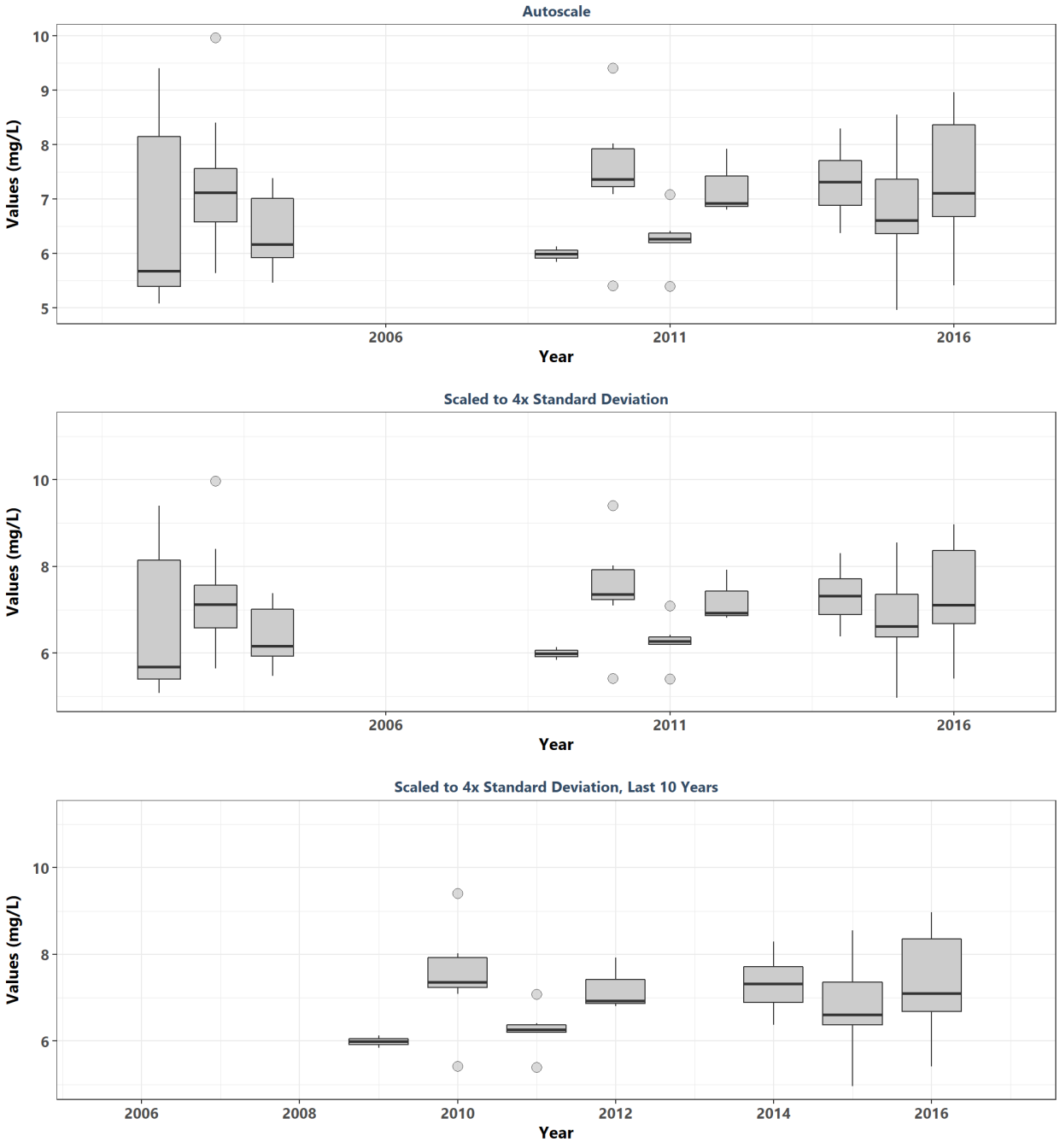


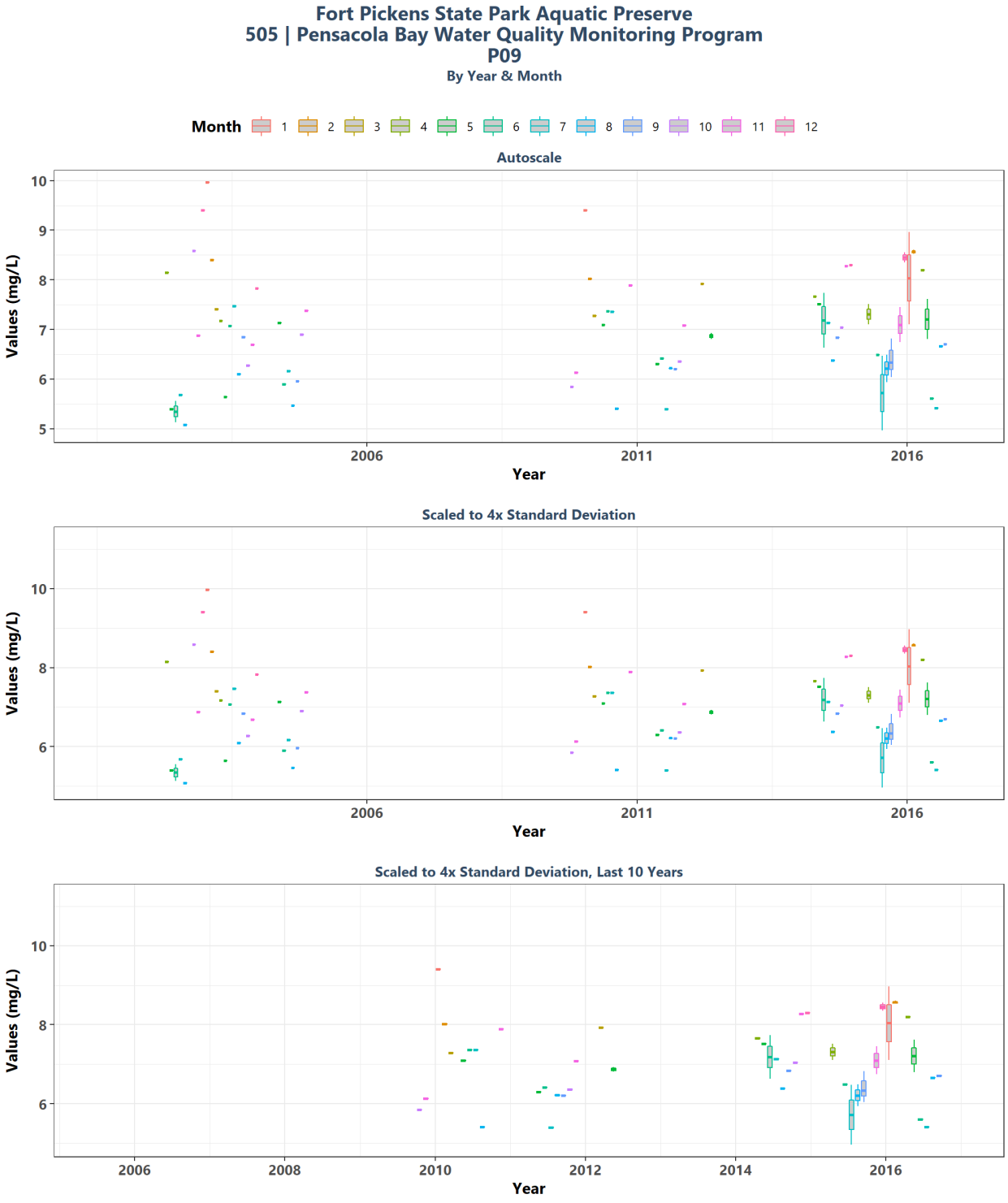

Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring
BBSSK
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
12.5
10.0
7.5
5.0
2.5
2005
2010
2015
Year
Scaled to 4x Standard Deviation
Values (mg/L)
10
5
2005
2010
2015
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
10
5
2005
2007
2009
2011
2013
2015
Year

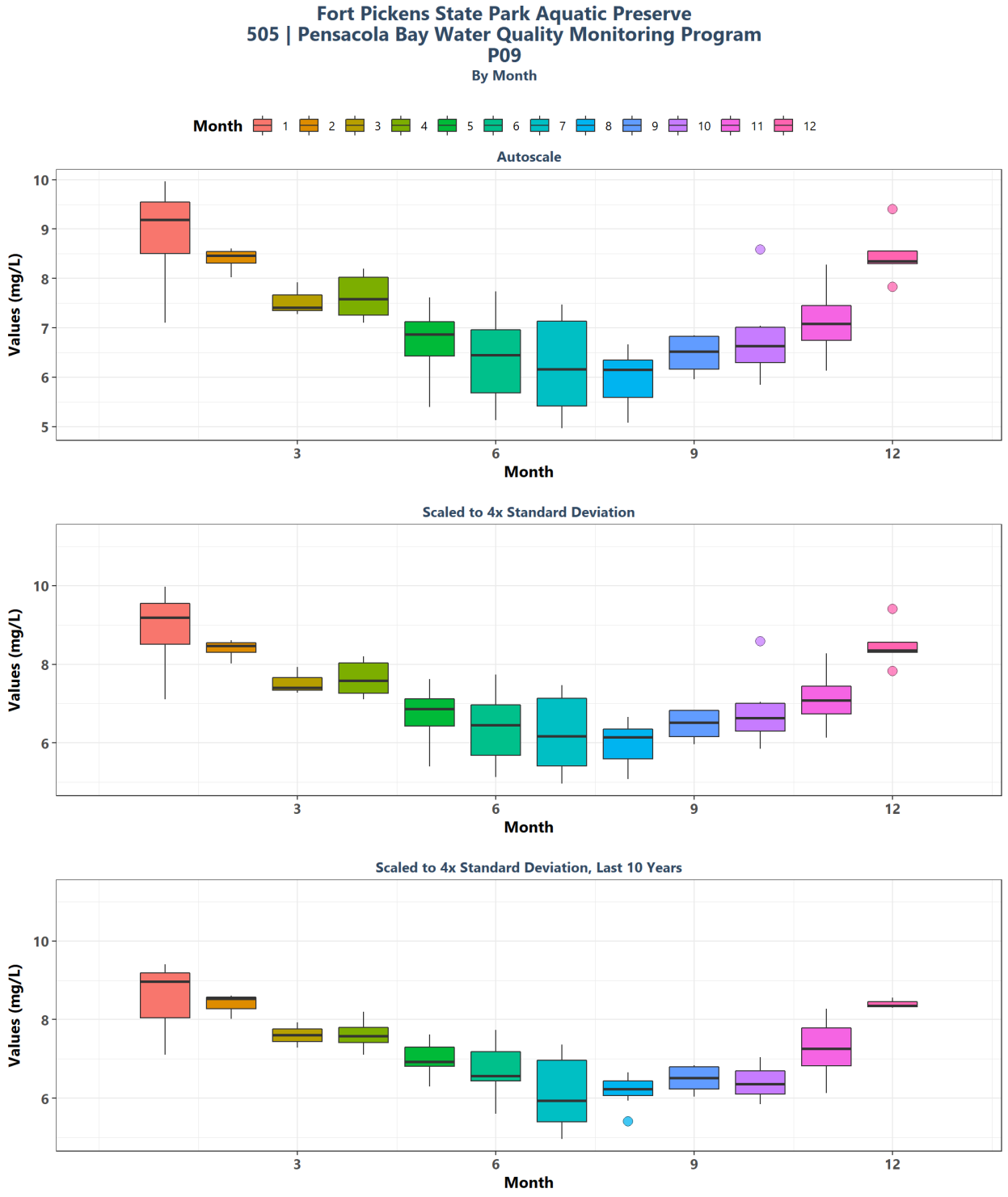
Big Bend Seagrasses Aquatic Preserve
471 | Big Bend Seagrasses Aquatic Preserves Continuous Water Quality Monitoring
BBSSK
By Month



Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program
P09
By Year




Fort Pickens State Park Aquatic Preserve
505 | Pensacola Bay Water Quality Monitoring Program
P09
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (mg/L)
Year
2006
2011
2016
Scaled to 4x Standard Deviation
Values (mg/L)
Year
2006
2011
2016
Scaled to 4x Standard Deviation, Last 10 Years
Values (mg/L)
Year
2006
2008
2010
2012
2014
2016




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