SEACAR Discrete Water Quality Analysis: Sample Surface Colored dissolved organic matter, CDOM

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

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(stringr)
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_Discrete_parameter_ReportCompile.R script.

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

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

```
#MA_All <- fread(here::here("WQ_Discrete/data/ManagedArea.csv"), sep = ",", header = TRUE, stringsAsFac
                #na.strings = "")

#file_in <- "C:/Users/steph/Dropbox/SEACAR_Panzik/SEACAR_Panzik/WQ_Discrete/data/Combined_WQ_WC_NUT_Wat
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="")

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

```
data[, `:=` (relyear = Year - min(Year), relyear_dd = DecDate - min(DecDate)), by = "ManagedAreaName"]
data <- data[ParameterName == parameter & str_detect(ActivityType, activity) & RelativeDepth == depth &
```

Data Filtering and Data Impacted by Specific Value Qualifiers

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

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

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

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

```
# param_name <- "Water_Temperature"
# out_dir <- here::here("WQ_Discrete/output/by_parameter/")
# APP_Plots <- TRUE

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

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

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

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

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

```
}

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


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

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

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

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

MA_Summ <- MA_Summ %>%
   select(AreaID, ManagedAreaName, ParameterName, RelativeDepth, ActivityType,
          SufficientData, everything())
MA_Summ <- as.data.frame(MA_Summ[order(MA_Summ$ManagedAreaName), ])


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

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

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

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

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

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

```
data$VQ_Plot <- data$ValueQualifier

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

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


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

```
## Number of Measurements: 10694, Number Passed Filter: 10694
## I Codes: 317 (2.964279%)
## Q Codes: 219 (2.047877%)
## U Codes: 225 (2.103984%)
```

```
data_summ <- data %>%
   group_by(AreaID, ManagedAreaName) %>%
   dplyr::summarize(ParameterName=parameter,
              RelativeDepth=depth,
              ActivityType=activity,
              N_Total=length(ResultValue),
              N_AnalysisUse=length(ResultValue[SufficientData==TRUE]),
```

```
            N_H=length(grep("H", data$ValueQualifier[data$ProgramID==476])),
            perc_H=100*N_H/length(data$ValueQualifier),
            N_I=length(grep("I", data$ValueQualifier)),
            perc_I=100*N_I/length(data$ValueQualifier),
            N_Q=length(grep("Q", data$ValueQualifier)),
            perc_Q=100*N_Q/length(data$ValueQualifier),
            N_S=length(grep("S", data$ValueQualifier)),
            perc_S=100*N_S/length(data$ValueQualifier),
            N_U=length(grep("U", data$ValueQualifier)),
            perc_U=100*N_U/length(data$ValueQualifier))

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

Managed Area Statistics

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

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

```
MA_YM_Stats <- data[data$Use_In_Analysis==TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, Year, Month) %>%
   dplyr::summarize(ParameterName=parameter,
            RelativeDepth=depth,
            ActivityType=activity,
            N_Data=length(ResultValue),
            Min=min(ResultValue),
            Max=max(ResultValue),
            Median=median(ResultValue),
```

```
            Mean=mean(ResultValue),
            StandardDeviation=sd(ResultValue),
            ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                             collapse=', '))
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                               MA_YM_Stats$Year,
                                               MA_YM_Stats$Month), ])
fwrite(MA_YM_Stats, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                           "_ManagedArea_YearMonth_Stats.txt"), sep="|")
rm(MA_YM_Stats)

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

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

Monitoring Location Statistics

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

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

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

Mon_Stats <- as.data.table(Mon_Stats[order(Mon_Stats$ManagedAreaName,
                                           Mon_Stats$ProgramName,
                                           Mon_Stats$ProgramID,
                                           Mon_Stats$ProgramLocationID), ])
fwrite(Mon_Stats, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                         "_MonitoringLoc_Stats.txt"), sep="|")
rm(Mon_Stats)
```

Seasonal Kendall Tau Analysis

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

The following steps are performed:

1. Define the functions used in the analysis
2. Check to see if there are any groups to run analysis on.
3. Take the `data` variable and only include rows that have a `SufficientData` value of `TRUE`
4. Group data that have the same `ManagedAreaName`.

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

```
tauSeasonal <- function(data, independent, stats.median, stats.minYear,
                        stats.maxYear, seasondata = MA_M_Stats[MA_M_Stats$ManagedAreaName == MA_Include
  setDT(data)
  tau <- NULL
   tryCatch({ken <- kendallSeasonalTrendTest(
      y = data$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(AreaID = unique(data$AreaID),
                ManagedAreaName = unique(data$ManagedAreaName),
                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[, `:=` (AreaID = unique(data$AreaID),
                      ManagedAreaName = unique(data$ManagedAreaName),
                      season = 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(nrow(seasondat_s) < 3 | length(unique(seasondat_s$Year)) < 3 | is.na(seasonresults[season == s,
    next

  } else{
    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, MA_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$relyear, na.rm = TRUE)
  # Calculate Kendall Tau and Slope stats, then update appropriate columns and table
  seasondata <- MA_M_Stats[MA_M_Stats$ManagedAreaName == MA_Include[i]]
  KT <- tauSeasonal(data, TRUE, stats.median,
                    stats.minYear, stats.maxYear, seasondata)
  # if (is.null(KT[9])) {
  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$ManagedAreaName
c_names <- c("AreaID", "ManagedAreaName", "Season", "Median", "Independent",
             "tau", "z", "p_z", "chi_sq", "p_chi_sq", "SennSlope", "SennIntercept", "Trend")
if(n==0){
   KT.Stats <- data.frame(matrix(ncol=length(c_names),
                                 nrow=length(MA_Summ$ManagedAreaName)))
   colnames(KT.Stats) <- c_names
   # KT.Stats[, c("AreaID", "ManagedAreaName")] <-
      # MA_Summ[, c("AreaID", "ManagedAreaName")]
} else{
   for (i in 1:n) {
      x <- nrow(data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_Include[i], ])
      if (x>0) {
         KT.Stats <- runStats(data[data$Use_In_Analysis == TRUE &
                                      data$ManagedAreaName ==
                                      MA_Include[i], ], MA_M_Stats)
      }
   }
   KT.Stats <- as.data.frame(KT.Stats)
   # c_names <- c("AreaID", "ManagedAreaName", "Season", "Median", "Independent",
   #              "tau", "z", "p_z", "chi_sq", "p_chi_sq", "SennSlope", "SennIntercept", "Trend")
   if(dim(KT.Stats)[2]==1){
      KT.Stats <- as.data.frame(t(KT.Stats))
   }
   colnames(KT.Stats) <- c_names
   rownames(KT.Stats) <- seq(1:nrow(KT.Stats))
   KT.Stats$tau <- round(as.numeric(KT.Stats$tau), digits=4)
   KT.Stats$z <- round(as.numeric(KT.Stats$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(MA_Summ, KT.Stats,
                              by=c("AreaID", "ManagedAreaName"), all=TRUE)

KT.Stats <- as.data.table(KT.Stats[order(KT.Stats$ManagedAreaName, KT.Stats$Season), ])
KT.Stats2 <- copy(KT.Stats)
KT.Stats[, `:=` (RelativeDepth = depth, Units = unit)]
KT.Stats_all <- rbind(KT.Stats_all, KT.Stats)
```

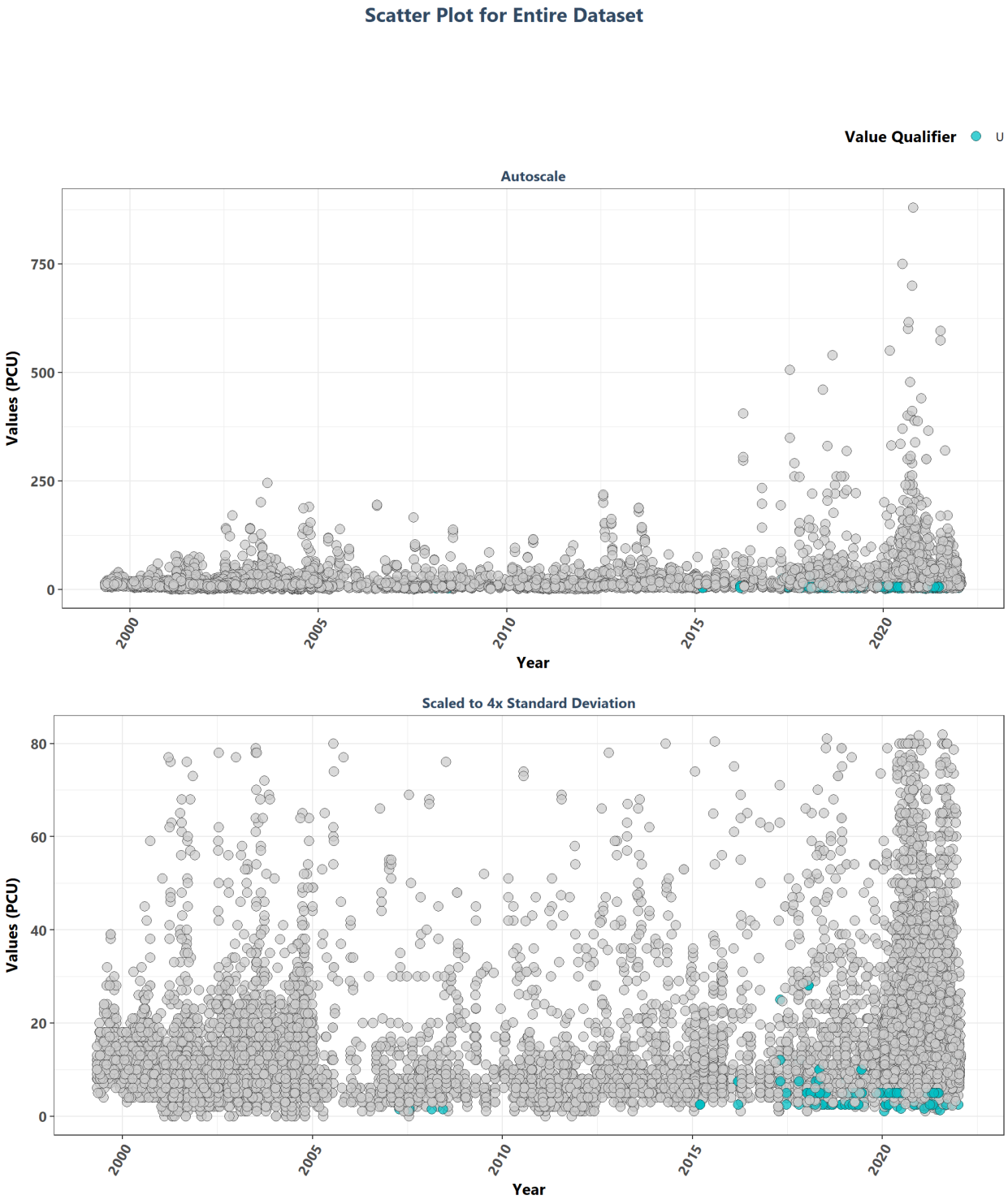
```
KT.Stats2$MonitoringID <- NULL
fwrite(KT.Stats2, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                         "_KendallTau_Stats.txt"), sep="|")
rm(KT.Stats2)
data <- data[!is.na(data$ResultValue),]
```

Appendix I: Scatter Plot of Entire Dataset

This part will create a scatter plot of the all data that passed initial filtering criteria with points colored based on specific value qualifiers. The values determined at the beginning (`year_lower`, `year_upper`, `min_RV`, `mn_RV`, `x_scale`, and `y_scale`) are solely for use by the plotting functions and are not output as part of the computed statistics.

```
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.y=element_text(face="bold"))

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


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

```
                                    "SU"="#7CAE00"), na.value="#cccccc")
   } else {
      scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
   }}


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

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

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

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



Appendix II: Dataset Summary Box Plots

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

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

This set of box plots are grouped by year.

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

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

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

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

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

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

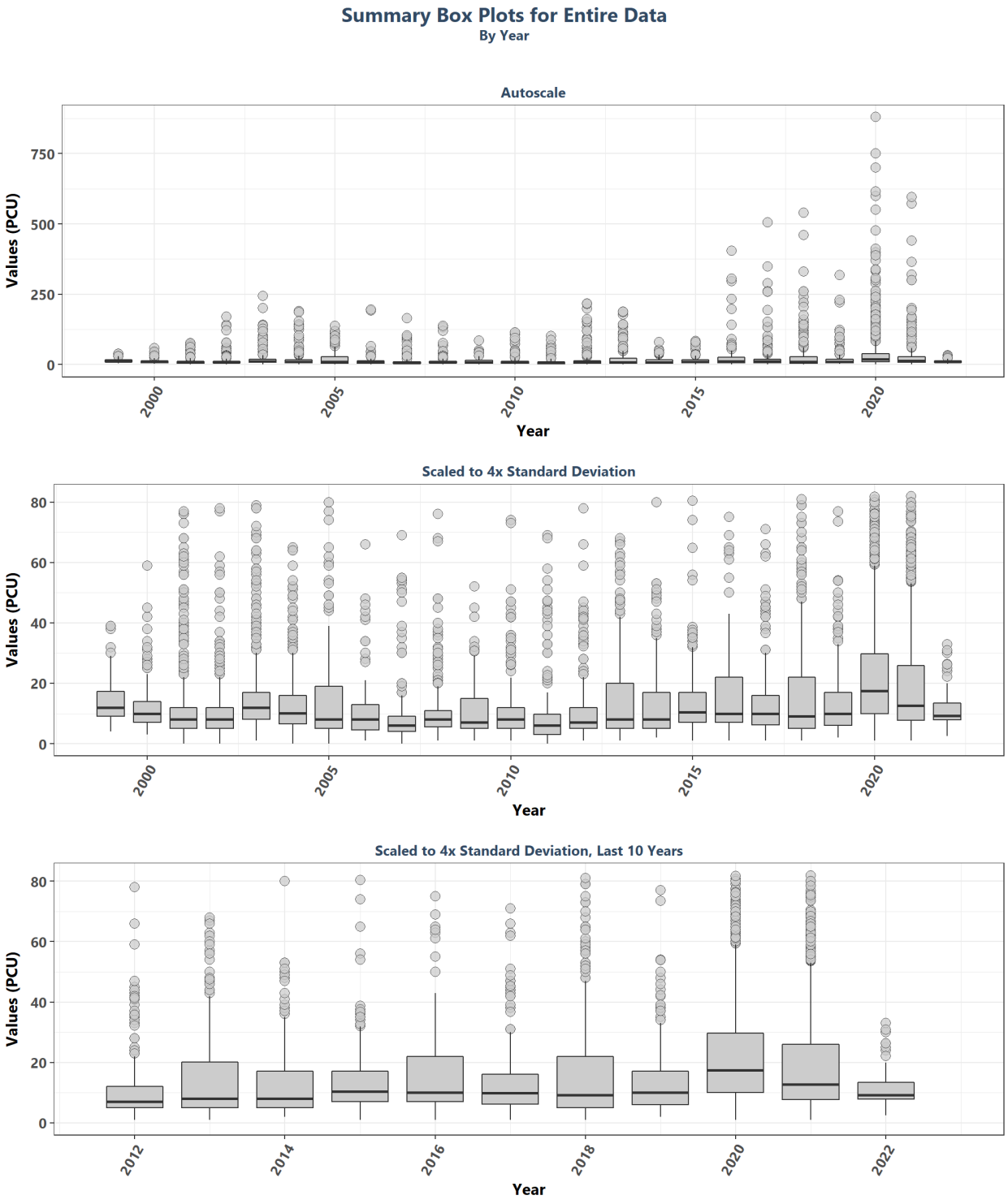
```
                      subtitle="By Year") + plot_theme +
   theme(panel.border=element_blank(), panel.grid.major=element_blank(),
         panel.grid.minor=element_blank(), axis.line=element_blank())

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

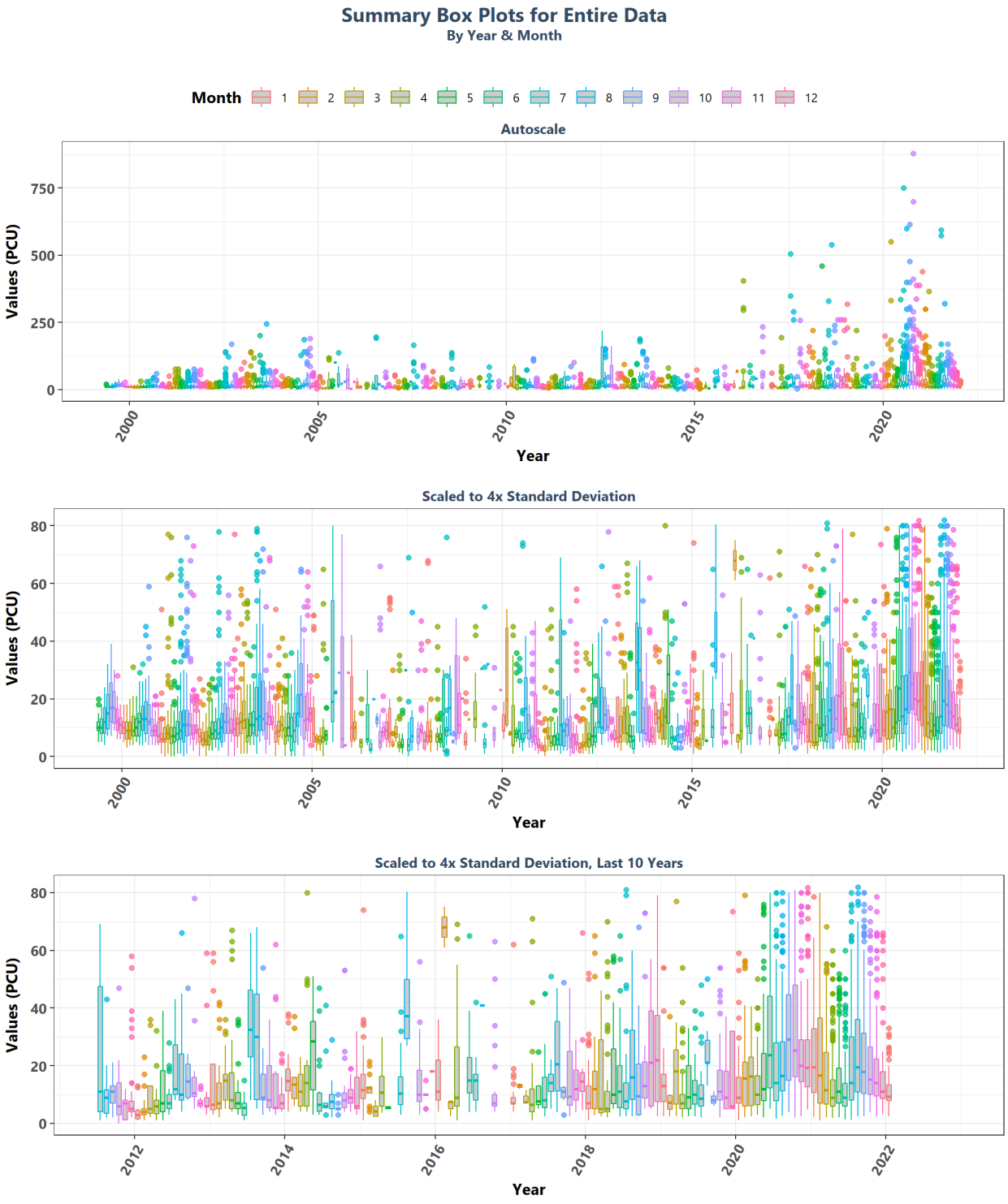
This set of box plots are grouped by year and month with the color being related to the month.

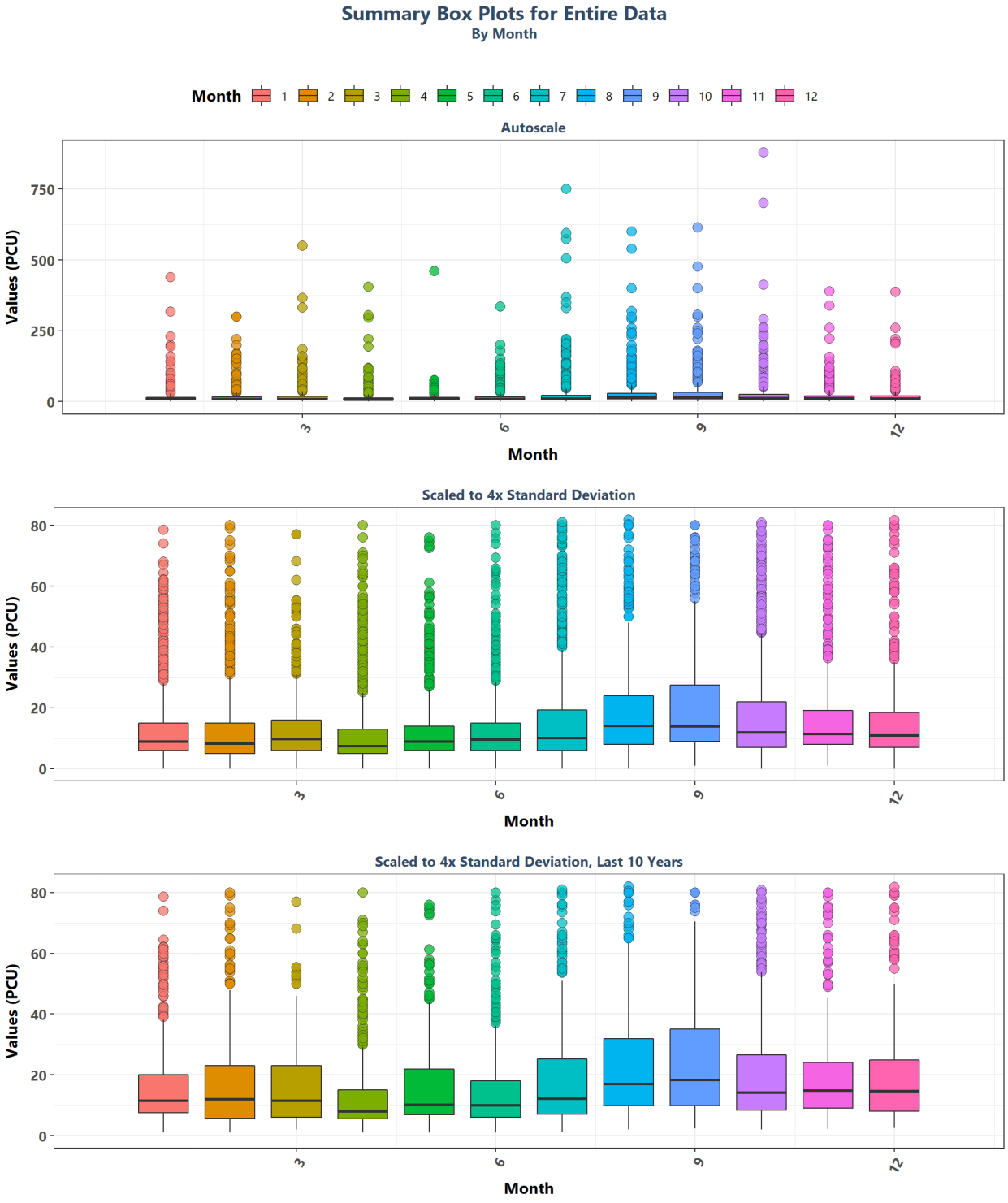
```
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Year",
        y=paste0("Values (", unit,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

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

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

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

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

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

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

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

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

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

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


Summary Box Plots for Entire Data
By Year
Autoscale
Values (PCU)
750
500
250
0
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation
Values (PCU)
80
60
40
20
0
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
80
60
40
20
0
2012
2014
2016
2018
2020
2022
Year

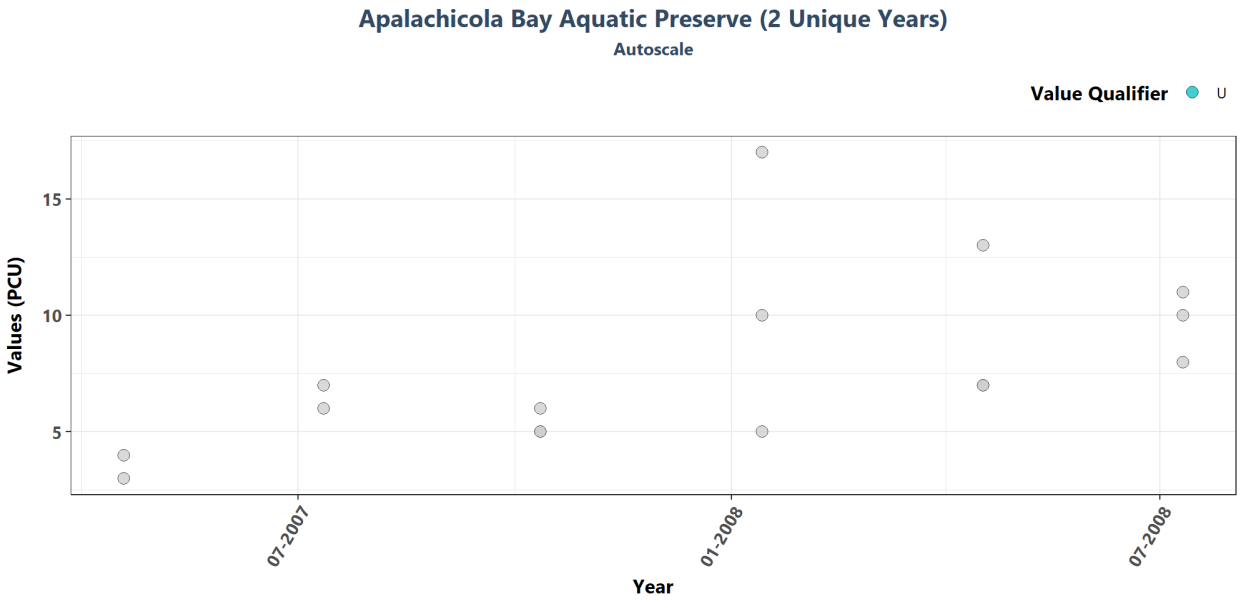

Summary Box Plots for Entire Data
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
750
500
250
0
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation
Values (PCU)
80
60
40
20
0
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
80
60
40
20
0
2012
2014
2016
2018
2020
2022
Year

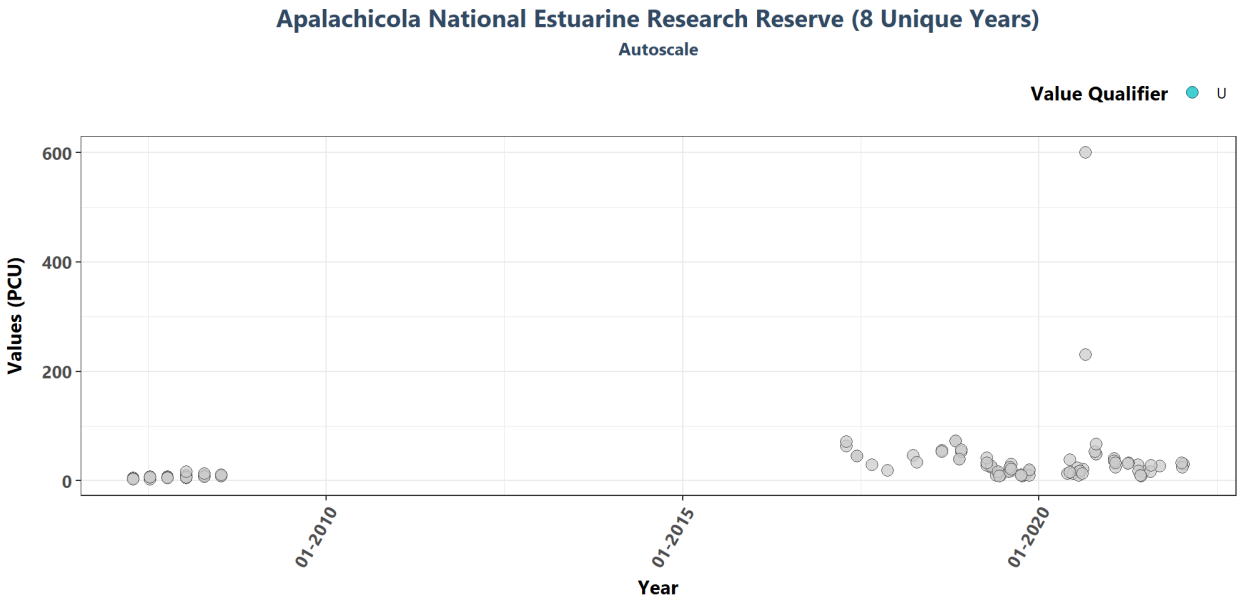


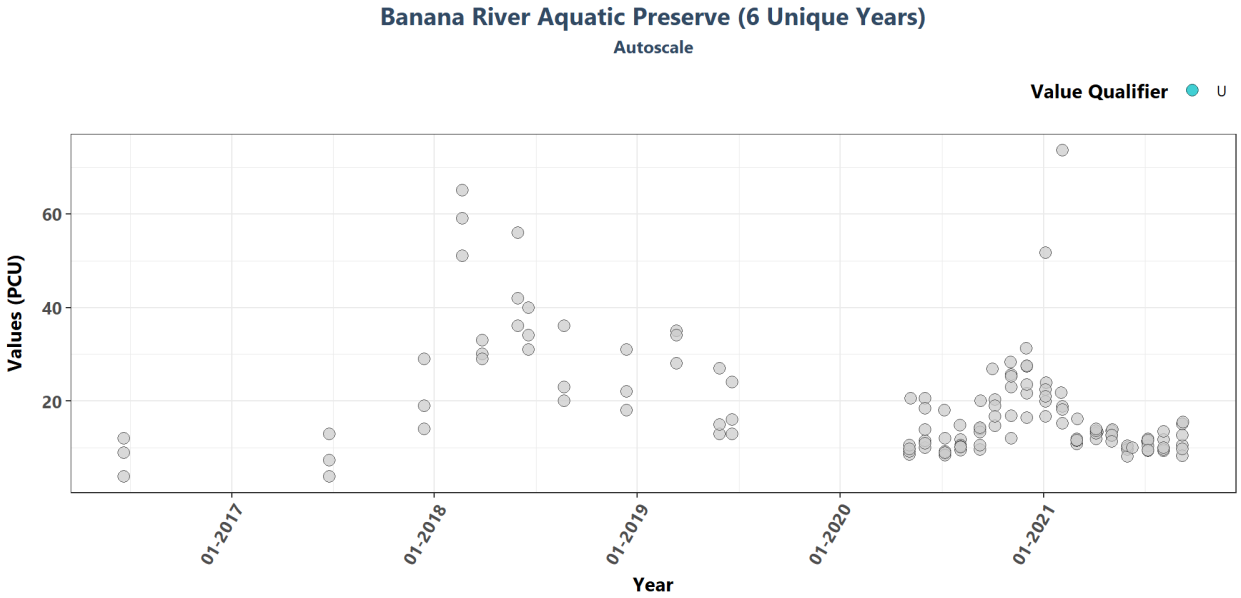
Appendix III: Excluded Managed Areas

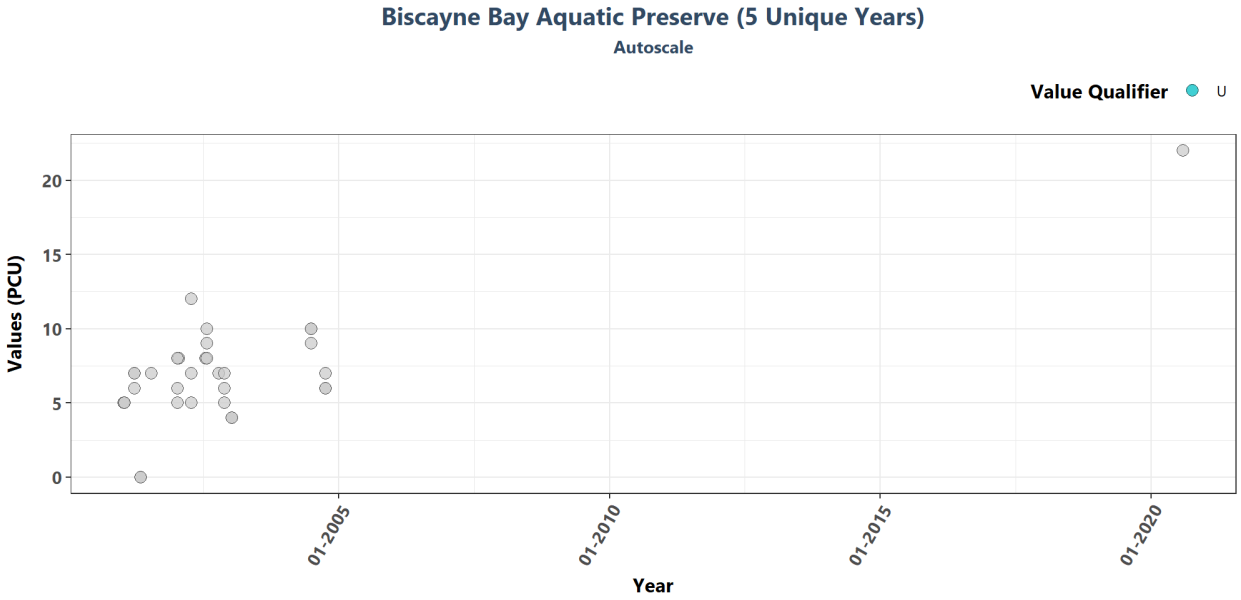
Scatter plots of data values are created for managed areas that have fewer than 10 separate years of data entries. Data points are colored based on specific value qualifiers of interest.

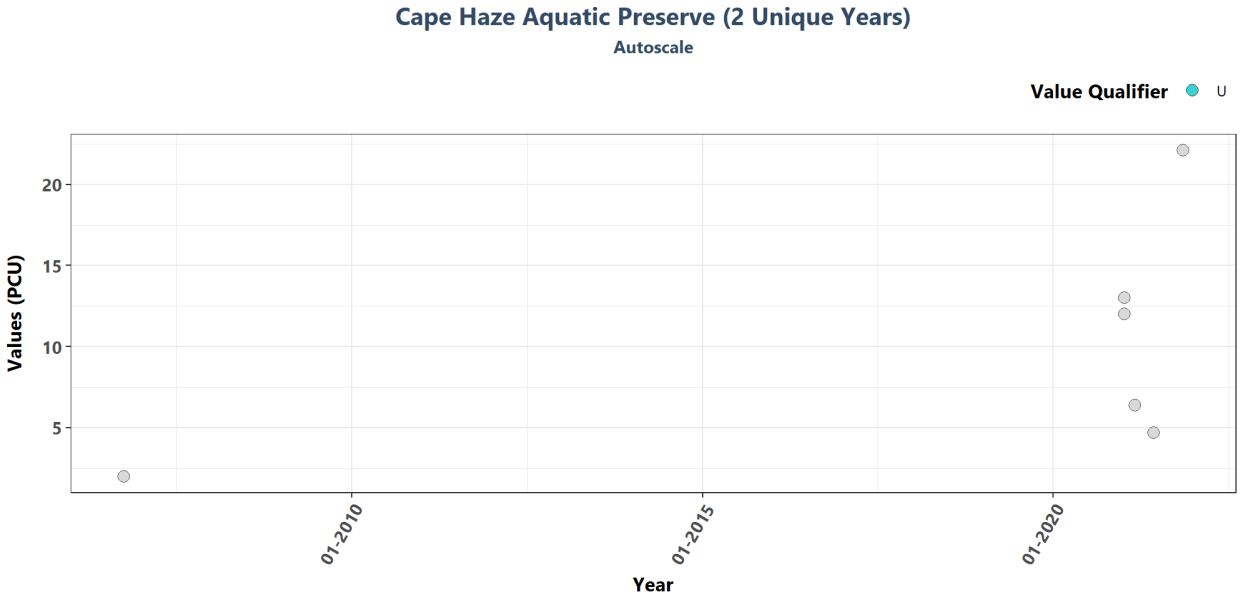
```
if(z==0){
   print("There are no managed areas that qualify.")
} else {
   for(i in 1:z){
      p1<-ggplot(data=data[data$ManagedAreaName==MA_Exclude$ManagedAreaName[i]&
                           data$Include==TRUE, ],
                 aes(x=SampleDate, y=ResultValue, fill=VQ_Plot)) +
         geom_point(shape=21,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

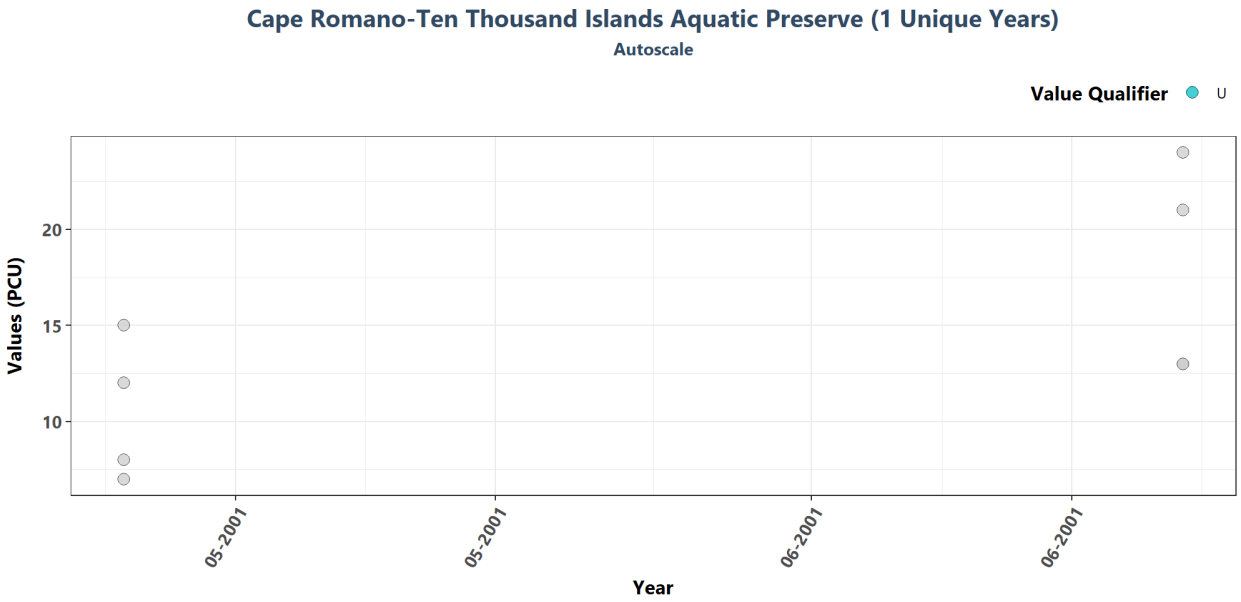


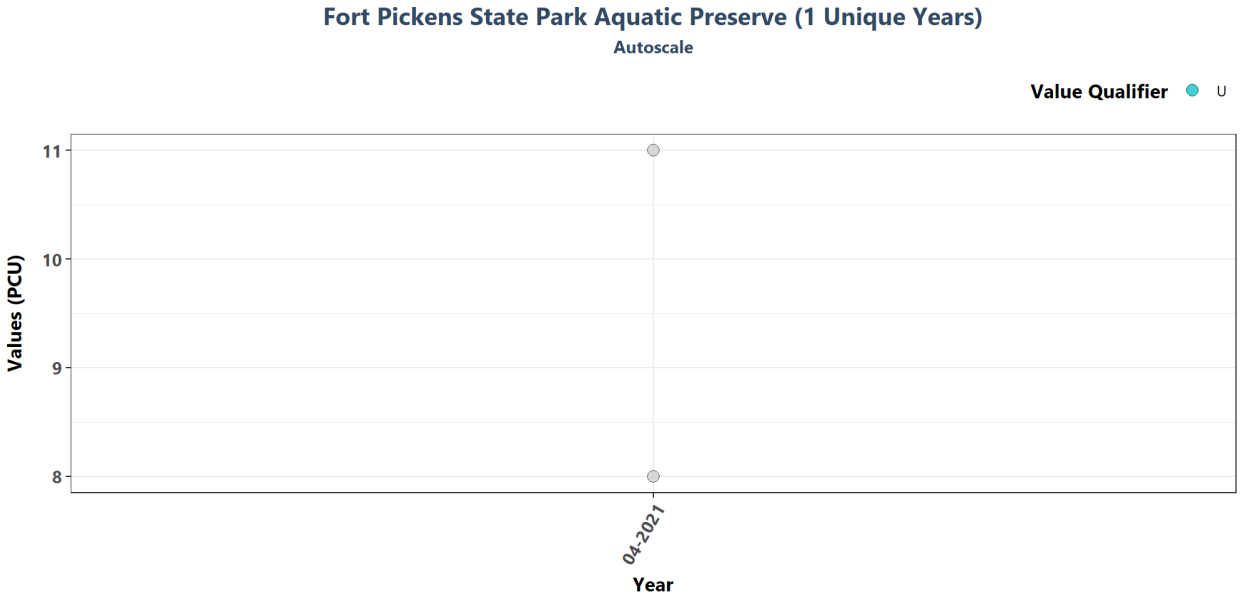

Apalachicola National Estuarine Research Reserve (8 Unique Years)
Autoscale
Value Qualifier
U
600
400
200
0
Values (PCU)
01-2010
01-2015
01-2020
Year

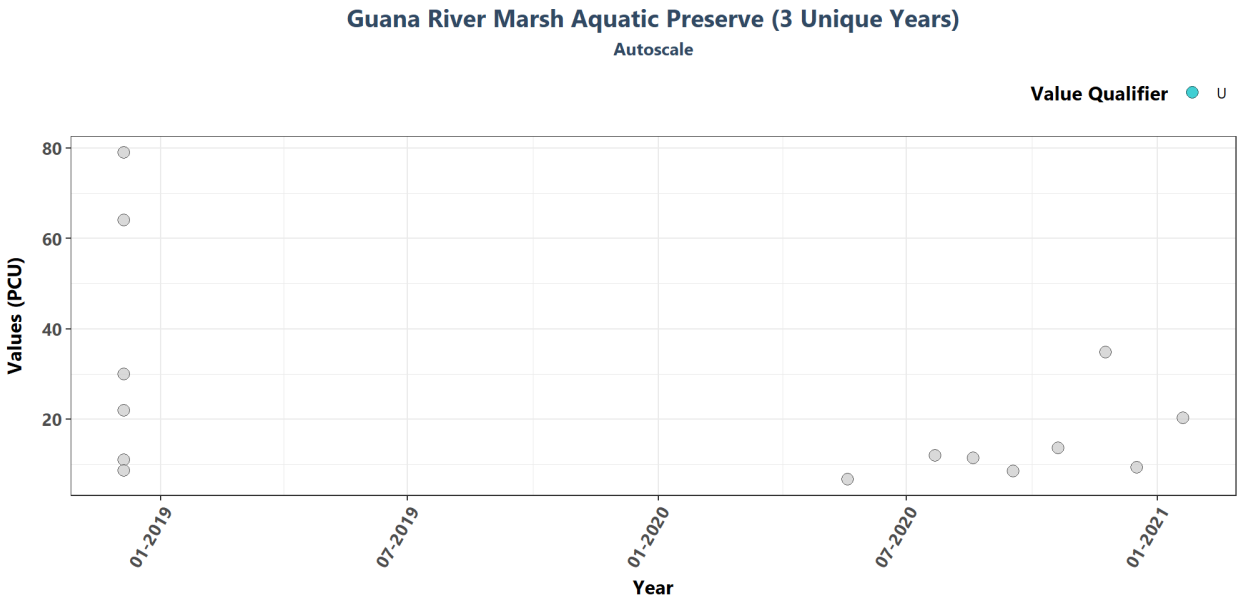

Banana River Aquatic Preserve (6 Unique Years)
Autoscale
Value Qualifier
U
60
40
20
Values (PCU)
01-2017
01-2018
01-2019
01-2020
01-2021
Year

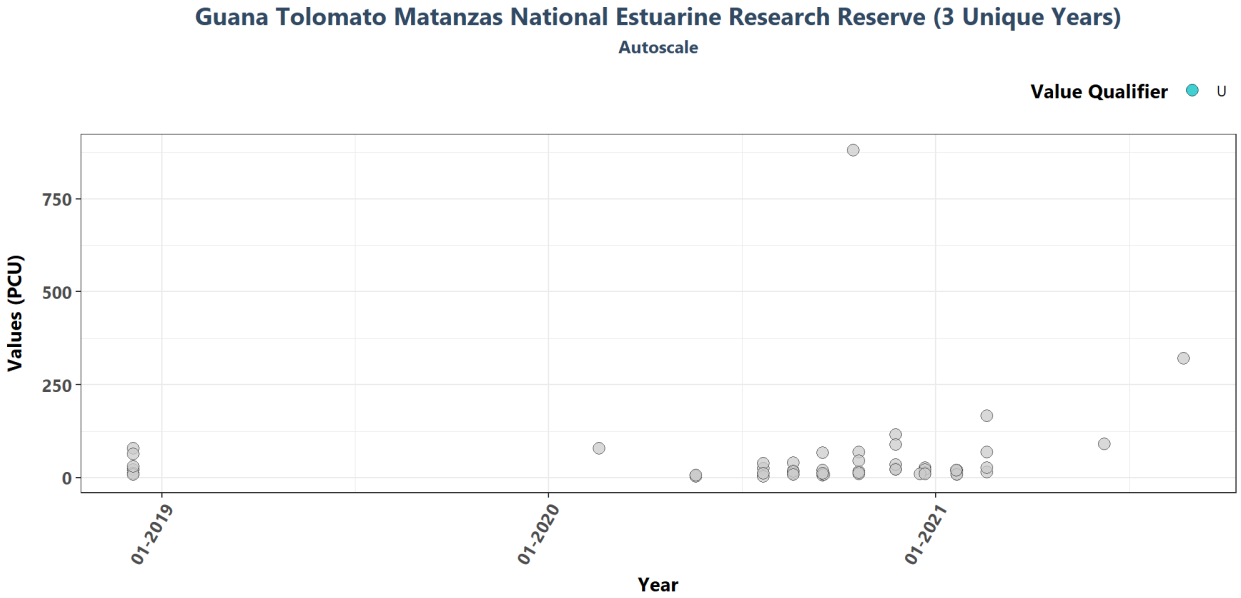

Biscayne Bay Aquatic Preserve (5 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
20
15
10
5
0
01-2005
01-2010
01-2015
01-2020
Year

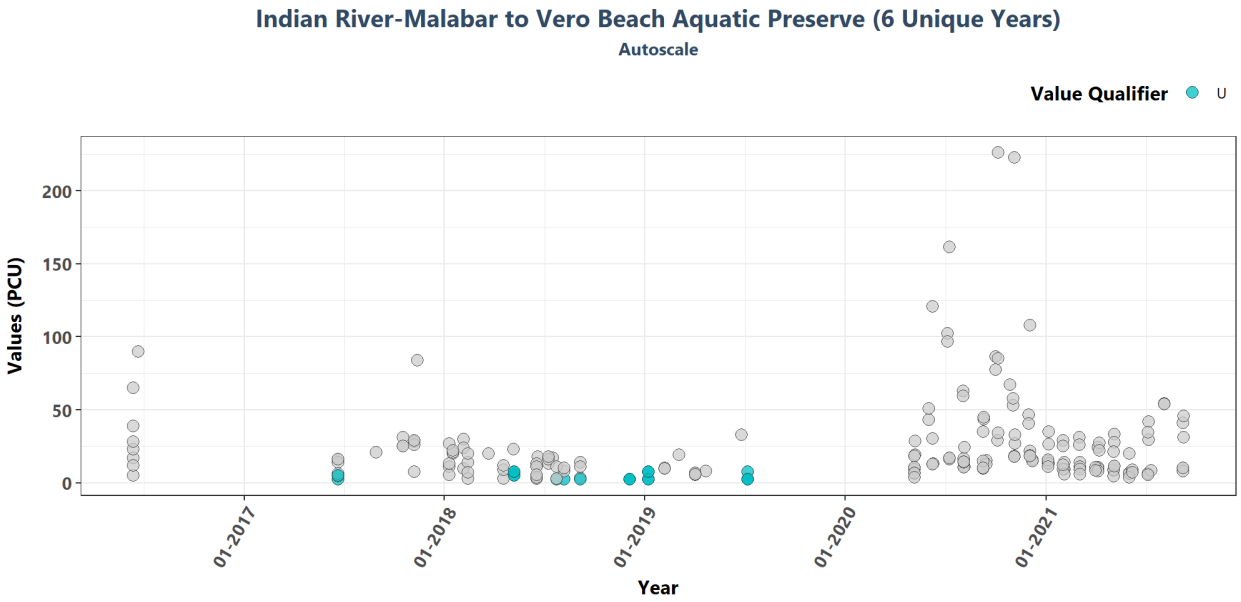

Cape Haze Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
20
15
10
5
01-2010
01-2015
01-2020
Year

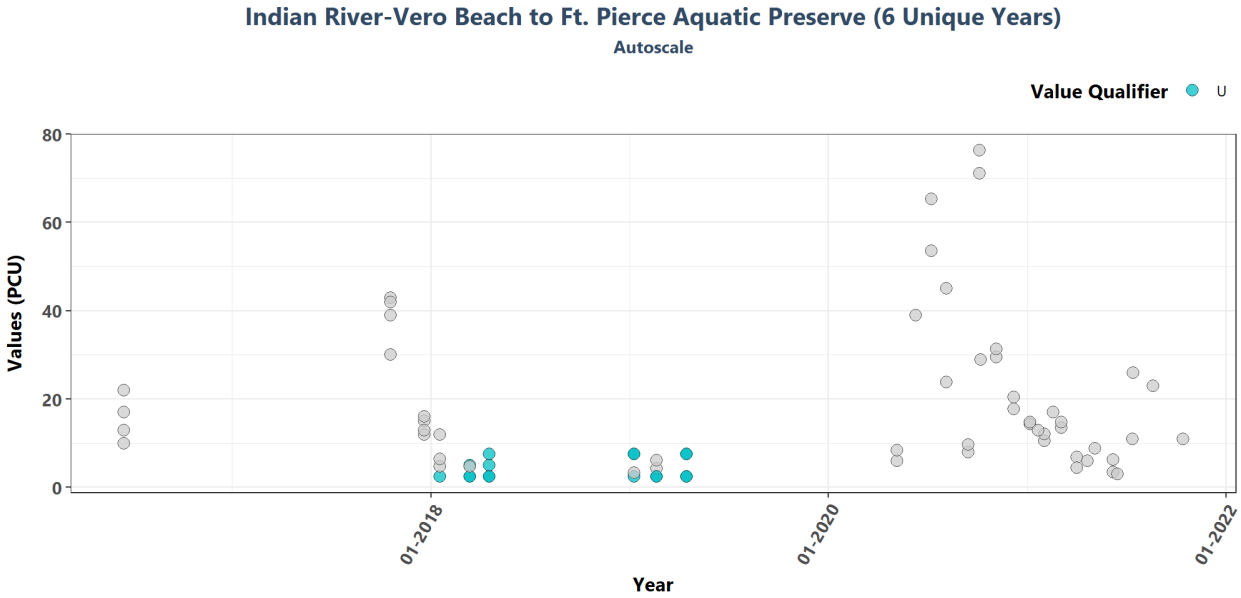

Cape Romano-Ten Thousand Islands Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
20
15
10
05-2001
05-2001
06-2001
06-2001
Year

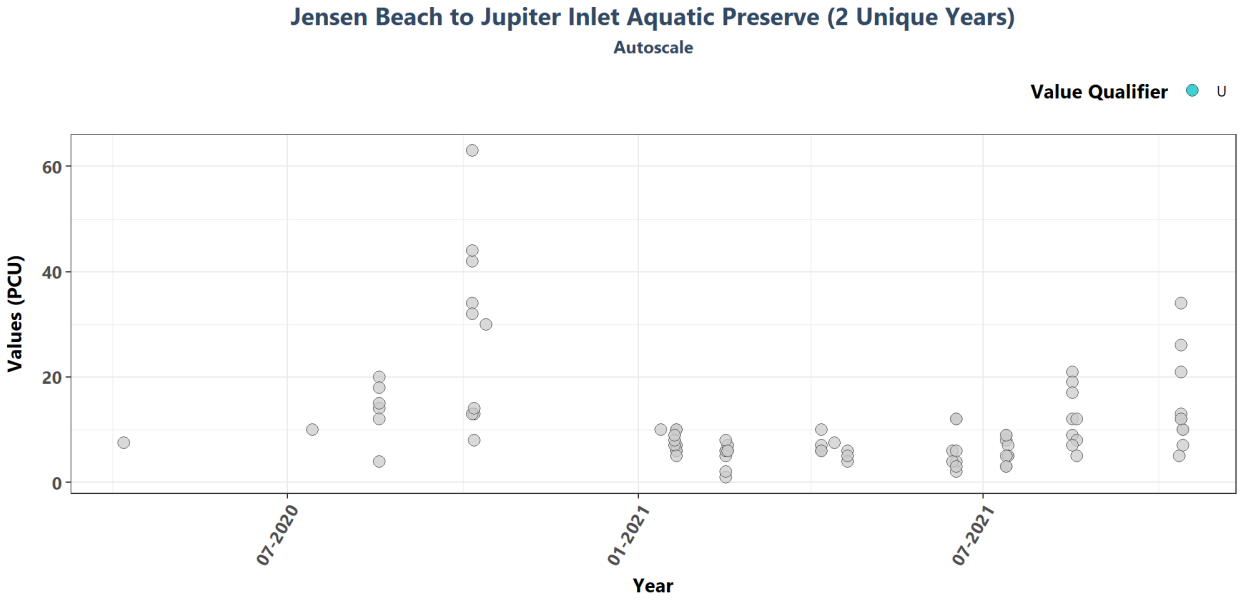

Fort Pickens State Park Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
11
10
9
8
04-2021
Year

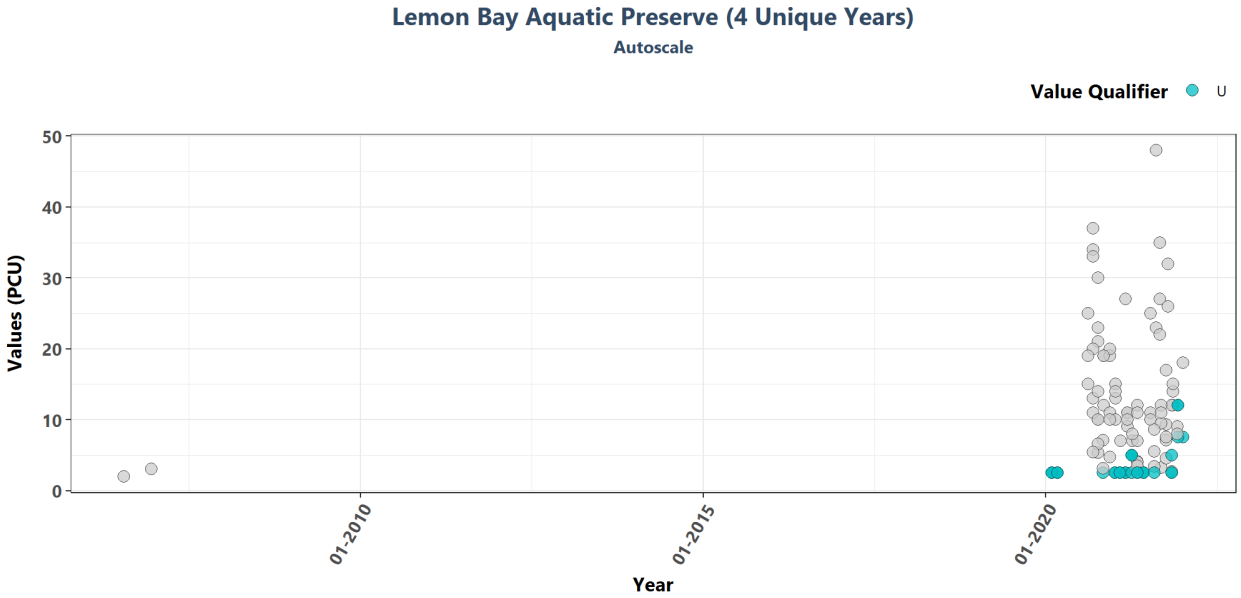

Guana River Marsh Aquatic Preserve (3 Unique Years)
Autoscale
Value Qualifier
U
80
60
40
20
Values (PCU)
01-2019
07-2019
01-2020
07-2020
01-2021
Year

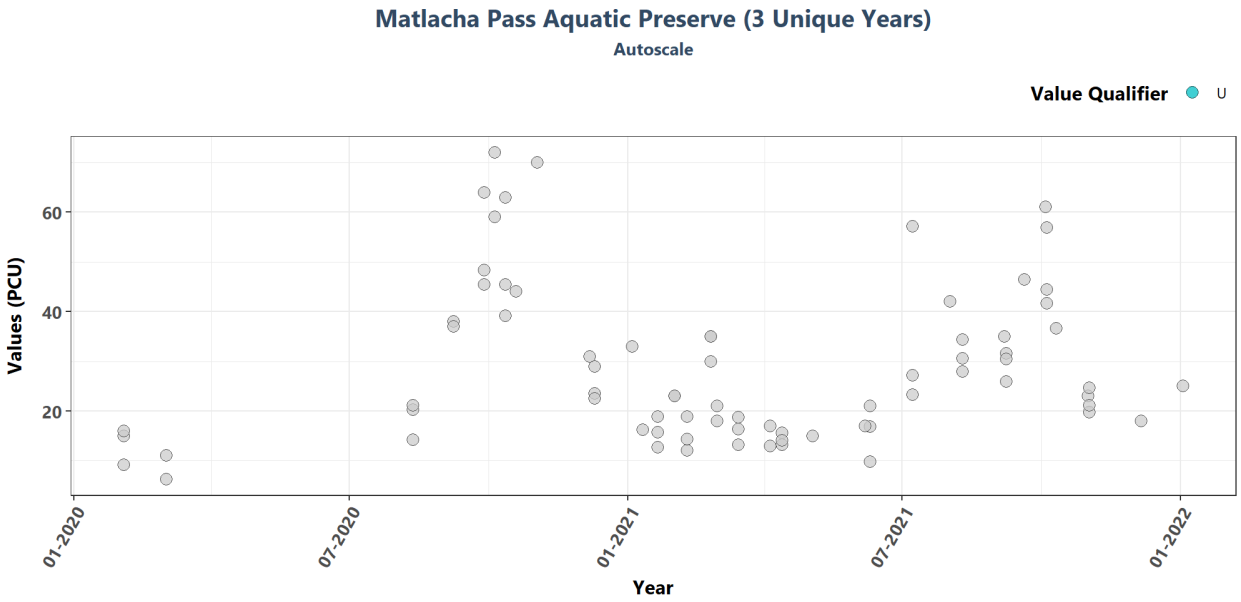

Guana Tolomato Matanzas National Estuarine Research Reserve (3 Unique Years)
Autoscale
Value Qualifier
U
750
500
250
0
Values (PCU)
01-2019
01-2020
01-2021
Year

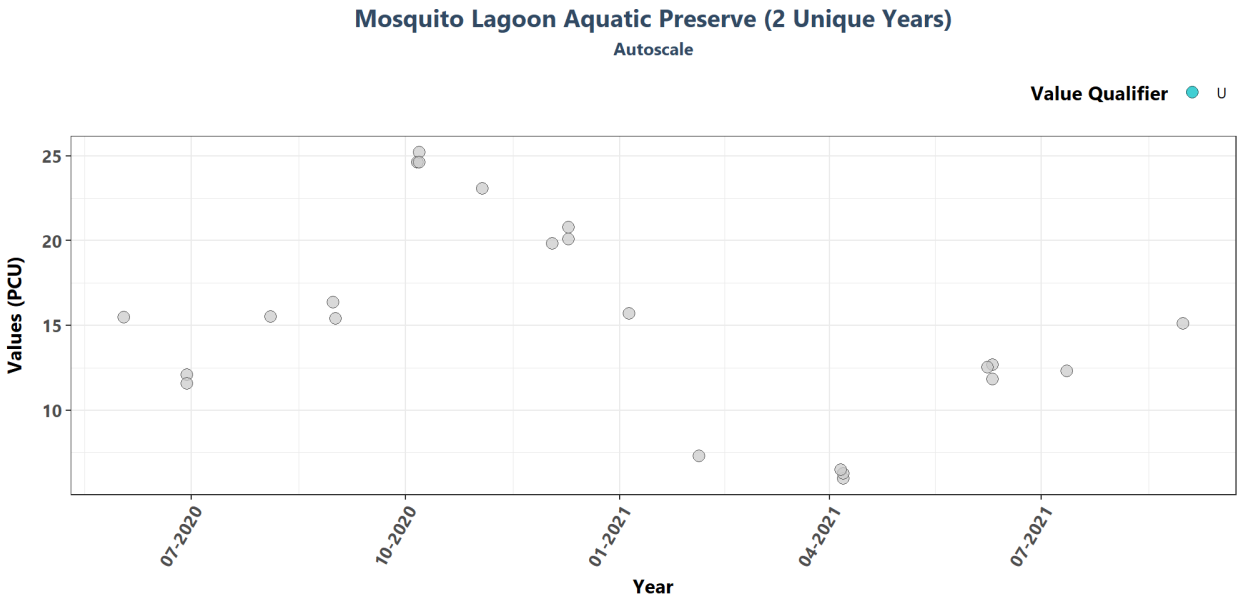

Indian River-Malabar to Vero Beach Aquatic Preserve (6 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
200
150
100
50
0
01-2017
01-2018
01-2019
01-2020
01-2021
Year

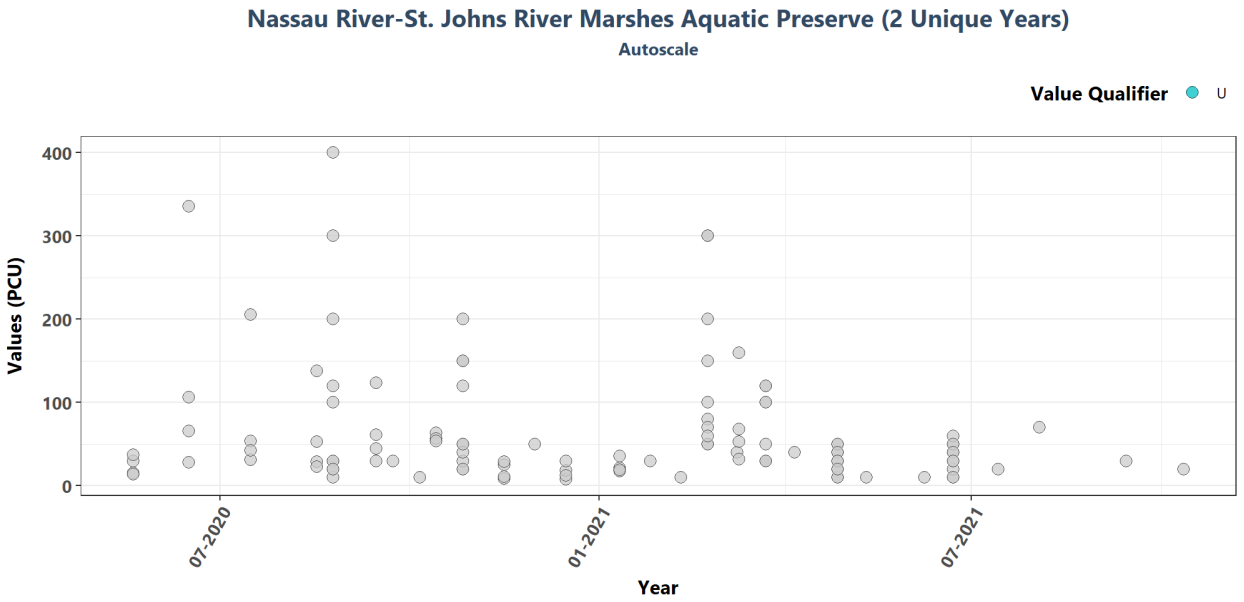

Indian River-Vero Beach to Ft. Pierce Aquatic Preserve (6 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
80
60
40
20
0
01-2018
01-2020
01-2022
Year

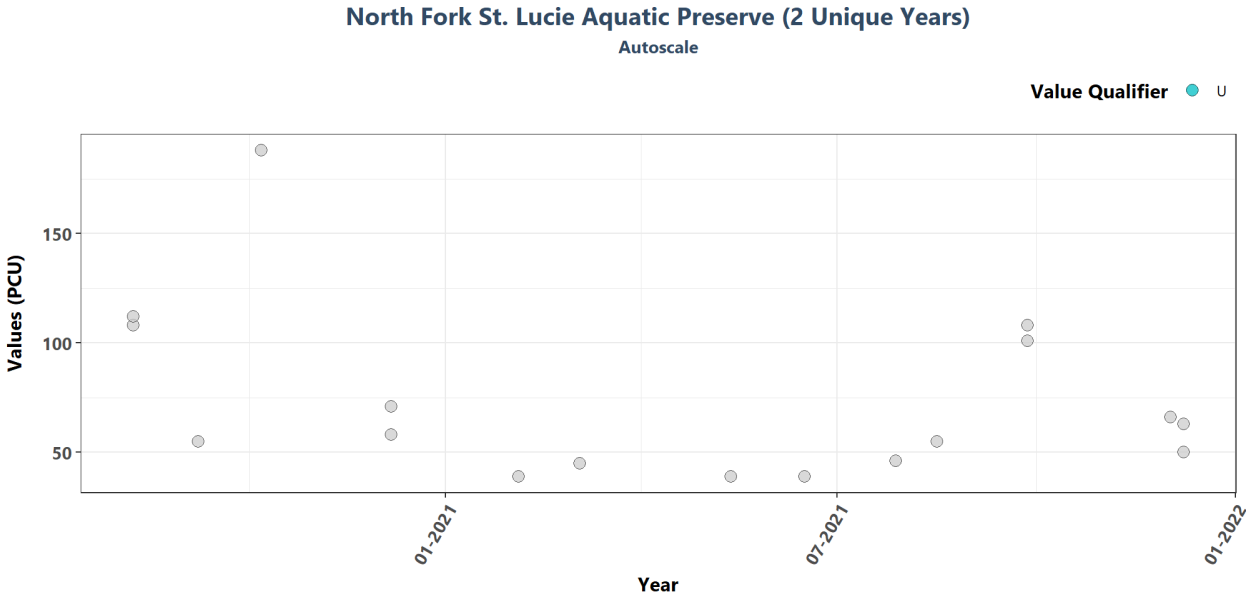

Jensen Beach to Jupiter Inlet Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
60
40
20
0
07-2020
01-2021
07-2021
Year

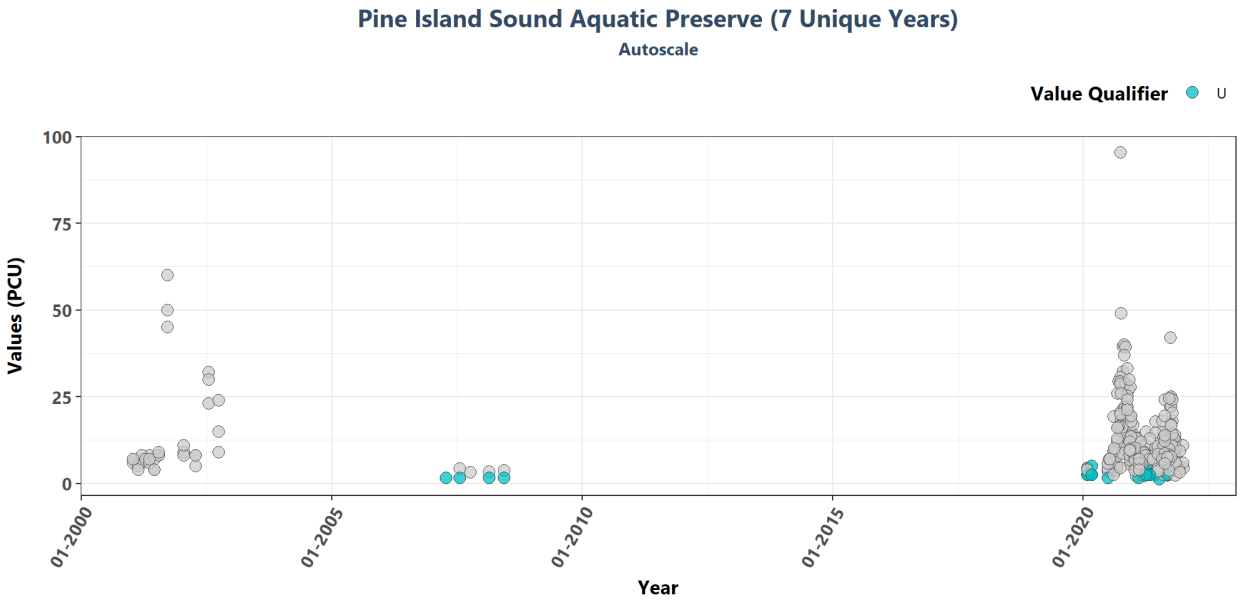

Lemon Bay Aquatic Preserve (4 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
50
40
30
20
10
0
01-2010
01-2015
01-2020
Year

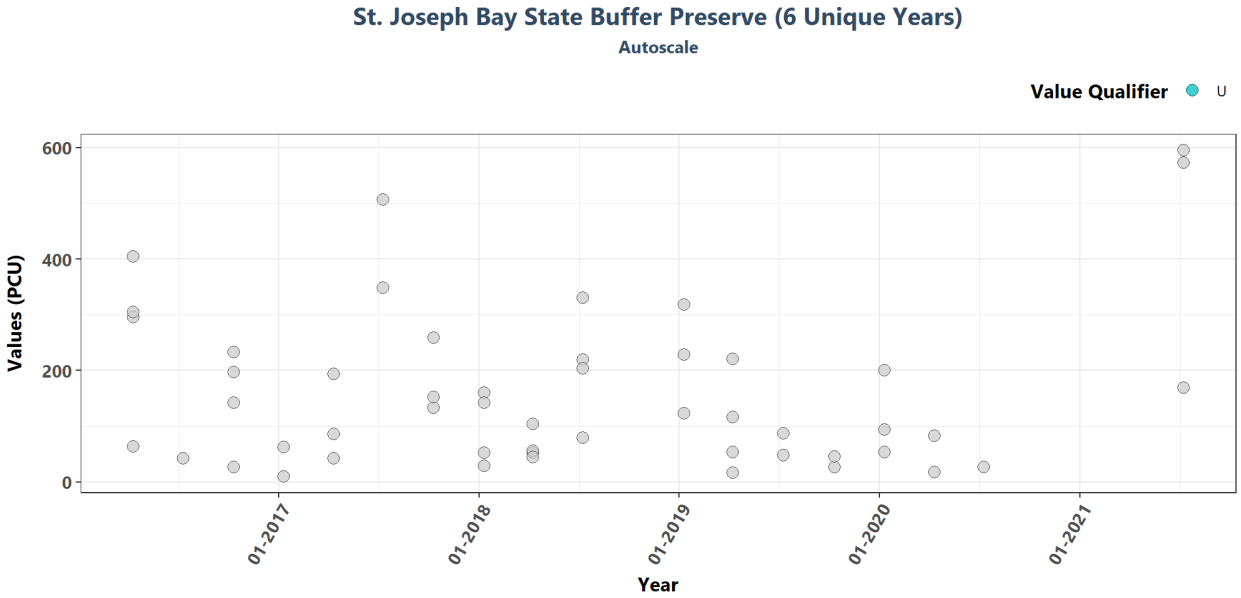

Matlacha Pass Aquatic Preserve (3 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
60
40
20
01-2020
07-2020
01-2021
07-2021
01-2022
Year

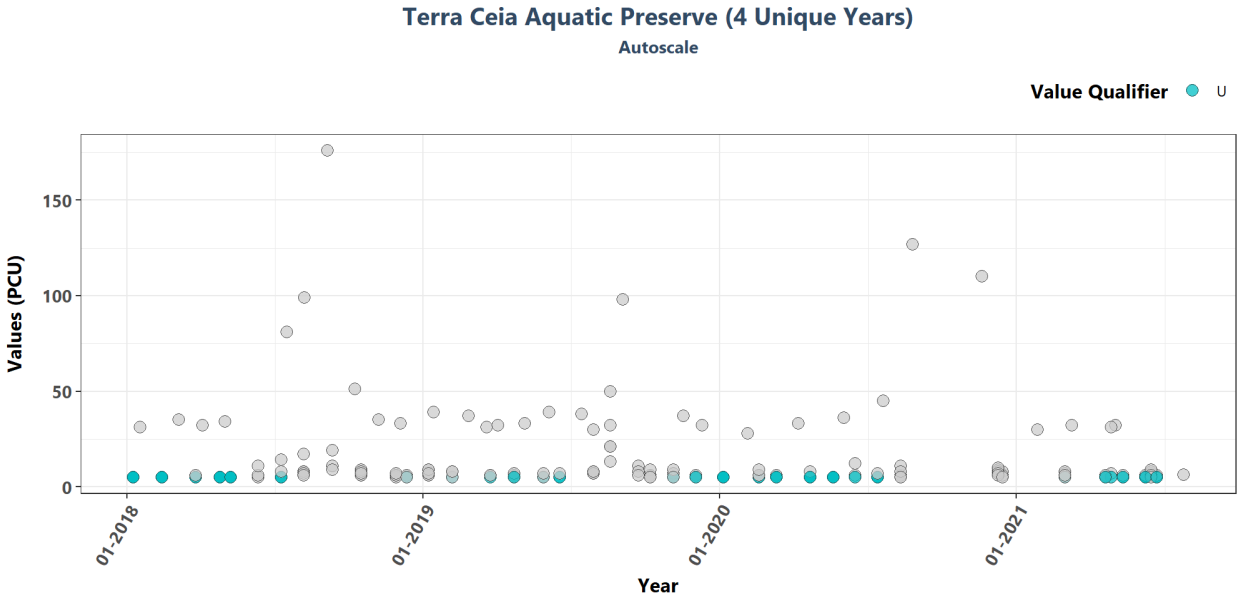

Mosquito Lagoon Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
25
20
15
10
07-2020
10-2020
01-2021
04-2021
07-2021
Year

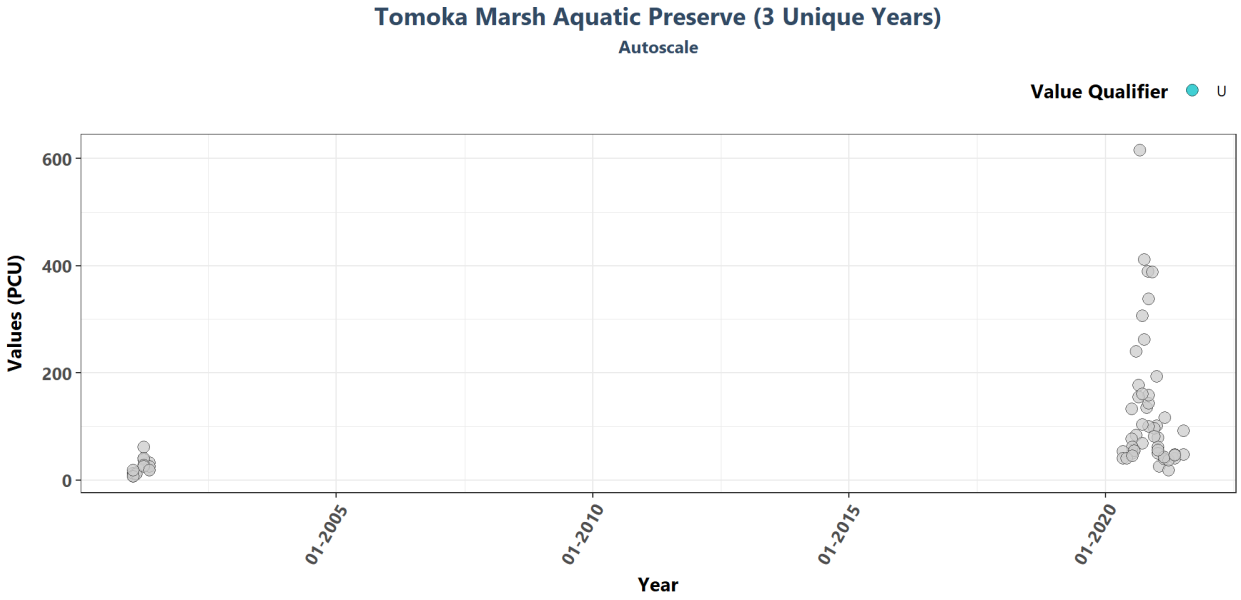

Nassau River-St. Johns River Marshes Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
400
300
200
100
0
Values (PCU)
07-2020
01-2021
07-2021
Year

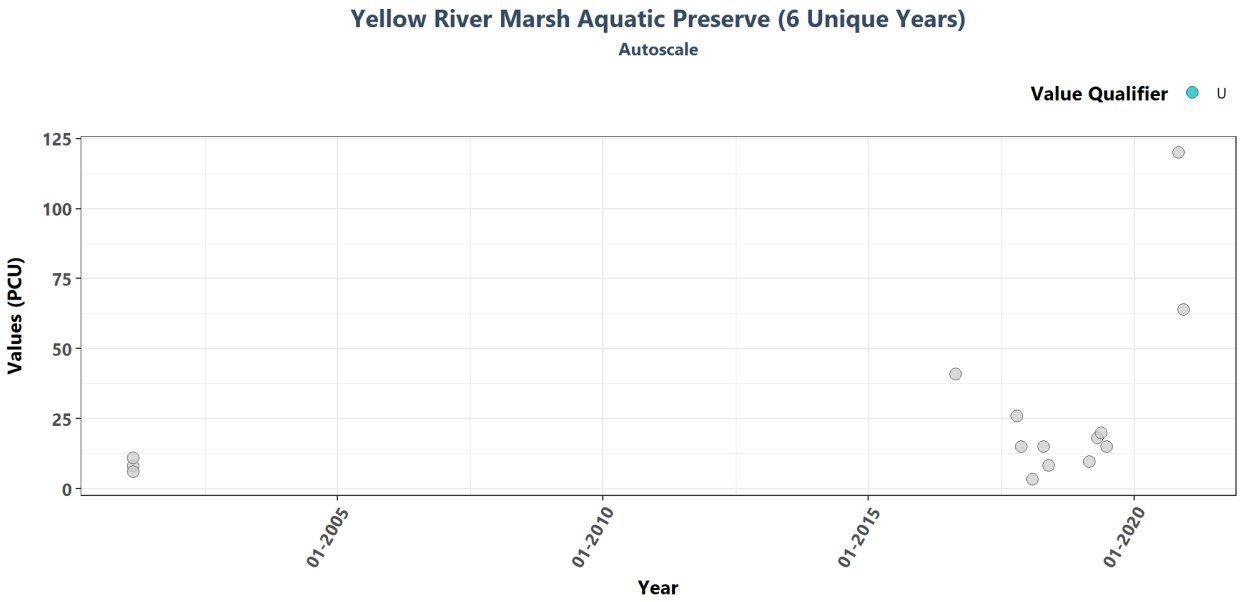

North Fork St. Lucie Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
U
150
100
50
Values (PCU)
01-2021
07-2021
01-2022
Year


Pine Island Sound Aquatic Preserve (7 Unique Years)
Autoscale
Value Qualifier
U
100
75
50
25
0
Values (PCU)
01-2000
01-2005
01-2010
01-2015
01-2020
Year


St. Joseph Bay State Buffer Preserve (6 Unique Years)
Autoscale
Value Qualifier
U
600
400
200
0
Values (PCU)
01-2017
01-2018
01-2019
01-2020
01-2021
Year


Terra Ceia Aquatic Preserve (4 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
150
100
50
0
01-2018
01-2019
01-2020
01-2021
Year


Tomoka Marsh Aquatic Preserve (3 Unique Years)
Autoscale
Value Qualifier
U
Values (PCU)
600
400
200
0
01-2005
01-2010
01-2015
01-2020
Year



Appendix IV: Managed Area Trendlines

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

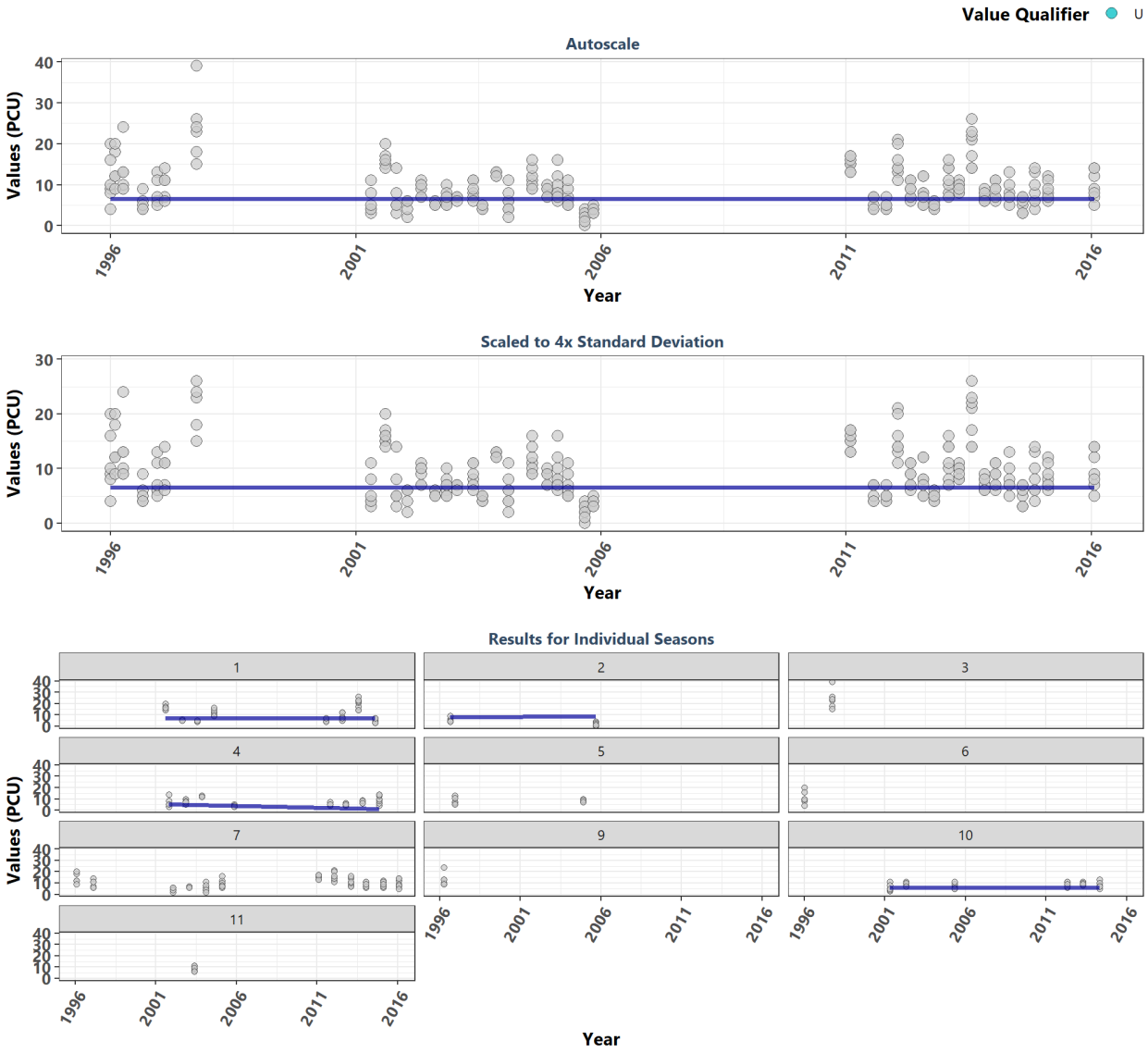
1. Use the data set that only has `SufficientData` of `TRUE` for the desired managed area
2. Determine the earliest and latest year of the data to create x-axis scale and intervals
3. Determine the minimum, mean, and standard deviation for the data to be used for y-axis scales
    - Excludes the top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

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

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

tau <- KT.Stats$tau[KT.Stats$ManagedAreaName==MA_Include[i]]
s_slope <- KT.Stats$SennSlope[KT.Stats$ManagedAreaName==MA_Include[i]]
s_int <- KT.Stats$SennIntercept[KT.Stats$ManagedAreaName==MA_Include[i]]
trend <- KT.Stats$Trend[KT.Stats$ManagedAreaName==MA_Include[i]]
z <- KT.Stats$z[KT.Stats$ManagedAreaName==MA_Include[i]]
p_z <- KT.Stats$p_z[KT.Stats$ManagedAreaName==MA_Include[i]]
chi_sq <- KT.Stats$chi_sq[KT.Stats$ManagedAreaName==MA_Include[i]]
p_chi_sq <- KT.Stats$p_chi_sq[KT.Stats$ManagedAreaName==MA_Include[i]]

# model <- lm(ResultValue ~ relyear_dd,
#             data=plot_data)
# m_int <- coef(model)[[1]]
# m_slope <- coef(model)[[2]]
# rm(model)

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

KT.Stats[, season := Season]
KT.Stats[ManagedAreaName==MA_Include[i] & season != "All", `:=` (N_Data = nrow(plot_data[Season ==
KT.Stats[ManagedAreaName==MA_Include[i] & season == "All", `:=` (relyear_dd_lower = min(plot_data
KT.Stats[, season := NULL]

# plot_data[is.na(VQ_Plot), VQ_Plot := "None"]
p1 <- ggplot(data=plot_data,
             aes(x=relyear_dd, y=ResultValue, fill = VQ_Plot)) +
   geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
   # geom_abline(aes(slope=s_slope, intercept=s_int),
   #             color="#000099", size=1.2, alpha=0.7) +
   geom_segment(data = KT.Stats[ManagedAreaName==MA_Include[i] & Season == "All", ], aes(x = rely
                                                                                         y = relyear_
                                                                                         xend = relyea
                                                                                         yend = relyea

                color="#000099", size=1.2, alpha=0.7, inherit.aes = FALSE) +
   labs(subtitle="Autoscale",
        x="Year", y=paste0("Values (", unit, ")"),
        fill="Value Qualifier") +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal",
         legend.justification="right") +
   {if(inc_H==TRUE){
      scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                                 "HU"="#7CAE00"), na.value="#cccccc")
```

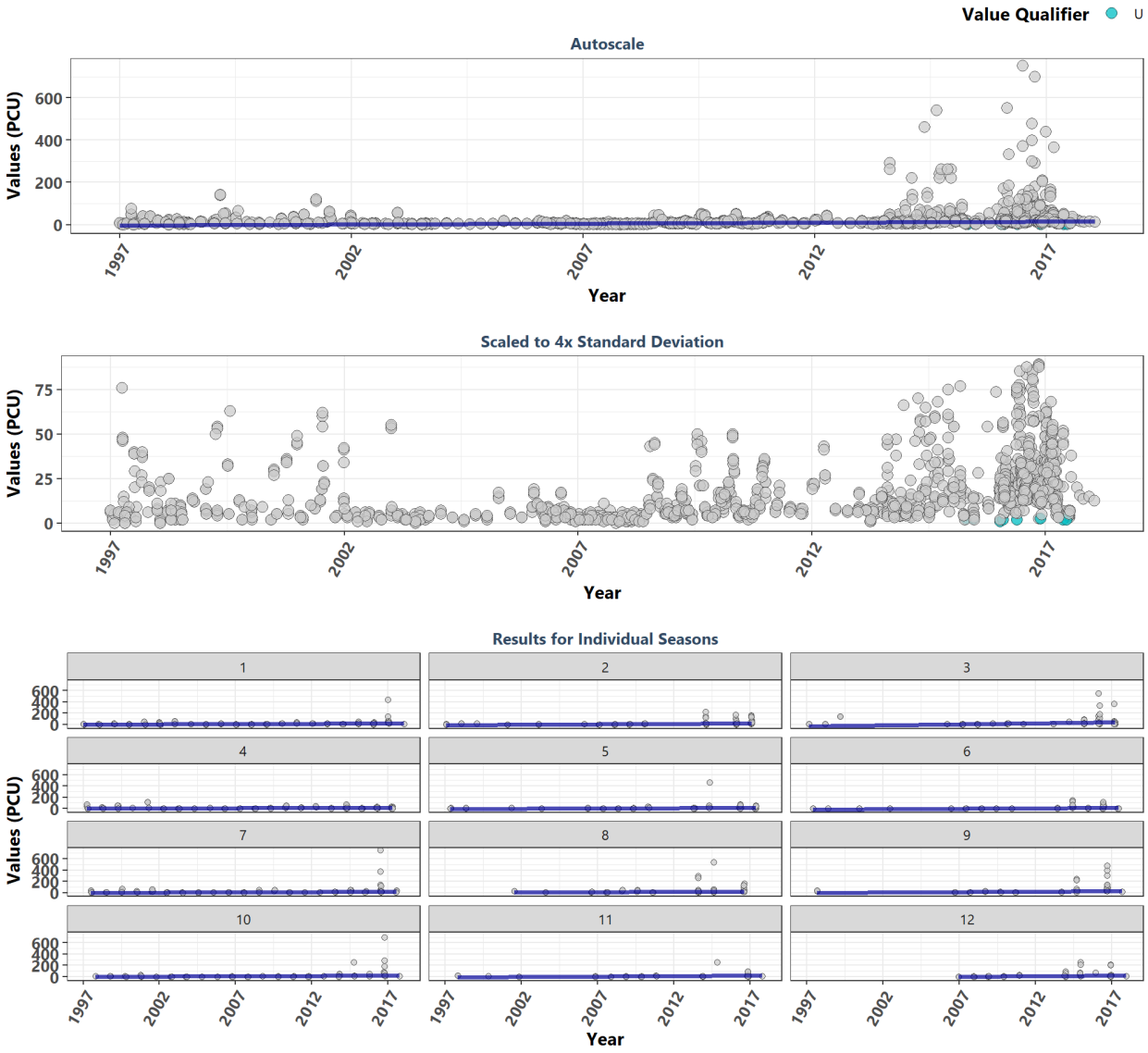
```
    } else if(param_name=="Secchi_Depth"){
       scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                                  "SU"="#7CAE00"), na.value="#cccccc")
    } else {
       scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
    }} +
    scale_x_continuous(breaks = xbrks,
                       labels = xlabs)

p2 <- ggplot(data=plot_data,
             aes(x=relyear_dd, y=ResultValue, fill=VQ_Plot)) +
    geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
    # geom_abline(aes(slope=s_slope, intercept=s_int),
    #             color="#000099", size=1.2, alpha=0.7) +
    geom_segment(data = KT.Stats[ManagedAreaName==MA_Include[i] & Season == "All", ], aes(x = relye
                                                                                          y = relyear_
                                                                                          xend = relyea
                                                                                          yend = relyea
                 color="#000099", size=1.2, alpha=0.7, inherit.aes = FALSE) +
    ylim(min_RV, y_scale) +
    labs(subtitle="Scaled to 4x Standard Deviation",
         x="Year", y=paste0("Values (", unit, ")")) +
    plot_theme +
    theme(legend.position="none") +
    {if(inc_H==TRUE){
       scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                                  "HU"="#7CAE00"), na.value="#cccccc")
    } else if(param_name=="Secchi_Depth"){
       scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                                  "SU"="#7CAE00"), na.value="#cccccc")
    } else {
       scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
    }} +
    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[ManagedAreaName==MA_Include[i] & Season != "All", ], aes(x = relye
                                                                                          y = relyear_
                                                                                          xend = relyea
                                                                                          yend = relyea
                 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 Seasons
    facet_wrap(~Season, ncol = 3) +
    plot_theme

leg <- get_legend(p1)
KTset <- ggarrange(leg, p1 + theme(legend.position="none"), p2,
                   splot, ncol=1, heights=c(0.1, 1, 1, 1.5))
```

```
      p0 <- ggplot() + labs(title=paste0(MA_Include[i])) +
         plot_theme + theme(panel.border=element_blank(),
                            panel.grid.major=element_blank(),
                            panel.grid.minor=element_blank(),
                            axis.line=element_blank())

      KT.Stats[ManagedAreaName==MA_Include[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$ManagedAreaName==MA_Include[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	284	8	0.0505	0.0000	6.5000	1.5	0.1398	14.5	0.0129	0
1	56	7	-0.0344	0.0000	7.0000	-0.4	0.7084	NA	NA	0
2	13	NA	0.0861	0.0714	7.9286	NA	NA	NA	NA	NA
3	6	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	56	6	-0.5000	-0.3333	7.3333	0.8	0.4312	NA	NA	0
5	13	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	6	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	82	NA	NA	NA	NA	NA	NA	NA	NA	NA
9	6	NA	0.2167	0.1818	5.7273	NA	NA	NA	NA	NA
10	40	8	0.0714	0.0000	6.0000	2.0	0.0446	NA	NA	0
11	6	NA	0.1410	0.1250	5.8750	NA	NA	NA	NA	NA

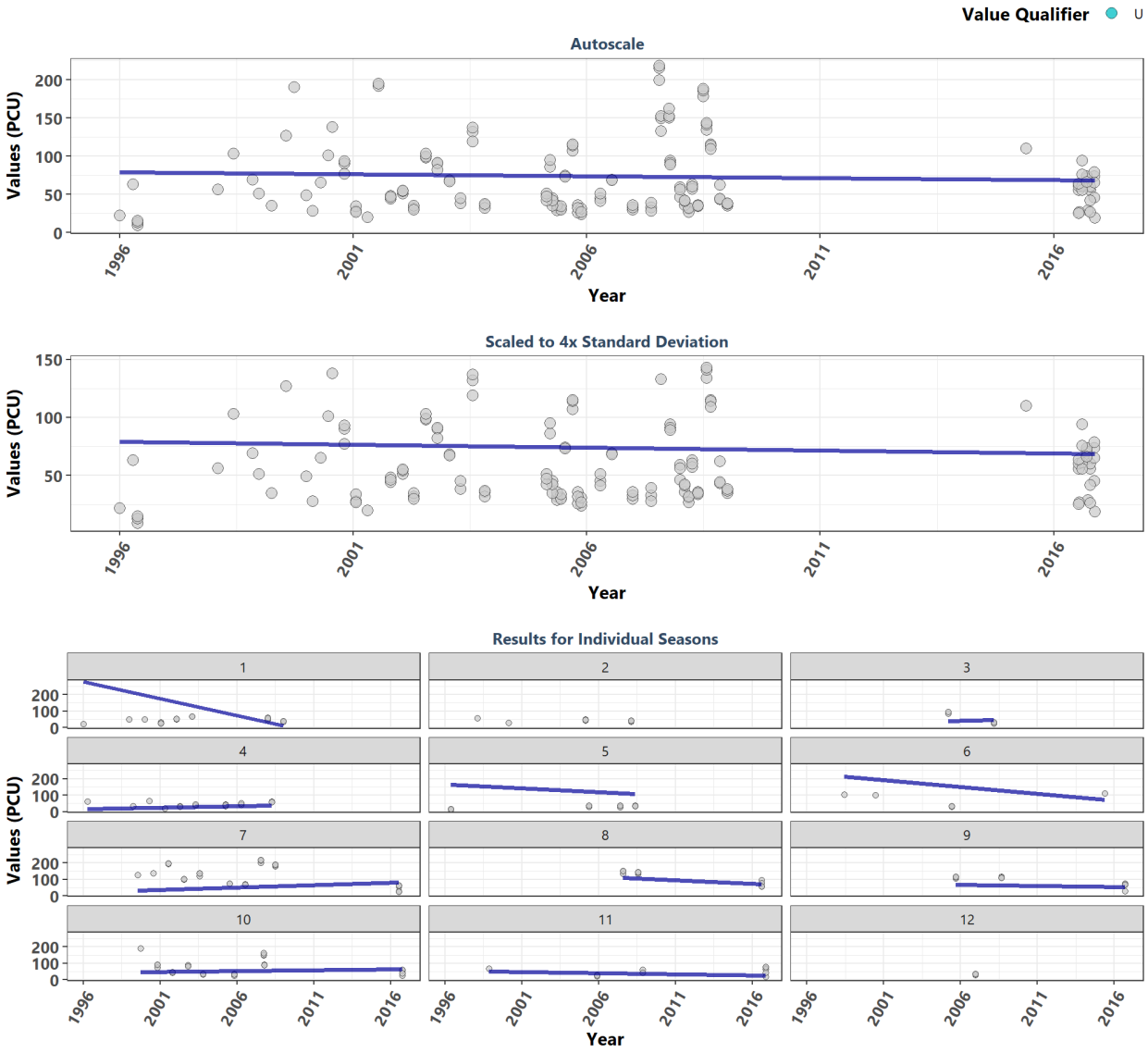
[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	1509	13.90	0.3510	1.0000	-5.0036	18.9	0.0000	51.6	0	1
1	197	14.40	0.4344	1.2500	-3.1000	9.1	0.0000	NA	NA	1
2	138	16.90	0.3442	1.7000	-11.1500	9.6	0.0000	NA	NA	2
3	134	19.75	0.4232	3.4450	-29.9150	5.6	0.0000	NA	NA	2
4	186	8.35	0.1255	0.2000	5.7500	2.6	0.0106	NA	NA	1
5	108	9.85	0.5037	1.1143	-5.6000	6.1	0.0000	NA	NA	2
6	88	8.00	0.4266	1.2857	-15.1429	6.1	0.0000	NA	NA	2
7	181	10.00	0.3879	0.7667	-3.1833	4.3	0.0000	NA	NA	1
8	99	19.30	0.2130	0.4743	3.8336	3.8	0.0001	NA	NA	1
9	70	28.65	0.2515	1.3333	-3.3667	5.4	0.0000	NA	NA	1
10	136	10.00	0.3180	1.2714	-4.4071	8.8	0.0000	NA	NA	2
11	80	12.40	0.3943	1.2071	-7.5179	5.5	0.0000	NA	NA	1
12	92	17.75	0.5357	2.1171	-23.3257	5.1	0.0000	NA	NA	2

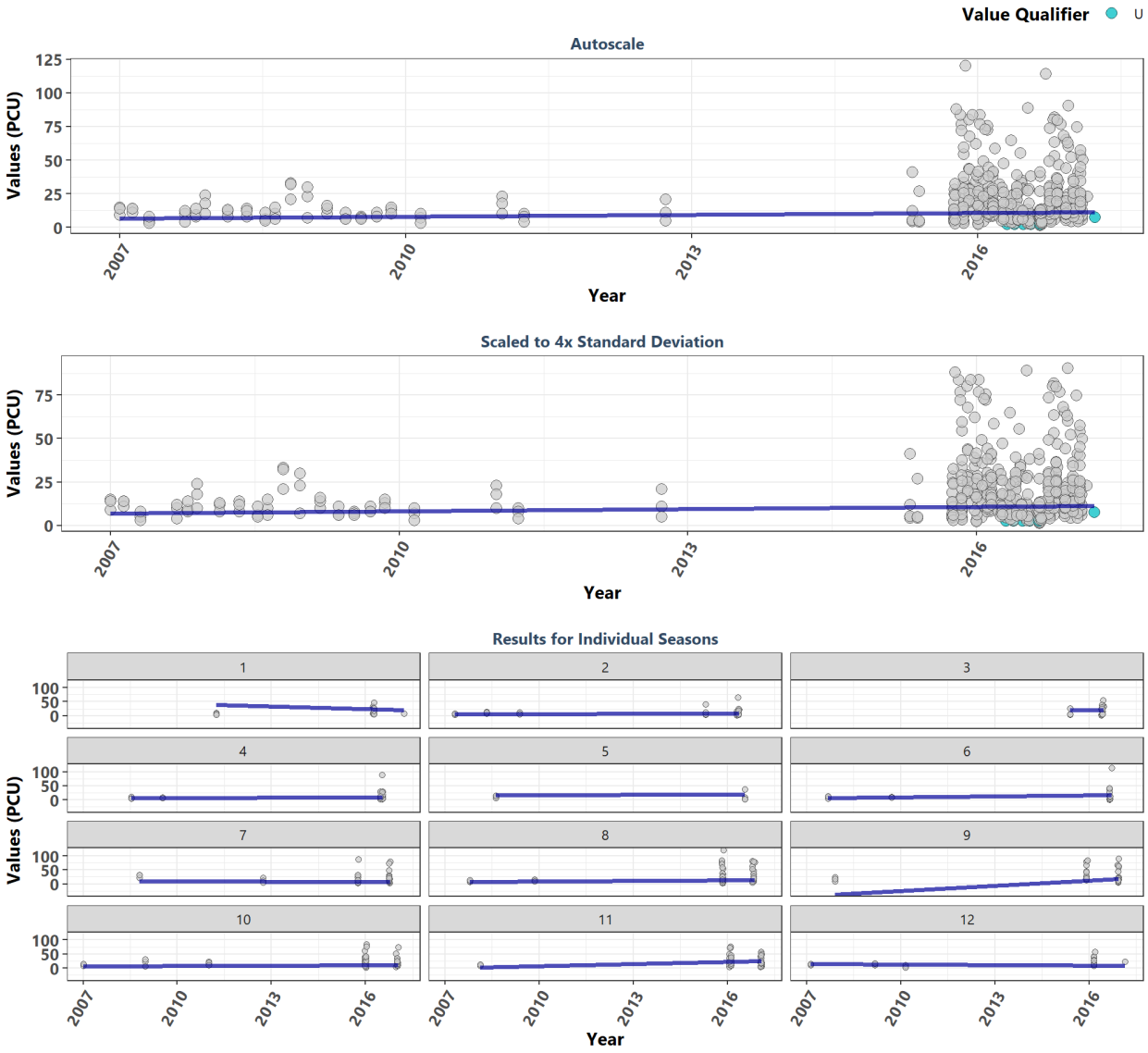
[a] $p < 0.00005$ appear as 0 due to rounding

Cockroach Bay Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	154	55.65	-0.0544	-0.5000	79.0033	-0.7	0.4711	36	0.0001	0
1	17	49.00	-0.6000	-20.3333	276.0000	1.0	0.2962	NA	NA	0
2	8	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	5	NA	0.2727	2.0057	20.5818	NA	NA	NA	NA	NA
4	18	43.50	0.5641	1.8182	15.6364	2.1	0.0375	NA	NA	0
5	13	32.00	-0.2900	-4.5077	164.0769	2.7	0.0063	NA	NA	0
6	6	67.50	-0.6000	-8.4000	234.3000	-0.2	0.8404	NA	NA	0
7	25	119.00	0.3595	3.0000	19.5000	-2.0	0.0426	NA	NA	0
8	10	133.50	-0.5273	-4.4125	159.9500	-2.5	0.0128	NA	NA	0
9	11	107.00	-0.1200	-1.3067	79.0033	-2.4	0.0178	NA	NA	0
10	26	68.55	0.1912	1.0000	43.0000	-0.8	0.3954	NA	NA	0
11	12	44.65	-0.3214	-1.4500	55.0500	1.2	0.2158	NA	NA	0
12	3	NA	-0.1333	-5.4286	116.3571	NA	NA	NA	NA	NA

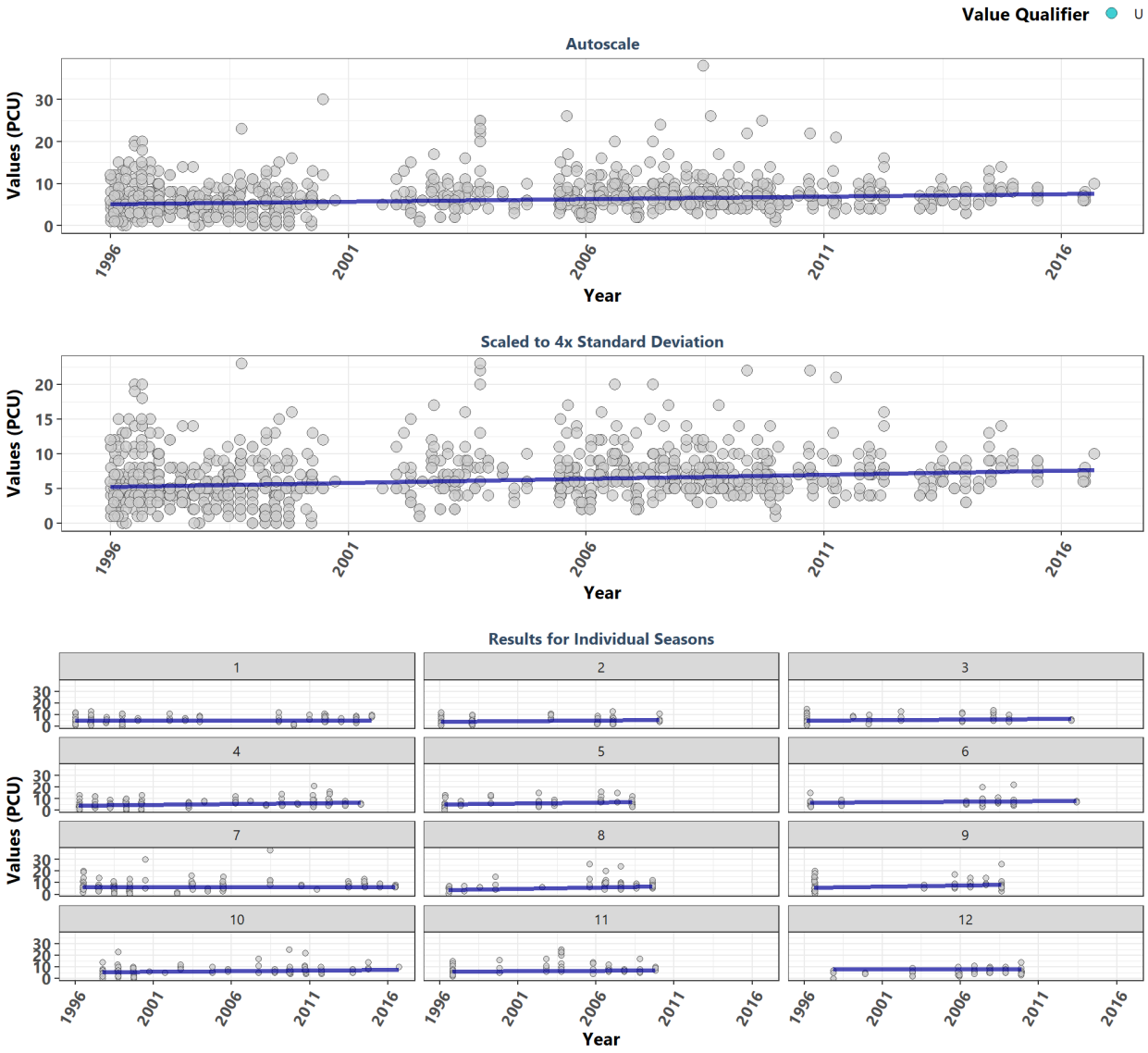
[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	445	14.80	0.0606	0.4337	6.8000	2.1	0.0332	16.1	0.1377	1
1	26	11.20	-0.1427	-3.1000	51.3000	0.8	0.4357	NA	NA	-2
2	38	8.00	0.0644	0.1625	5.6900	0.7	0.4636	NA	NA	1
3	28	NA	0.0454	0.6667	13.6167	NA	NA	NA	NA	NA
4	30	7.31	0.0711	0.1590	6.4097	0.7	0.4870	NA	NA	1
5	7	NA	0.0168	0.1548	16.9071	NA	NA	NA	NA	NA
6	31	8.16	0.2799	1.0722	6.4500	-0.2	0.8425	NA	NA	2
7	46	13.75	-0.0194	-0.0756	8.9156	0.6	0.5395	NA	NA	-1
8	53	21.10	0.0580	0.7333	7.1500	2.4	0.0169	NA	NA	1
9	44	20.30	0.0820	6.0000	-42.3000	-1.5	0.1253	NA	NA	2
10	54	18.30	0.0708	0.5400	5.8000	0.2	0.8523	NA	NA	1
11	54	19.95	0.2054	2.5000	-1.4000	0.5	0.5891	NA	NA	2
12	34	16.10	-0.1905	-0.4188	13.4875	2.9	0.0038	NA	NA	-1

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

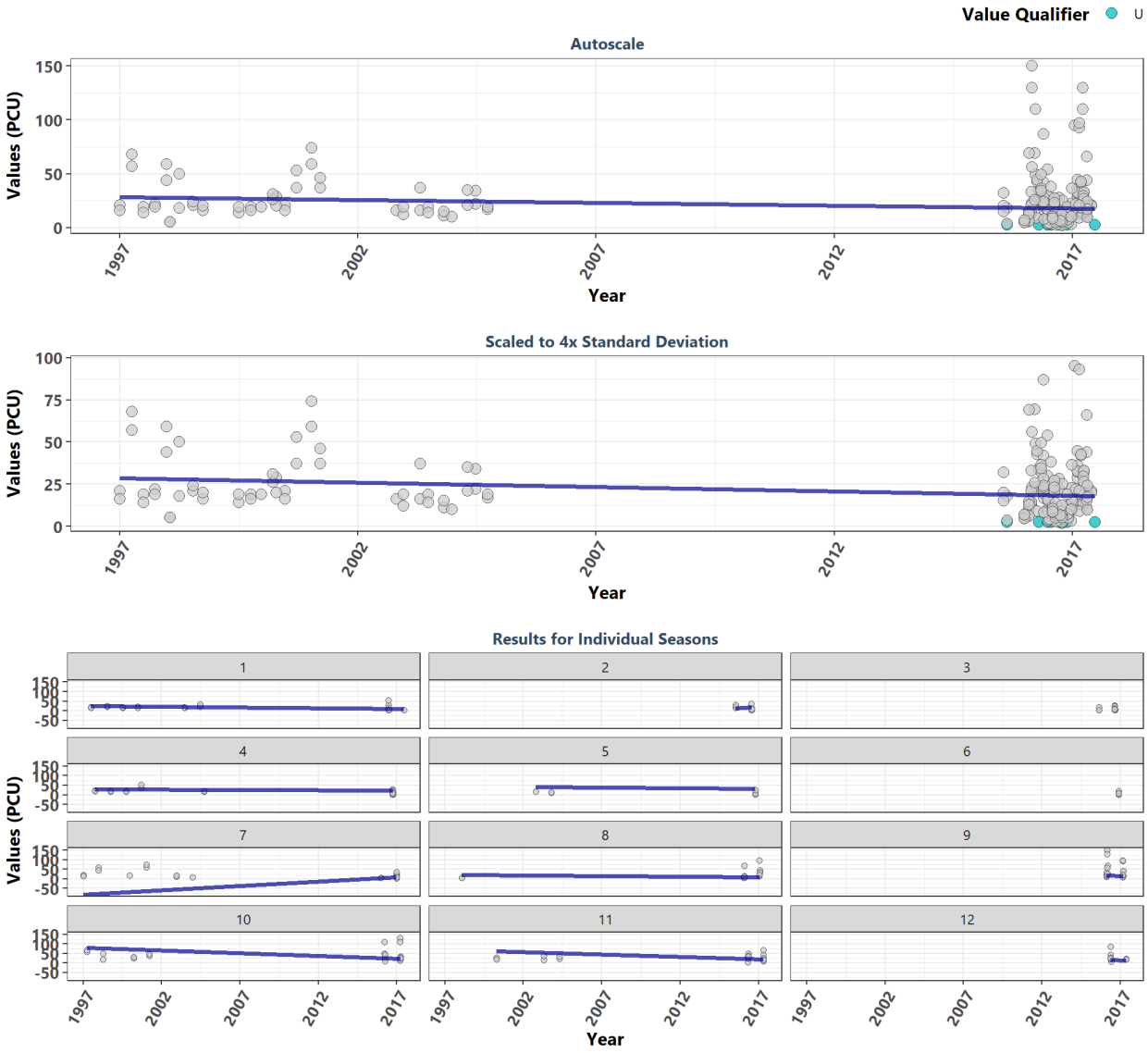
Florida Keys National Marine Sanctuary



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	954	6.0	0.1440	0.1176	5.2000	6.7	0.0000	16.7	0.1159	1
1	112	5.0	-0.0110	0.0000	5.0000	1.9	0.0615	NA	NA	-1
2	56	5.0	0.1393	0.1000	3.9000	-0.1	0.9070	NA	NA	1
3	43	6.0	0.1179	0.0801	4.7596	0.0	0.9914	NA	NA	1
4	138	5.0	0.1975	0.1429	4.0000	3.5	0.0005	NA	NA	1
5	54	7.0	0.1971	0.1818	4.9091	1.0	0.3243	NA	NA	1
6	48	6.5	0.0908	0.0833	6.4167	1.9	0.0537	NA	NA	1
7	144	6.0	0.0022	0.0000	6.0000	2.9	0.0044	NA	NA	-1
8	56	6.0	0.2701	0.2222	3.7778	3.0	0.0026	NA	NA	1
9	60	7.5	0.1678	0.2000	5.7000	2.0	0.0484	NA	NA	1
10	120	6.0	0.1871	0.1000	5.4000	3.2	0.0012	NA	NA	1
11	56	8.0	0.1586	0.1111	5.6111	0.5	0.6231	NA	NA	1
12	67	5.0	0.0448	0.0000	8.0000	1.7	0.0864	NA	NA	-1

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

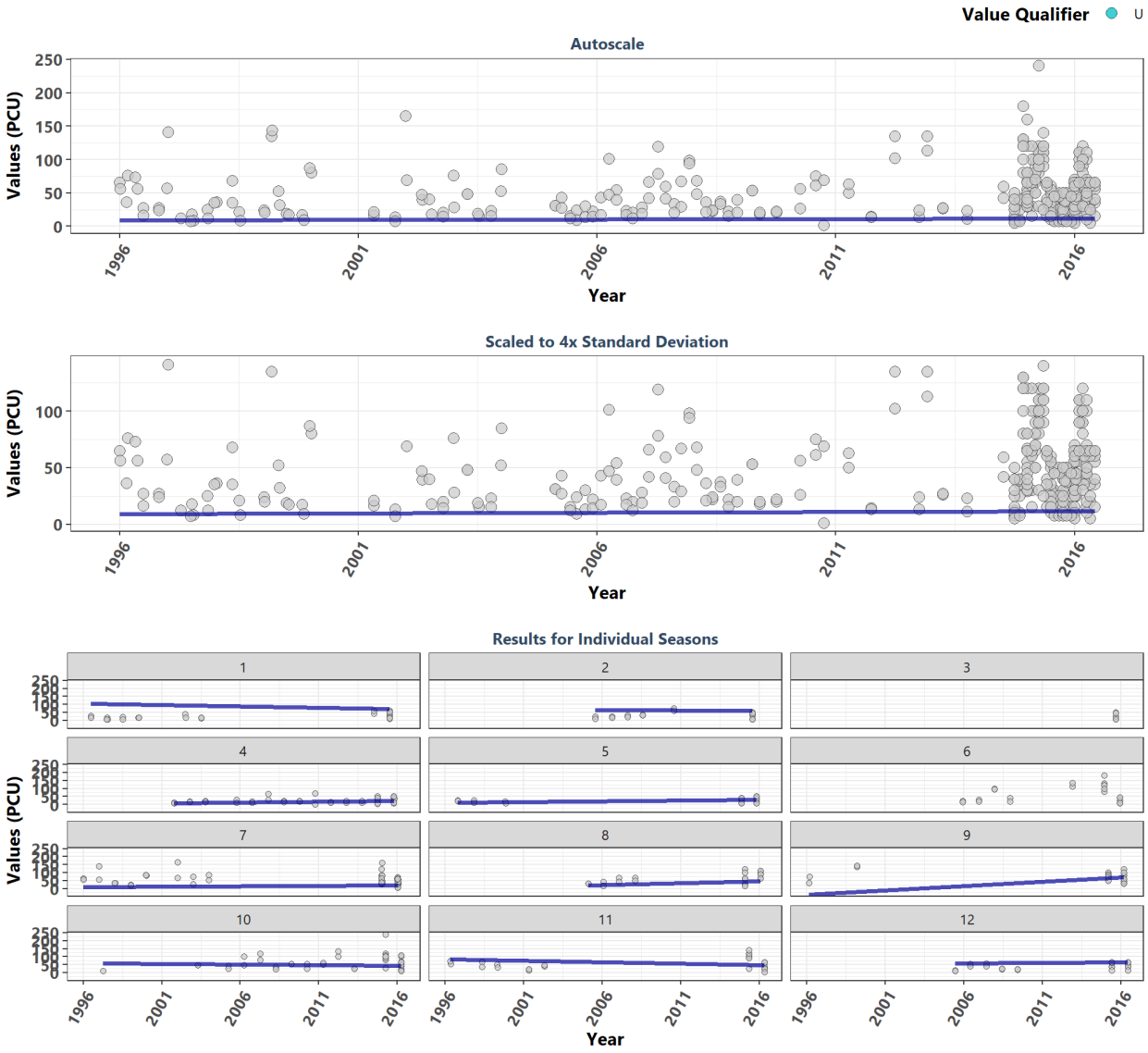
Gasparilla Sound-Charlotte Harbor Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	198	19.0	-0.1234	-0.5214	28.4673	-2.7	0.0064	28.9	0.0013	-1
1	23	15.9	-0.3810	-0.6538	24.0769	-1.9	0.0552	NA	NA	-1
2	12	NA	0.5147	8.5050	-146.8950	NA	NA	NA	NA	NA
3	14	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	21	16.0	-0.0857	-0.2404	28.4673	-2.6	0.0102	NA	NA	-1
5	7	11.0	-0.1542	-0.6587	45.5147	-1.2	0.2299	NA	NA	-1
6	9	NA	-0.3182	-11.6800	244.2500	NA	NA	NA	NA	NA
7	21	15.0	0.1868	4.4400	-80.4800	-2.1	0.0348	NA	NA	2
8	17	14.7	-0.2767	-0.5120	19.9958	3.2	0.0016	NA	NA	-1
9	19	NA	-0.3762	-0.8117	32.2350	NA	NA	NA	NA	NA
10	23	33.0	-0.0468	-2.8500	79.8500	-1.1	0.2935	NA	NA	-1
11	21	23.9	-0.0727	-2.6000	70.4000	-0.5	0.5879	NA	NA	-2
12	11	NA	-0.3286	-0.6071	26.5357	NA	NA	NA	NA	NA

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

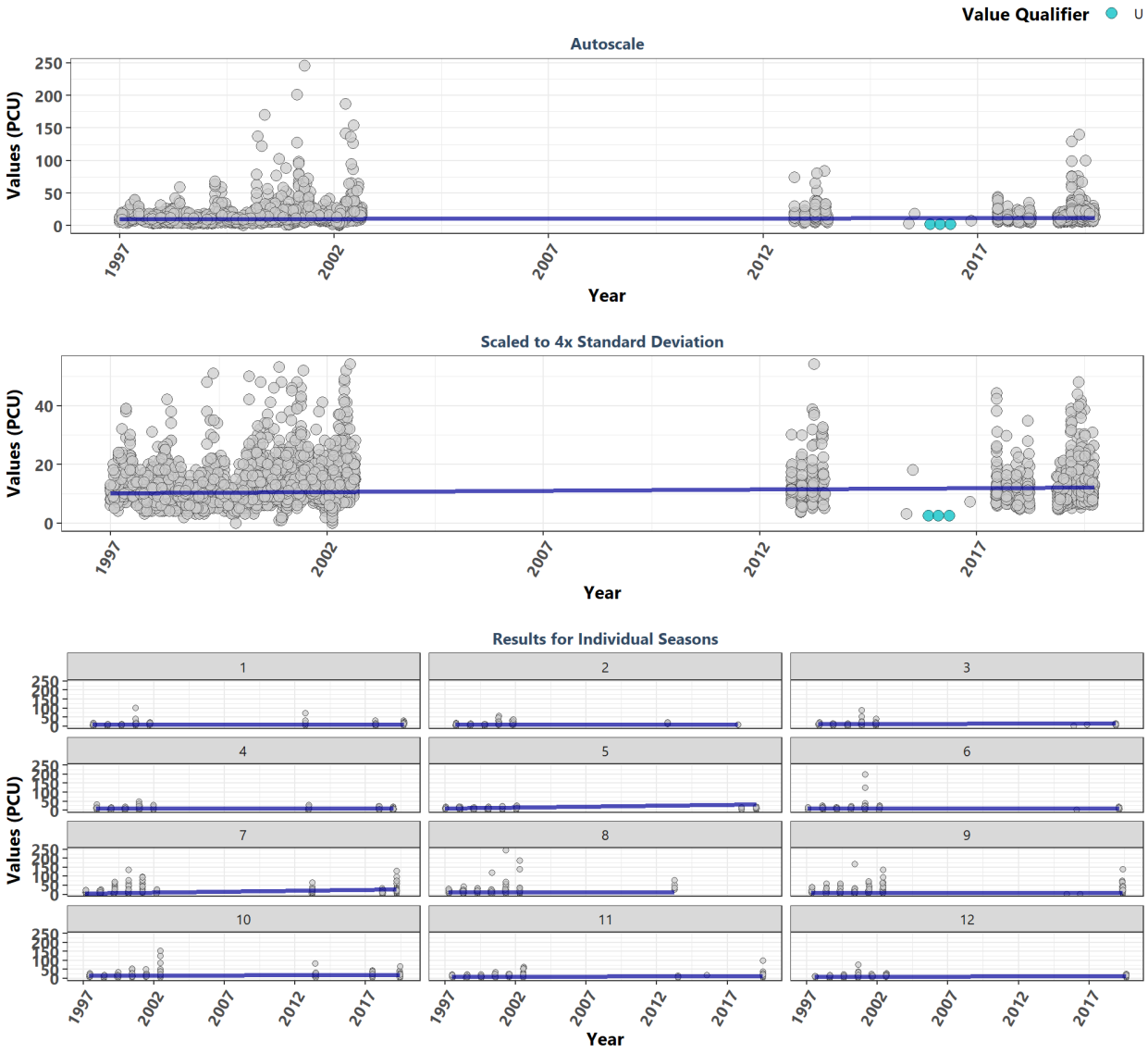
Loxahatchee River-Lake Worth Creek Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	392	44.0	0.0798	0.1250	8.9167	0.8	0.4284	34.1	0.0002	0
1	30	20.5	-0.1077	-1.8000	105.1000	2.5	0.0119	NA	NA	0
2	19	33.0	-0.0291	-0.2857	65.4286	1.6	0.1162	NA	NA	0
3	9	NA	0.2963	2.1429	6.7857	NA	NA	NA	NA	NA
4	58	20.0	0.3034	0.8125	4.6562	1.2	0.2309	NA	NA	0
5	25	30.0	0.2867	1.1111	8.8889	2.1	0.0339	NA	NA	0
6	27	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	65	50.0	0.1053	0.5833	8.9167	-2.9	0.0032	NA	NA	0
8	26	65.0	0.2515	2.1000	1.5000	3.5	0.0005	NA	NA	0
9	26	70.0	0.4615	5.3636	-36.9091	-0.8	0.4077	NA	NA	0
10	51	63.0	-0.0228	-0.6583	57.5083	0.5	0.6215	NA	NA	0
11	28	60.0	-0.2317	-1.6875	82.0625	-0.2	0.8370	NA	NA	0
12	28	47.5	0.0471	0.6364	50.9091	2.3	0.0209	NA	NA	0

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

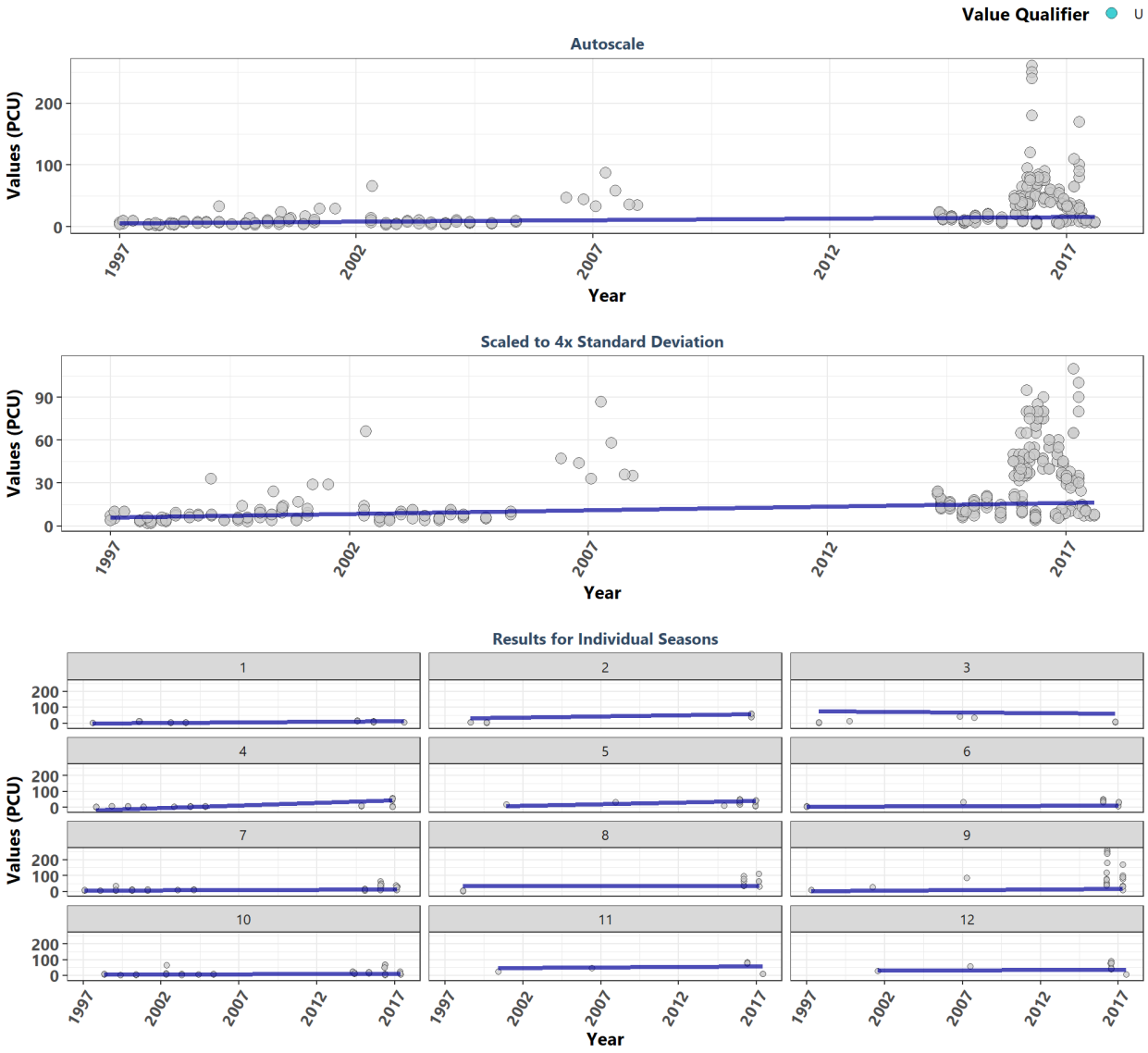
Nature Coast Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	3149	10.09	0.1009	0.0842	10.2010	8.1	0.0000	27.9	0.0033	1
1	298	9.05	-0.0063	0.0000	10.0000	3.9	0.0001	NA	NA	-1
2	191	9.00	0.0520	0.0298	7.8508	5.4	0.0000	NA	NA	1
3	233	8.38	0.0943	0.1161	12.9721	1.3	0.1973	NA	NA	1
4	292	8.00	0.0368	0.0039	8.9844	1.3	0.1812	NA	NA	1
5	268	9.00	0.1384	1.0000	11.0000	0.9	0.3633	NA	NA	2
6	256	10.00	0.1519	0.0861	8.6149	-0.2	0.8787	NA	NA	1
7	324	12.05	0.2587	1.0000	6.0000	1.5	0.1328	NA	NA	1
8	208	14.00	0.0556	0.0600	11.8100	3.0	0.0026	NA	NA	1
9	253	15.00	0.0559	0.0284	8.2619	2.9	0.0040	NA	NA	1
10	310	13.44	0.1200	0.2601	14.2198	2.5	0.0126	NA	NA	1
11	263	11.00	0.1432	0.1436	10.5690	4.1	0.0000	NA	NA	1
12	253	11.00	0.1676	0.1495	10.4020	3.4	0.0006	NA	NA	1

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

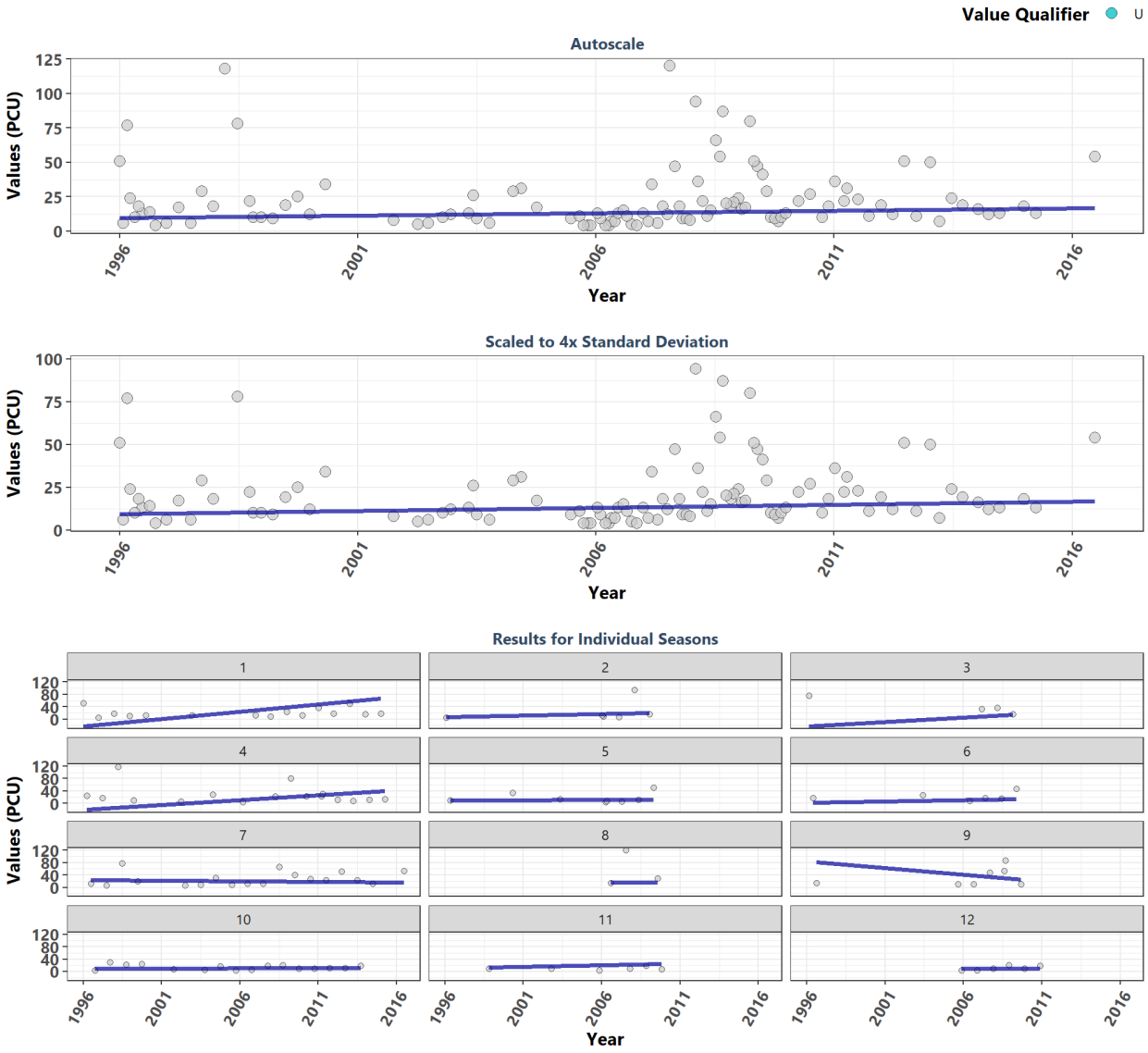
Pinellas County Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	312	14.0	0.2790	0.5000	5.9730	7.3	0.0000	36.9	0.0001	1
1	33	9.0	0.4833	0.6250	-0.2500	1.3	0.2070	NA	NA	1
2	13	40.0	0.1200	1.3000	30.3000	2.3	0.0191	NA	NA	1
3	8	7.5	-0.1071	-0.8035	76.2659	1.2	0.2479	NA	NA	-2
4	38	6.0	0.4103	3.2105	-21.0000	4.9	0.0000	NA	NA	2
5	22	35.0	0.4487	2.1176	-2.3529	0.6	0.5491	NA	NA	1
6	20	33.0	0.5434	0.4000	2.8000	0.1	0.9155	NA	NA	1
7	53	11.0	0.3571	0.2893	5.6163	5.2	0.0000	NA	NA	1
8	13	40.0	0.0211	0.0000	33.0000	2.1	0.0389	NA	NA	-1
9	25	55.0	0.2597	0.5625	3.4375	0.9	0.3736	NA	NA	1
10	66	13.0	0.1515	0.1484	6.3297	3.1	0.0017	NA	NA	1
11	8	61.0	0.0128	0.7333	44.0667	-0.3	0.7872	NA	NA	1
12	13	58.0	0.0866	0.2111	30.9889	0.0	1.0000	NA	NA	1

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

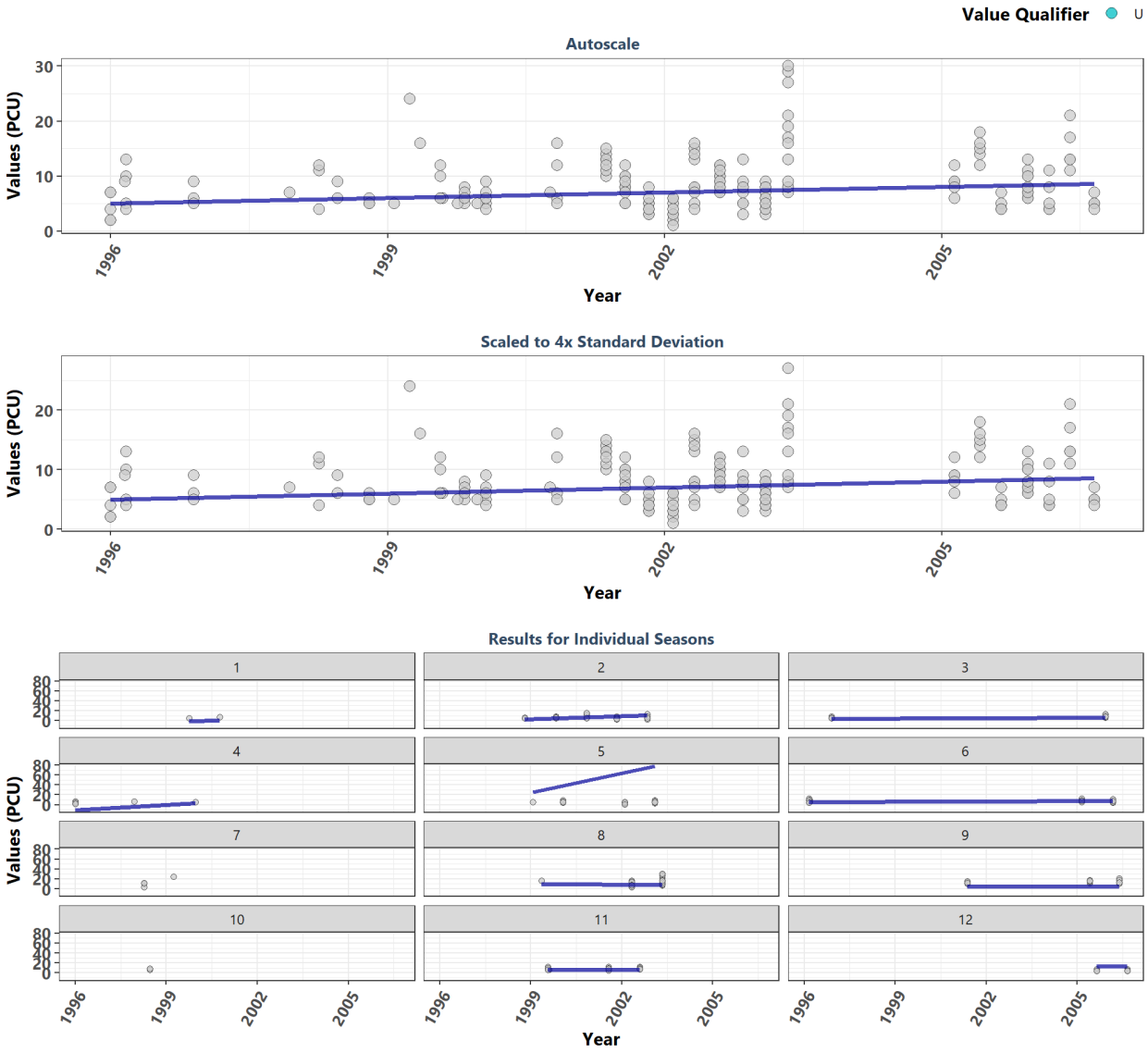
Rocky Bayou State Park Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	111	15.0	0.1263	0.3571	9.4271	1.6	0.1182	11.8	0.3782	0
1	15	16.0	0.3333	4.6667	-22.3333	1.4	0.1497	NA	NA	0
2	6	11.0	0.1333	1.1429	6.0000	1.3	0.1806	NA	NA	0
3	4	35.0	0.6667	2.8000	-22.7000	-1.0	0.3082	NA	NA	0
4	16	19.5	0.1429	3.1667	-20.8333	-0.8	0.4150	NA	NA	0
5	8	10.5	0.0167	0.0500	10.5250	0.0	1.0000	NA	NA	0
6	6	18.0	0.5333	0.8846	1.7115	0.2	0.8483	NA	NA	0
7	18	21.0	-0.1583	-0.3333	23.6667	1.6	0.1020	NA	NA	0
8	3	29.0	0.2857	0.5000	10.0000	0.0	1.0000	NA	NA	0
9	7	14.0	-0.6667	-4.2622	84.0157	0.3	0.7587	NA	NA	0
10	16	11.0	0.0357	0.0833	9.6667	0.0	0.9640	NA	NA	0
11	6	9.5	0.2876	1.1250	9.1875	-0.2	0.8483	NA	NA	0
12	6	9.5	-0.1333	-0.1111	10.6111	1.7	0.0852	NA	NA	0

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

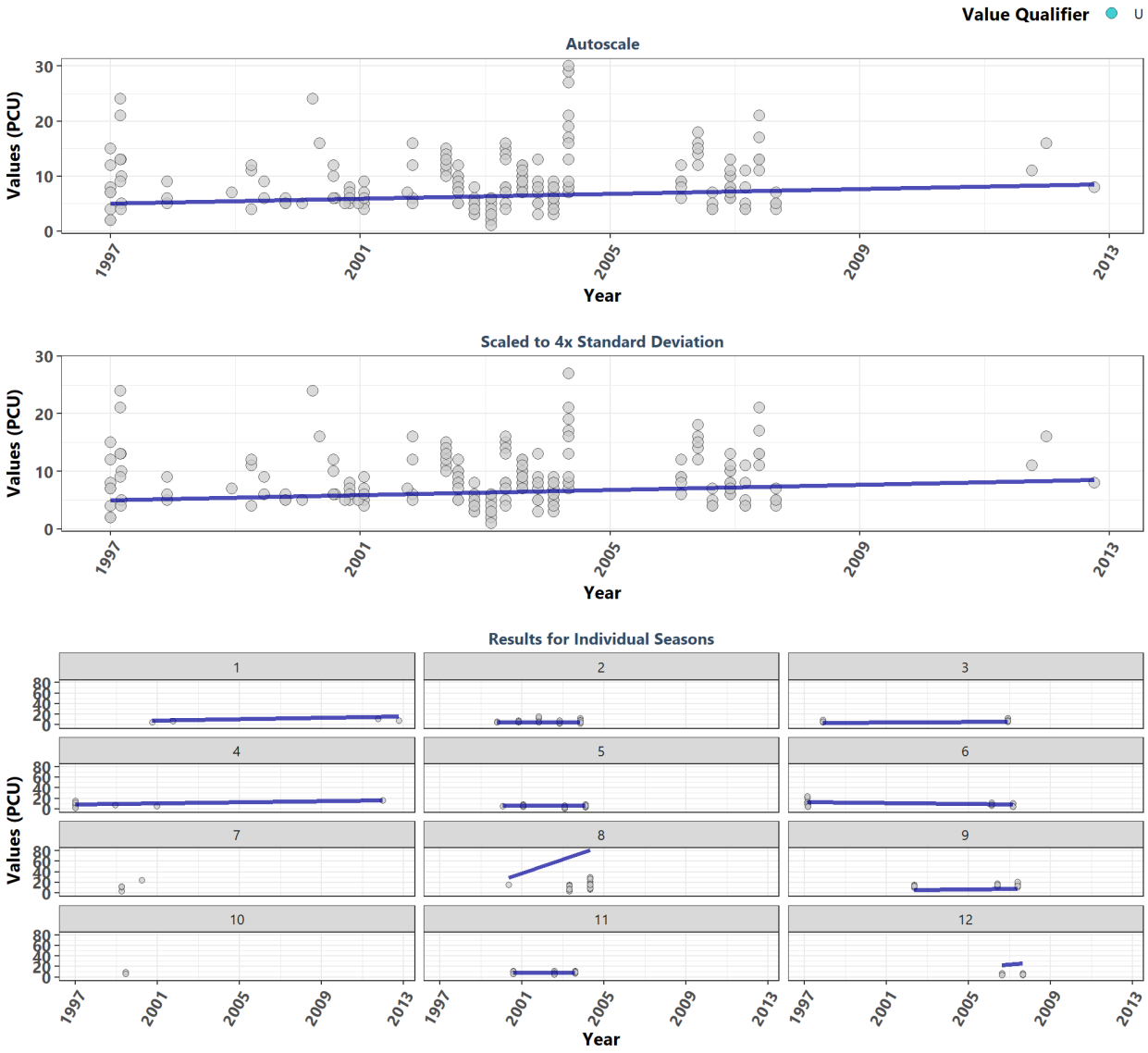
Rookery Bay Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	182	7.0	0.1343	0.3333	5.0000	2.2	0.0306	6.4	0.7801	1
1	2	NA	0.1818	1.0000	-5.0000	NA	NA	NA	NA	NA
2	32	5.0	1.0000	2.0000	-3.0000	0.1	0.8913	NA	NA	2
3	14	NA	0.1032	0.3333	3.0000	NA	NA	NA	NA	NA
4	7	5.0	0.2251	3.5000	-11.0000	0.4	0.6897	NA	NA	2
5	28	5.0	0.5000	13.0000	-14.5000	0.8	0.4131	NA	NA	2
6	15	8.0	0.1429	0.2500	5.0000	-1.1	0.2860	NA	NA	1
7	4	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	22	13.5	-0.2000	-0.2000	9.8000	1.7	0.0970	NA	NA	-1
9	17	13.0	0.0181	0.0000	5.0000	1.3	0.1807	NA	NA	-1
10	2	NA	0.1878	0.6667	5.1667	NA	NA	NA	NA	NA
11	28	8.5	0.1868	0.2222	5.2778	1.5	0.1290	NA	NA	1
12	11	NA	0.2279	0.5000	8.5000	NA	NA	NA	NA	NA

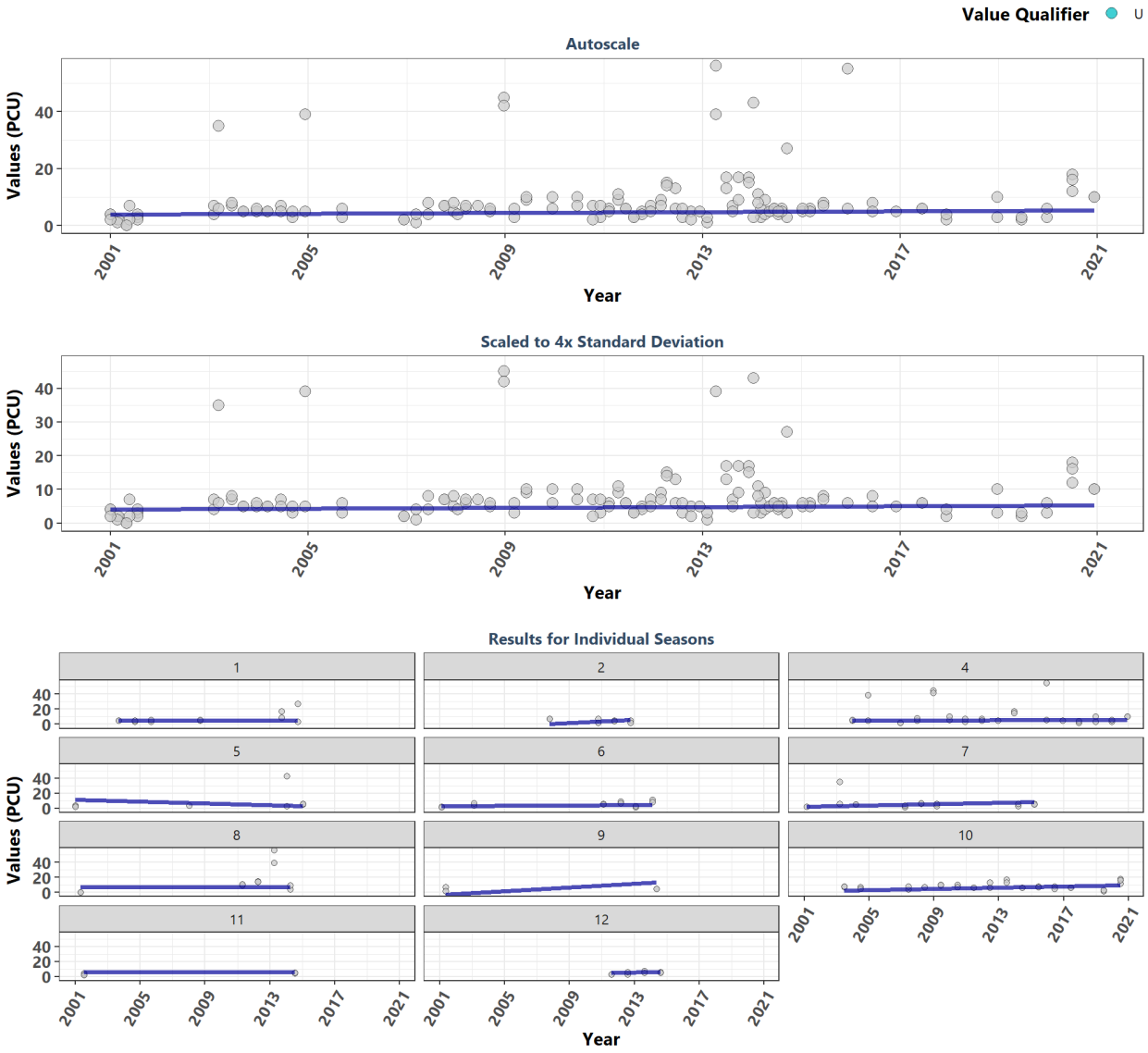
[a] $p < 0.00005$ appear as 0 due to rounding

Rookery Bay National Estuarine Research Reserve



Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	193	8.0	0.1117	0.2222	5.0000	1.8	0.0735	12.9	0.2304	0
1	4	7.5	0.1878	0.6667	5.1667	1.0	0.3082	NA	NA	0
2	32	5.0	0.0181	0.0000	5.0000	0.1	0.8913	NA	NA	0
3	14	NA	0.1032	0.3333	3.0000	NA	NA	NA	NA	NA
4	12	7.0	0.2279	0.5000	8.5000	0.6	0.5772	NA	NA	0
5	28	5.0	0.1868	0.2222	5.2778	0.8	0.4131	NA	NA	0
6	19	9.0	-0.3567	-0.5000	13.5000	-2.3	0.0216	NA	NA	0
7	4	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	22	13.5	0.5000	13.0000	-14.5000	1.7	0.0970	NA	NA	0
9	17	13.0	0.6667	0.3250	4.2500	1.3	0.1807	NA	NA	0
10	2	NA	0.1818	1.0000	-5.0000	NA	NA	NA	NA	NA
11	28	8.5	0.1061	0.2583	7.0000	1.5	0.1290	NA	NA	0
12	11	NA	0.2251	3.5000	-11.0000	NA	NA	NA	NA	NA

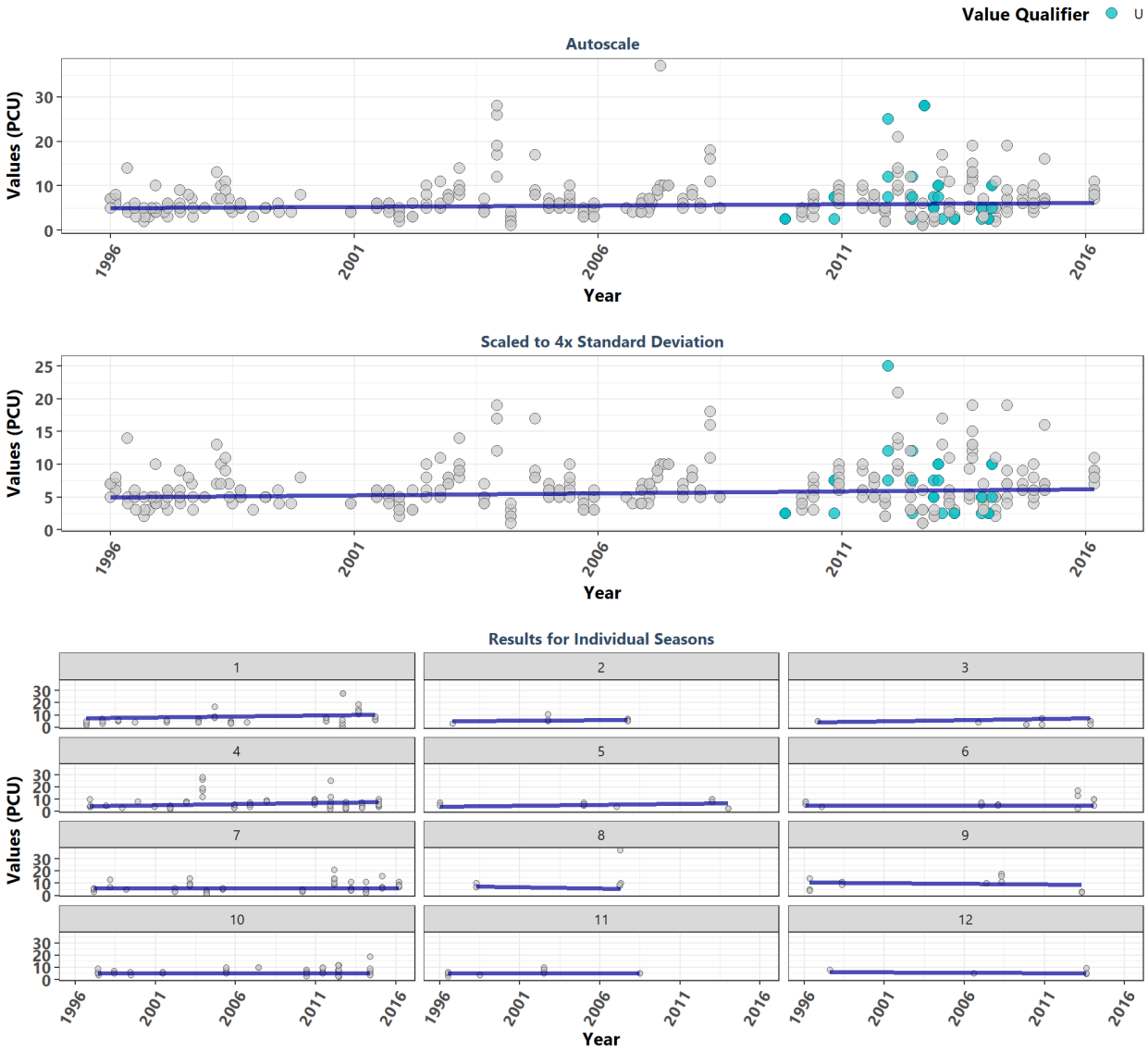
[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	143	6.0	0.1410	0.0625	4.0000	1.2	0.2173	11.5	0.3193	0
1	12	5.0	0.0095	0.0000	5.0000	1.6	0.1144	NA	NA	0
2	8	5.0	0.4286	1.0000	-6.5000	-1.5	0.1429	NA	NA	0
4	31	6.0	0.0000	0.0385	4.7500	0.3	0.7833	NA	NA	0
5	8	4.0	-0.4286	-0.6250	11.5625	1.3	0.1836	NA	NA	0
6	12	5.5	0.3929	0.1429	2.5714	1.7	0.0807	NA	NA	0
7	15	5.0	0.3485	0.4722	1.9306	0.0	1.0000	NA	NA	0
8	10	10.0	0.0184	0.0000	7.0000	1.5	0.1411	NA	NA	0
9	4	NA	0.3778	1.1864	-3.0500	NA	NA	NA	NA	NA
10	30	7.0	0.3939	0.4273	1.0136	0.1	0.8992	NA	NA	0
11	5	NA	0.0366	0.0000	6.0000	NA	NA	NA	NA	NA
12	8	5.0	0.5000	0.1154	4.0000	1.5	0.1429	NA	NA	0

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

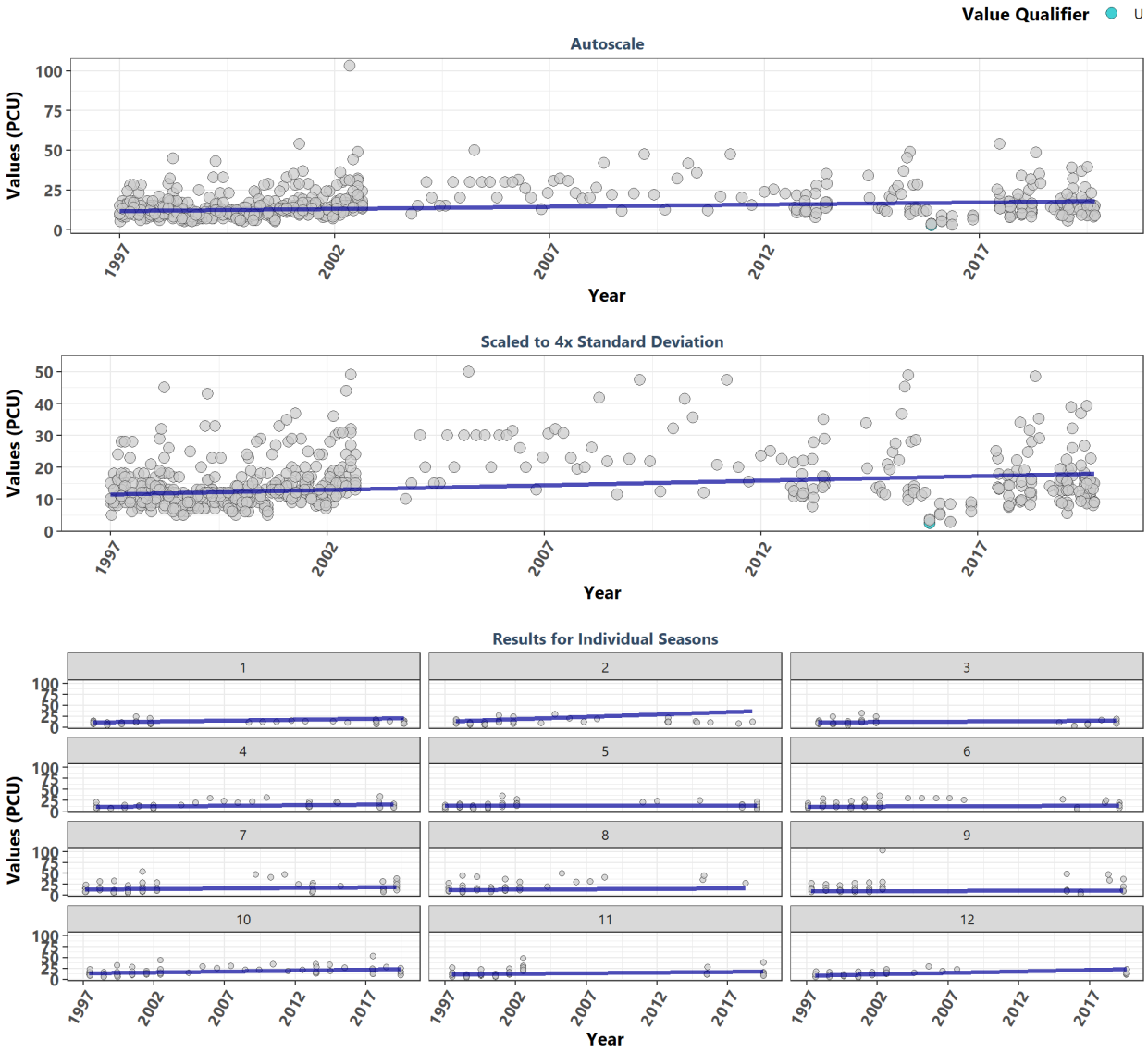
St. Joseph Bay Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	336	6.0	0.0768	0.0588	5.0000	2.2	0.0283	16.3	0.1314	1
1	60	6.0	0.1429	0.1667	7.1667	2.6	0.0085	NA	NA	1
2	8	5.5	0.1232	0.0714	5.0000	1.0	0.3206	NA	NA	1
3	14	5.0	0.2305	0.1818	4.1818	-0.5	0.6280	NA	NA	1
4	74	5.0	0.2857	0.2000	4.1000	-0.8	0.4390	NA	NA	1
5	16	6.0	0.1909	0.1429	4.0000	-1.2	0.2179	NA	NA	1
6	18	6.0	-0.0611	0.0000	5.0000	0.4	0.6940	NA	NA	-1
7	62	6.0	0.0719	0.0000	6.0000	2.2	0.0270	NA	NA	-1
8	7	NA	-0.2250	-0.2143	7.9286	NA	NA	NA	NA	NA
9	12	9.5	-0.1364	-0.1144	10.8154	-0.6	0.5706	NA	NA	-1
10	48	6.0	-0.0989	0.0000	5.0000	1.3	0.2104	NA	NA	-1
11	12	5.0	0.1061	0.0000	5.0000	0.5	0.6420	NA	NA	-1
12	5	5.3	-0.1000	-0.0429	6.0286	0.0	1.0000	NA	NA	-1

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

St. Martins Marsh Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	598	13.00	0.2228	0.2859	11.5470	8.0	0.0000	15.7	0.1526	1
1	52	11.40	0.3024	0.4086	10.7743	2.2	0.0286	NA	NA	1
2	45	12.00	0.3224	1.0550	12.8350	1.7	0.0911	NA	NA	1
3	43	10.00	0.1737	0.1594	11.3622	0.8	0.3966	NA	NA	1
4	46	11.90	0.2715	0.2923	9.7077	2.7	0.0075	NA	NA	1
5	49	13.00	-0.0073	0.0000	13.0000	3.0	0.0031	NA	NA	-1
6	51	13.00	0.2081	0.1343	10.7284	1.5	0.1370	NA	NA	1
7	59	15.00	0.1601	0.2500	14.0000	1.8	0.0724	NA	NA	1
8	44	16.00	0.1427	0.2333	12.0667	3.1	0.0019	NA	NA	1
9	50	13.00	0.0897	0.0811	9.6758	-0.1	0.9461	NA	NA	1
10	68	16.09	0.3310	0.4429	13.8757	4.0	0.0001	NA	NA	1
11	46	12.00	0.2900	0.3170	11.7318	3.0	0.0028	NA	NA	1
12	45	11.00	0.3798	0.6667	9.0000	3.7	0.0002	NA	NA	1

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

Appendix V: Managed Area Summary Box Plots

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

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

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

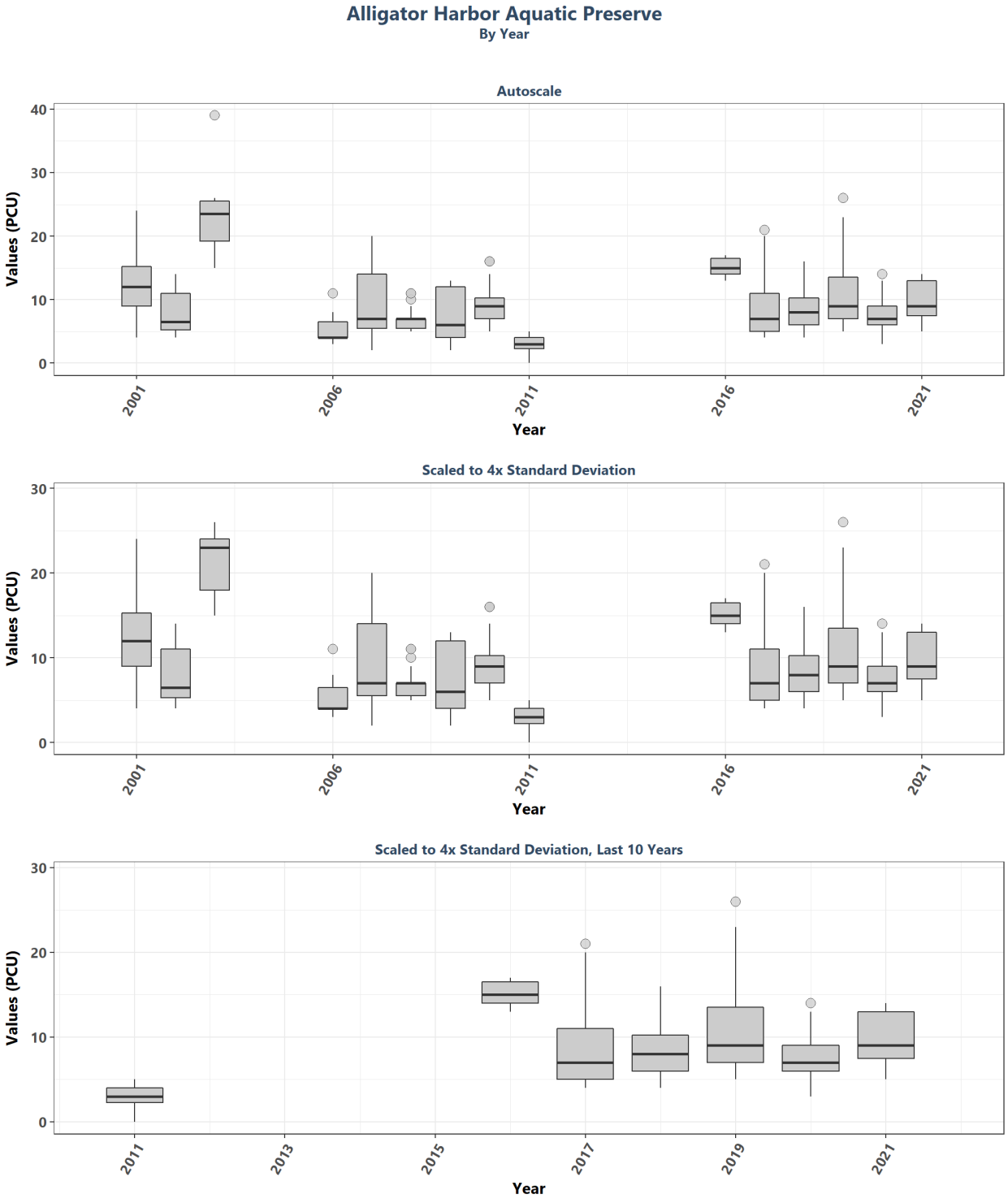
1. Y-axis autoscaled
2. Y-axis set to be mean + 4 times the standard deviation
3. Y-axis set to be mean + 4 times the standard deviation for most recent 10 years of data

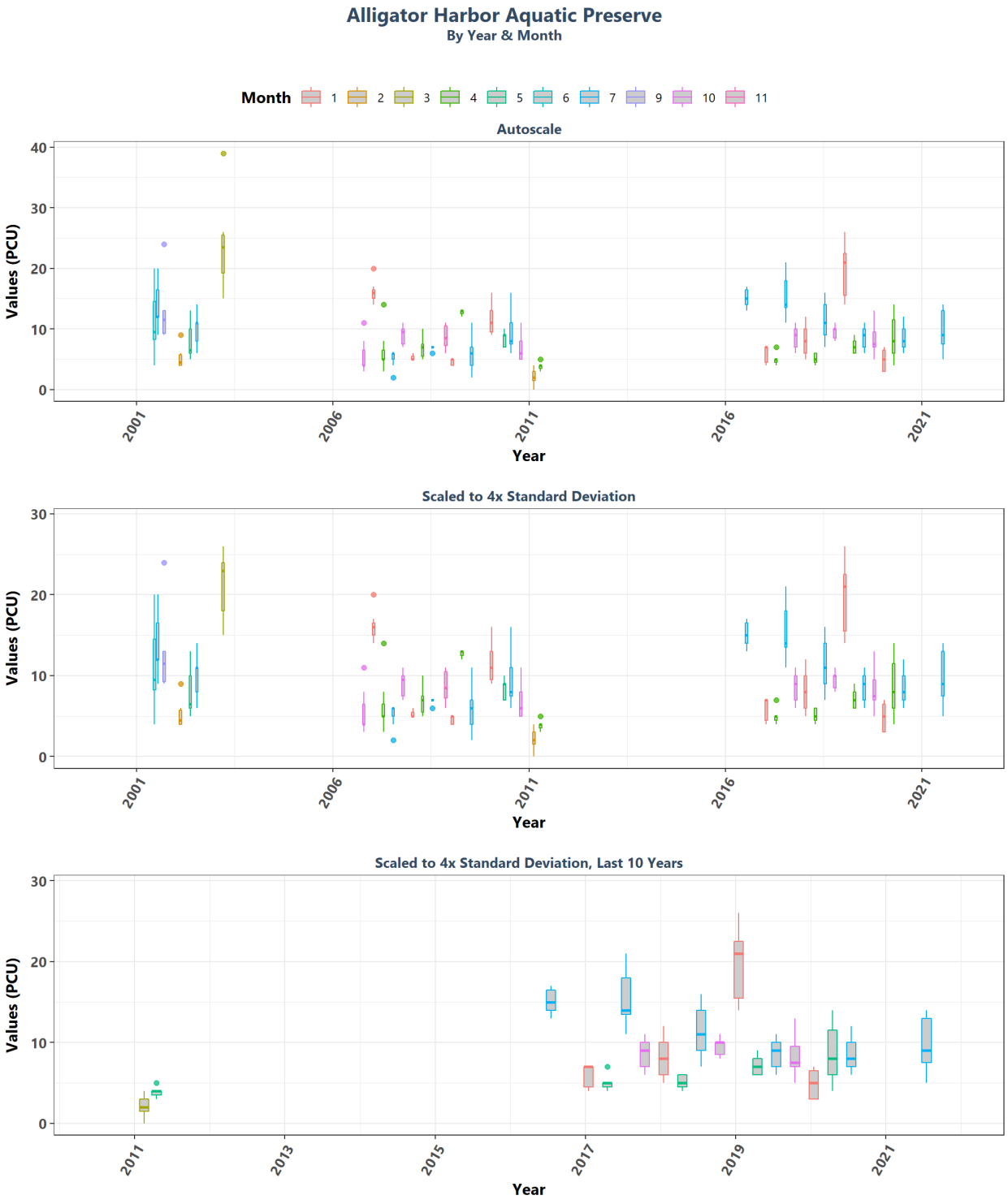
```
if(n==0){
   print("There are no managed areas that qualify.")
} else {
   for (i in 1:n) {
      plot_data <- data[data$SufficientData==TRUE &
                           data$ManagedAreaName==MA_Include[i],]
      year_lower <- min(plot_data$Year)
      year_upper <- max(plot_data$Year)
      min_RV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

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

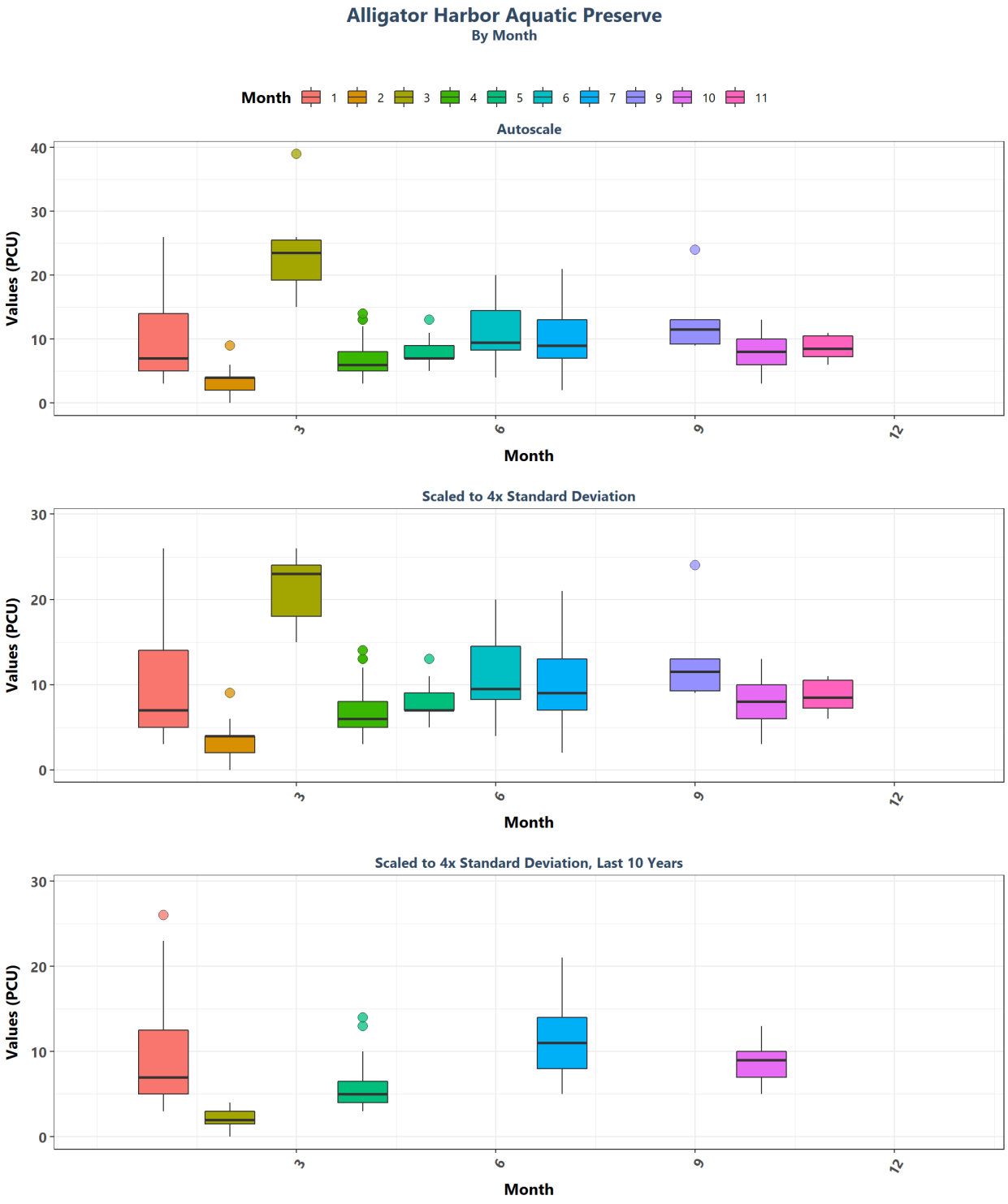
```
p2 <- ggplot(data=plot_data,
             aes(x=Year, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Scaled to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

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

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

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

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

p5 <- ggplot(data=plot_data,
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Scaled to 4x Standard Deviation",
        x="Year", y=paste0("Values (", unit, ")"), color="Month") +
```

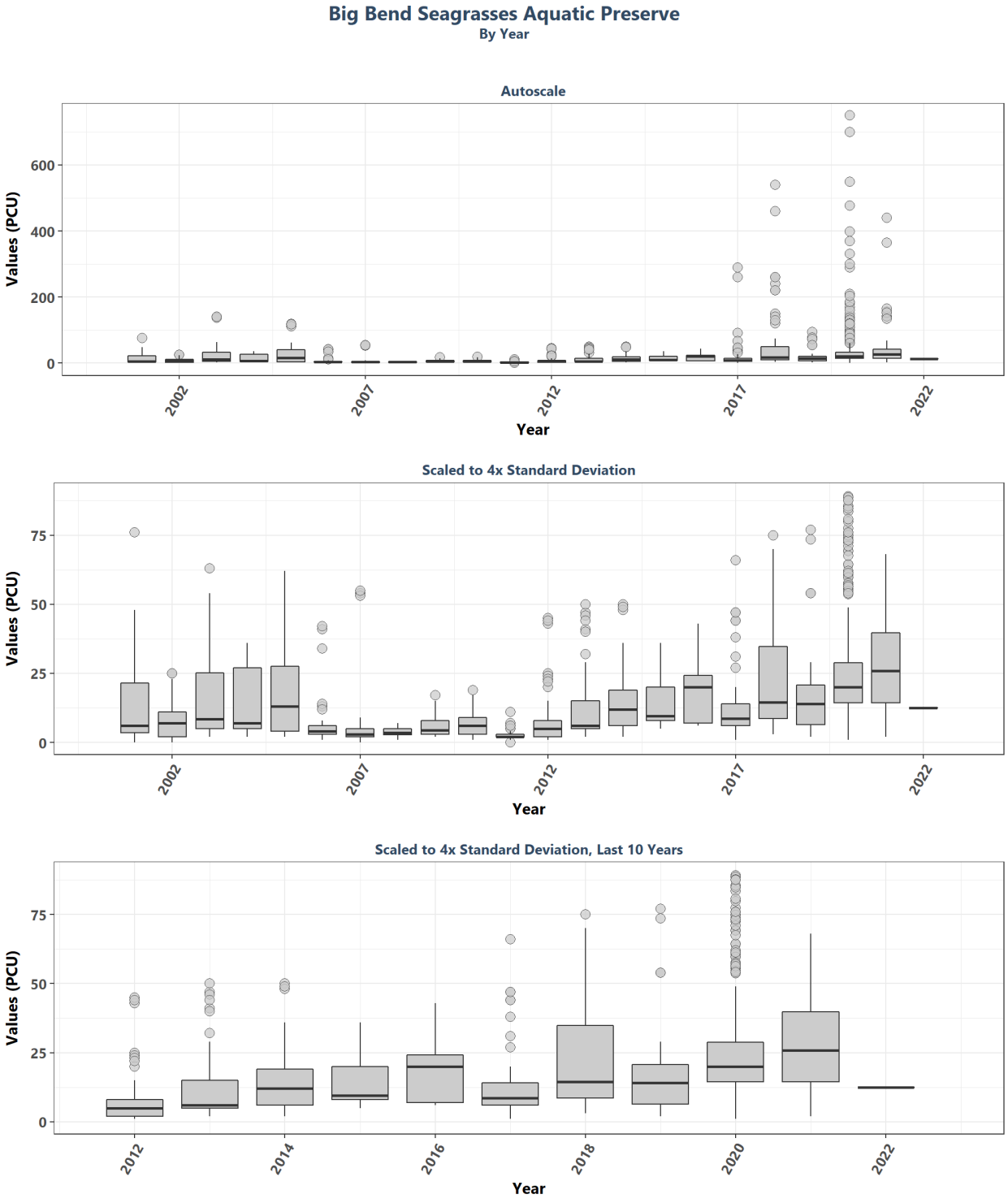
```
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(year_lower - 1, year_upper + 1),
                      breaks=rev(seq(year_upper,
                                     year_lower, -x_scale))) +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal") +
   guides(color=guide_legend(nrow=1))

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

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

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

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

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

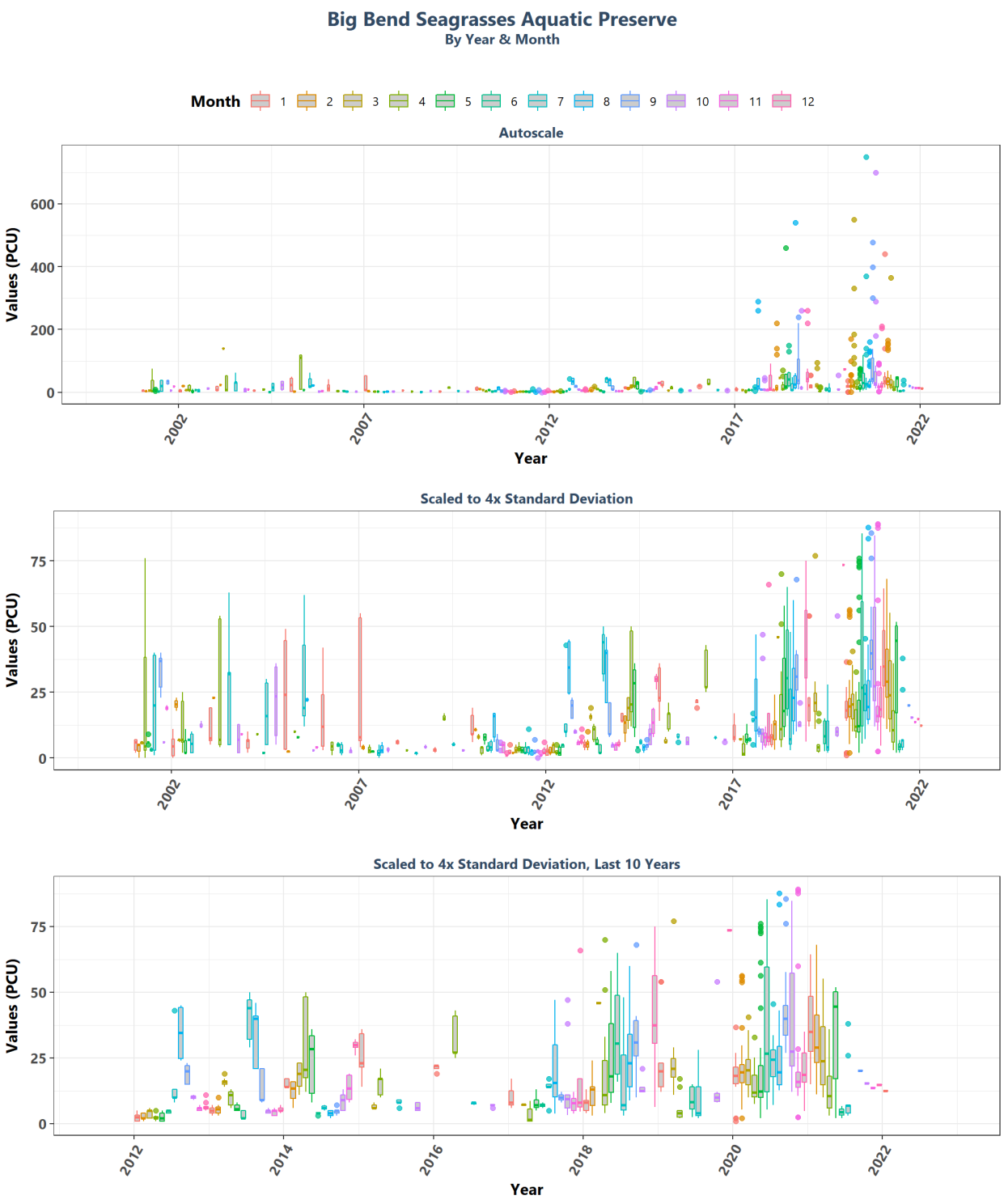
```
        guides(fill=guide_legend(nrow=1))

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

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

    p000 <- ggplot() + labs(title=paste0(MA_Include[i]),
                            subtitle="By Month") + plot_theme +
        theme(panel.border=element_blank(),
              panel.grid.major=element_blank(),
              panel.grid.minor=element_blank(), axis.line=element_blank())

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

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

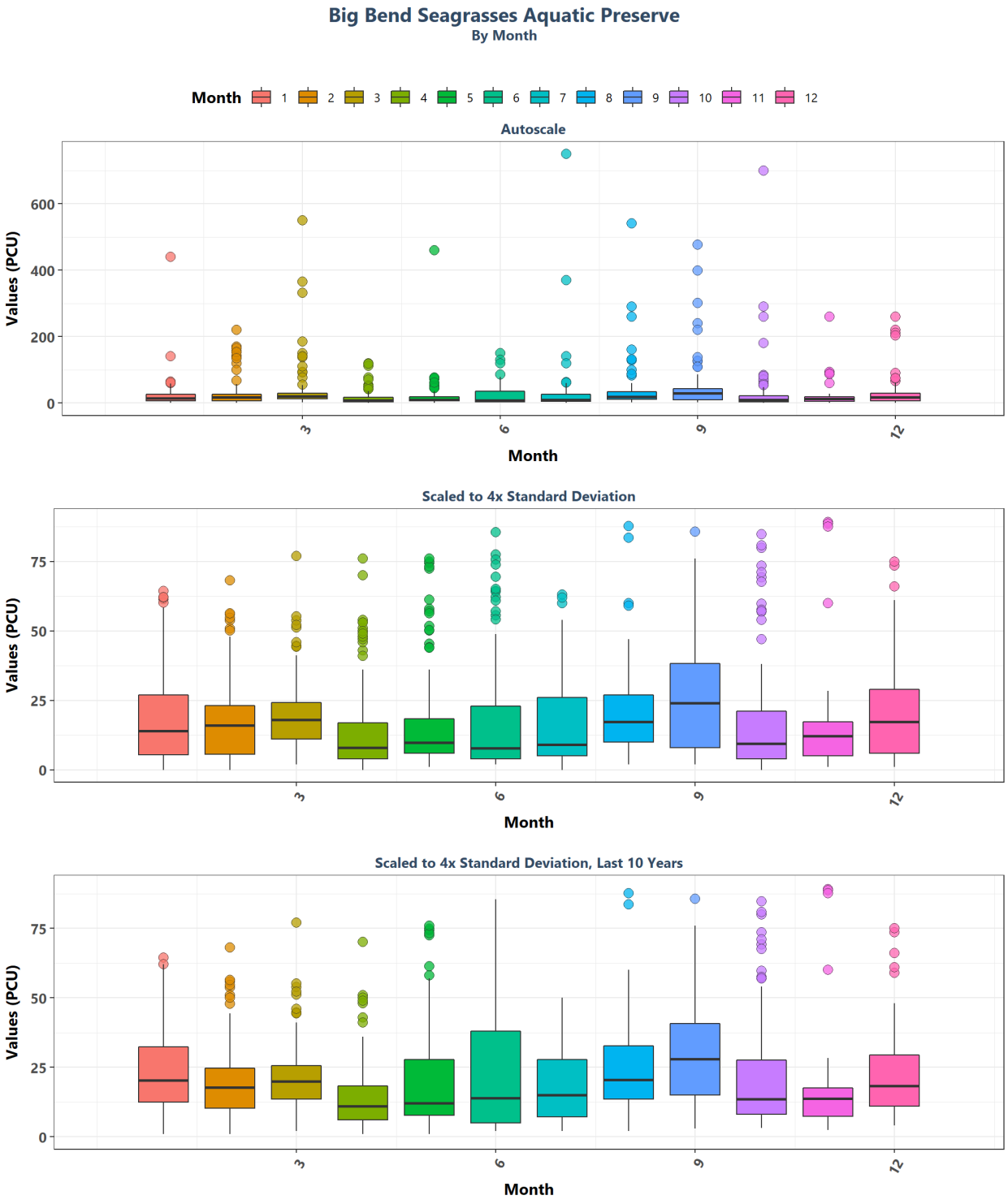

Alligator Harbor Aquatic Preserve
By Year
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year


Alligator Harbor Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 9 10 11
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year


Alligator Harbor Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 9 10 11
Autoscale
Values (PCU)
Month
Scaled to 4x Standard Deviation
Values (PCU)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Month

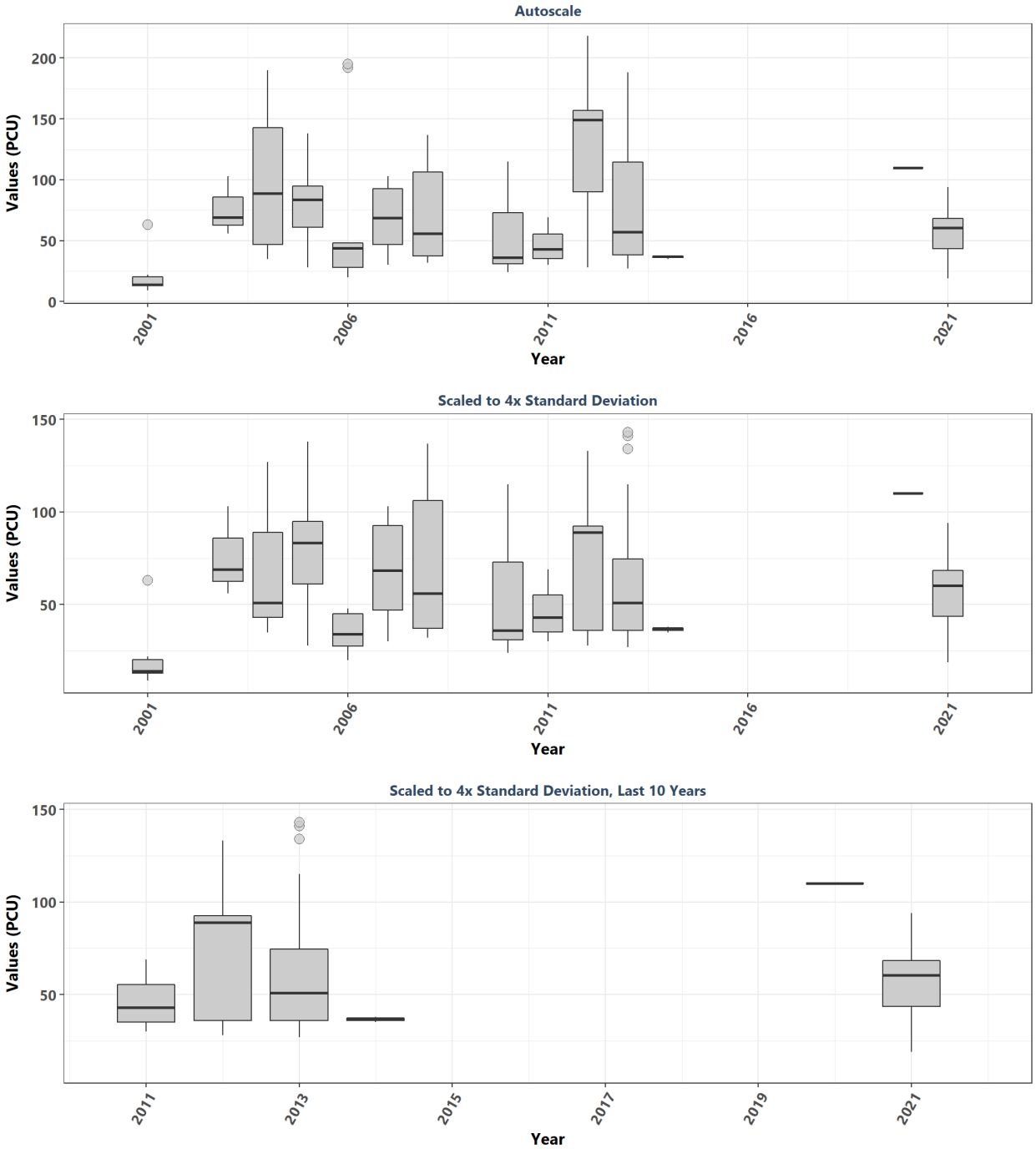

Big Bend Seagrasses Aquatic Preserve
By Year
Autoscale
Values (PCU)
600
400
200
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (PCU)
75
50
25
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
75
50
25
0
2012
2014
2016
2018
2020
2022
Year

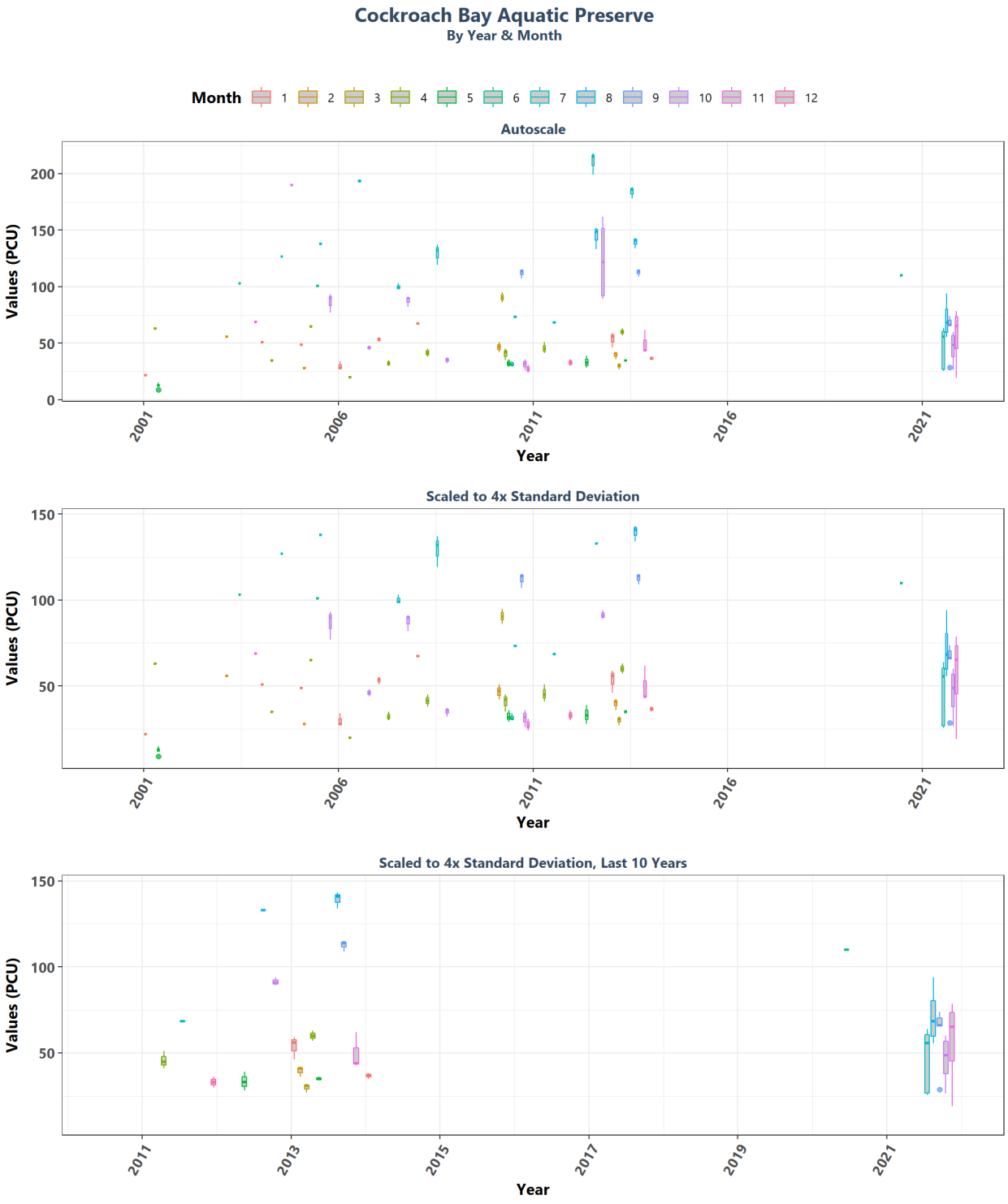

Big Bend Seagrasses Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
600
400
200
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (PCU)
75
50
25
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
75
50
25
0
2012
2014
2016
2018
2020
2022
Year

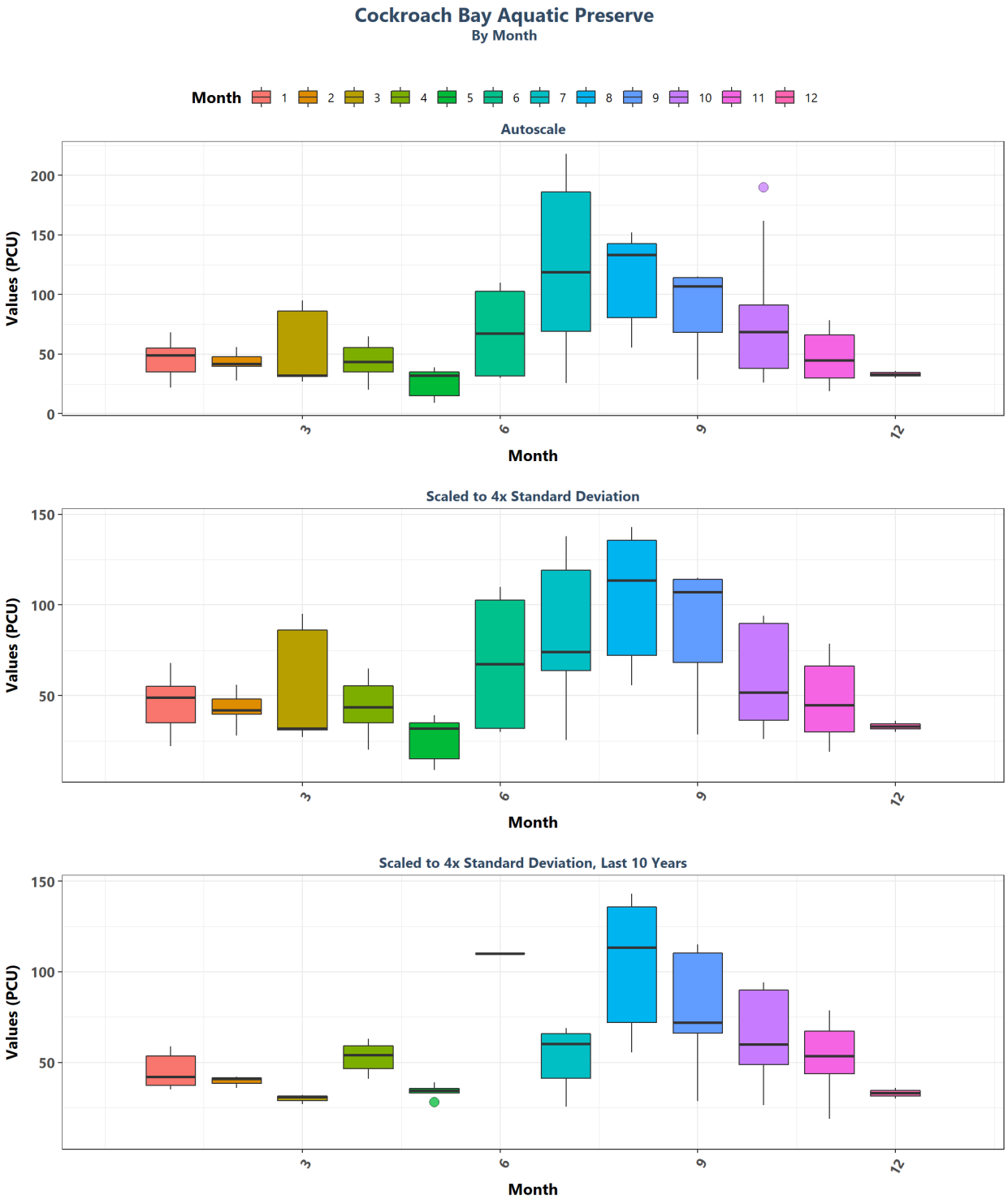

Big Bend Seagrasses Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
600
400
200
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (PCU)
75
50
25
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
75
50
25
0
3
6
9
12
Month

Cockroach Bay Aquatic Preserve

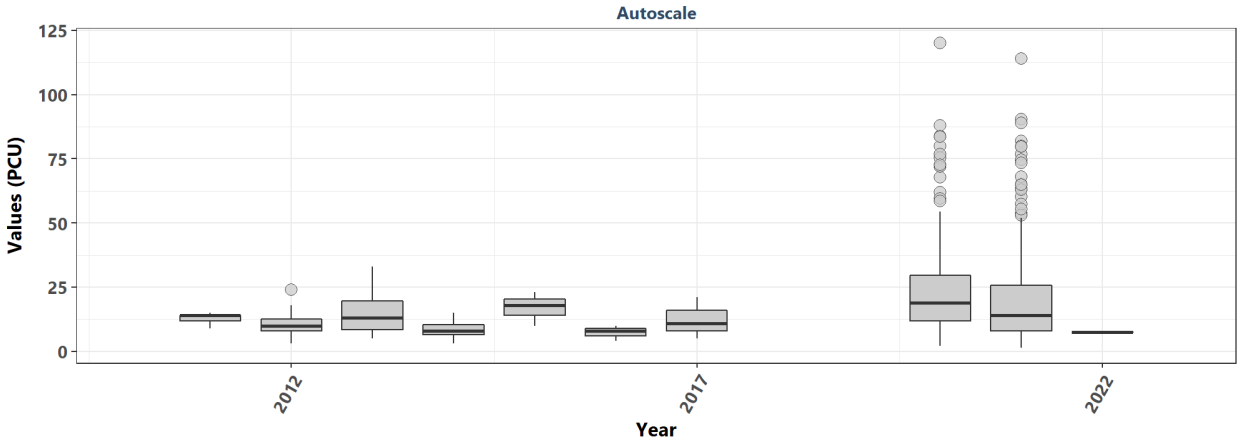
By Year

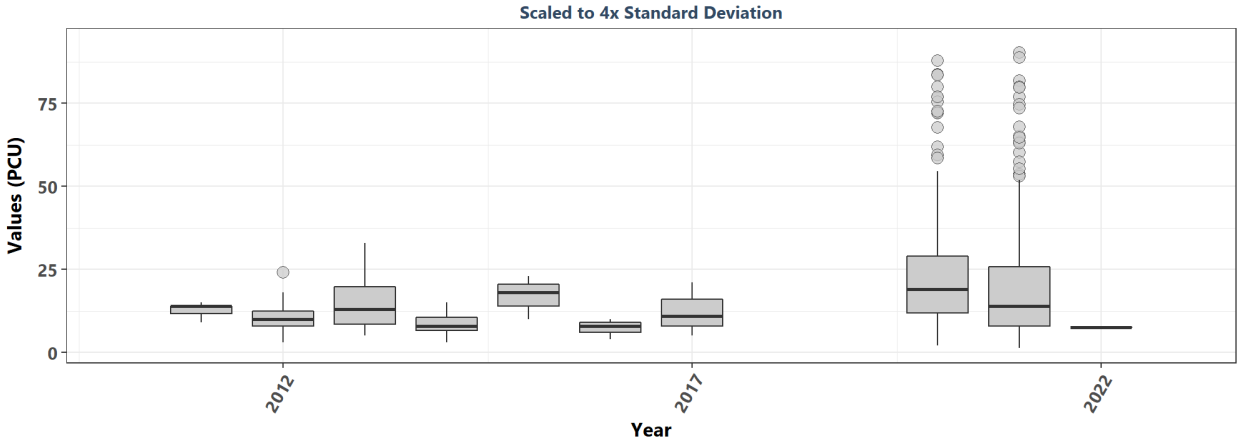


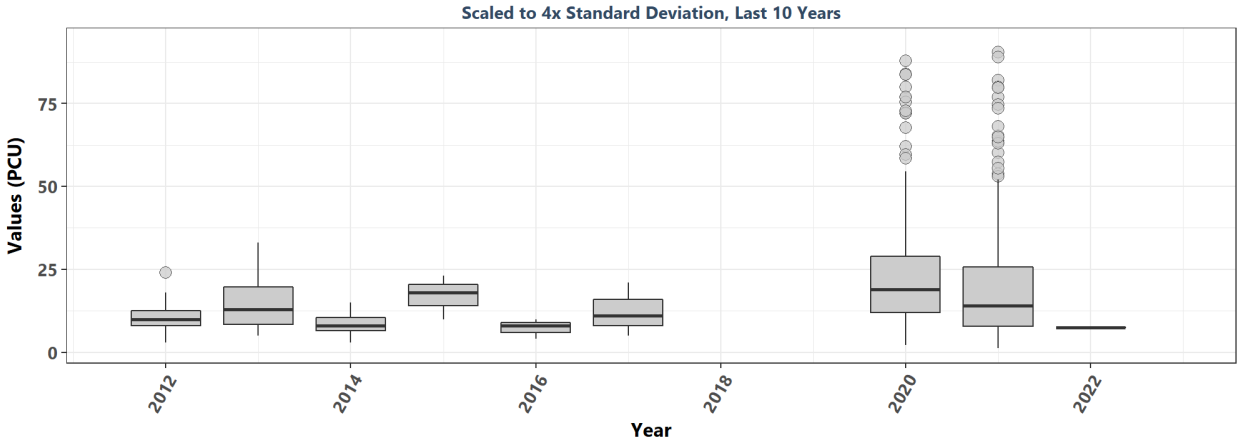

Cockroach Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year

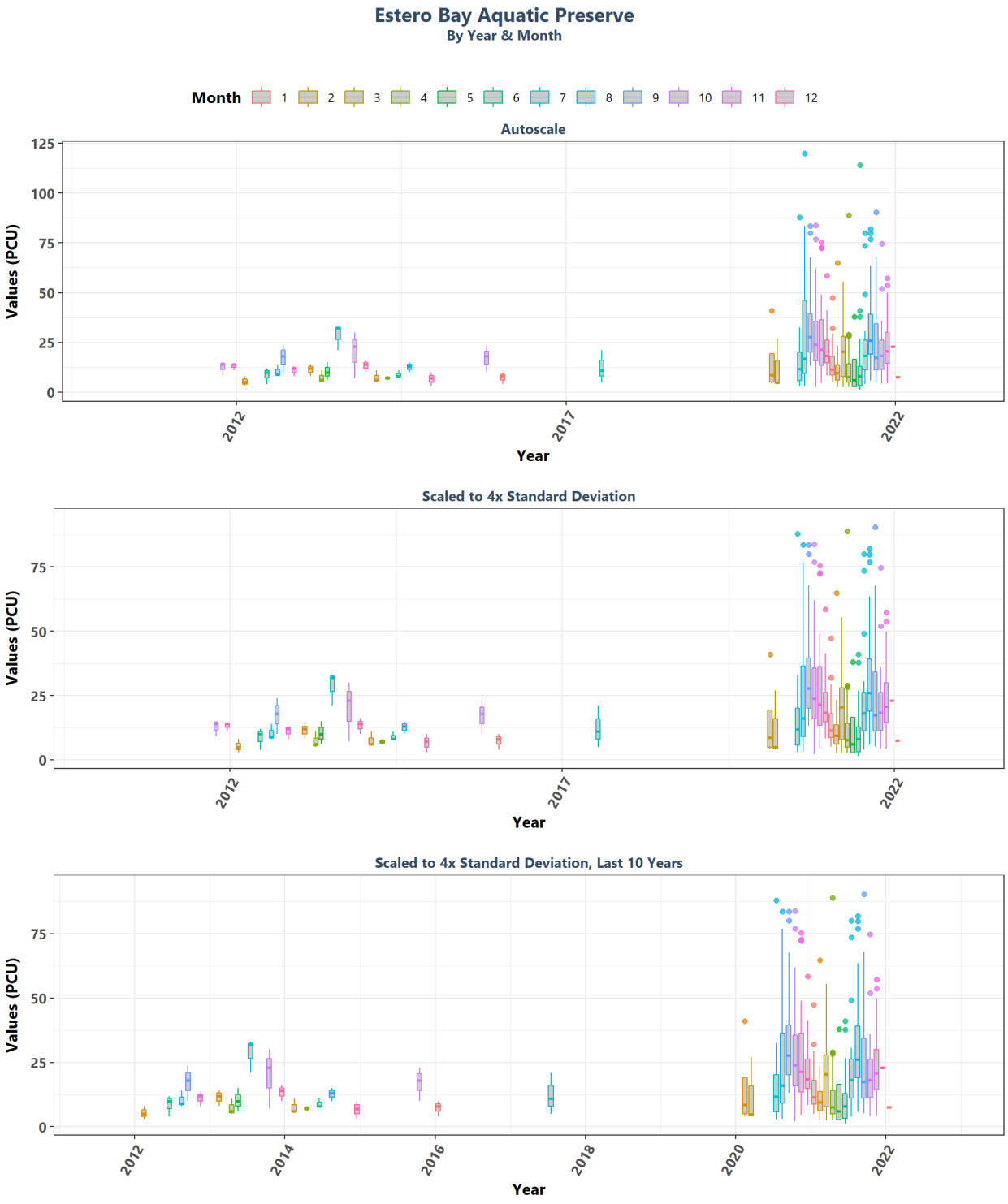

Cockroach Bay Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Month
Scaled to 4x Standard Deviation
Values (PCU)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Month

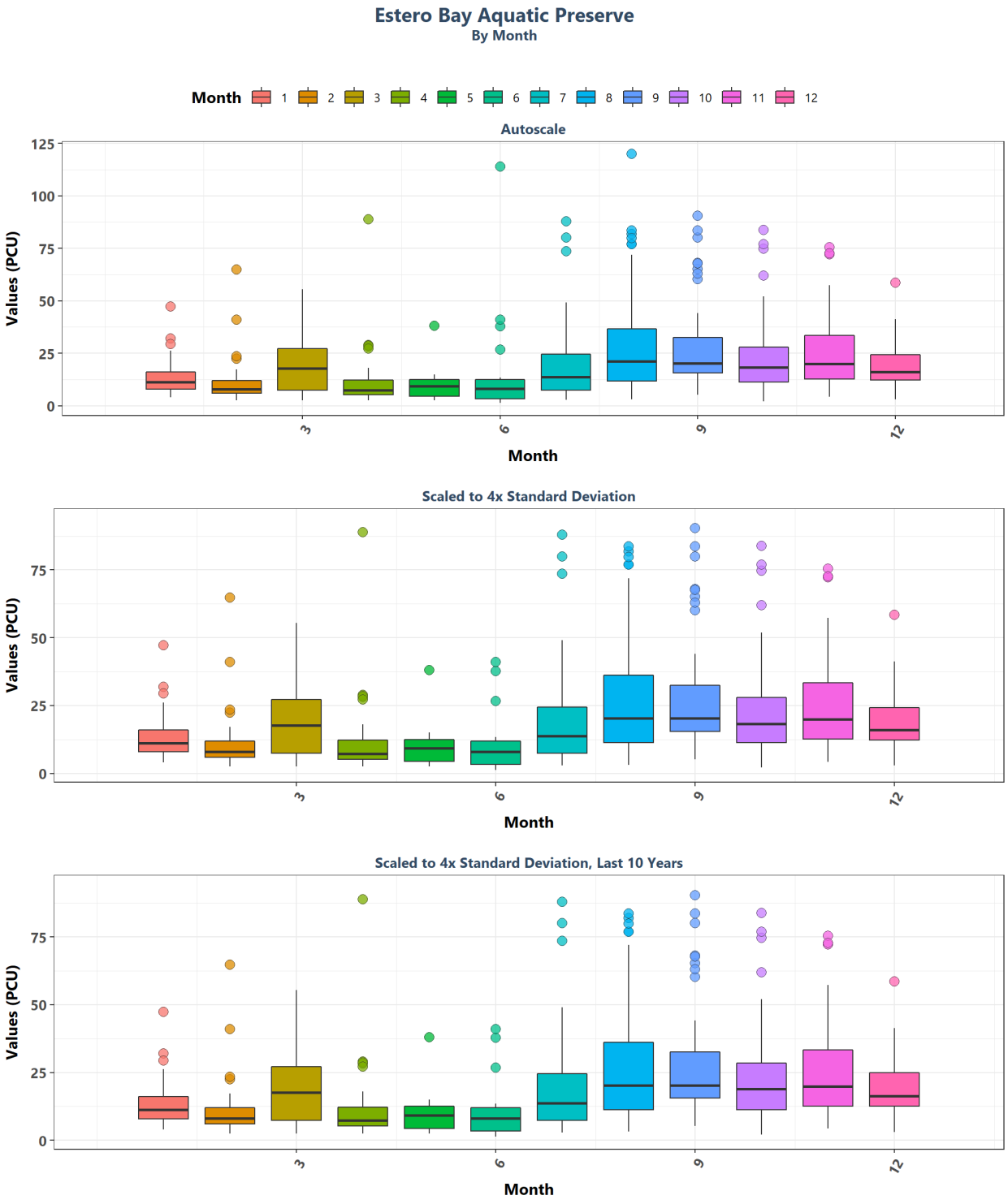
Estero Bay Aquatic Preserve
By Year





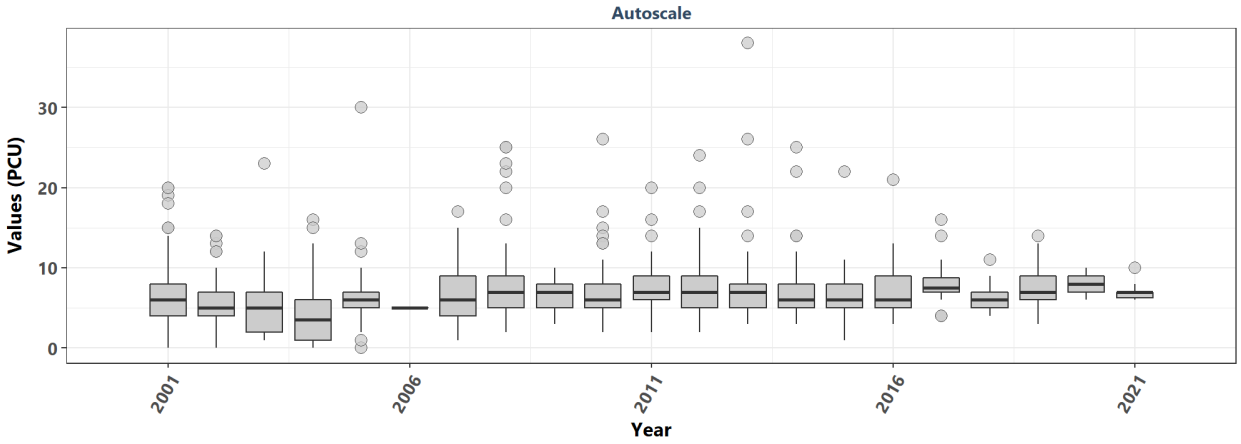


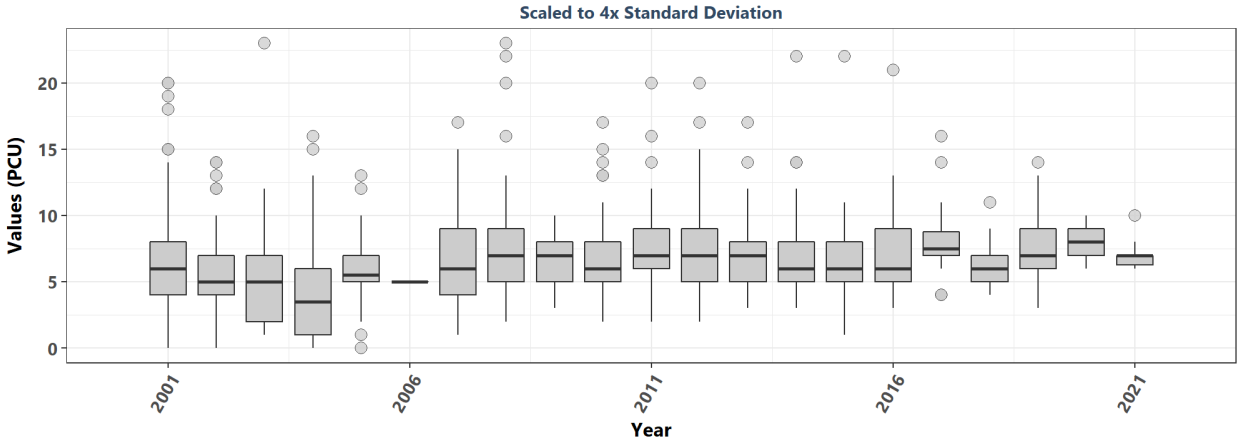

Estero Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
125
100
75
50
25
0
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (PCU)
75
50
25
0
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
75
50
25
0
2012
2014
2016
2018
2020
2022
Year

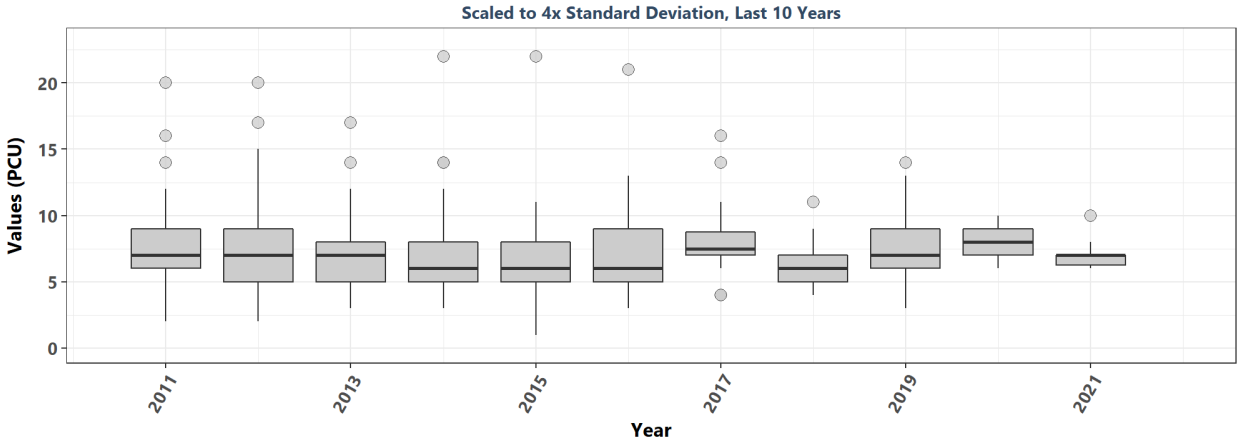

Estero Bay Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
125
100
75
50
25
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (PCU)
75
50
25
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
75
50
25
0
3
6
9
12
Month

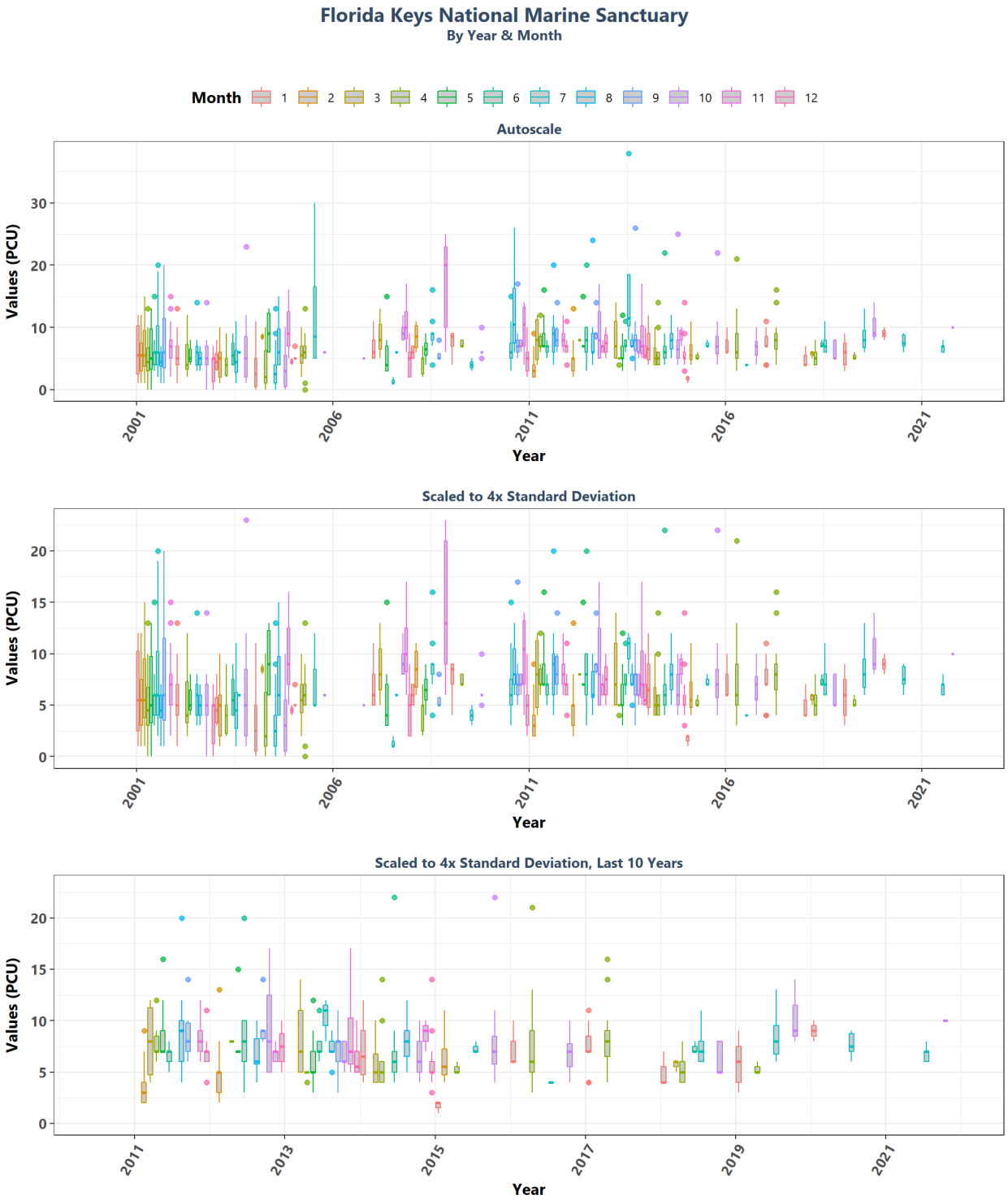
Florida Keys National Marine Sanctuary

By Year



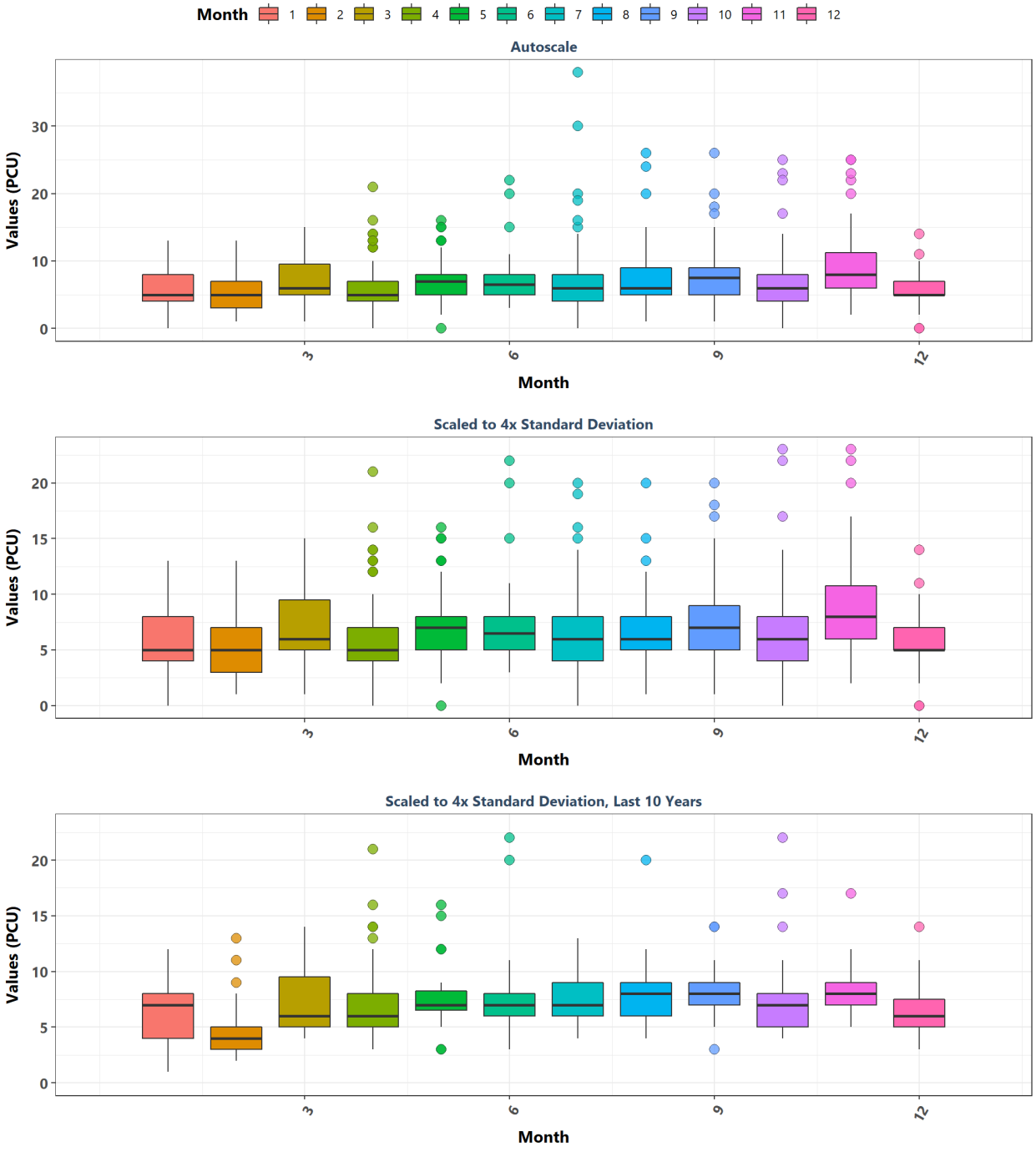





Florida Keys National Marine Sanctuary
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year

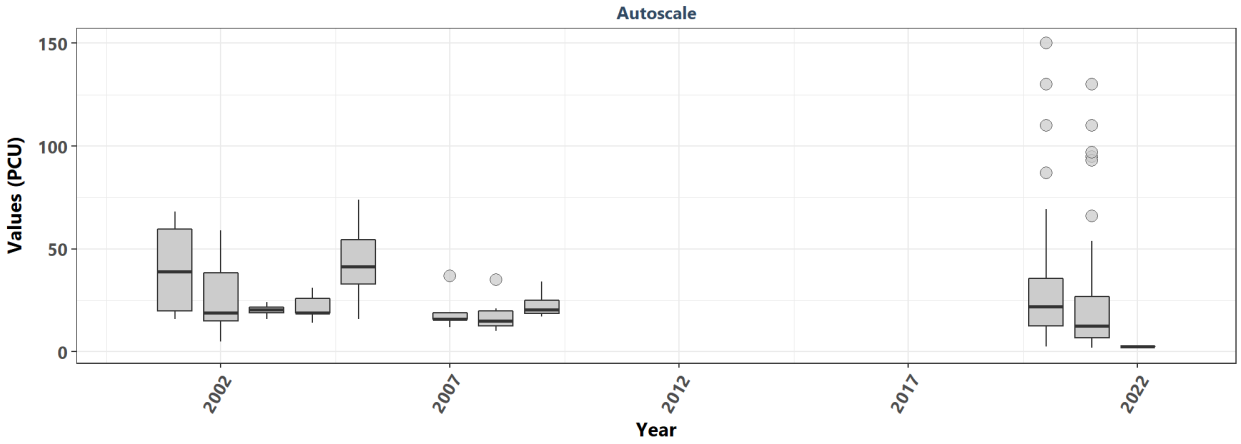
Florida Keys National Marine Sanctuary

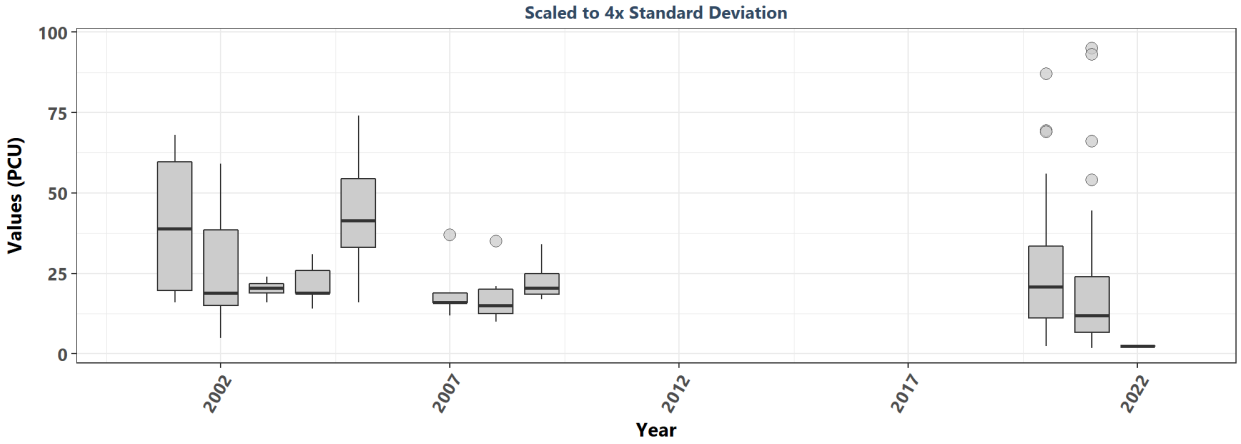
By Month

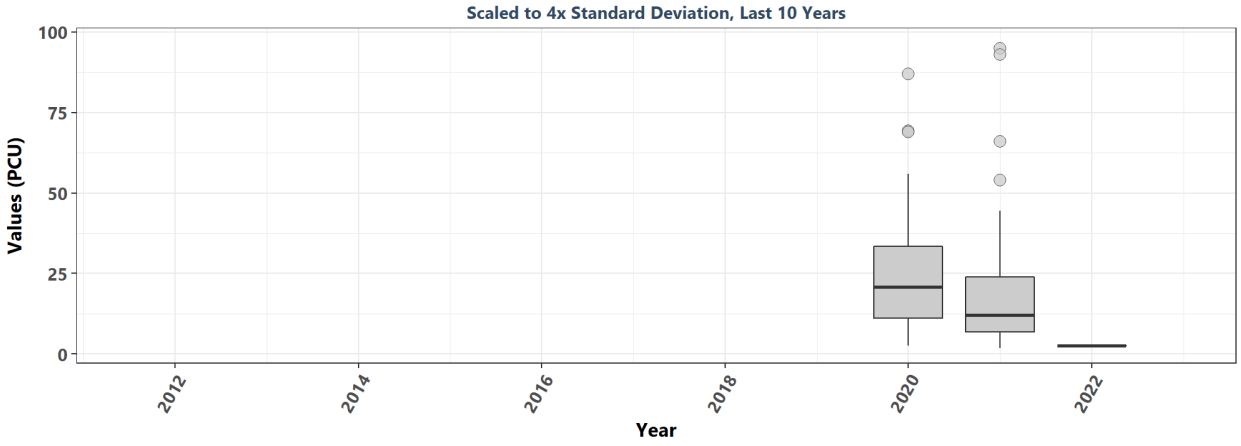


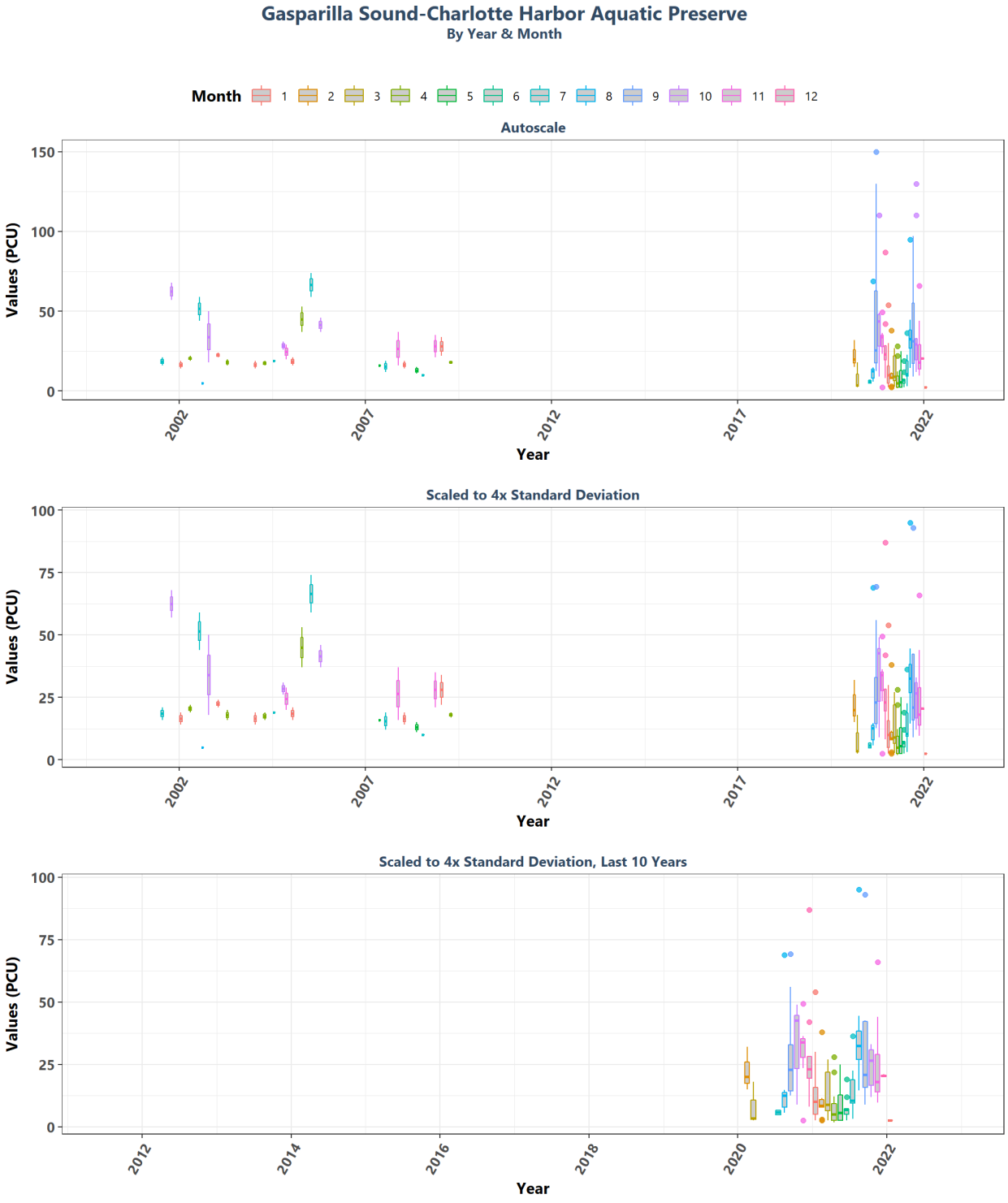
Gasparilla Sound-Charlotte Harbor Aquatic Preserve

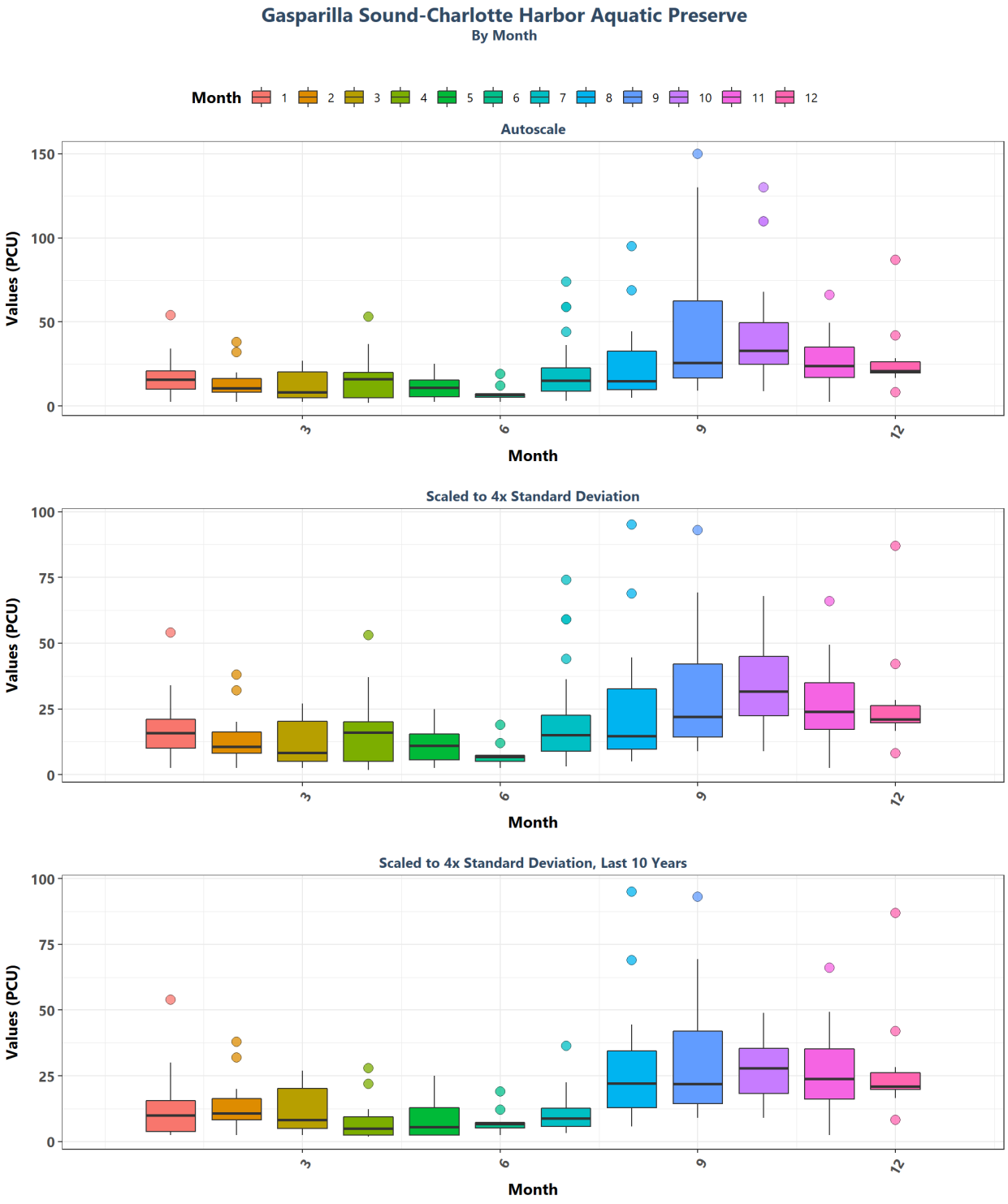
By Year





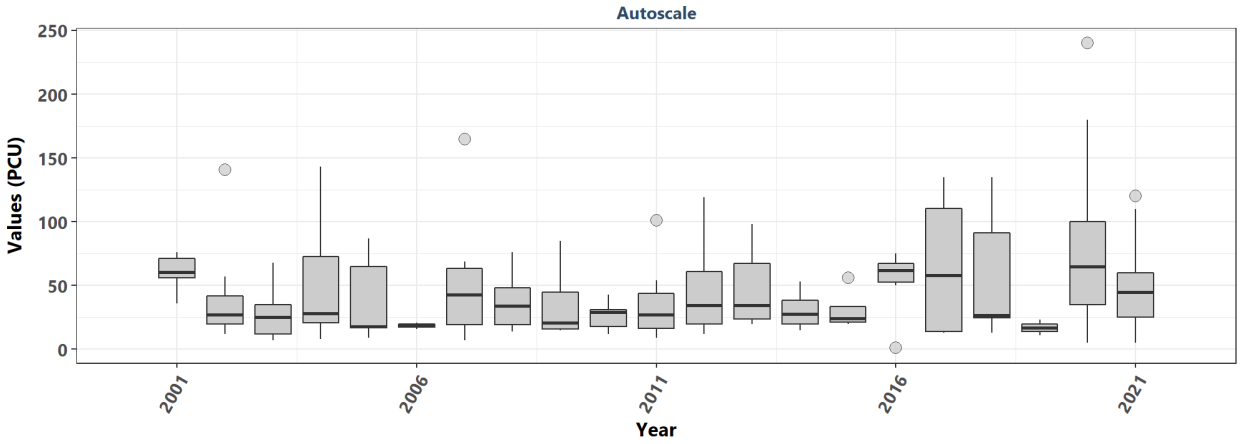


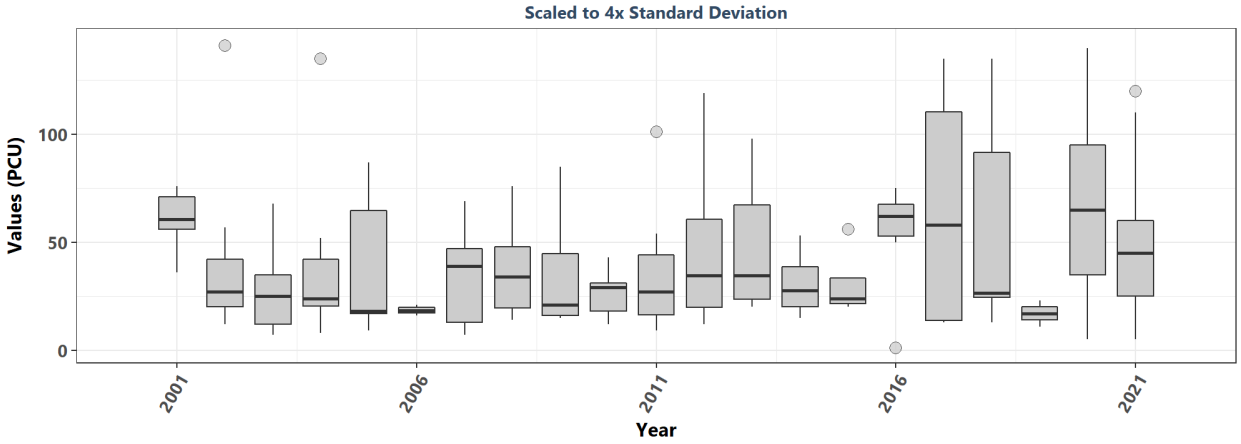

Gasparilla Sound-Charlotte Harbor Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
150
100
50
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (PCU)
100
75
50
25
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
100
75
50
25
0
2012
2014
2016
2018
2020
2022
Year

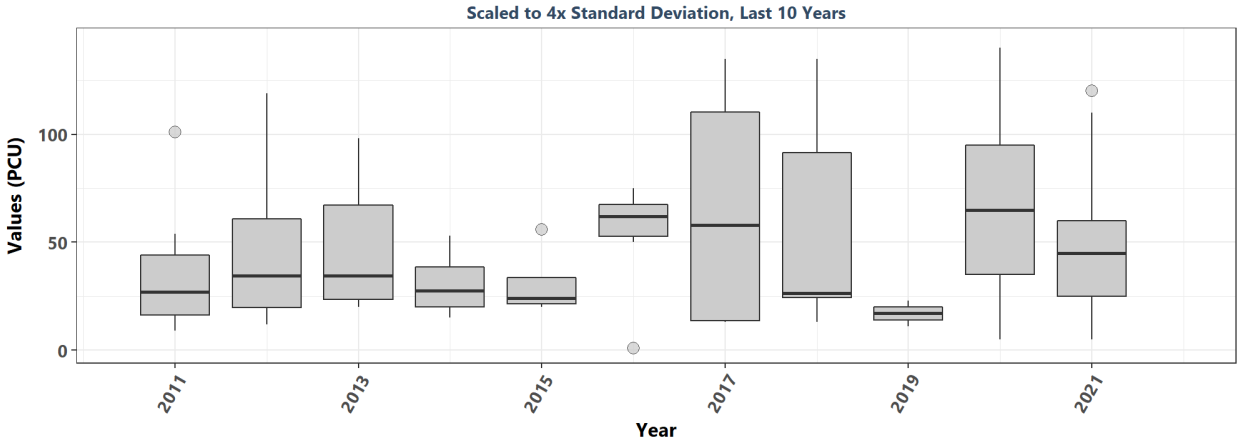

Gasparilla Sound-Charlotte Harbor Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (PCU)
100
75
50
25
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
100
75
50
25
0
3
6
9
12
Month

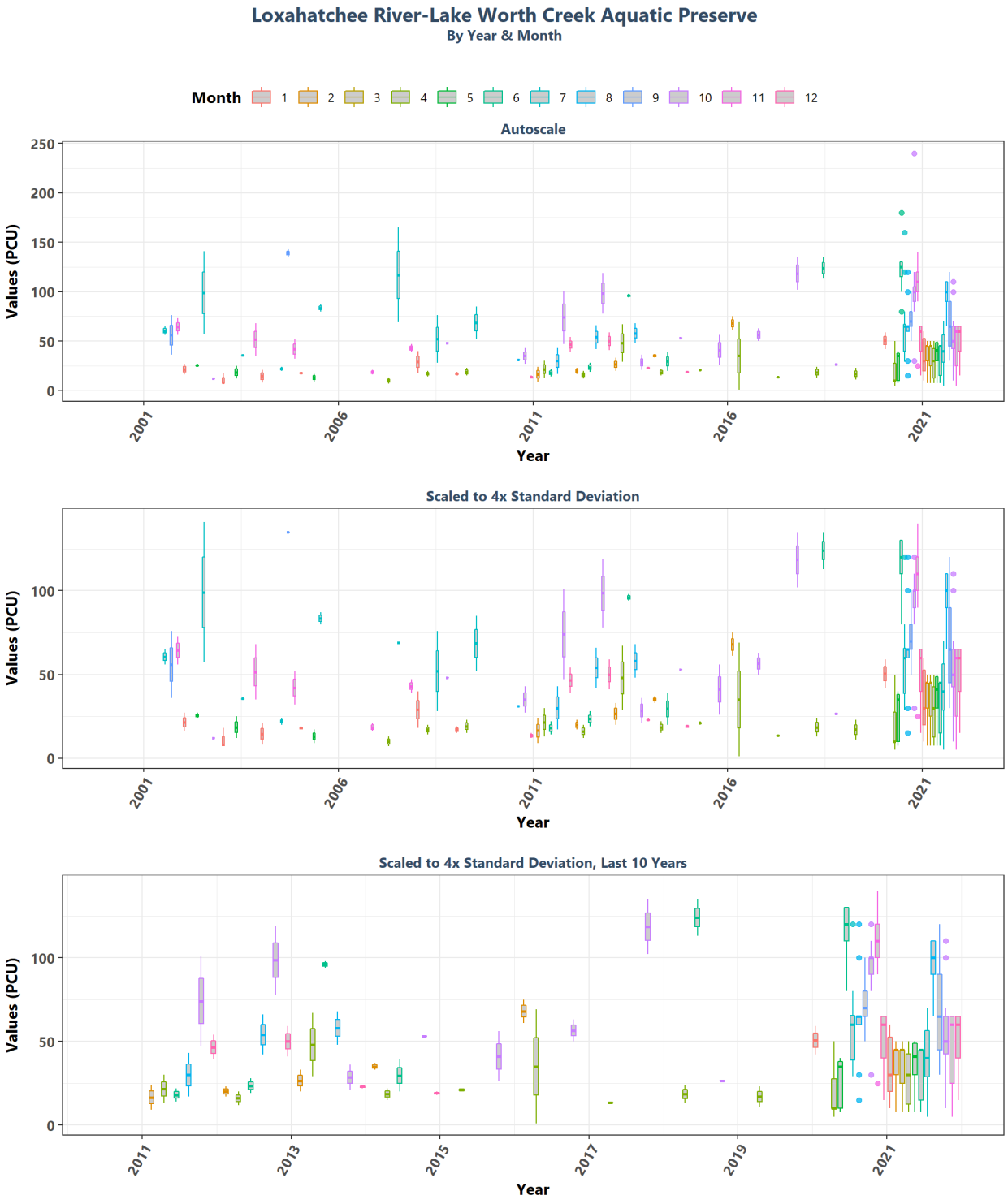
Loxahatchee River-Lake Worth Creek Aquatic Preserve

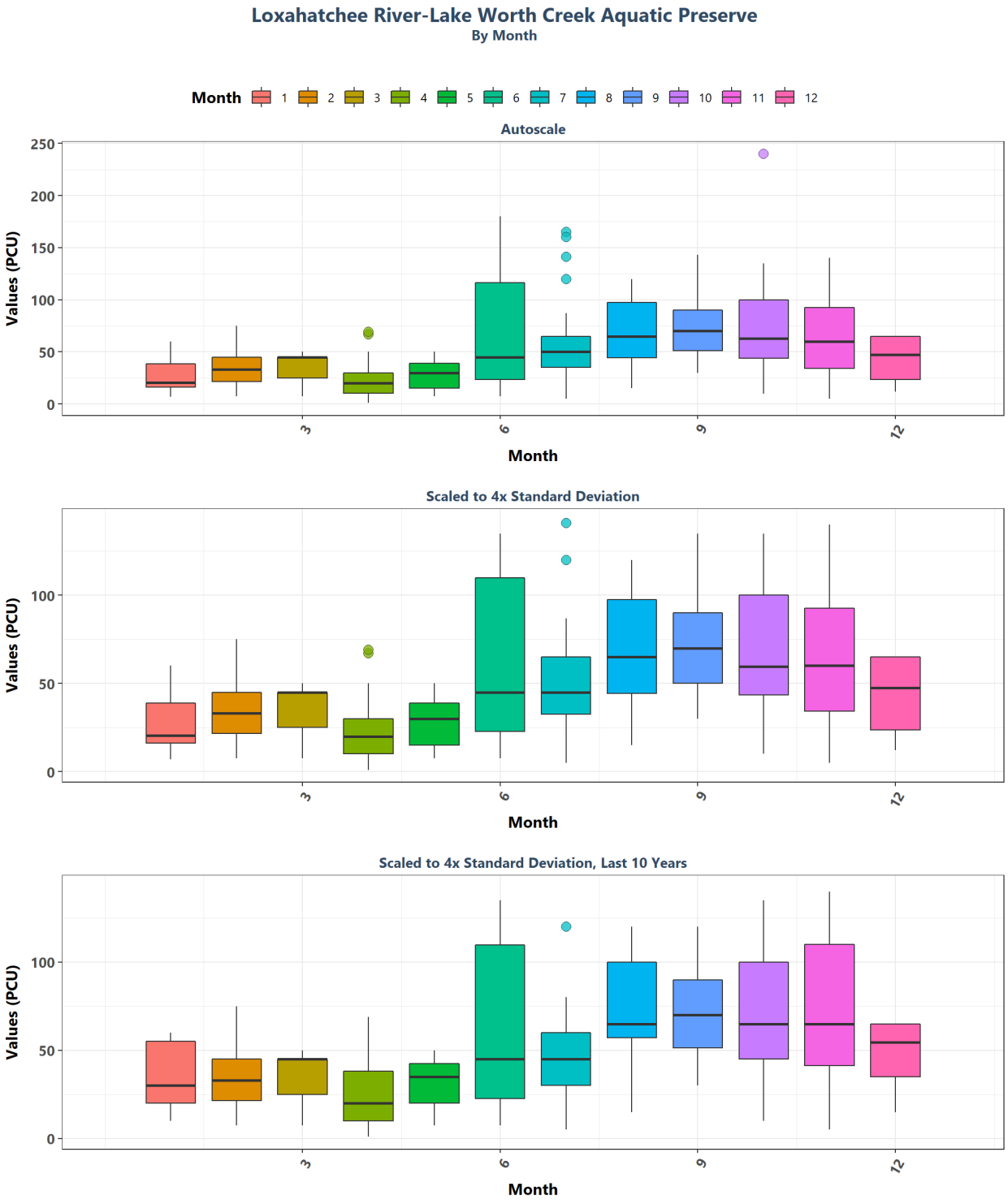
By Year





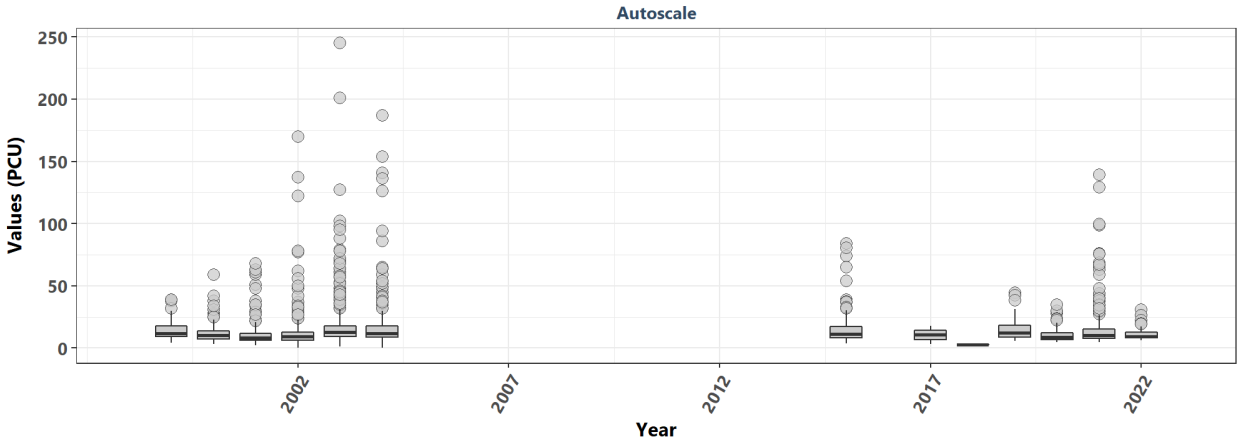


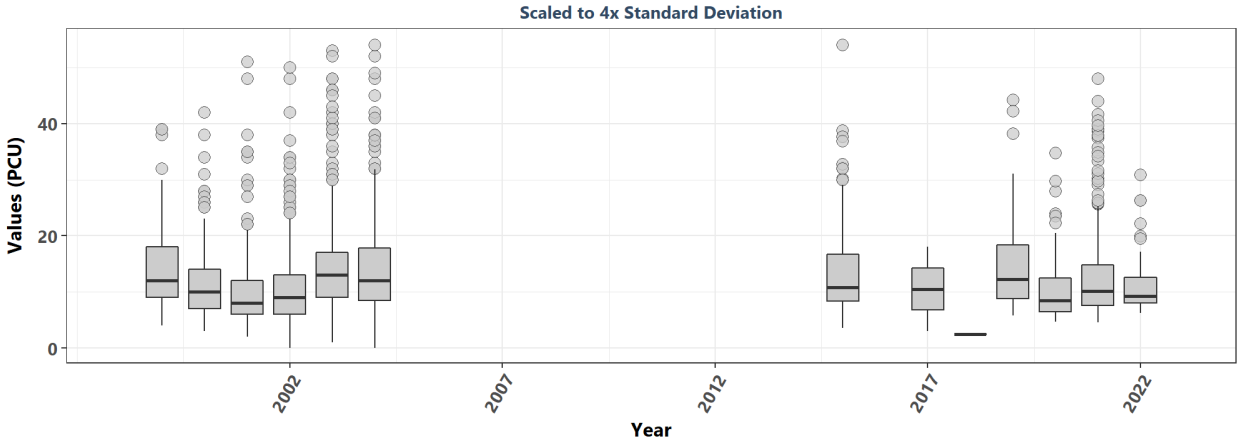

Loxahatchee River-Lake Worth Creek Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
250
200
150
100
50
0
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (PCU)
100
50
0
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
100
50
0
2011
2013
2015
2017
2019
2021
Year

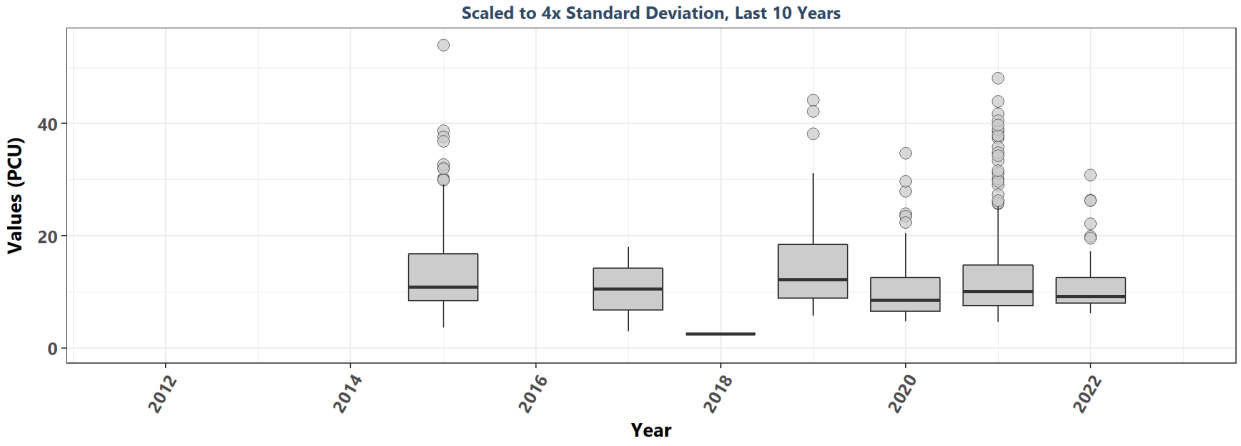

Loxahatchee River-Lake Worth Creek Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
250
200
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (PCU)
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
100
50
0
3
6
9
12
Month

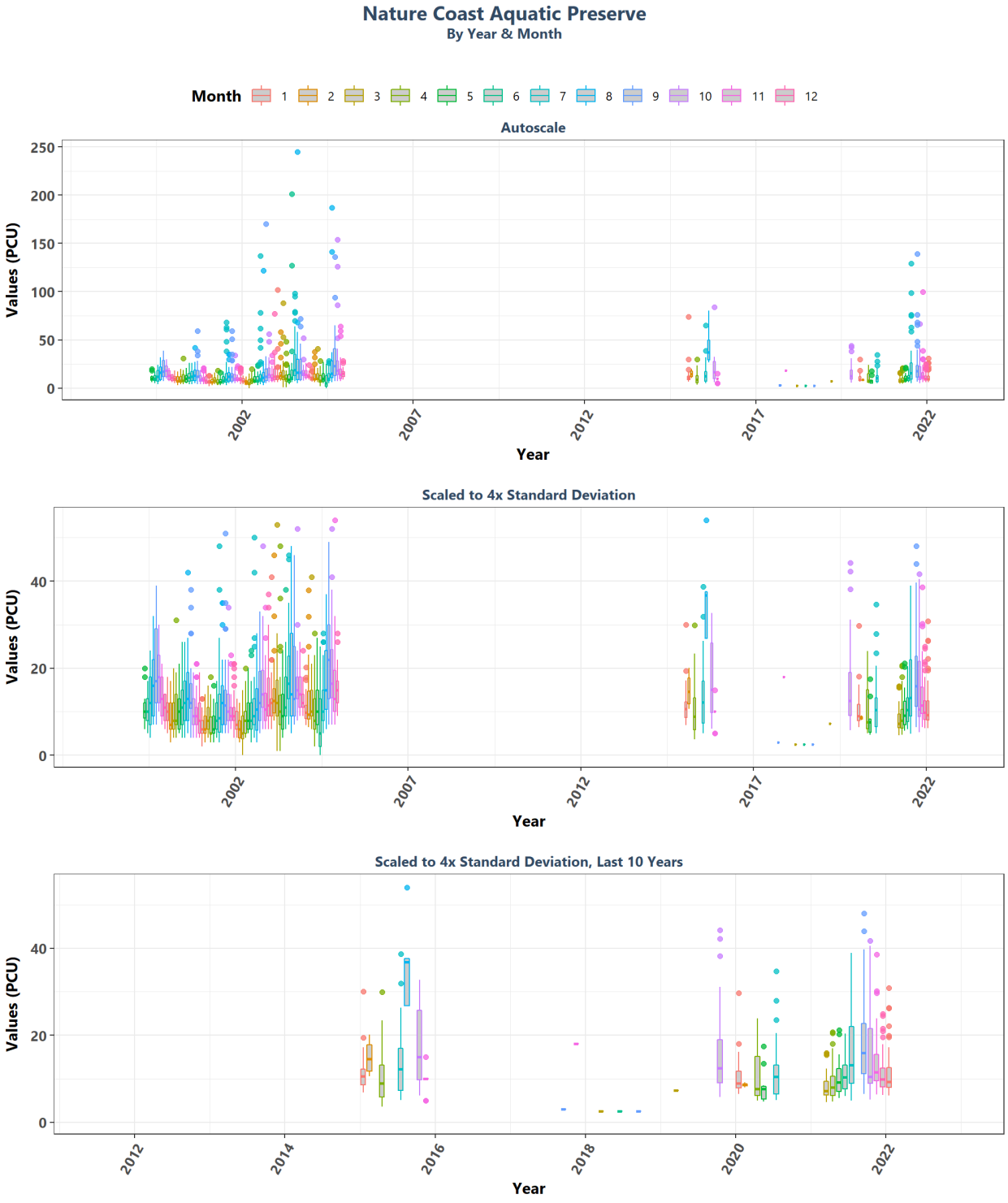
Nature Coast Aquatic Preserve

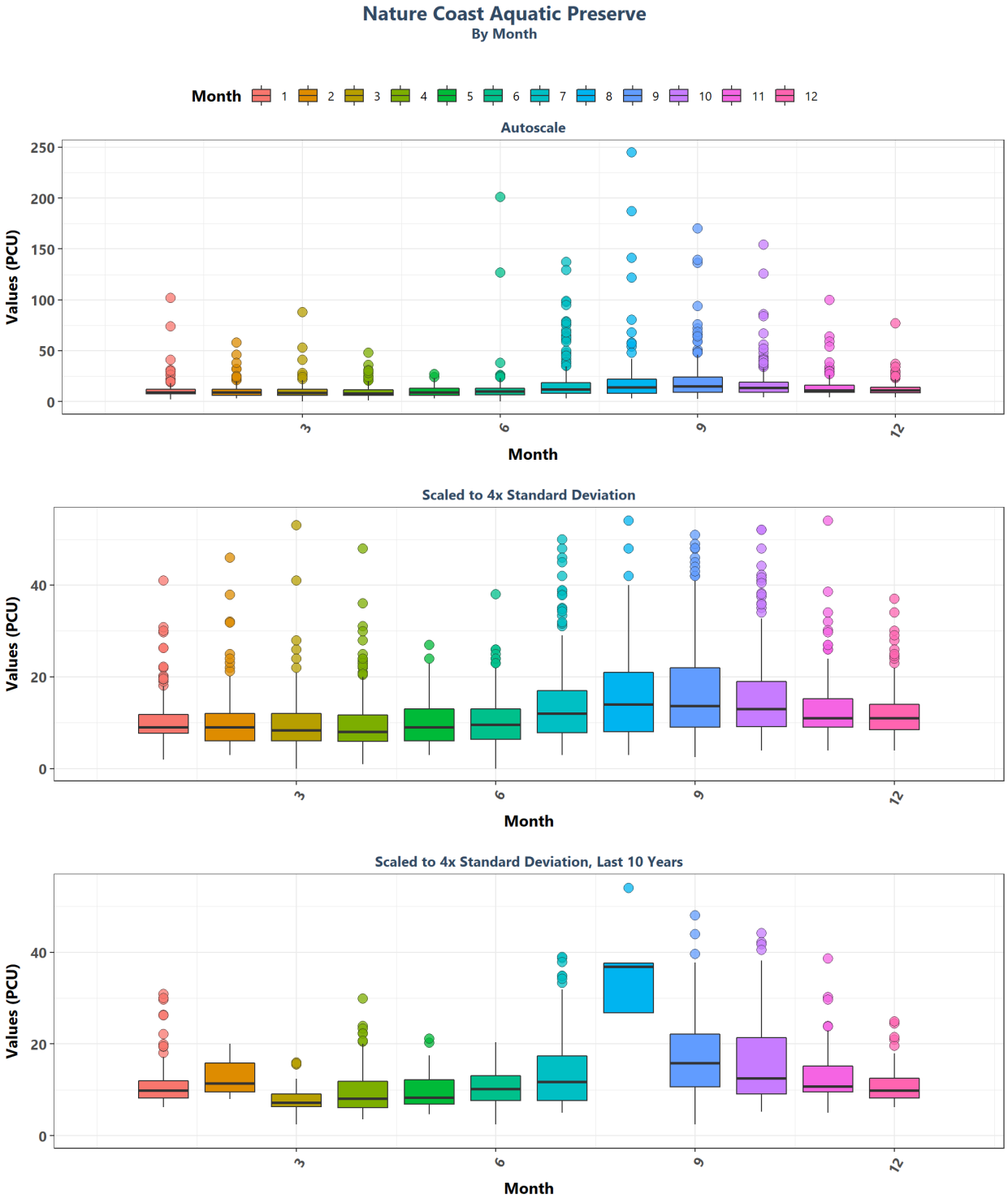
By Year





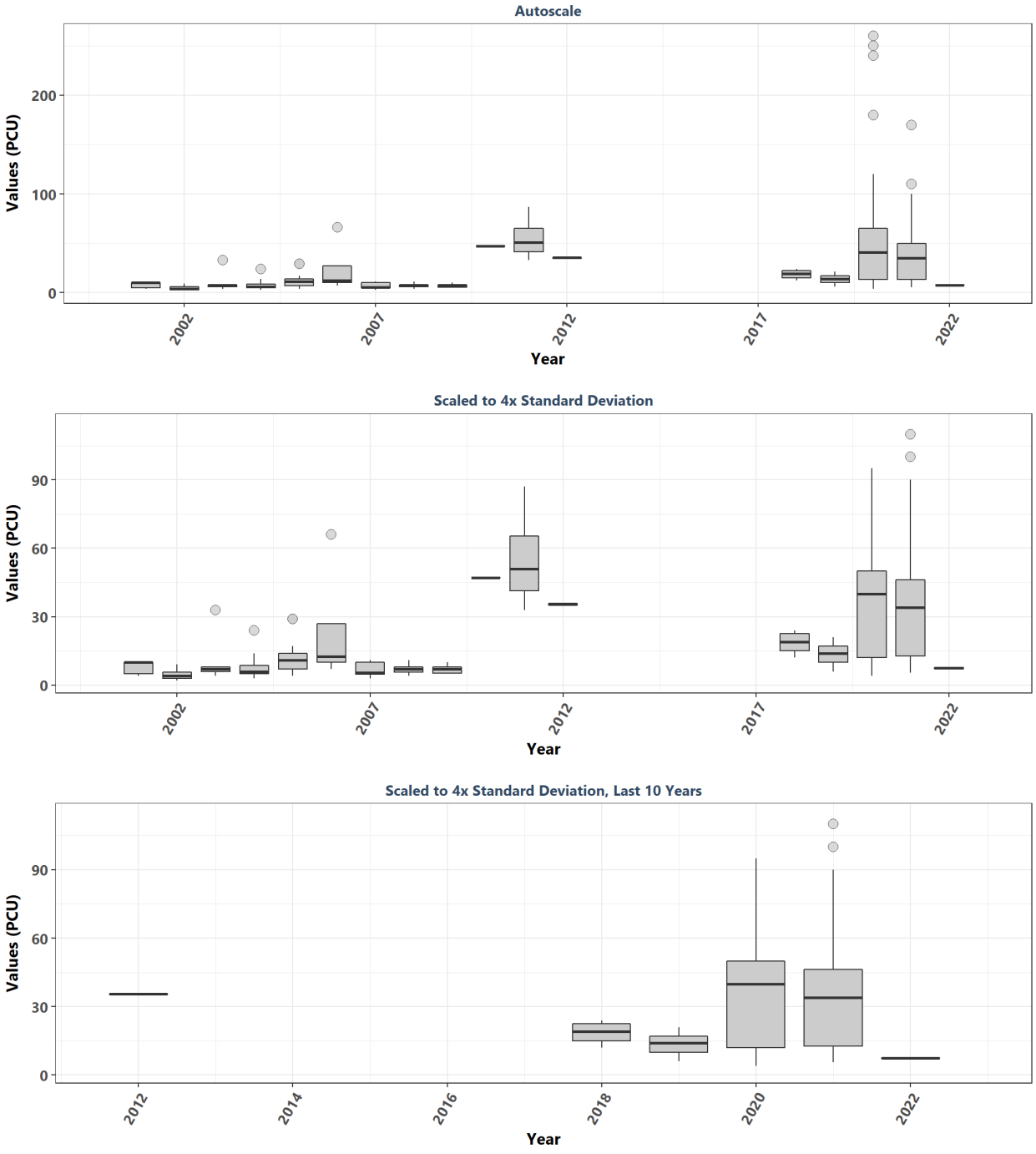


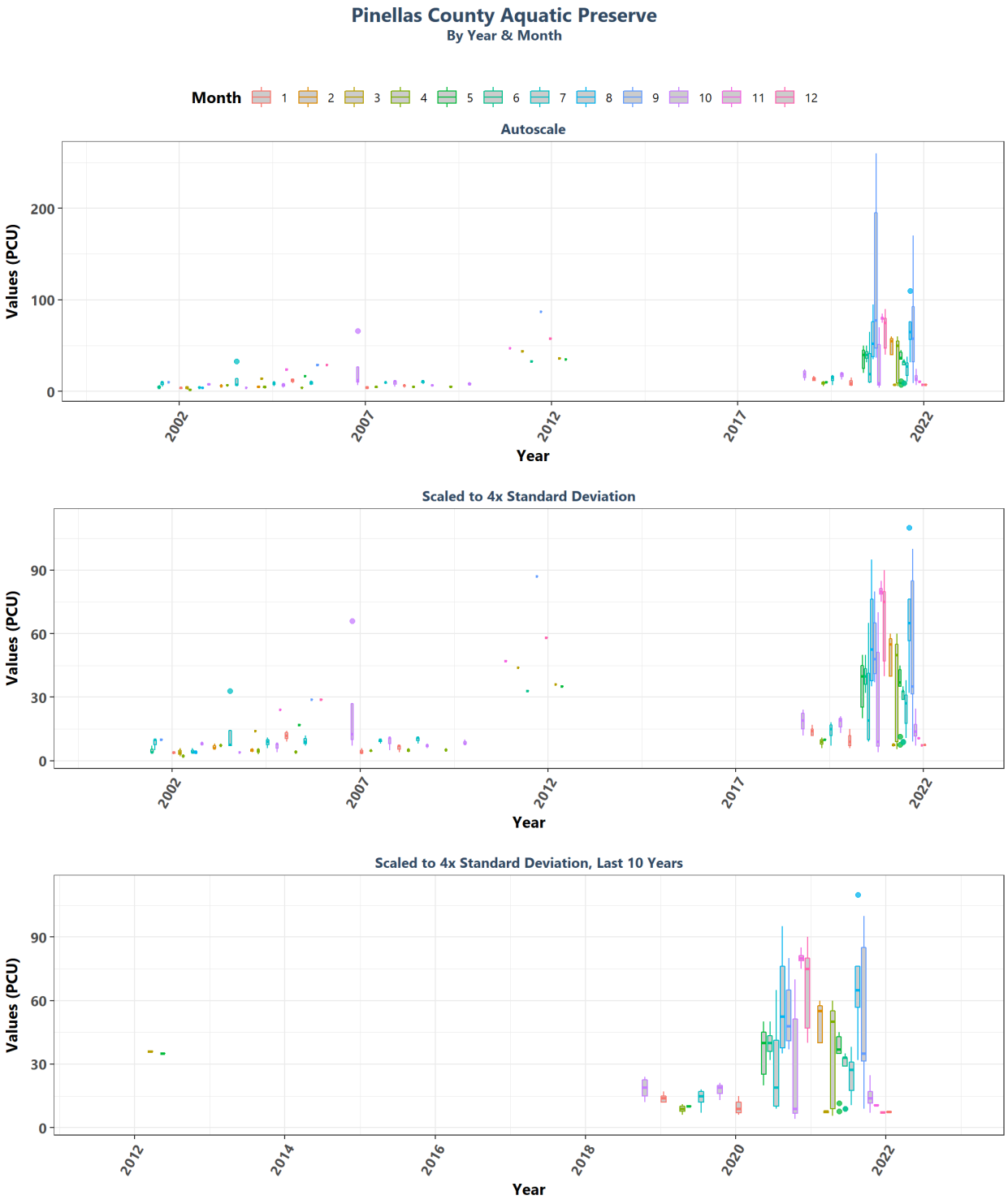

Nature Coast Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
250
200
150
100
50
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (PCU)
40
20
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
40
20
0
2012
2014
2016
2018
2020
2022
Year

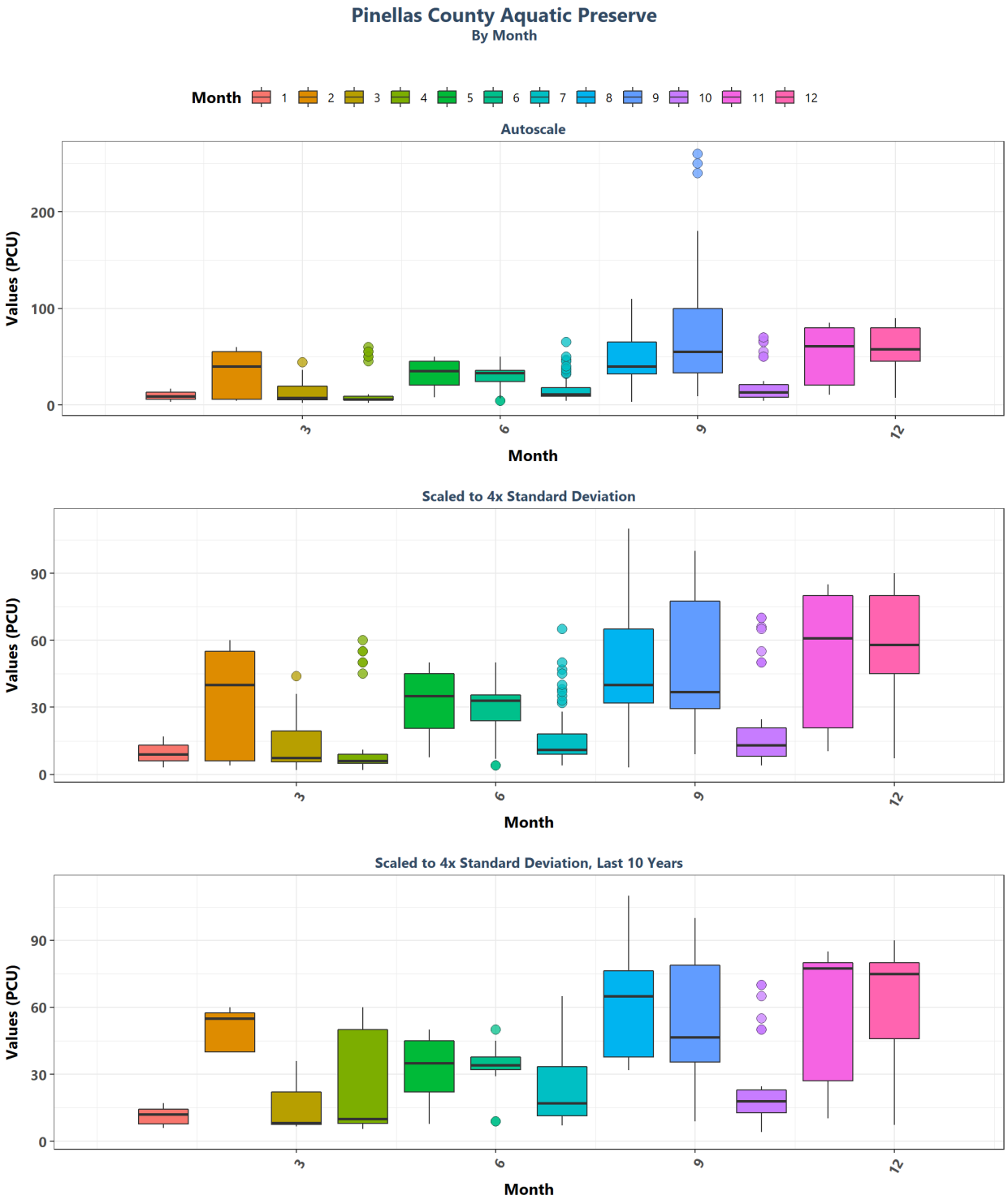

Nature Coast Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
250
200
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (PCU)
40
20
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
40
20
0
3
6
9
12
Month

Pinellas County Aquatic Preserve

By Year

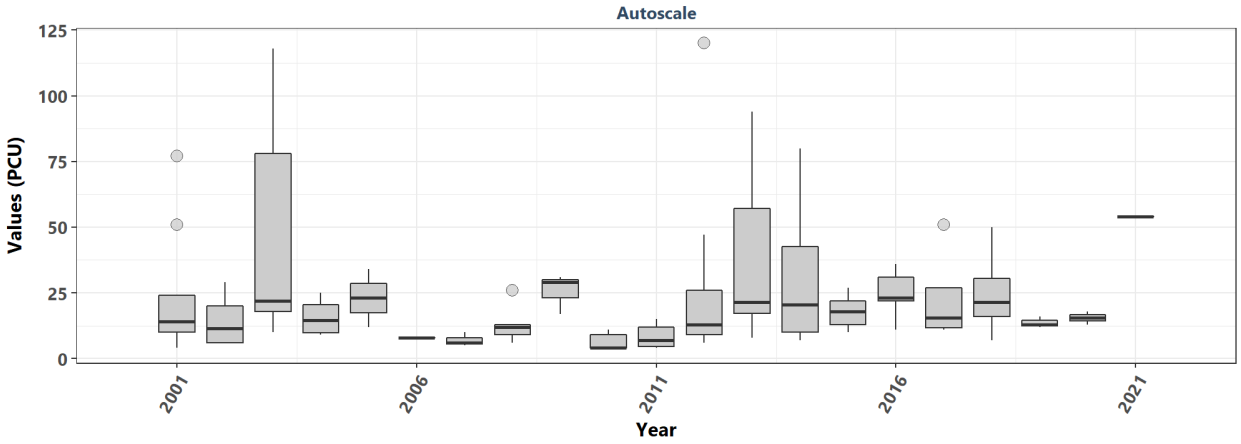


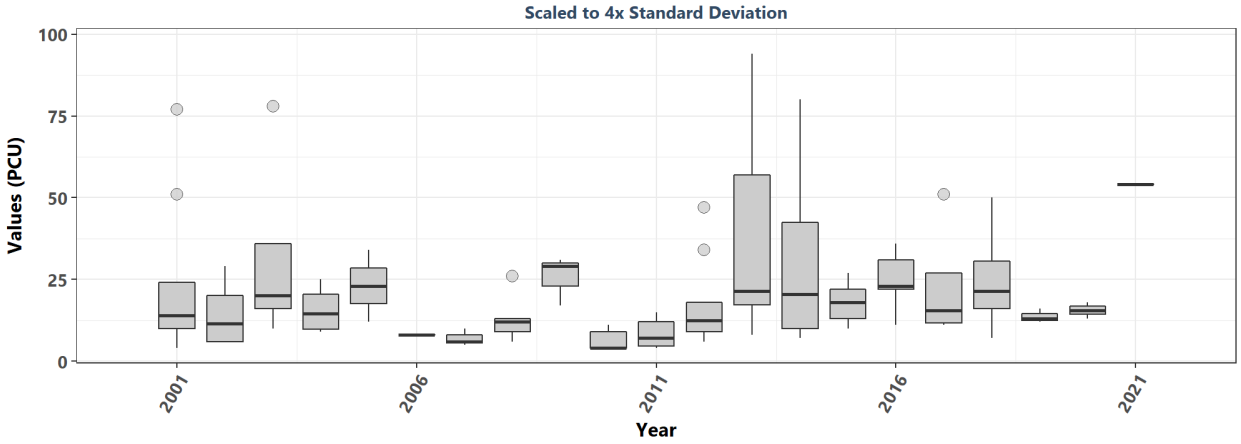

Pinellas County Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
200
100
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (PCU)
90
60
30
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
90
60
30
0
2012
2014
2016
2018
2020
2022
Year

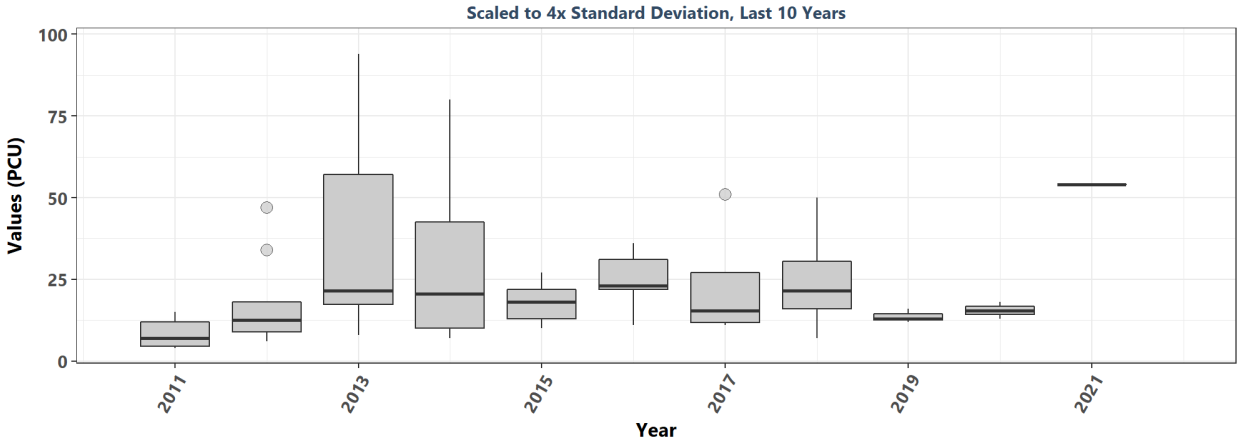

Pinellas County Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
200
100
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (PCU)
90
60
30
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
90
60
30
0
3
6
9
12
Month

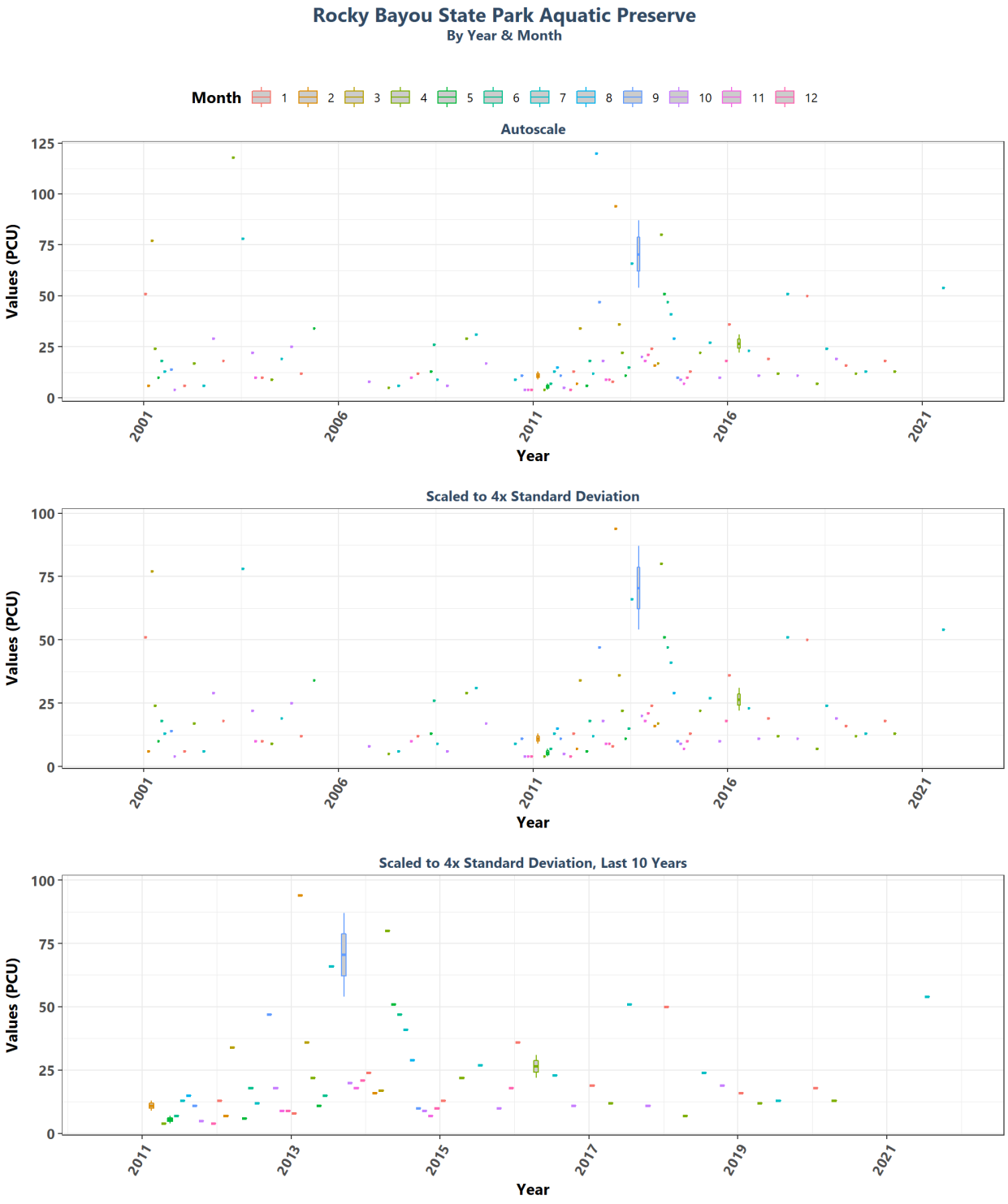
Rocky Bayou State Park Aquatic Preserve

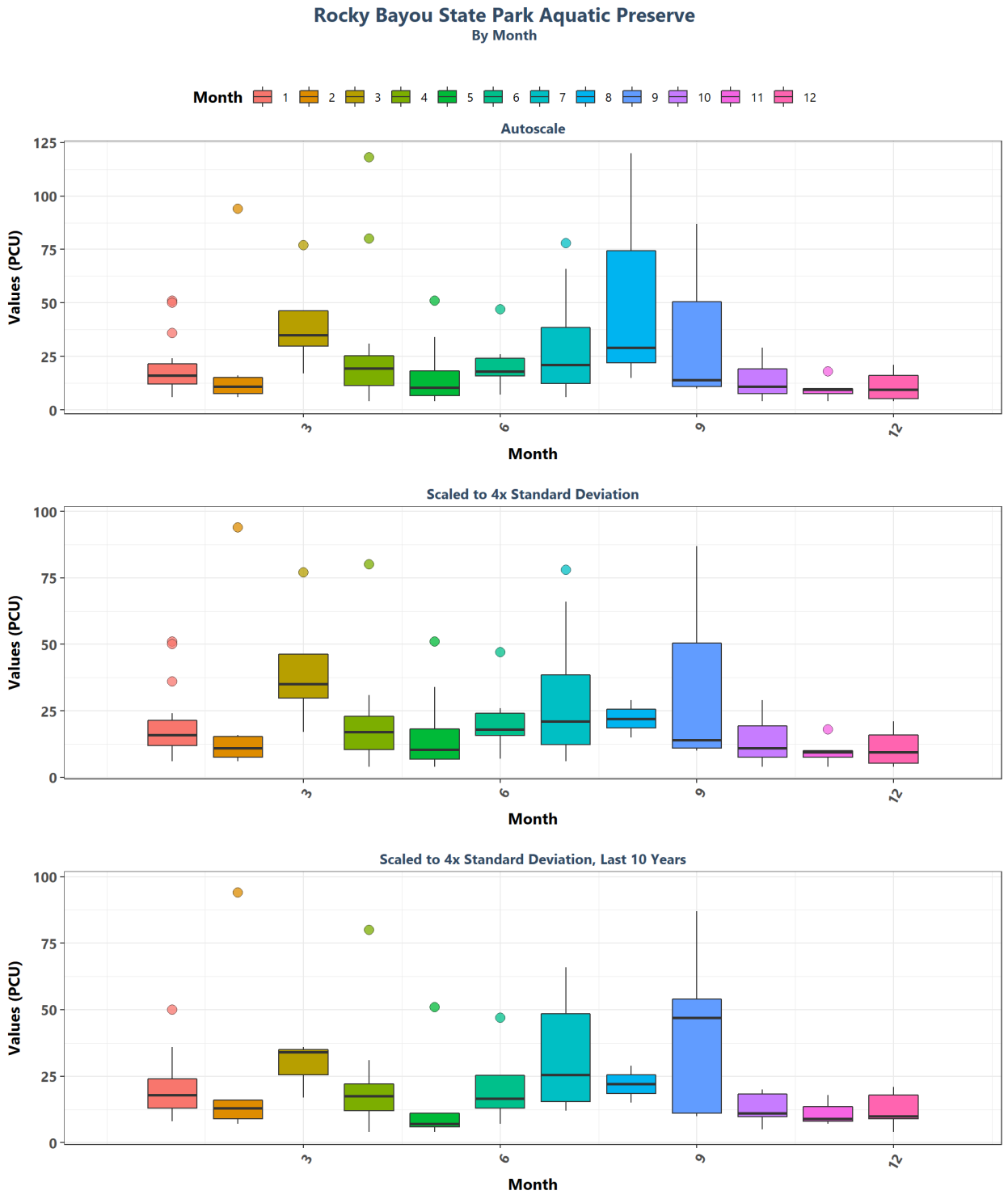
By Year



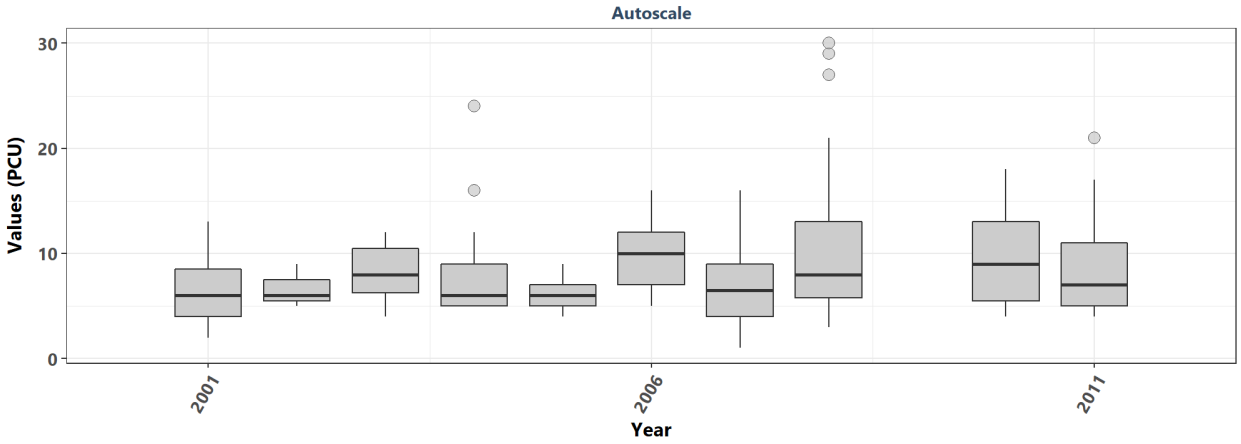


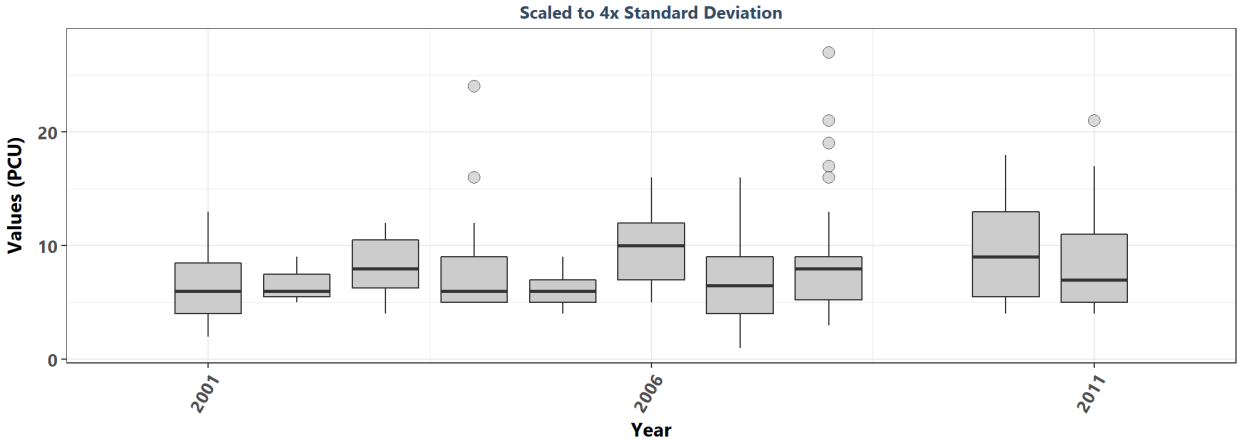


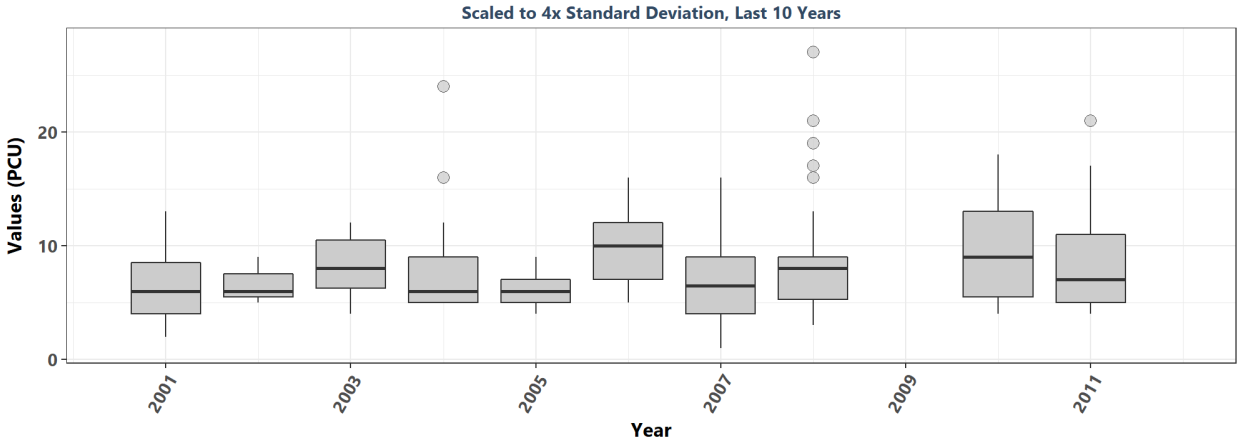

Rocky Bayou State Park Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
125
100
75
50
25
0
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (PCU)
100
75
50
25
0
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
100
75
50
25
0
2011
2013
2015
2017
2019
2021
Year

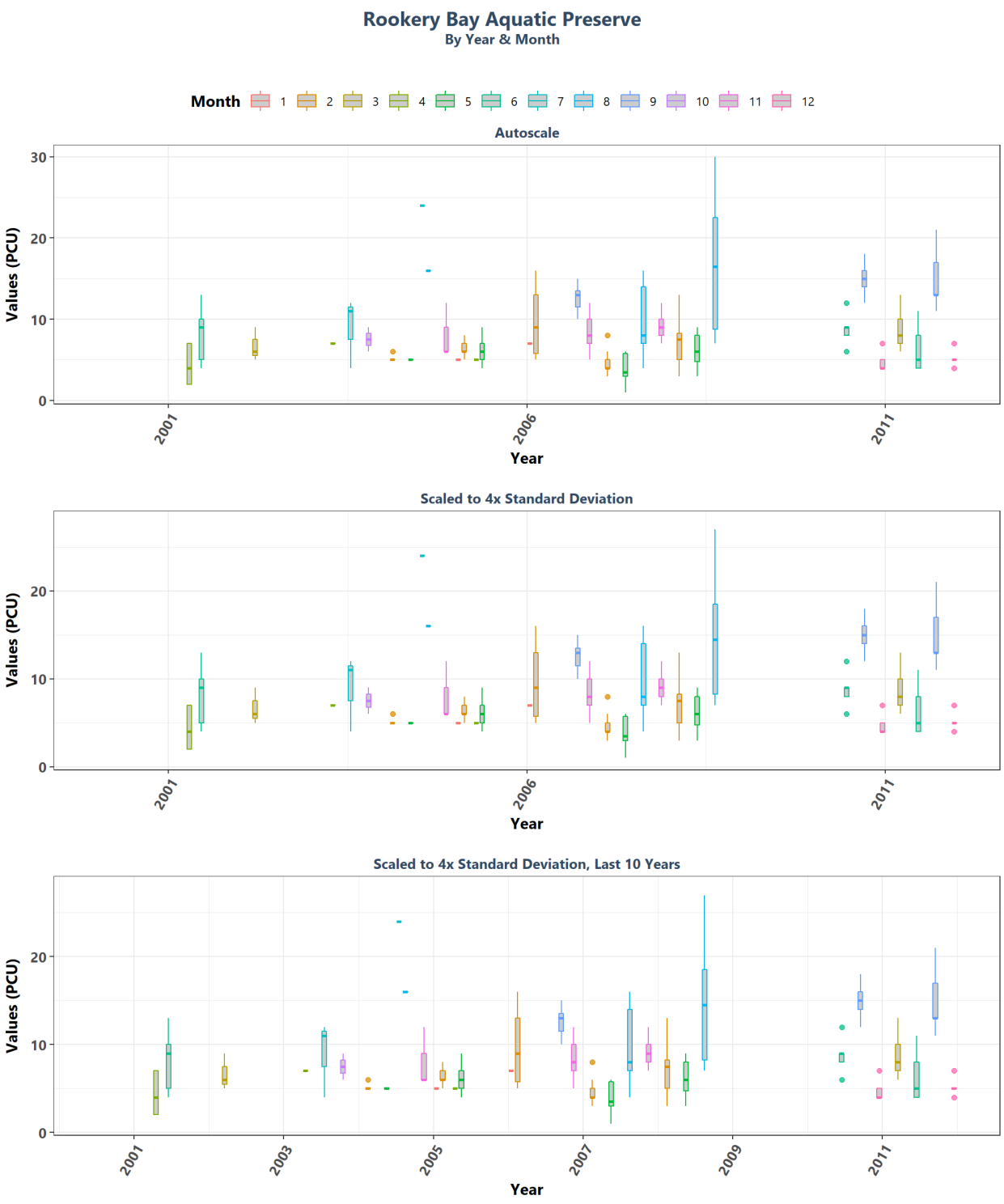

Rocky Bayou State Park Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
125
100
75
50
25
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (PCU)
100
75
50
25
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
100
75
50
25
0
3
6
9
12
Month

Rookery Bay Aquatic Preserve
By Year

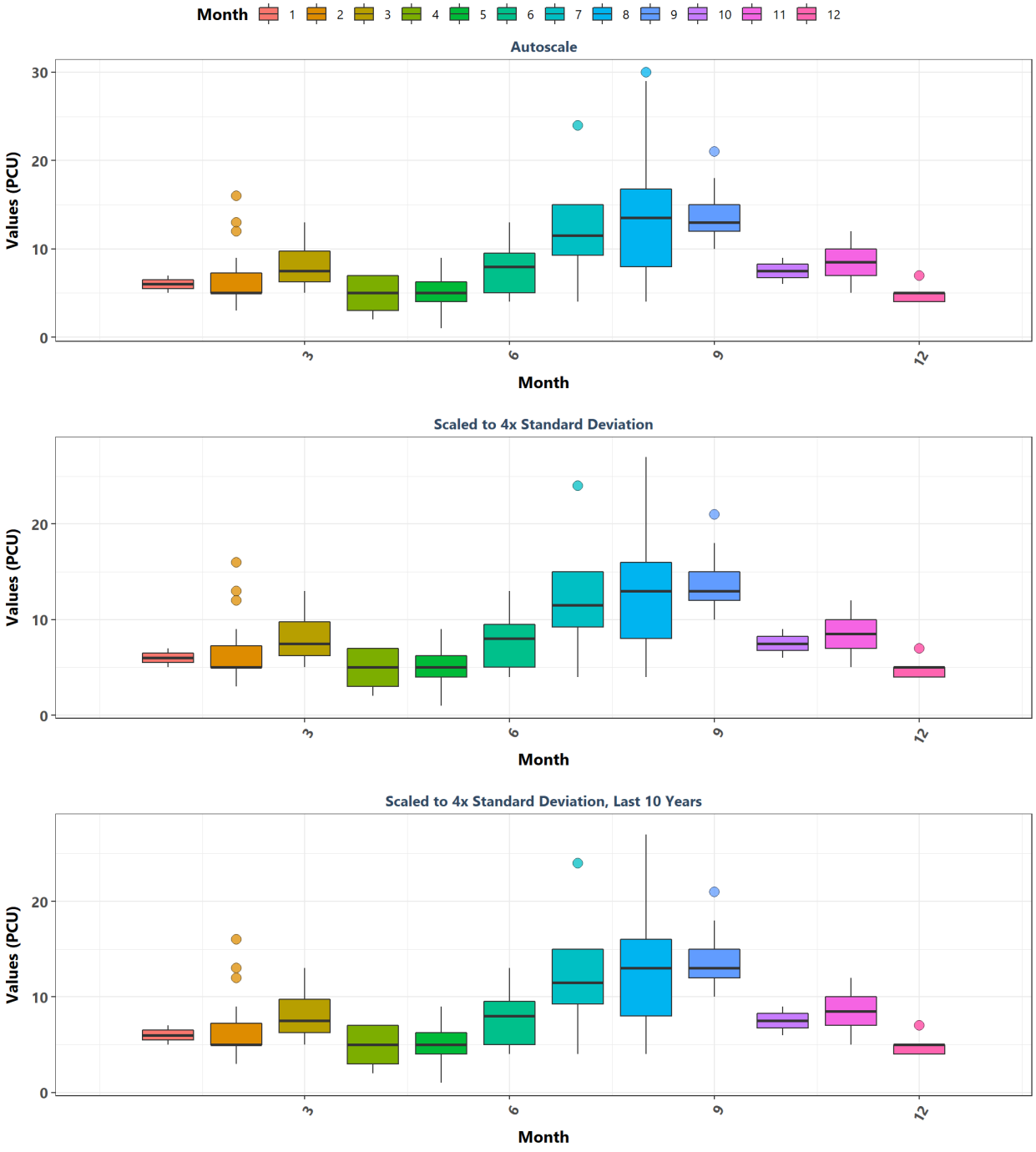


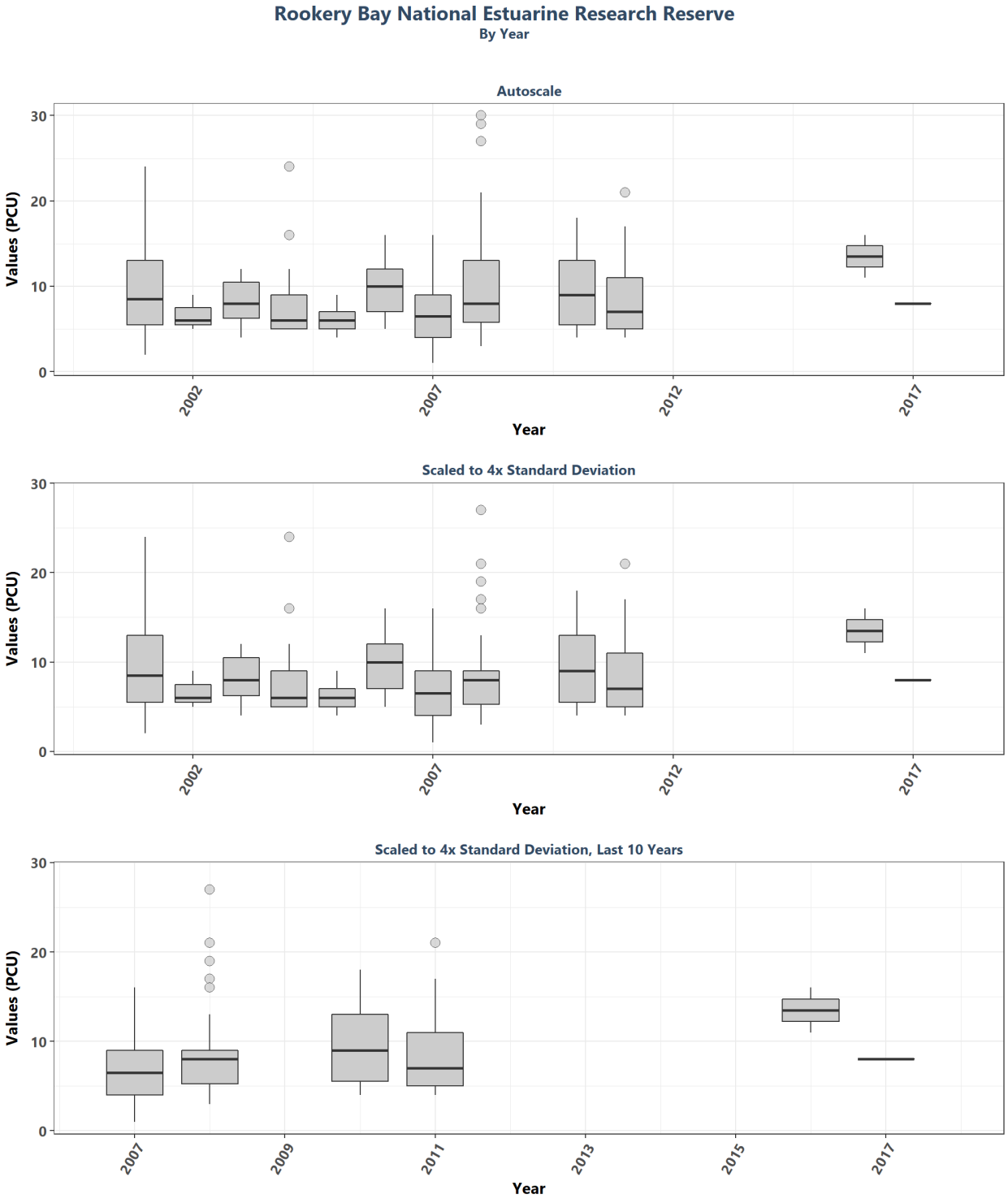


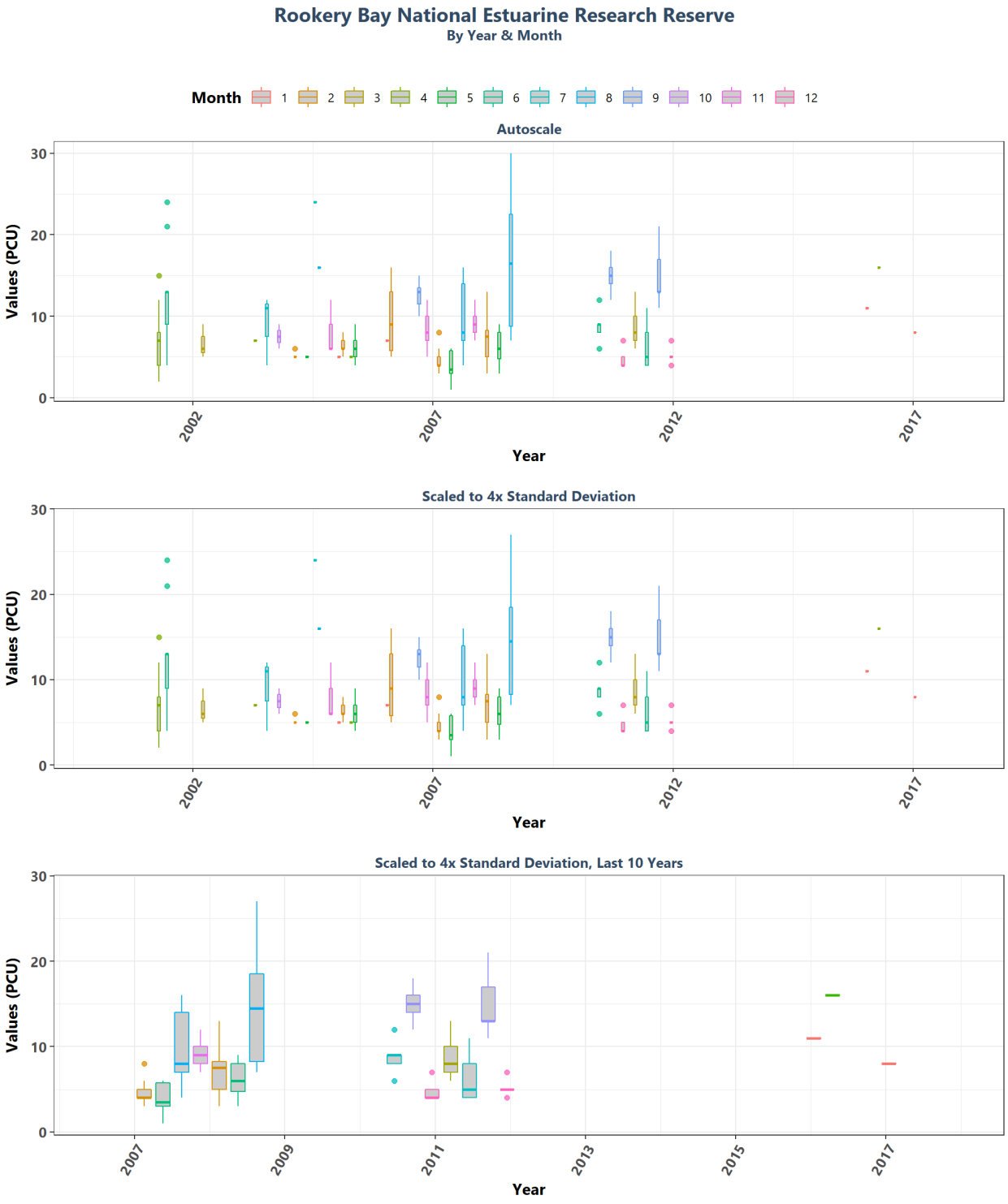


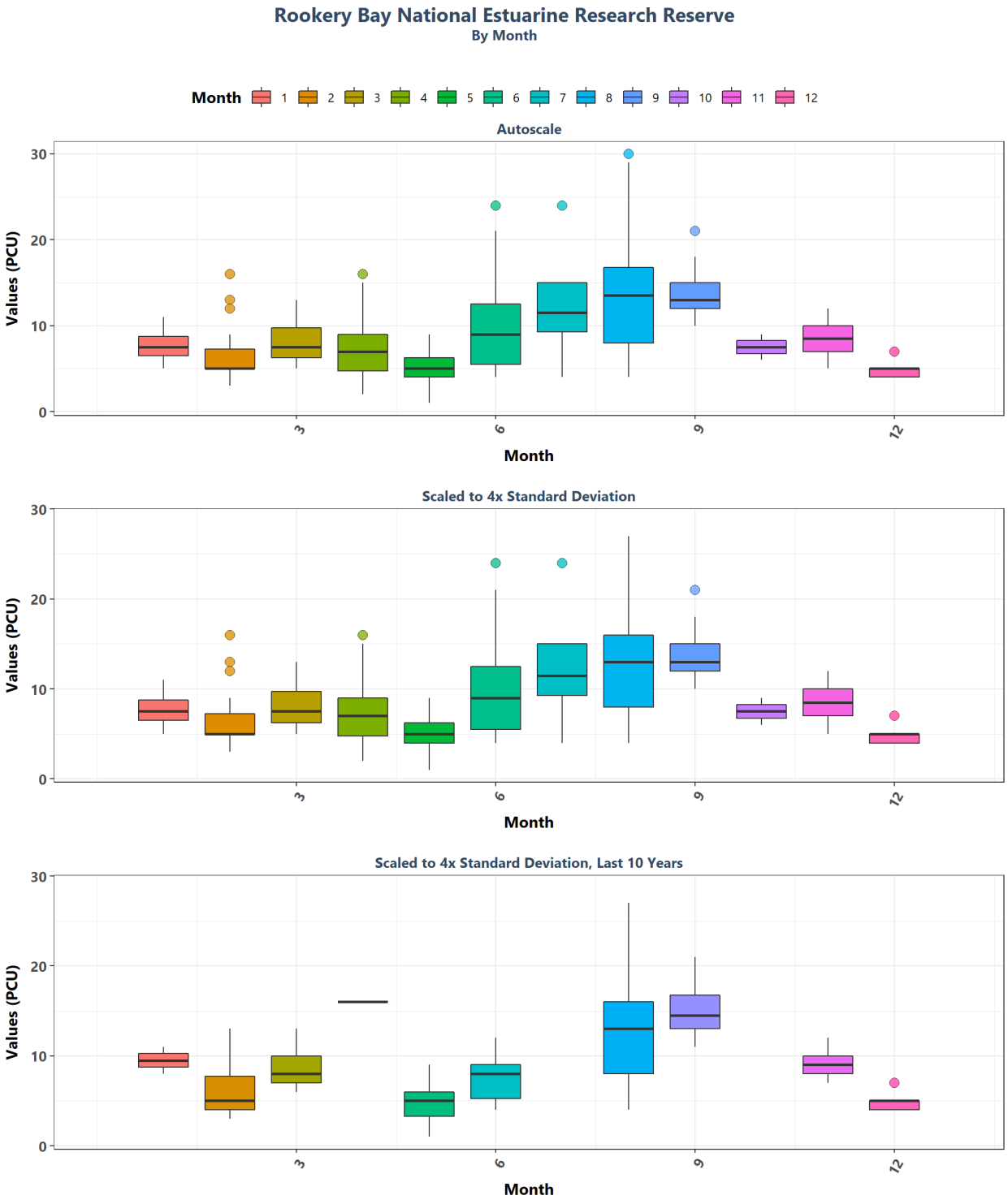

Rookery Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year

Rookery Bay Aquatic Preserve
By Month

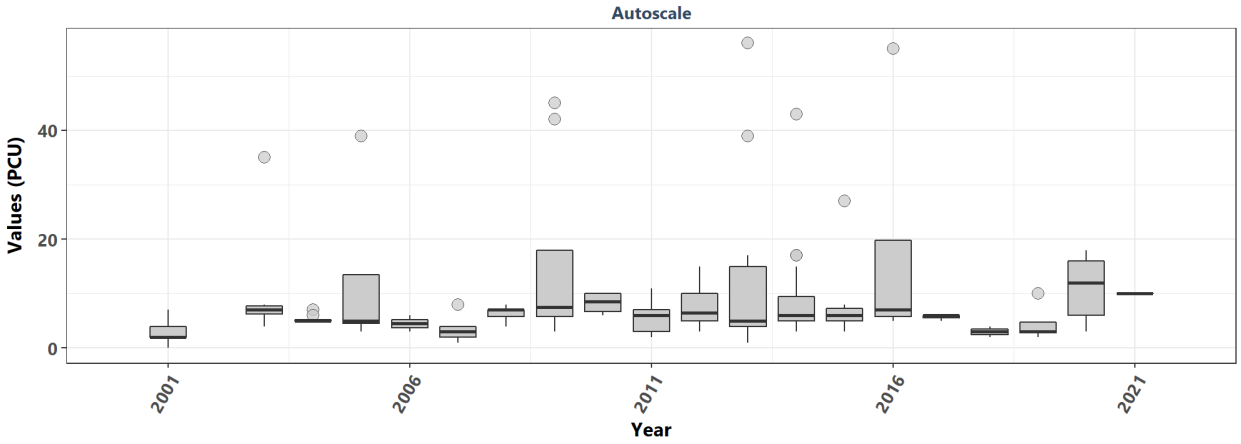


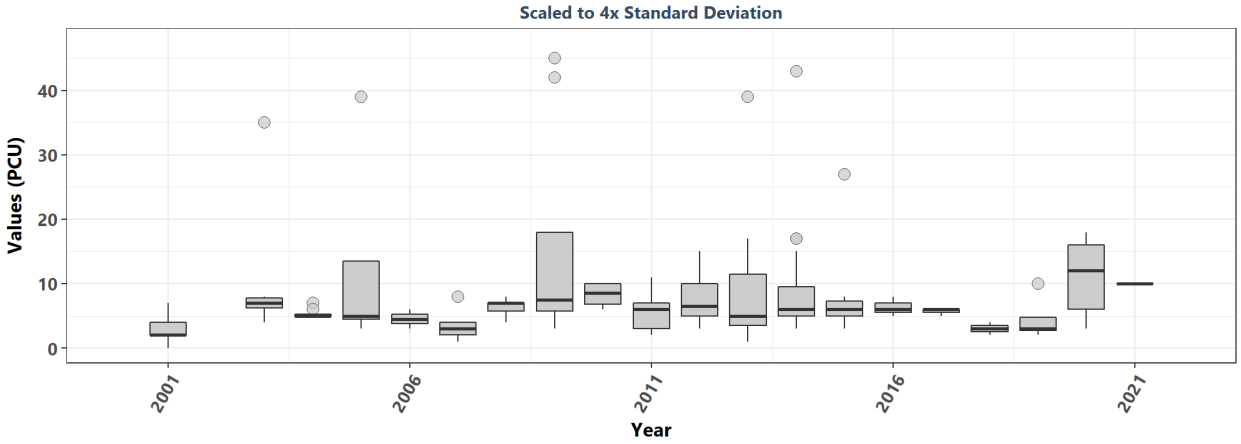

Rookery Bay National Estuarine Research Reserve
By Year
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year

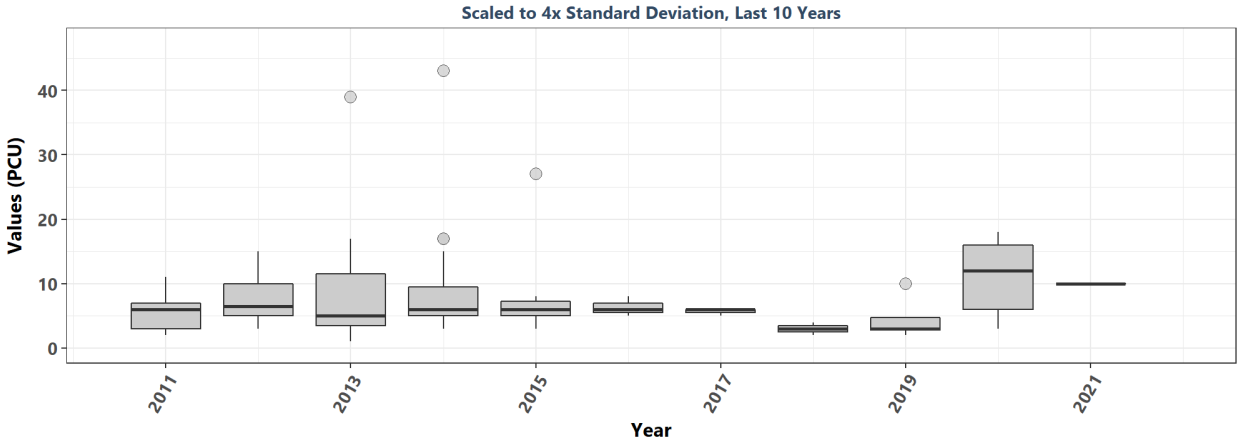

Rookery Bay National Estuarine Research Reserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year

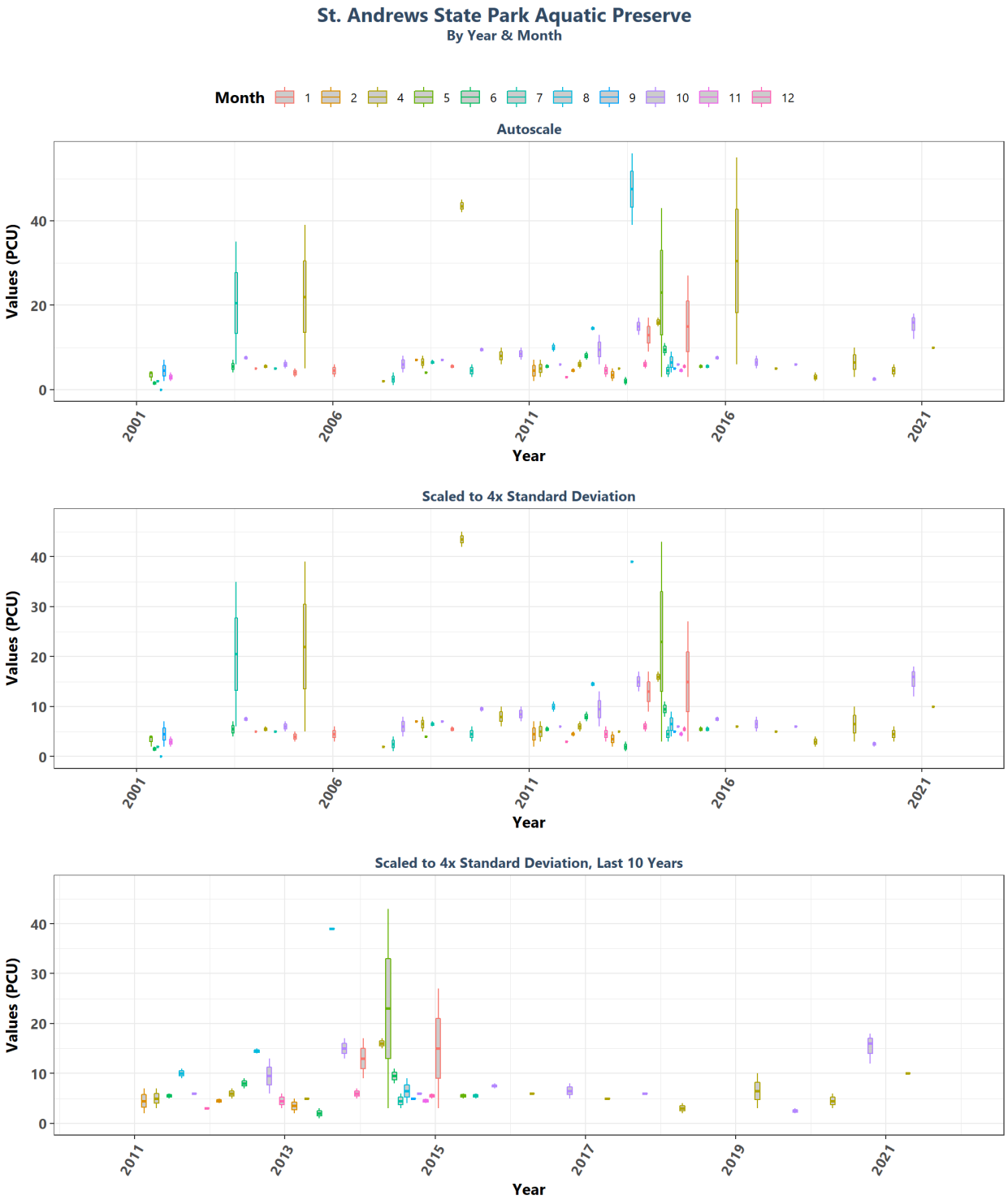

Rookery Bay National Estuarine Research Reserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Month
Scaled to 4x Standard Deviation
Values (PCU)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Month

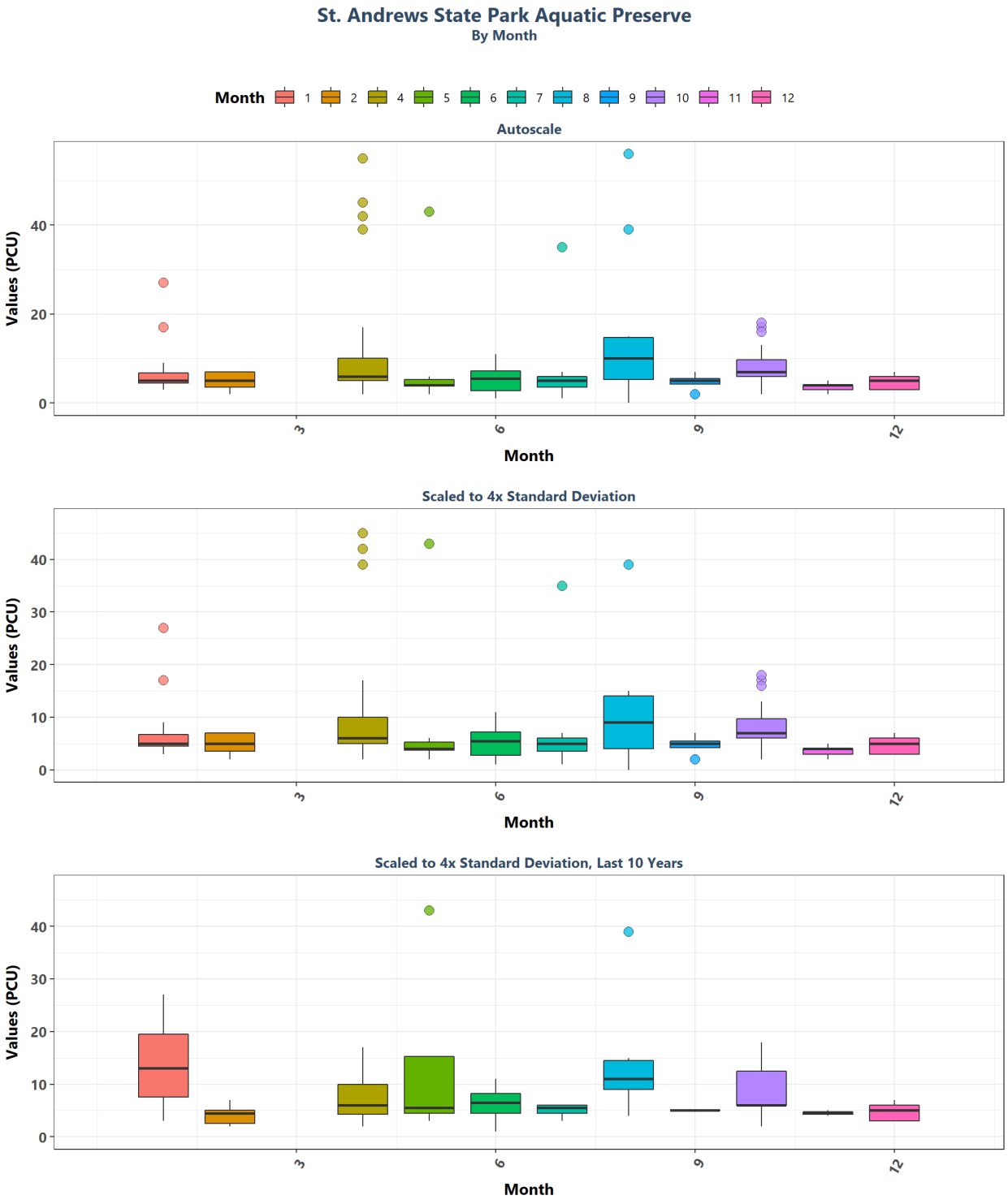
St. Andrews State Park Aquatic Preserve
By Year





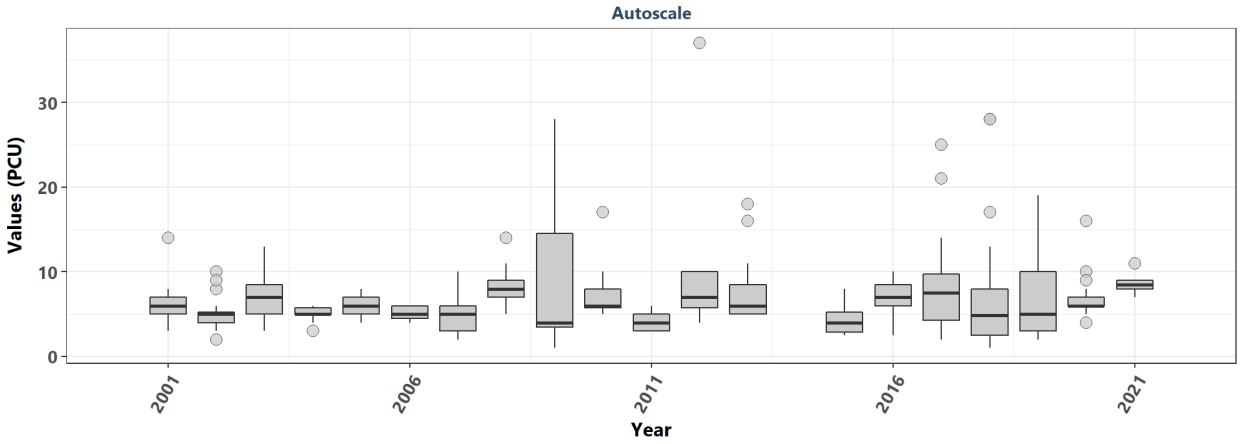


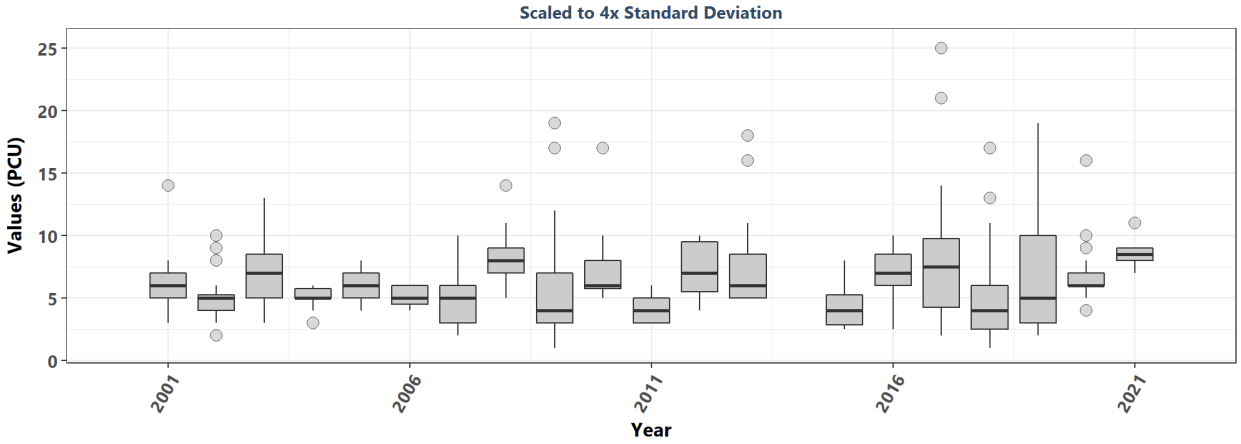

St. Andrews State Park Aquatic Preserve
By Year & Month
Month 1 2 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year

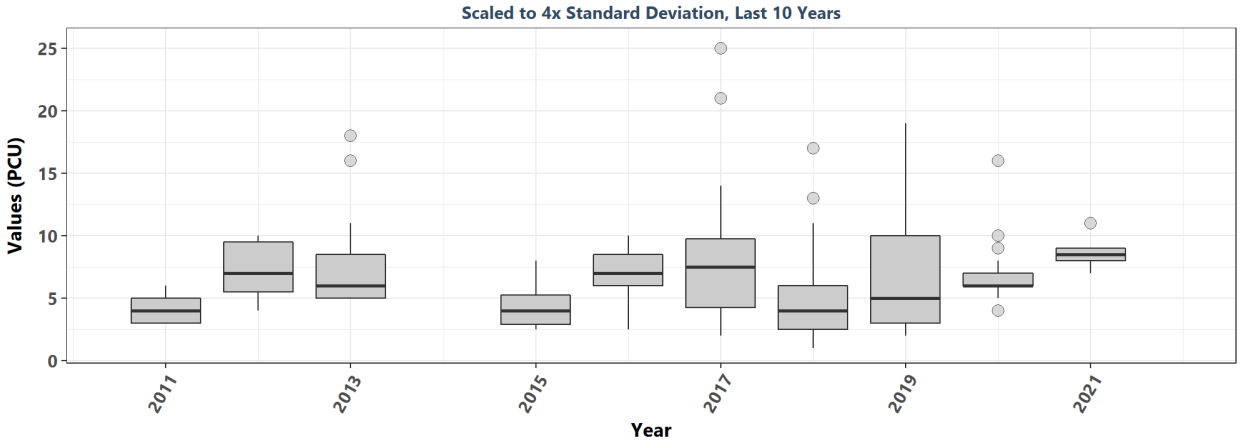

St. Andrews State Park Aquatic Preserve
By Month
Month 1 2 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Month
Scaled to 4x Standard Deviation
Values (PCU)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Month

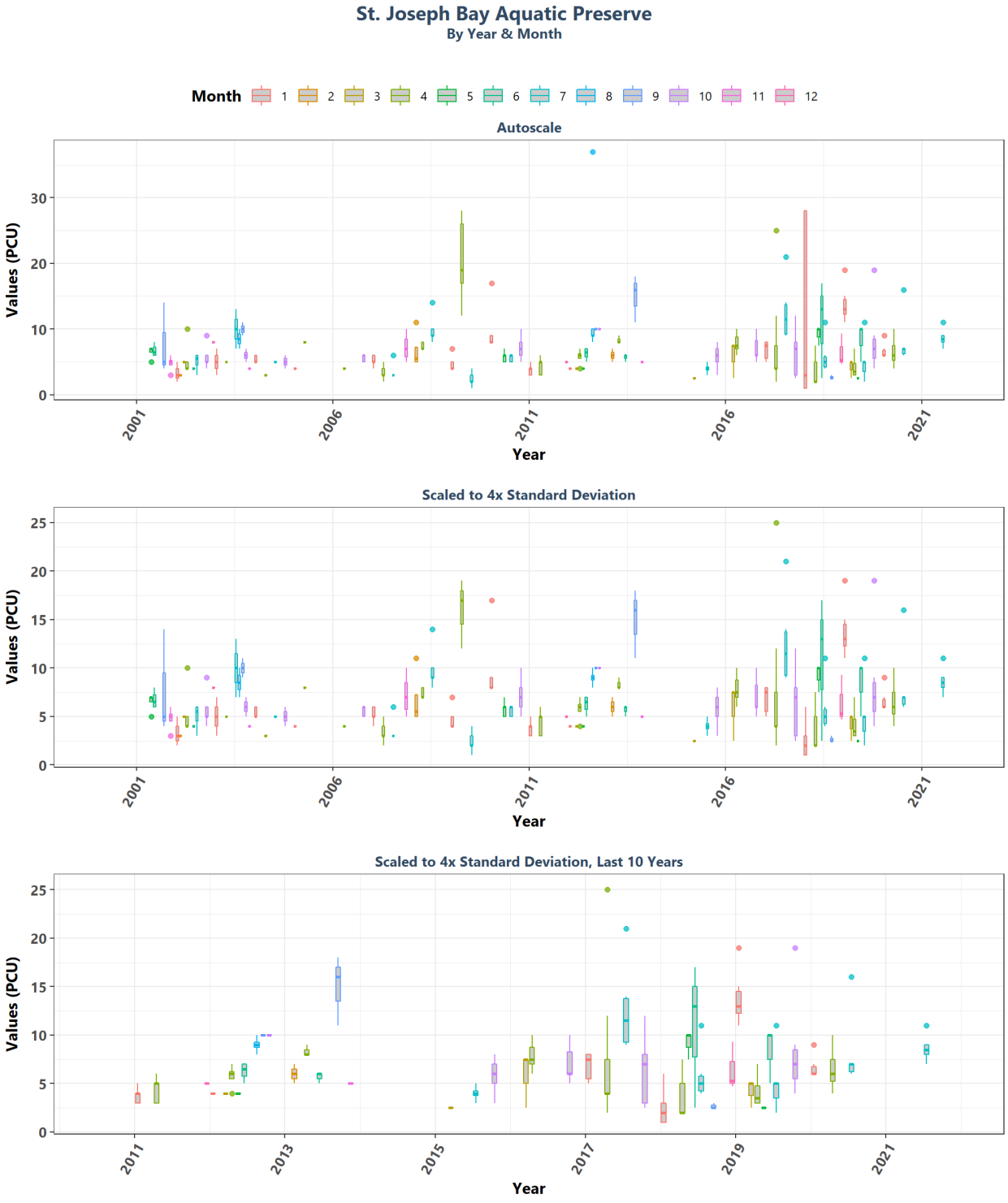
St. Joseph Bay Aquatic Preserve

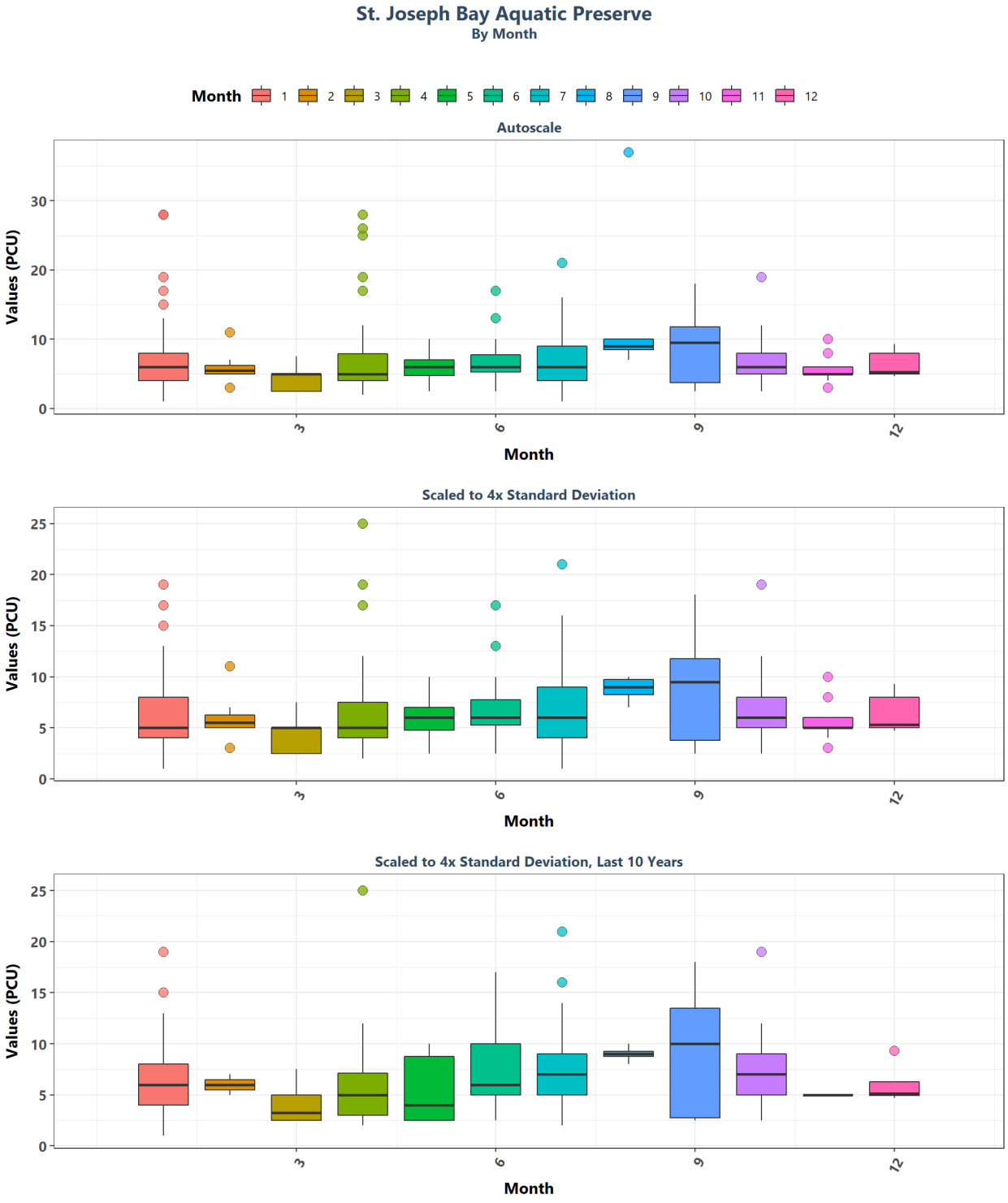
By Year





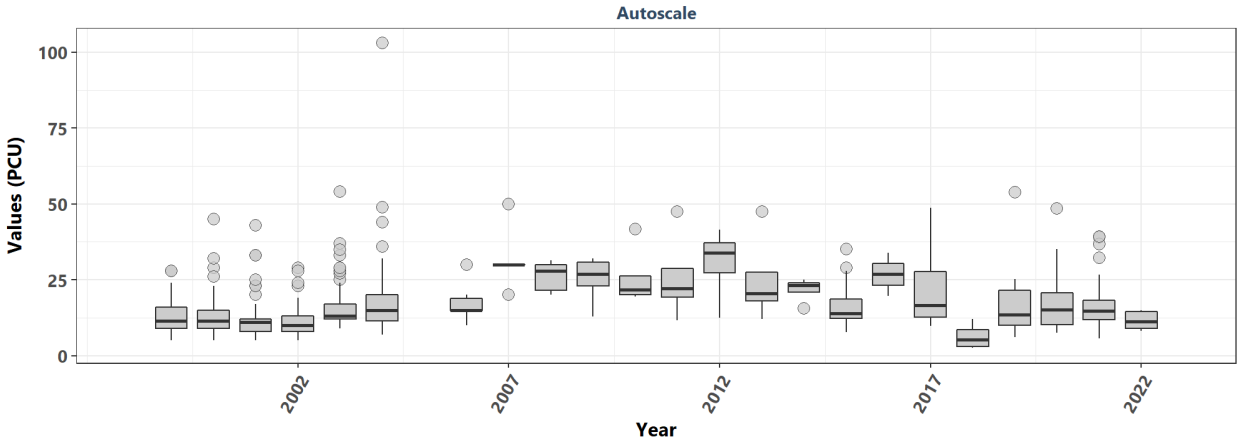


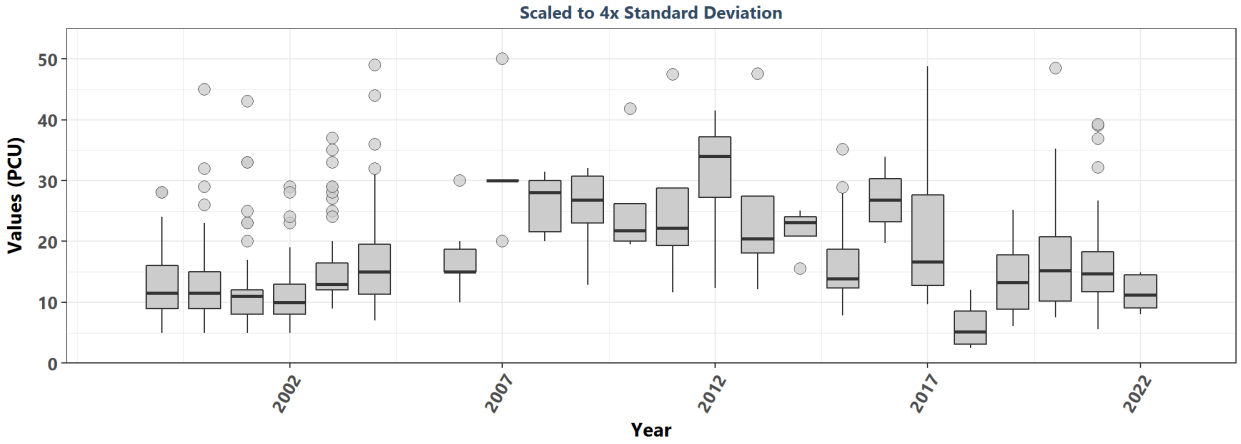

St. Joseph Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year

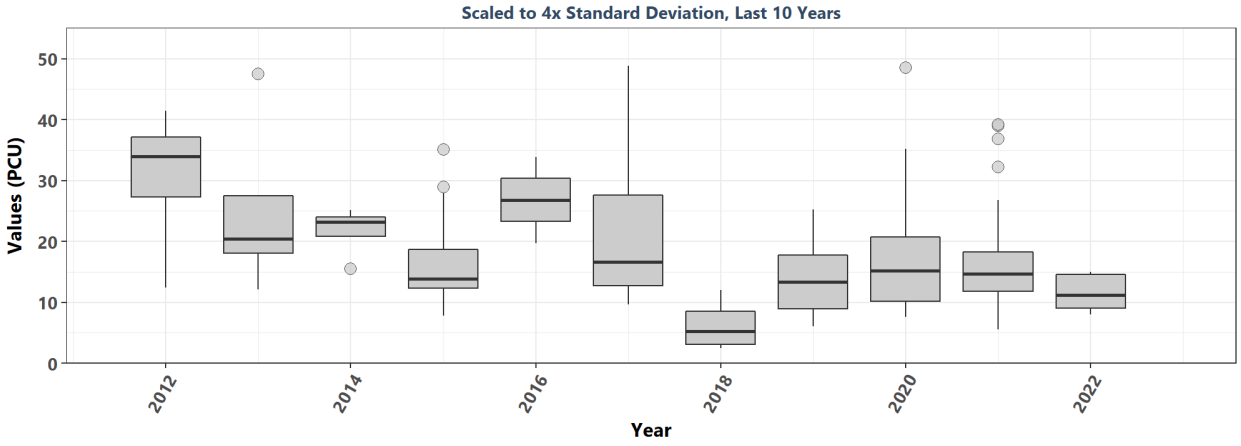

St. Joseph Bay Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Month
Scaled to 4x Standard Deviation
Values (PCU)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Month

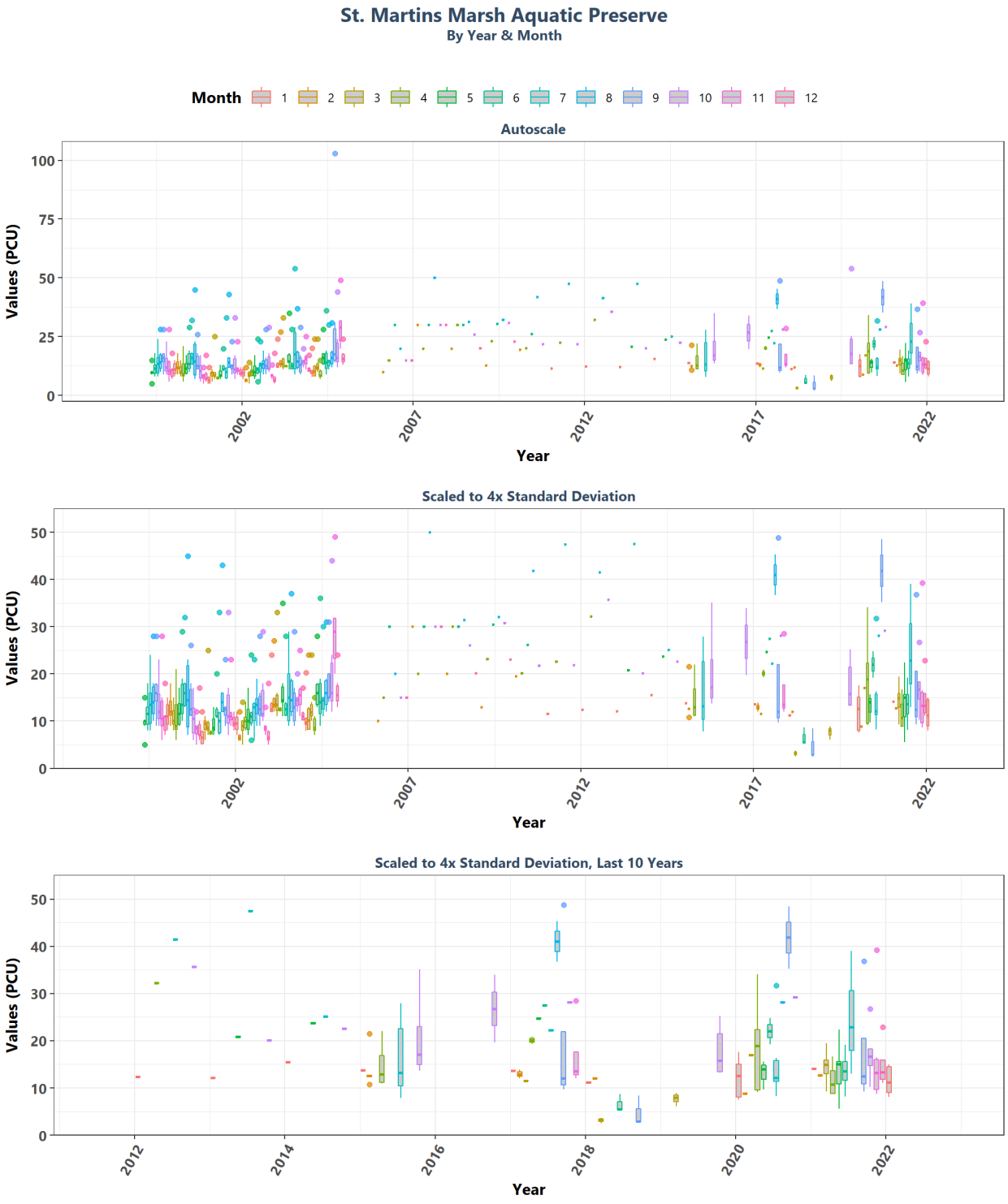
St. Martins Marsh Aquatic Preserve

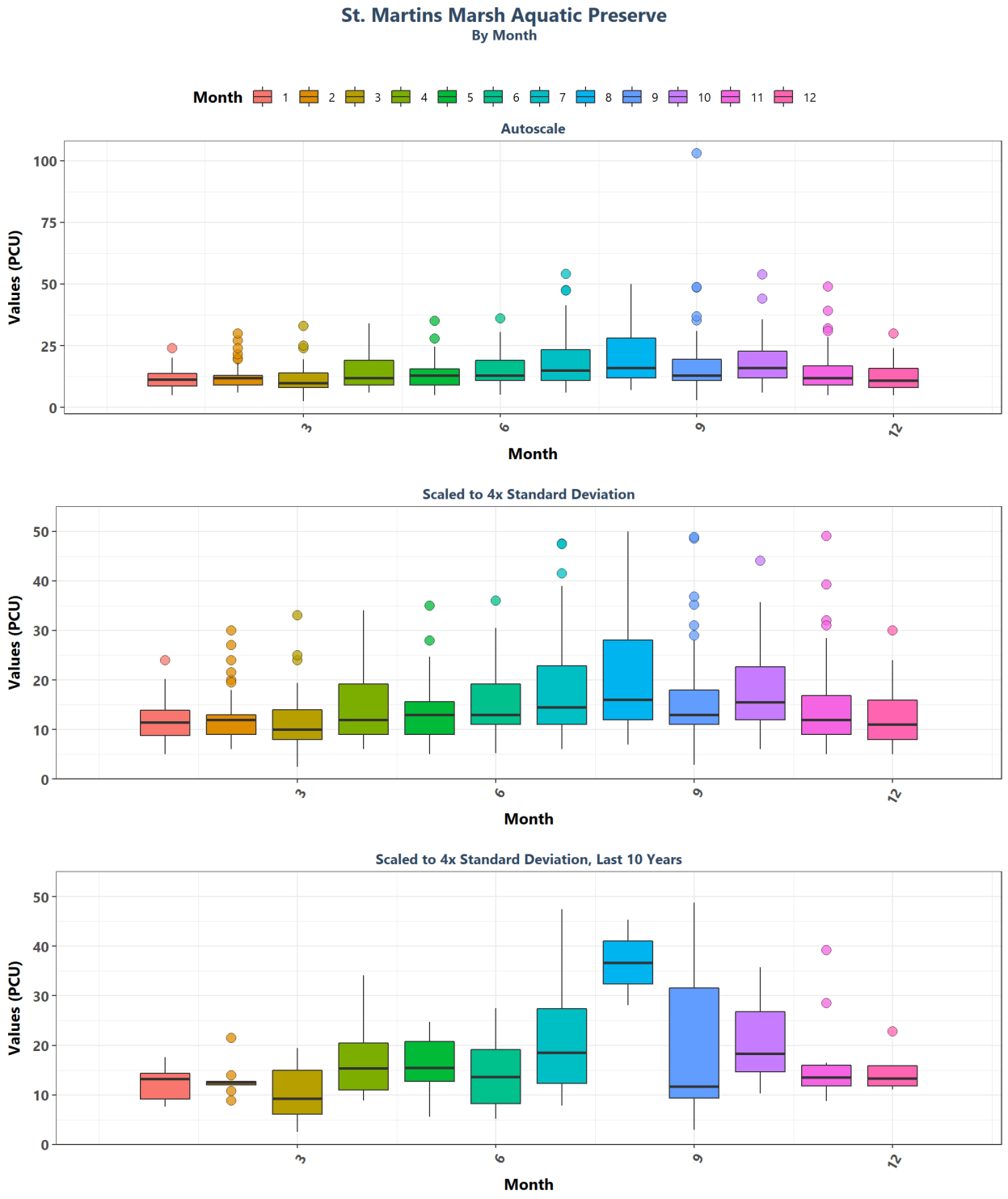
By Year








St. Martins Marsh Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (PCU)
Year
Scaled to 4x Standard Deviation
Values (PCU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (PCU)
Year




```
rm(list = setdiff(ls(), c("all_params", "all_depths", "all_activity", "param_name", "depth", "activity"
```