SEACAR Discrete Water Quality Analysis: Field Bottom Dissolved Oxygen Saturation

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: 41035, Number Passed Filter: 41035
## Program 476 H Codes: 0 (0%)
## I Codes: 0 (0%)
## Q Codes: 0 (0%)
## U Codes: 0 (0%)
```

```
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)
```

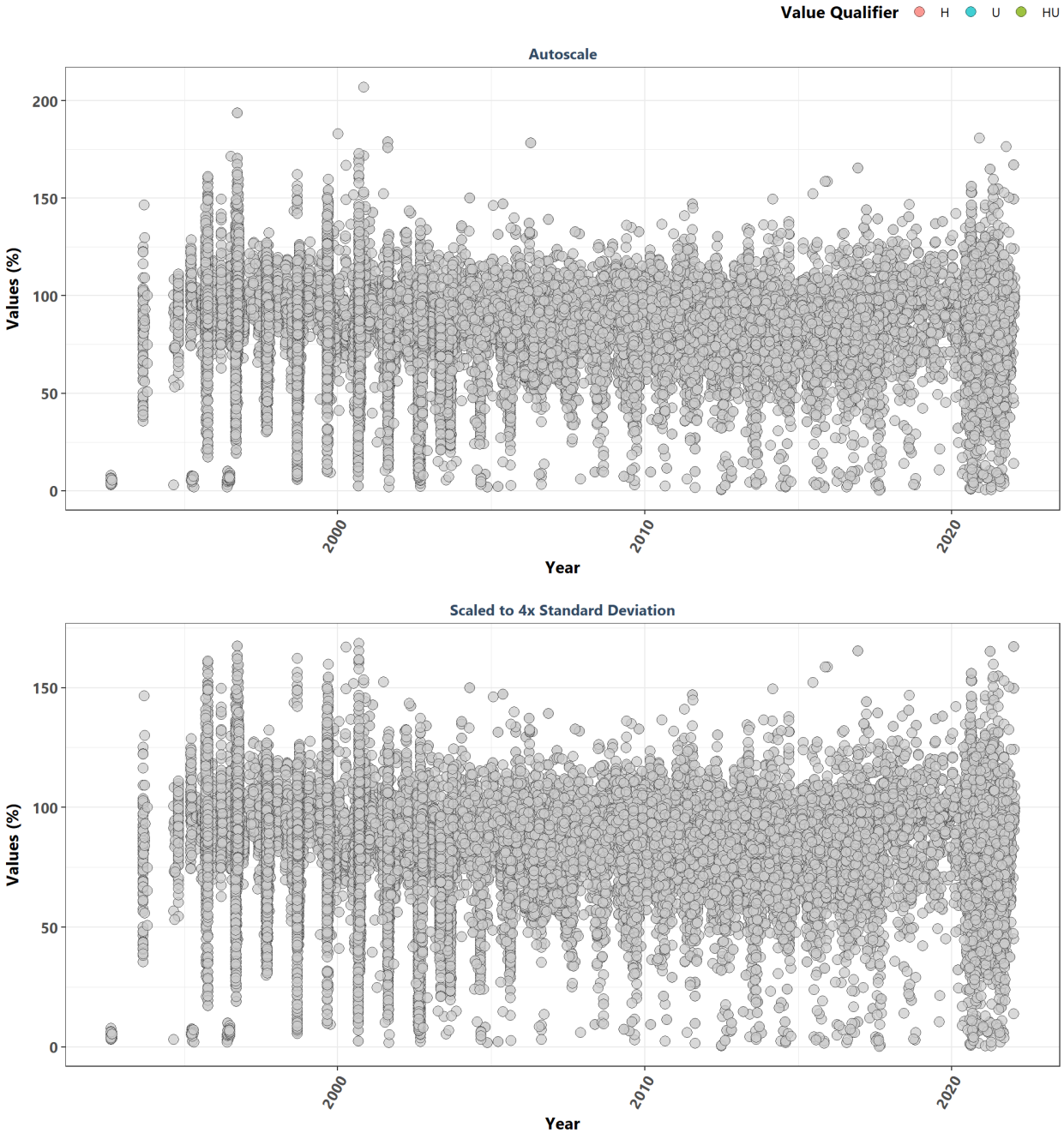
```
KT.Stats2$MonitoringID <- NULL
fwrite(KT.Stats2, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                         "_KendallTau_Stats.txt"), sep="|")
rm(KT.Stats2)
data <- data[!is.na(data$ResultValue),]
```

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))
```

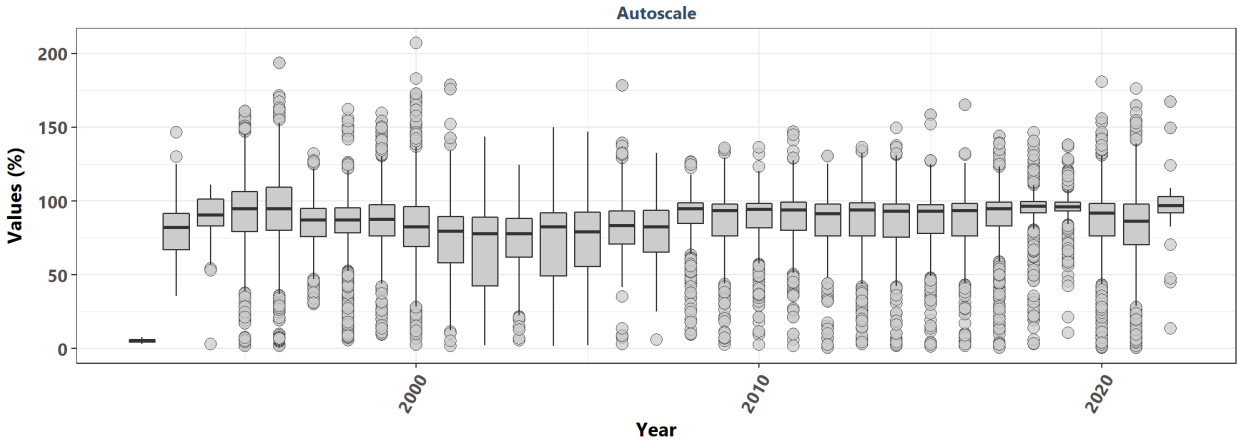
This set of box plots are grouped by year and month with the color being related to the month.

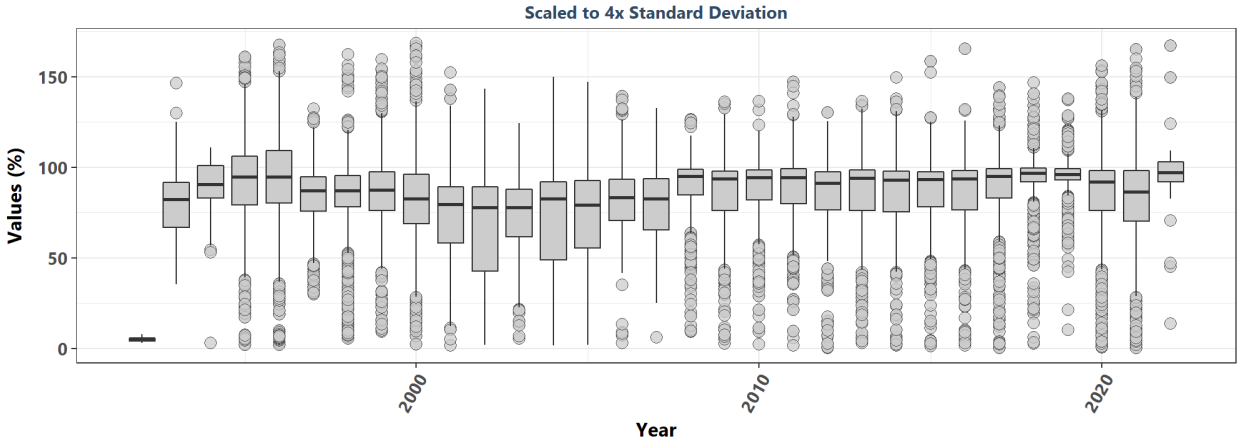
```
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Year",
        y=paste0("Values (", unit,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

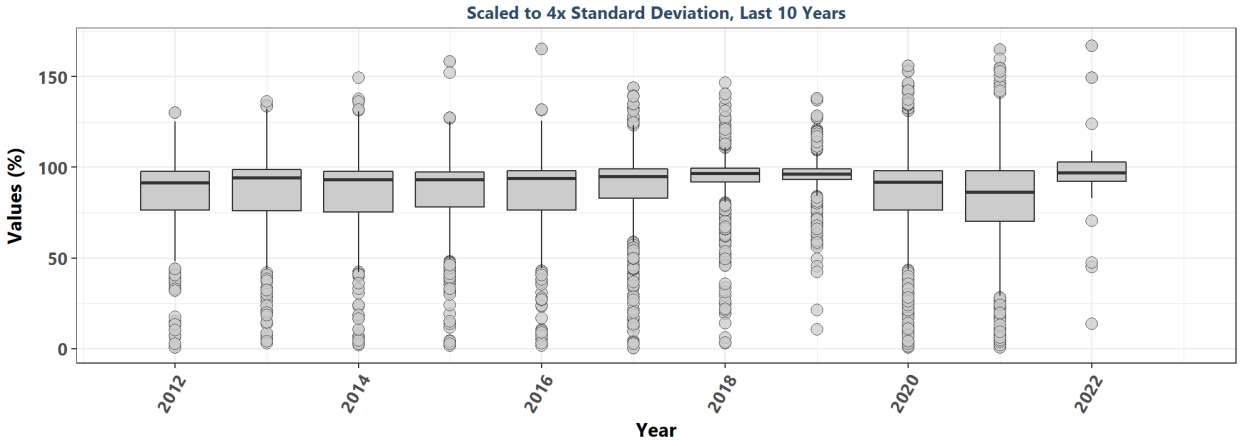
The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

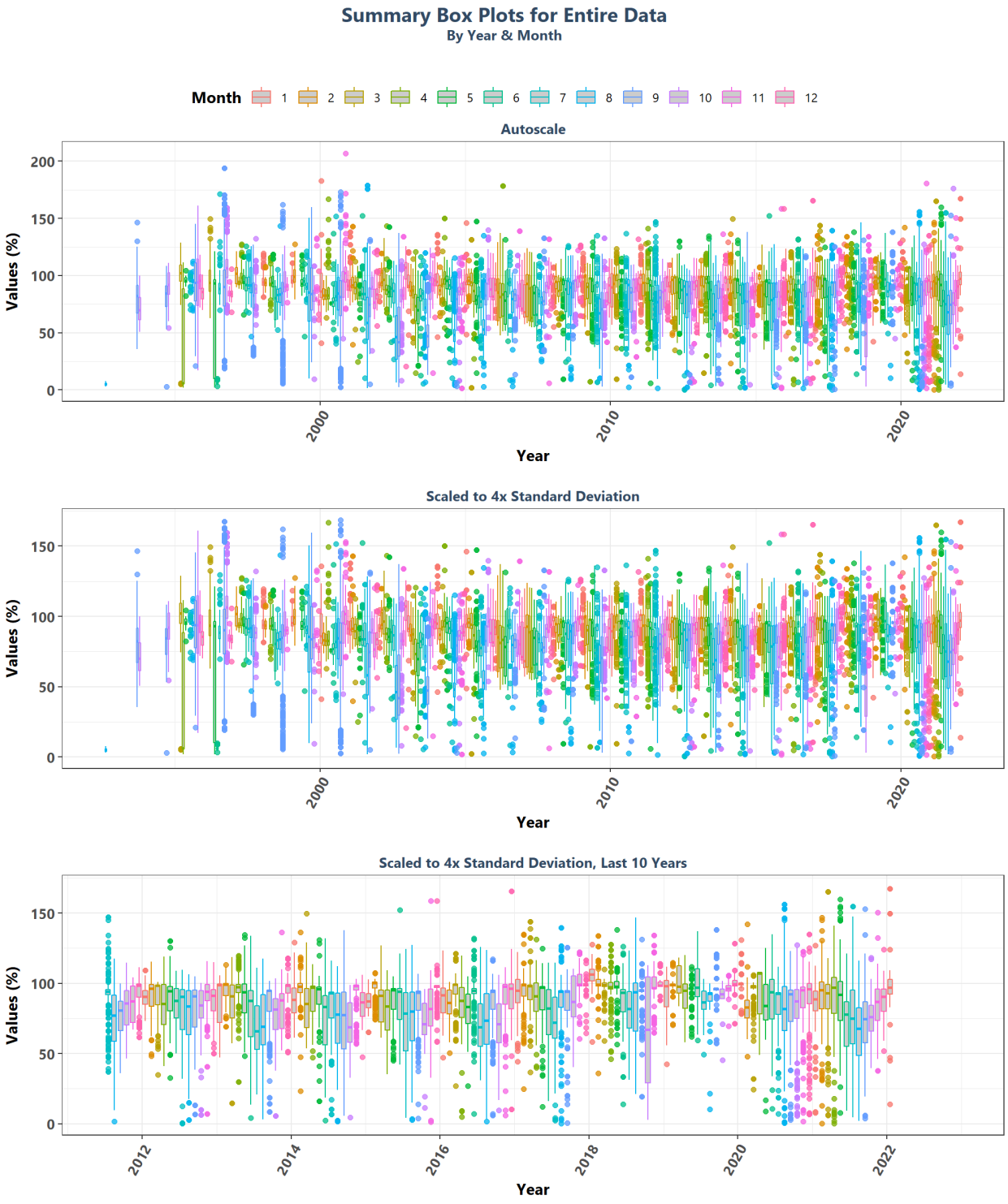
```
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
   geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
                outlier.color="#333333", outlier.alpha=0.75) +
   labs(subtitle="Autoscale", 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))
```

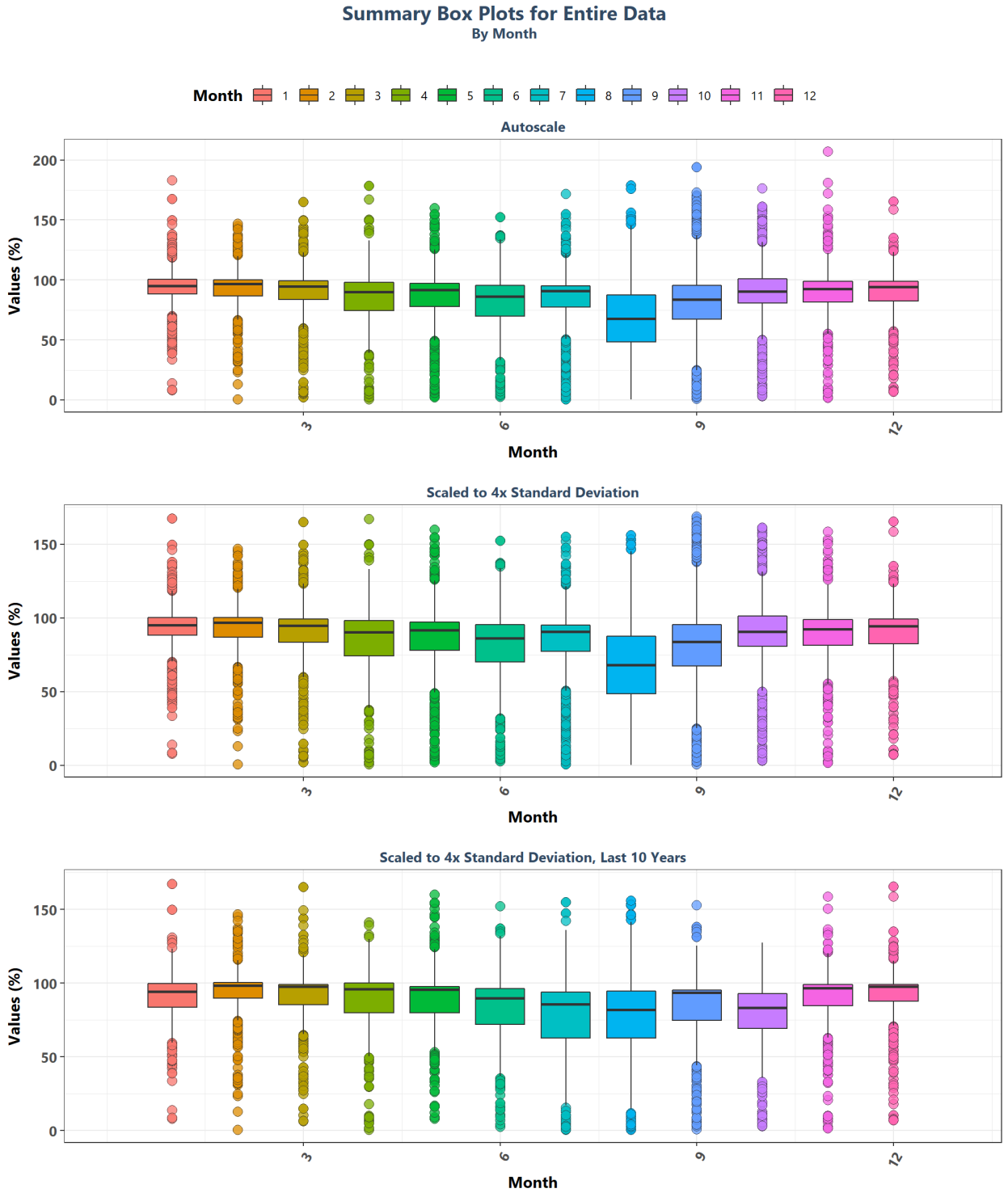
Summary Box Plots for Entire Data
By Year







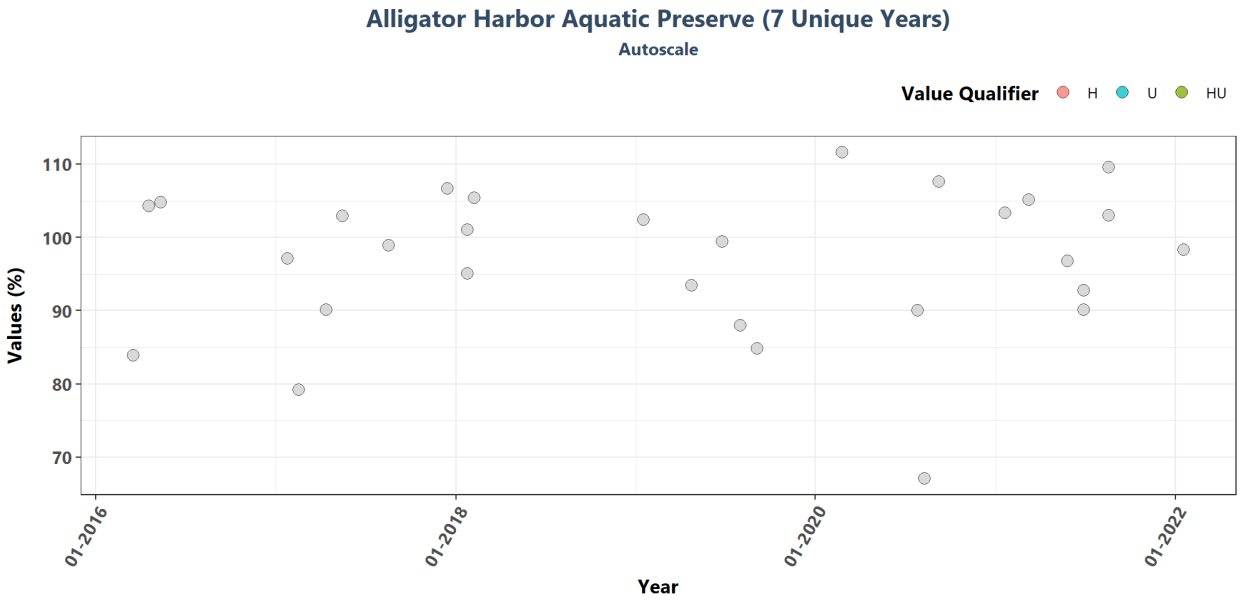

Summary Box Plots for Entire Data
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Year
Scaled to 4x Standard Deviation
Values (%)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Year

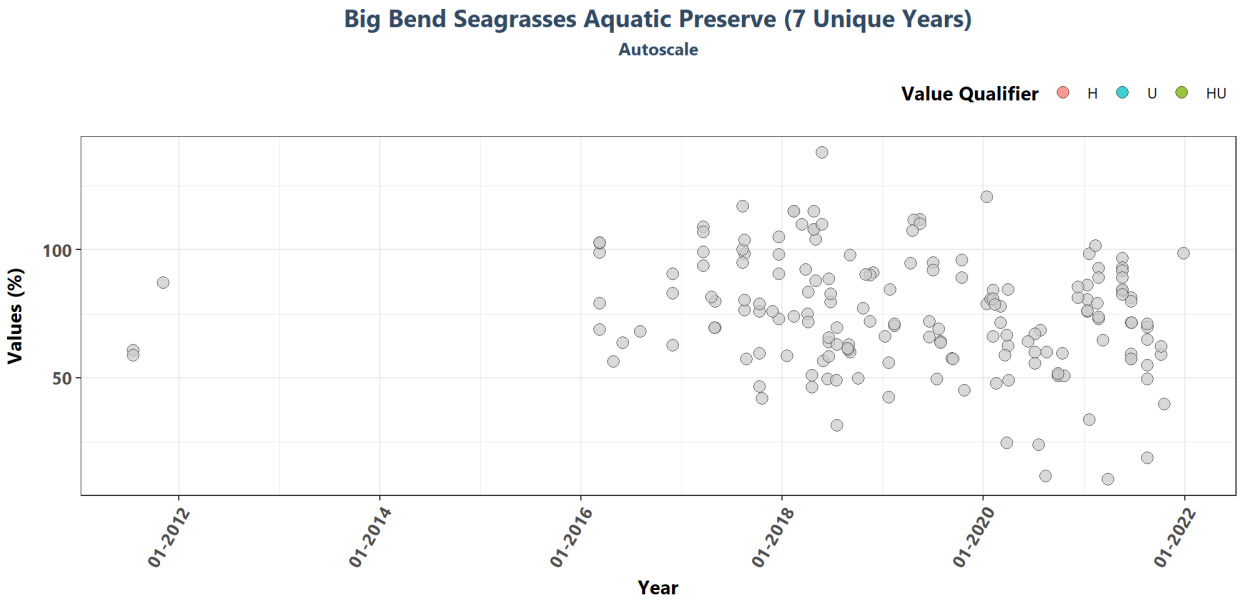


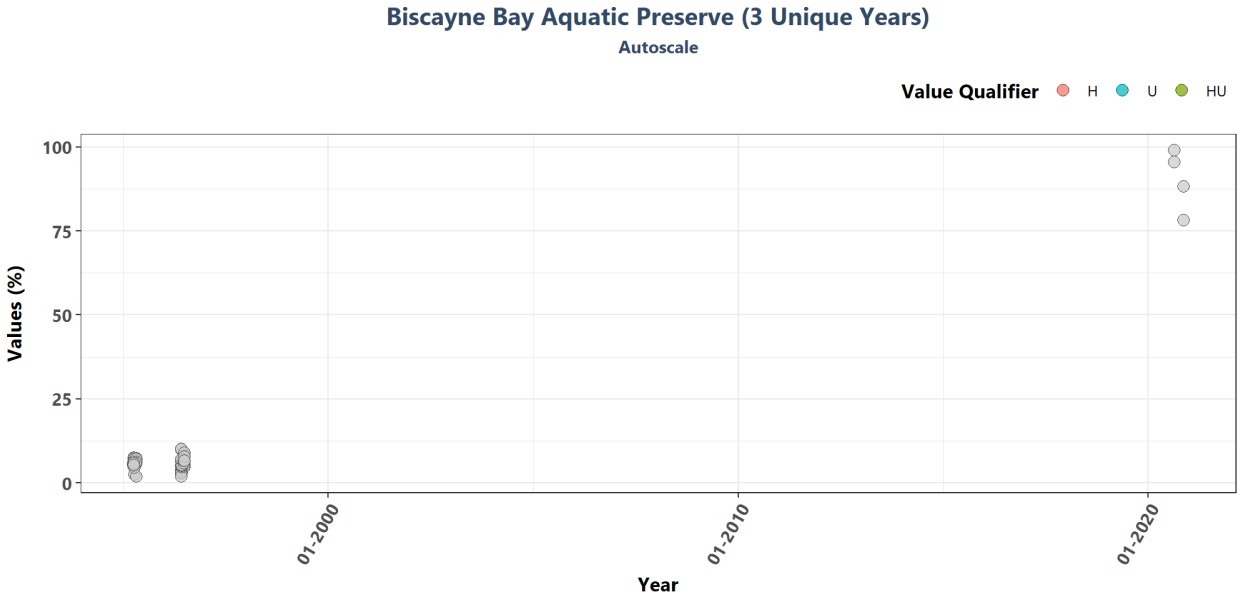
Appendix III: Excluded Managed Areas

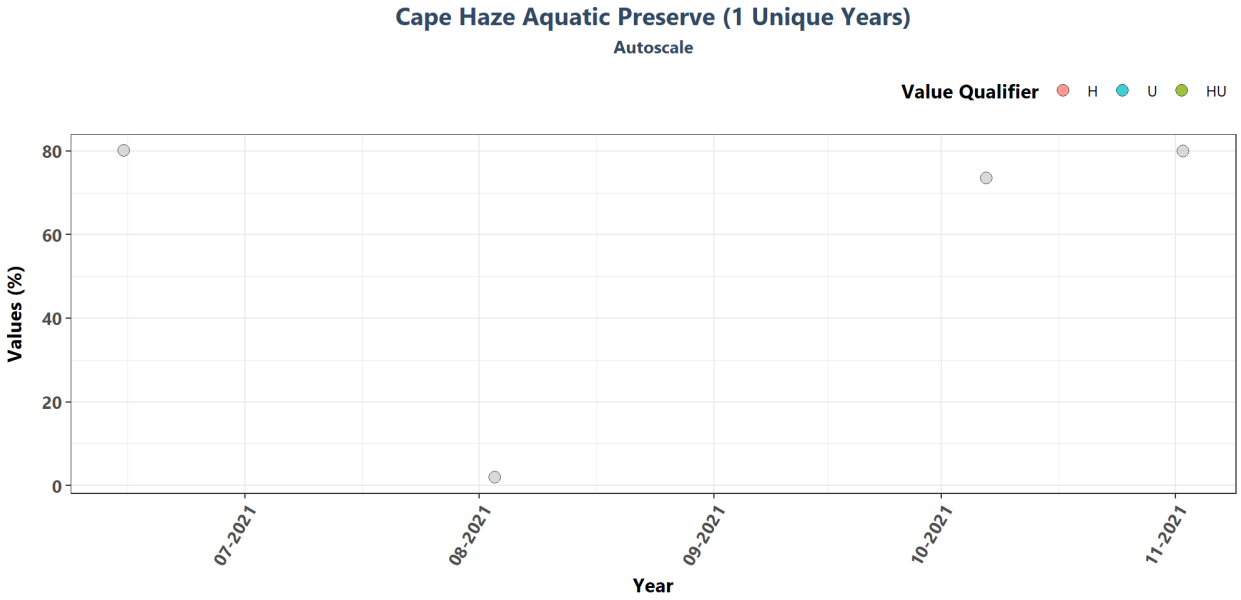
Scatter plots of data values are created for managed areas that have fewer than 10 separate years of data entries. Data points are colored based on specific value qualifiers of interest.

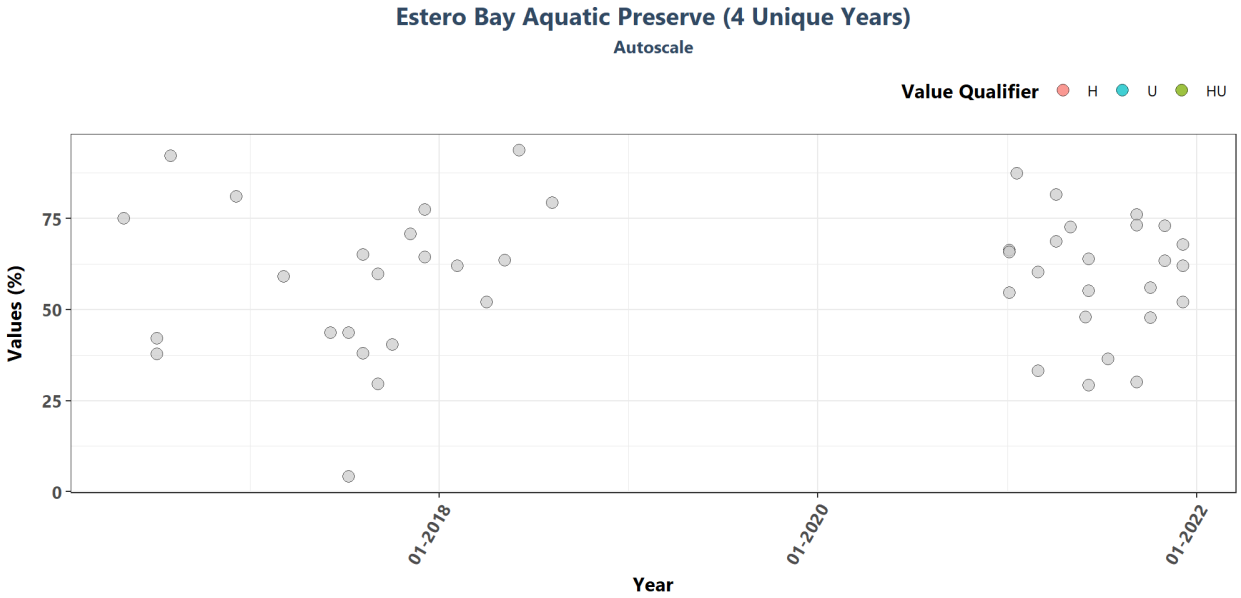
```
if(z==0){
   print("There are no managed areas that qualify.")
} else {
   for(i in 1:z){
      p1<-ggplot(data=data[data$ManagedAreaName==MA_Exclude$ManagedAreaName[i]&
                           data$Include==TRUE, ],
                 aes(x=SampleDate, y=ResultValue, fill=VQ_Plot)) +
         geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
         labs(title=paste0(MA_Exclude$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)
   }
}
```

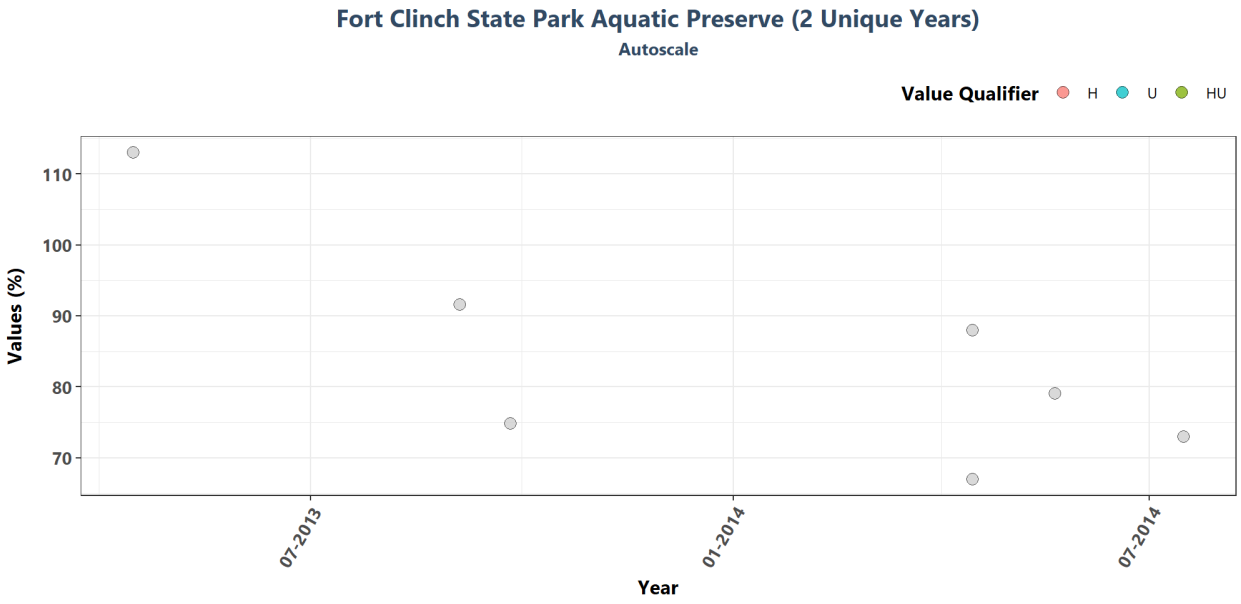


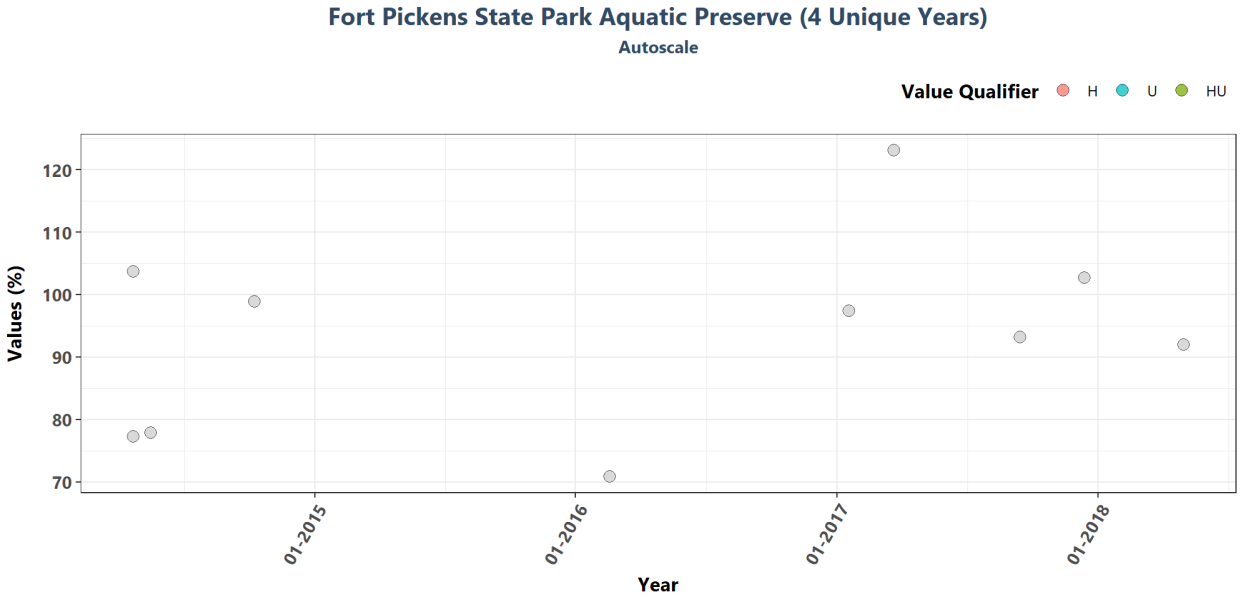

Big Bend Seagrasses Aquatic Preserve (7 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
100
50
01-2012
01-2014
01-2016
01-2018
01-2020
01-2022
Year

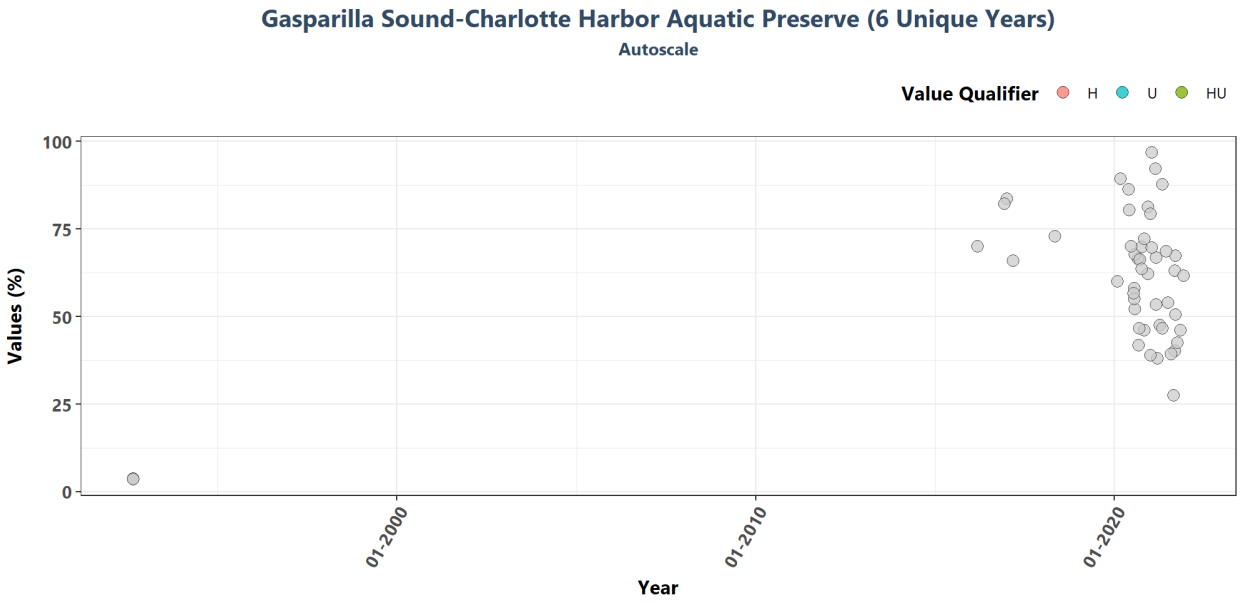

Biscayne Bay Aquatic Preserve (3 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
100
75
50
25
0
01-2000
01-2010
01-2020
Year

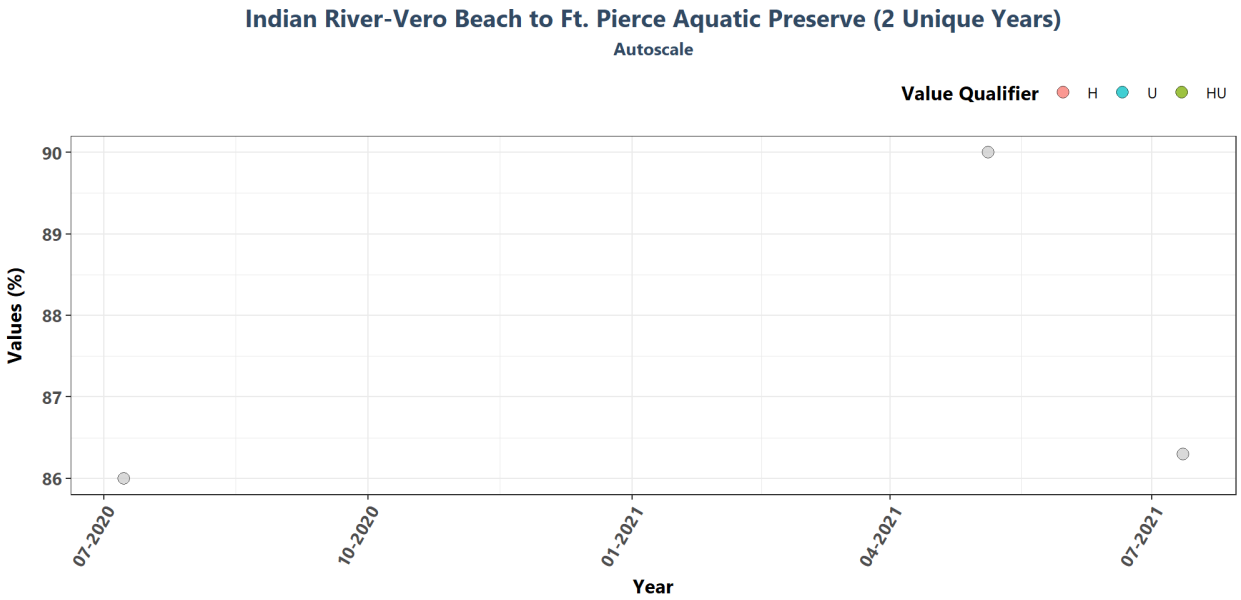

Cape Haze Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
80
60
40
20
0
07-2021
08-2021
09-2021
10-2021
11-2021
Year

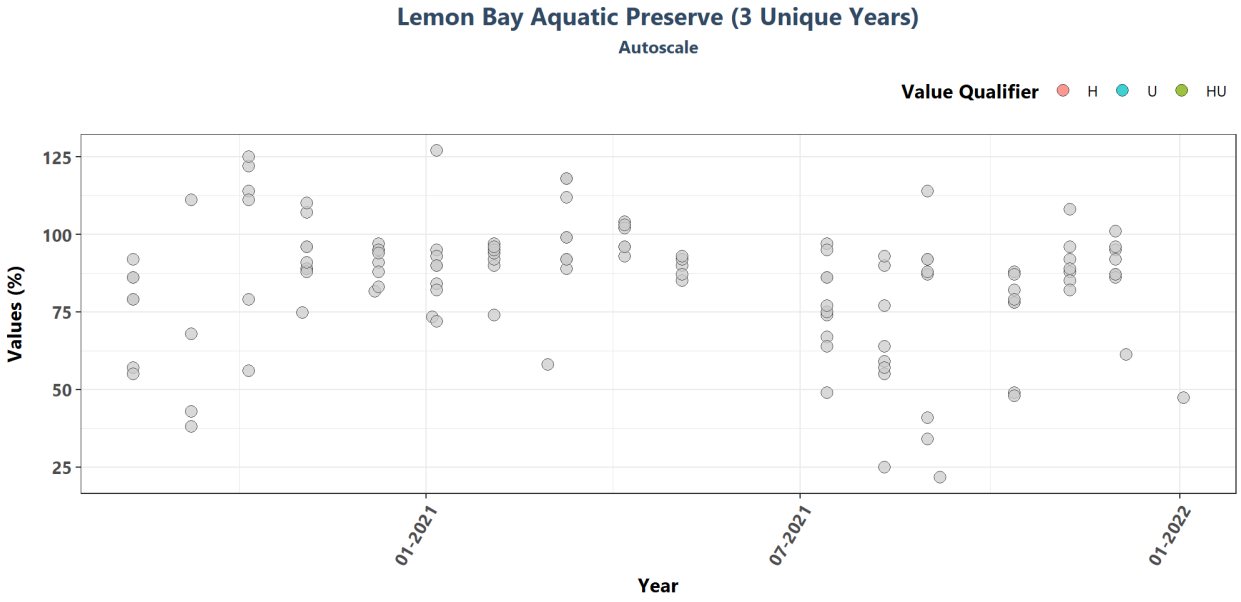

Estero Bay Aquatic Preserve (4 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
75
50
25
0
01-2018
01-2020
01-2022
Year

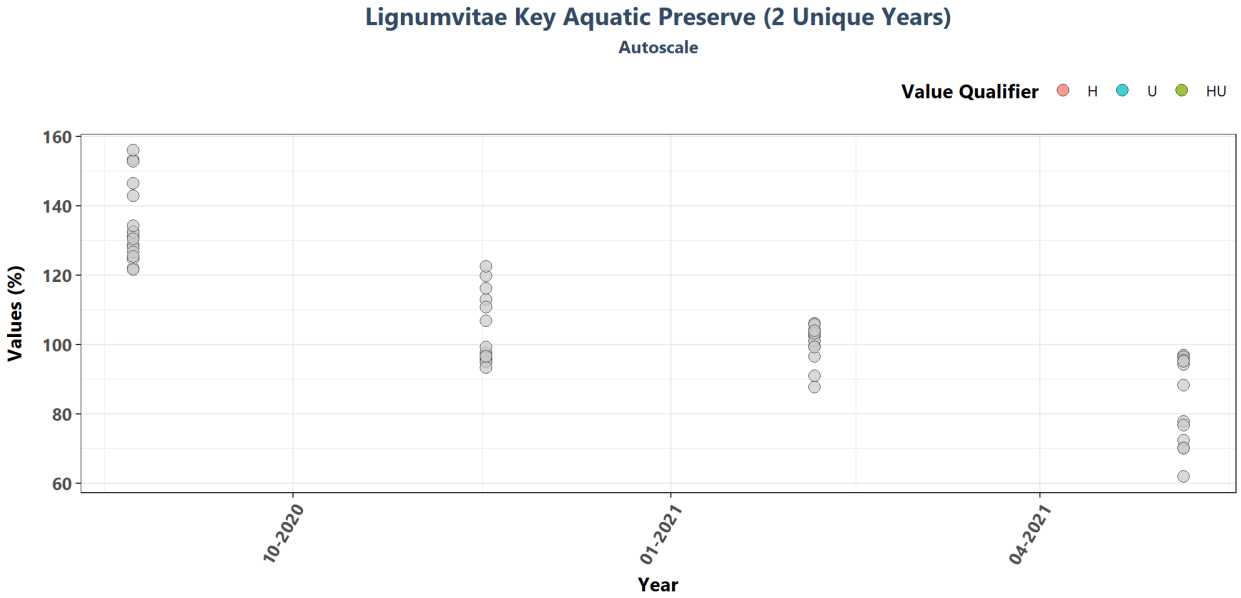

Fort Clinch State Park Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
H
U
HU
110
100
90
80
70
Values (%)
07-2013
01-2014
07-2014
Year

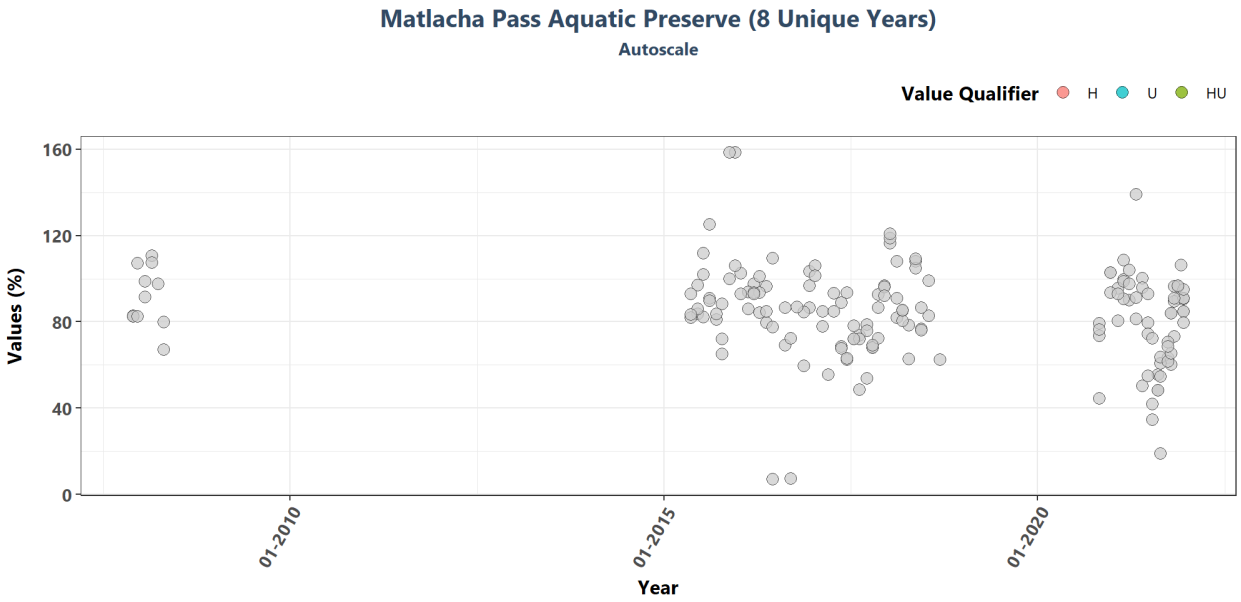

Fort Pickens State Park Aquatic Preserve (4 Unique Years)
Autoscale
Value Qualifier
H
U
HU
120
110
100
90
80
70
Values (%)
01-2015
01-2016
01-2017
01-2018
Year

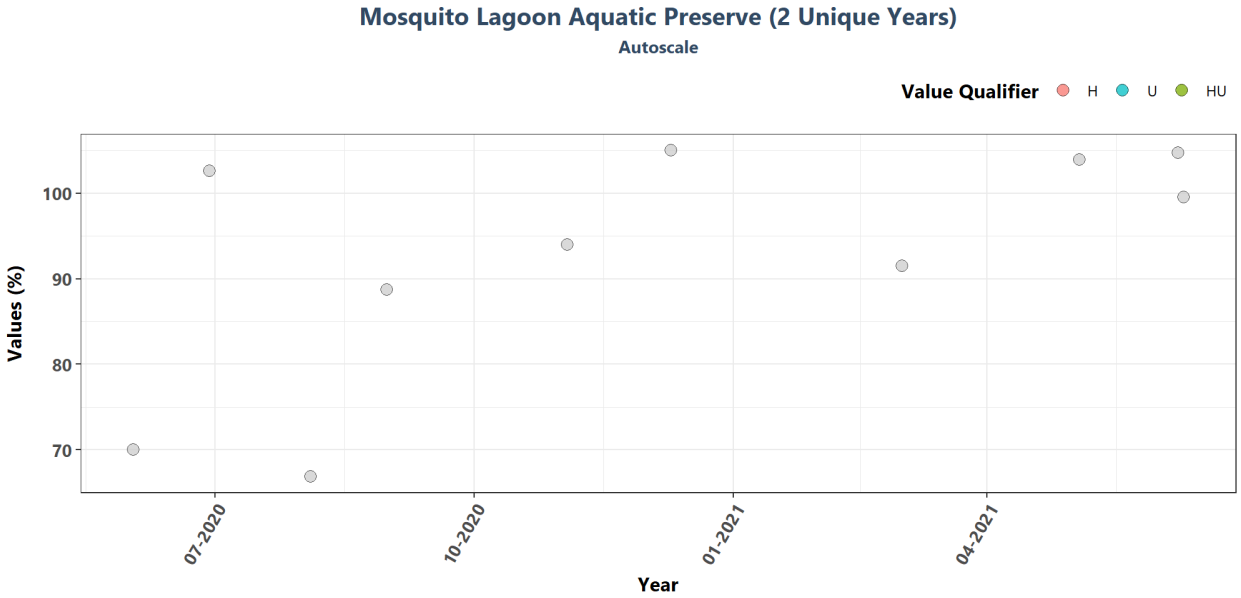

Gasparilla Sound-Charlotte Harbor Aquatic Preserve (6 Unique Years)
Autoscale
Value Qualifier H U HU
100
75
50
25
0
Values (%)
01-2000
01-2010
01-2020
Year

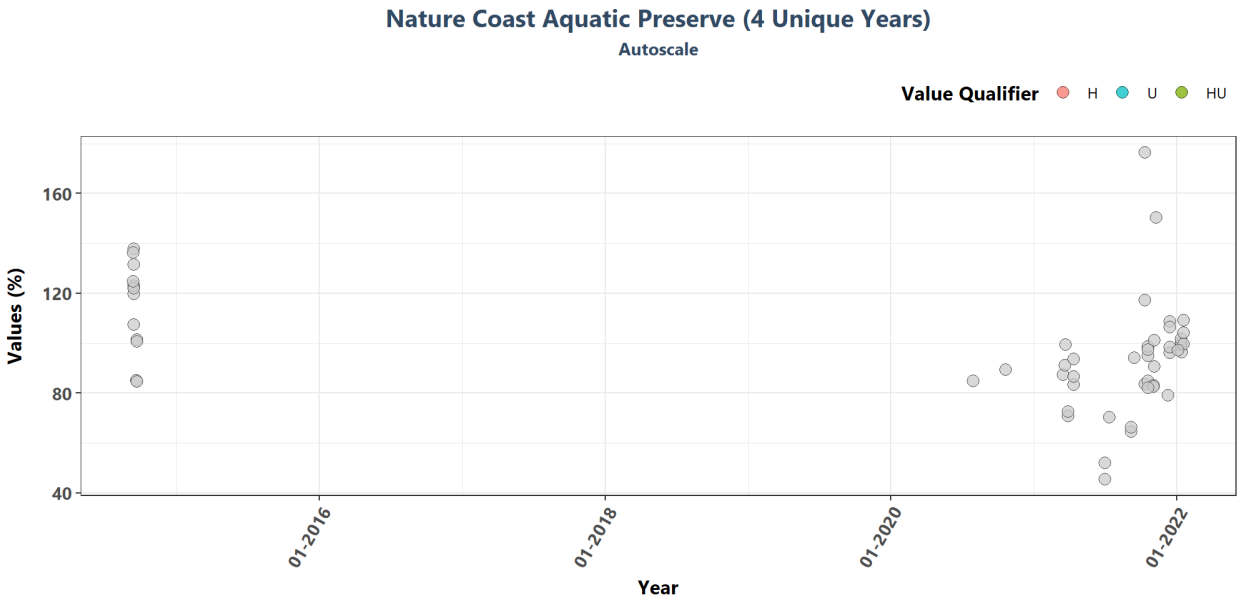

Indian River-Vero Beach to Ft. Pierce Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier H U HU
90
89
88
87
86
Values (%)
07-2020
10-2020
01-2021
04-2021
07-2021
Year

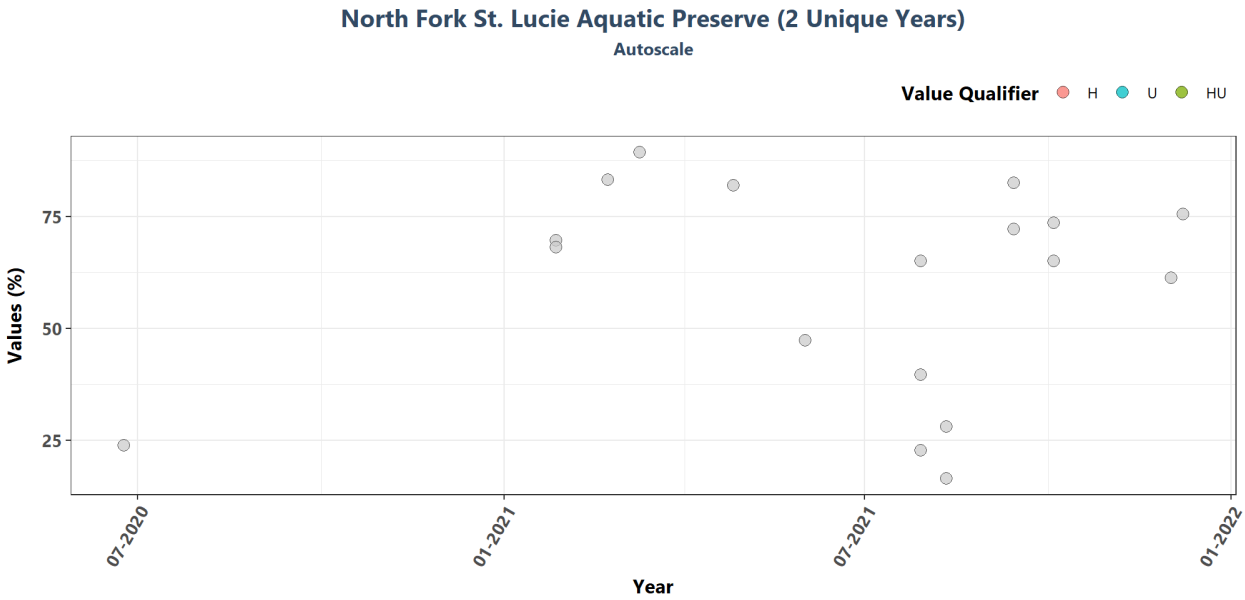

Lemon Bay Aquatic Preserve (3 Unique Years)
Autoscale
Value Qualifier
H
U
HU
125
100
75
50
25
Values (%)
01-2021
07-2021
01-2022
Year

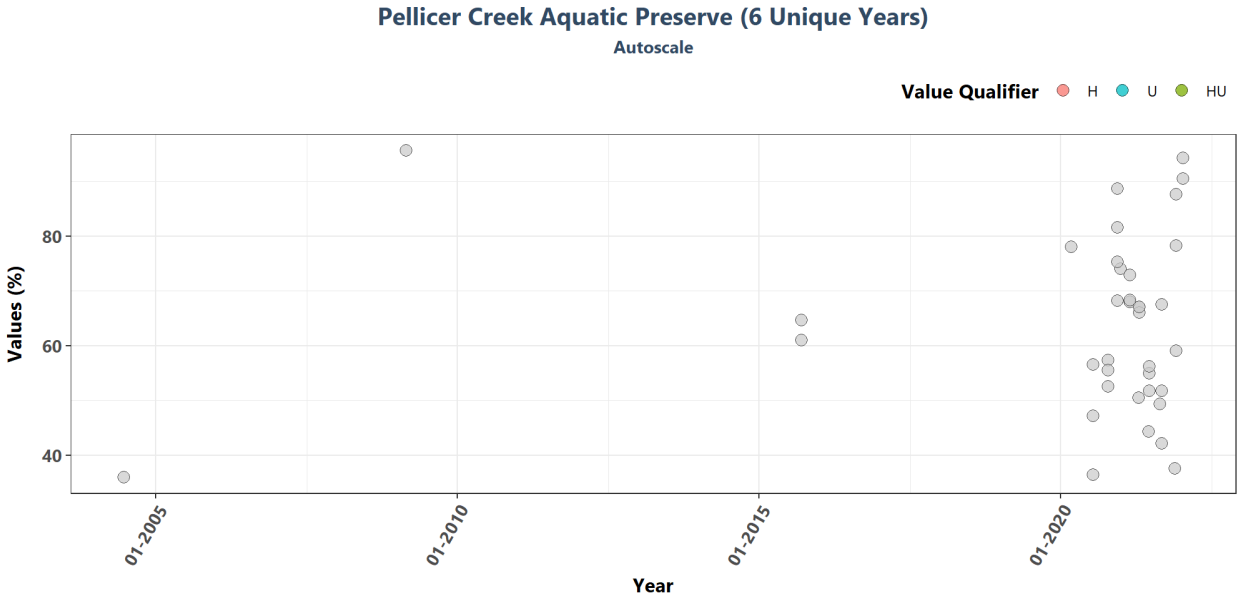

Lignumvitae Key Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
H
U
HU
160
140
120
100
80
60
Values (%)
10-2020
01-2021
04-2021
Year

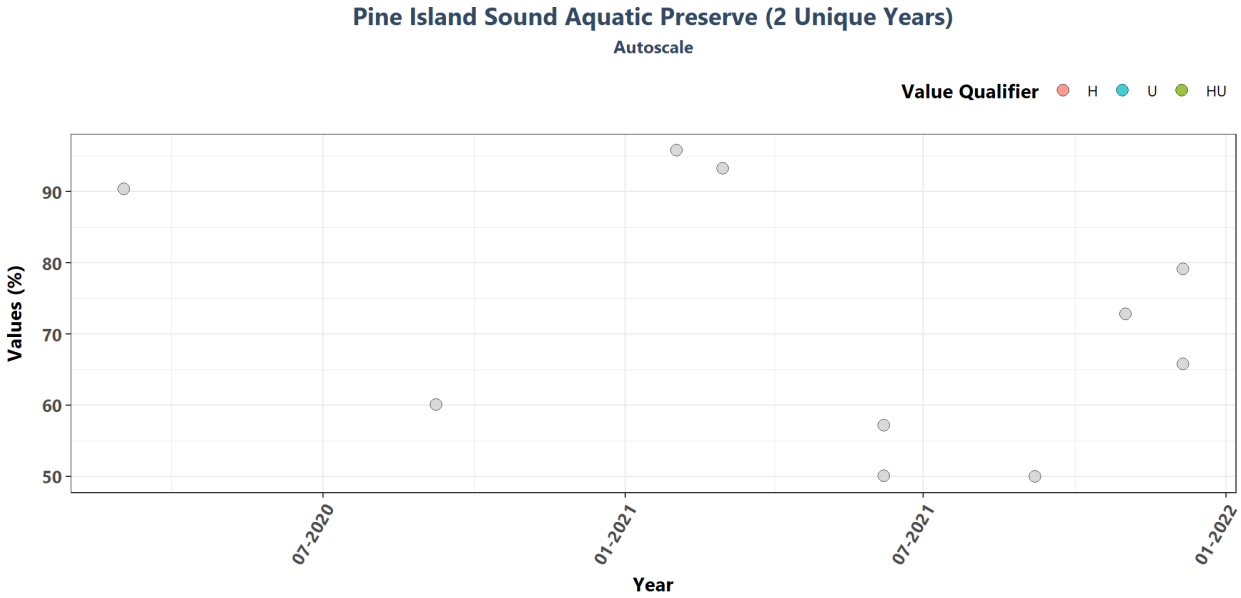

Matlacha Pass Aquatic Preserve (8 Unique Years)
Autoscale
Value Qualifier H U HU
160
120
80
40
0
Values (%)
01-2010
01-2015
01-2020
Year

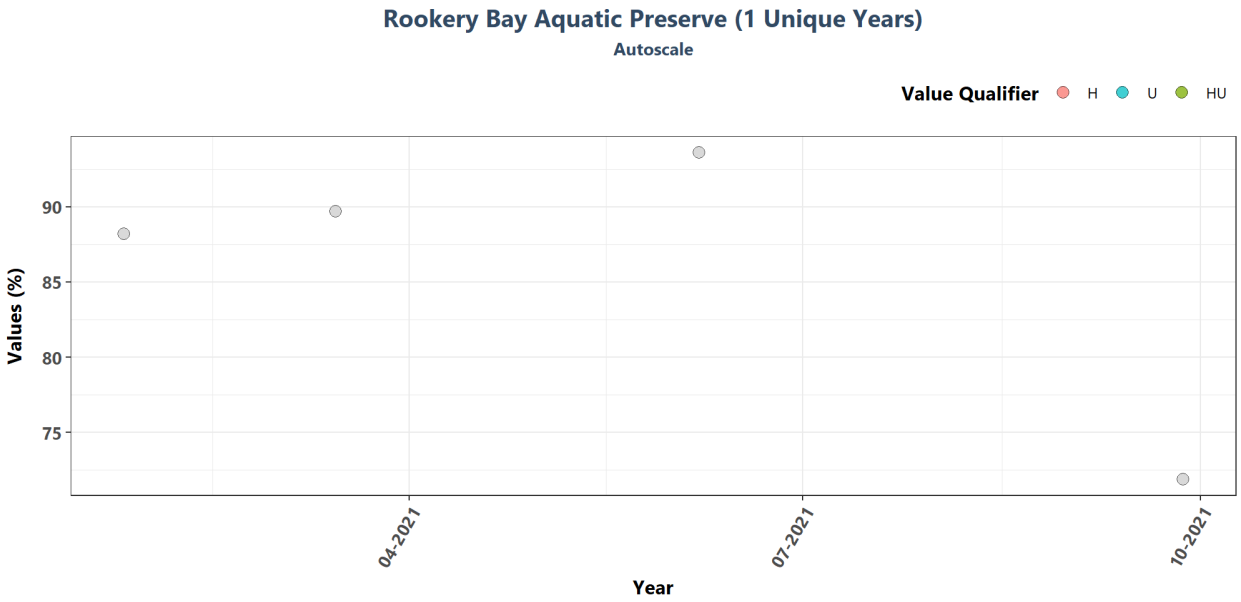

Mosquito Lagoon Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier H U HU
100
90
80
70
Values (%)
07-2020
10-2020
01-2021
04-2021
Year

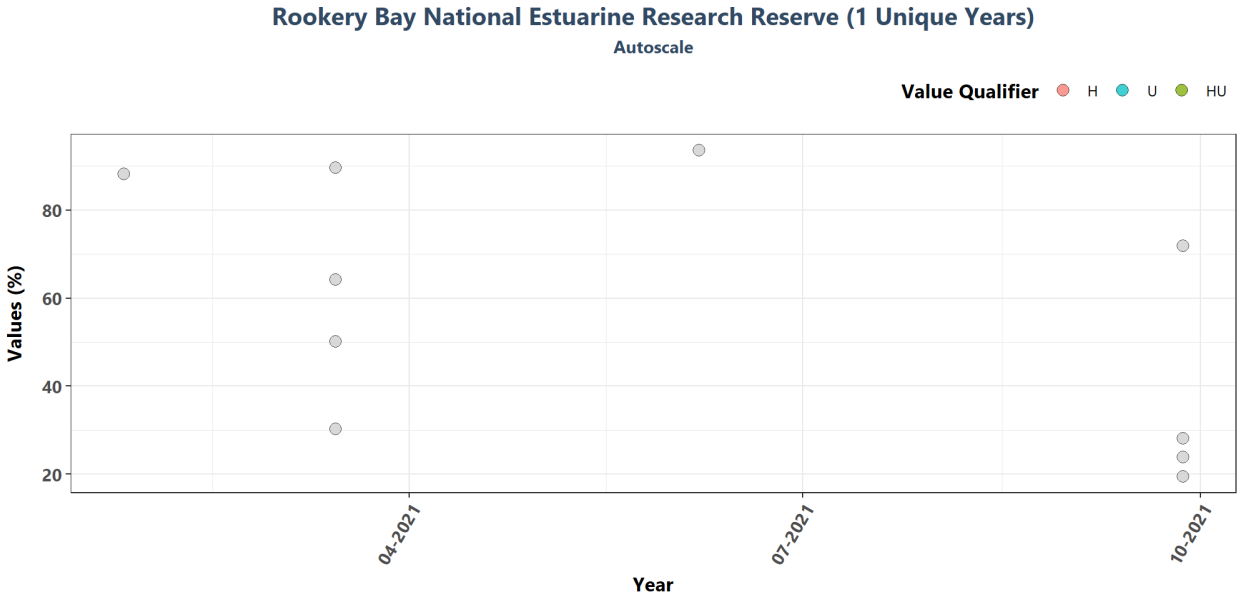

Nature Coast Aquatic Preserve (4 Unique Years)
Autoscale
Value Qualifier
H
U
HU
160
120
80
40
Values (%)
01-2016
01-2018
01-2020
01-2022
Year

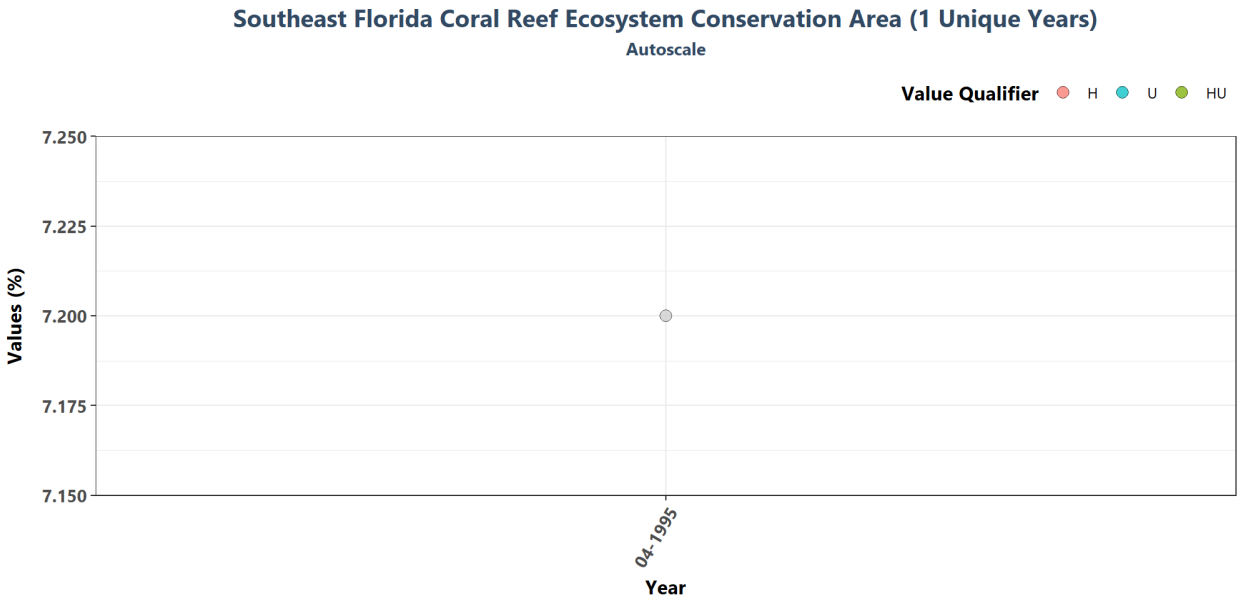

North Fork St. Lucie Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier
H
U
HU
75
50
25
Values (%)
07-2020
01-2021
07-2021
01-2022
Year

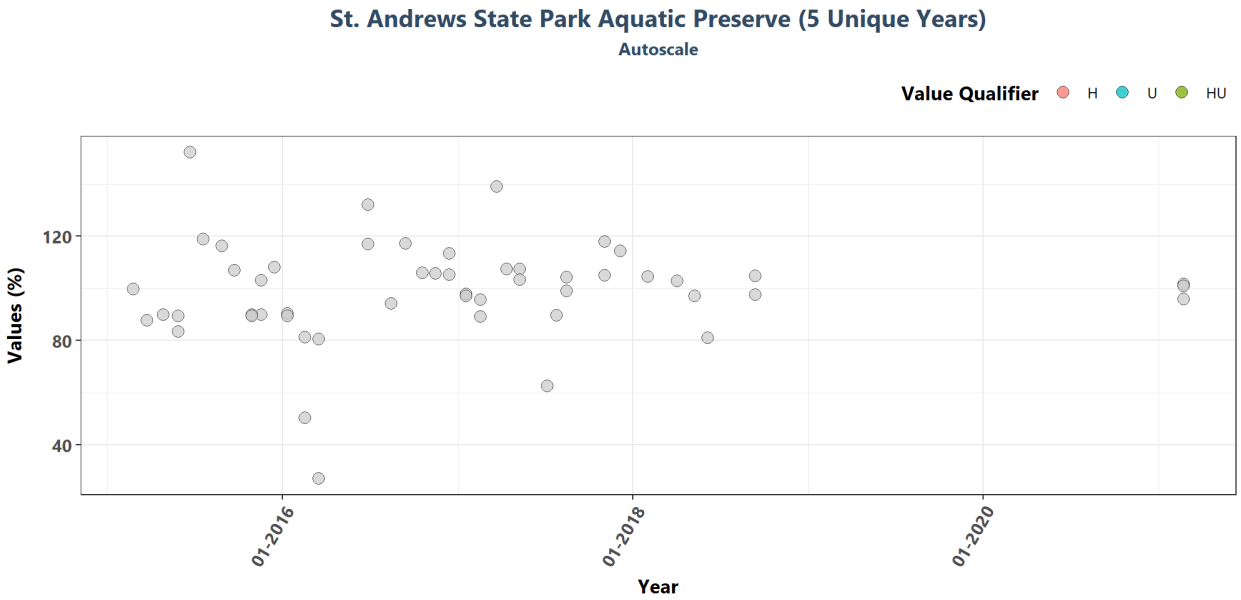

Pellicer Creek Aquatic Preserve (6 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
80
60
40
01-2005
01-2010
01-2015
01-2020
Year

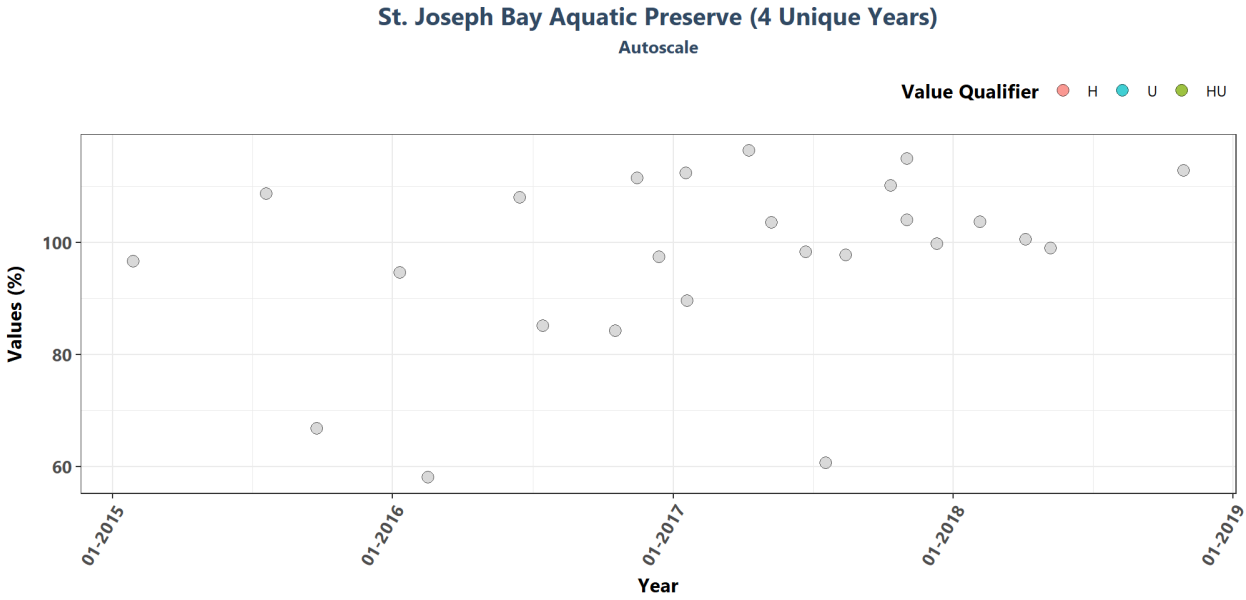

Pine Island Sound Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
90
80
70
60
50
07-2020
01-2021
07-2021
01-2022
Year

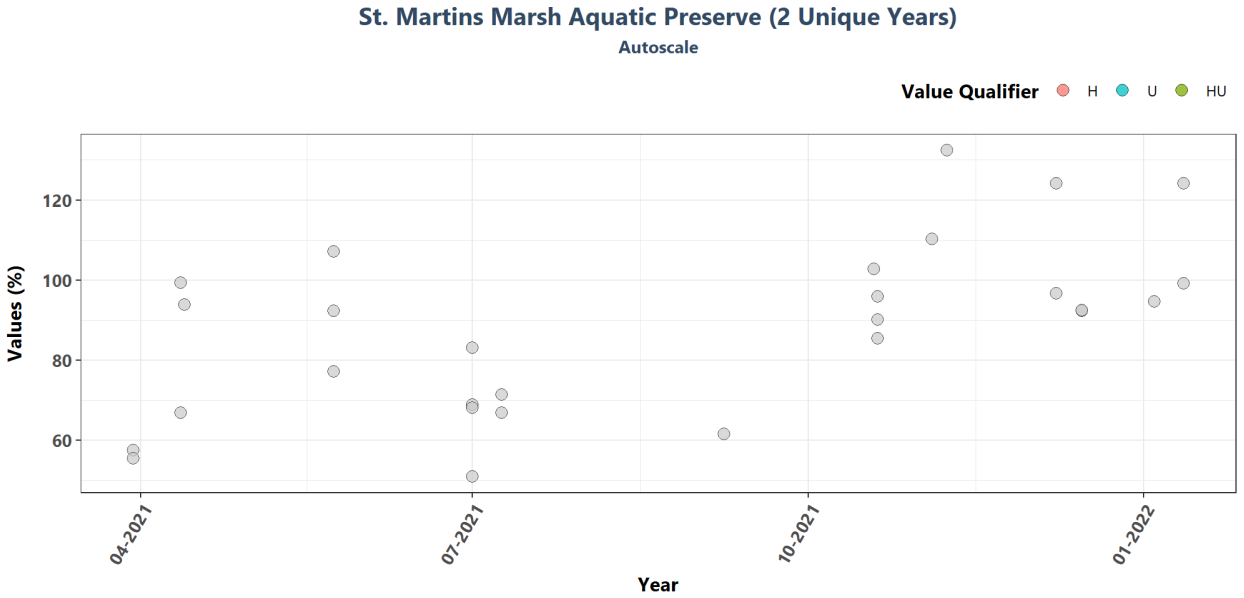

Rookery Bay Aquatic Preserve (1 Unique Years)
Autoscale
Value Qualifier
H
U
HU
Values (%)
90
85
80
75
04-2021
07-2021
10-2021
Year

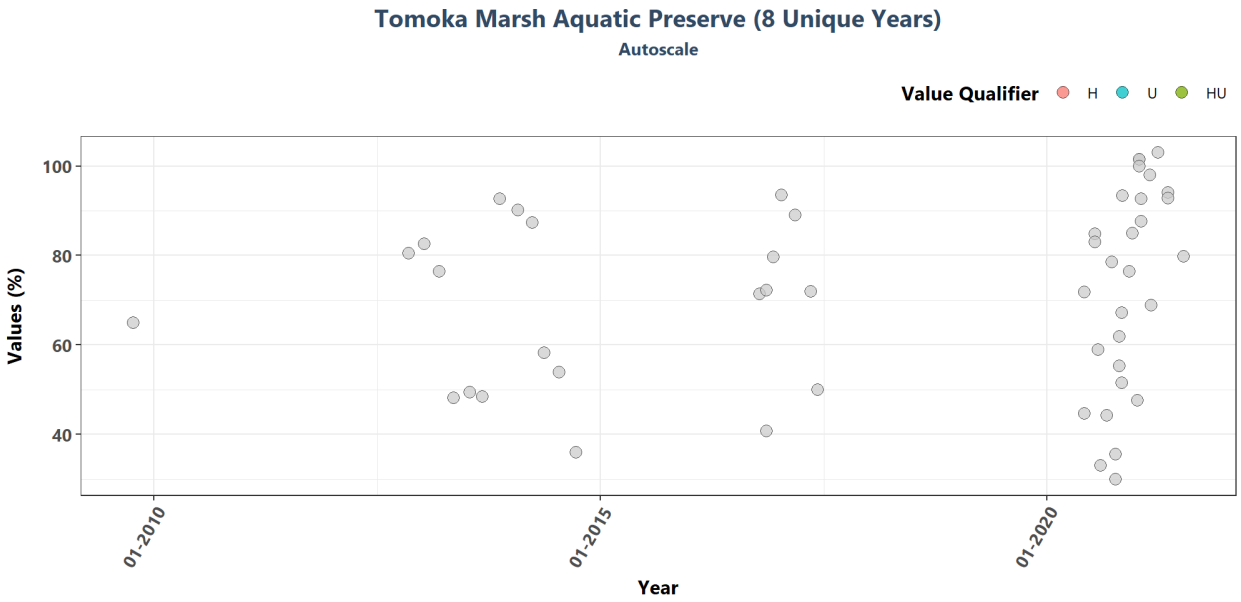

Rookery Bay National Estuarine Research Reserve (1 Unique Years)
Autoscale
Value Qualifier
H
U
HU
Values (%)
80
60
40
20
04-2021
07-2021
10-2021
Year

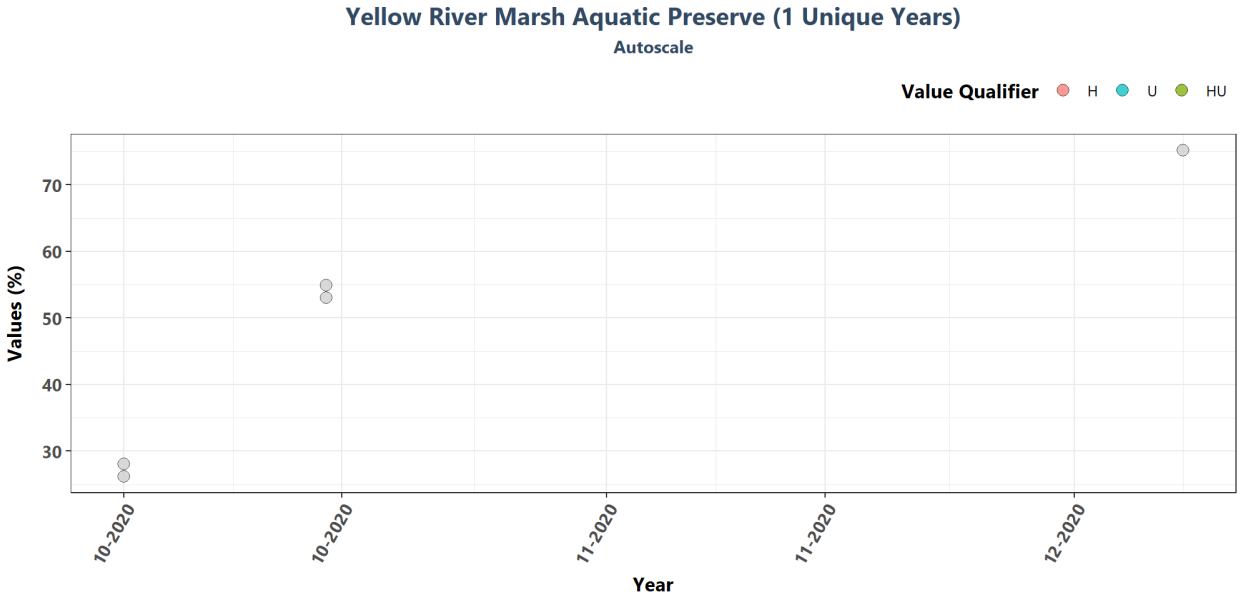

Southeast Florida Coral Reef Ecosystem Conservation Area (1 Unique Years)
Autoscale
Value Qualifier H U HU
7.250
7.225
7.200
7.175
7.150
Values (%)
04-1995
Year


St. Andrews State Park Aquatic Preserve (5 Unique Years)
Autoscale
Value Qualifier H U HU
120
80
40
Values (%)
01-2016
01-2018
01-2020
Year


St. Joseph Bay Aquatic Preserve (4 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
100
80
60
01-2015
01-2016
01-2017
01-2018
01-2019
Year


St. Martins Marsh Aquatic Preserve (2 Unique Years)
Autoscale
Value Qualifier H U HU
Values (%)
120
100
80
60
04-2021
07-2021
10-2021
01-2022
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

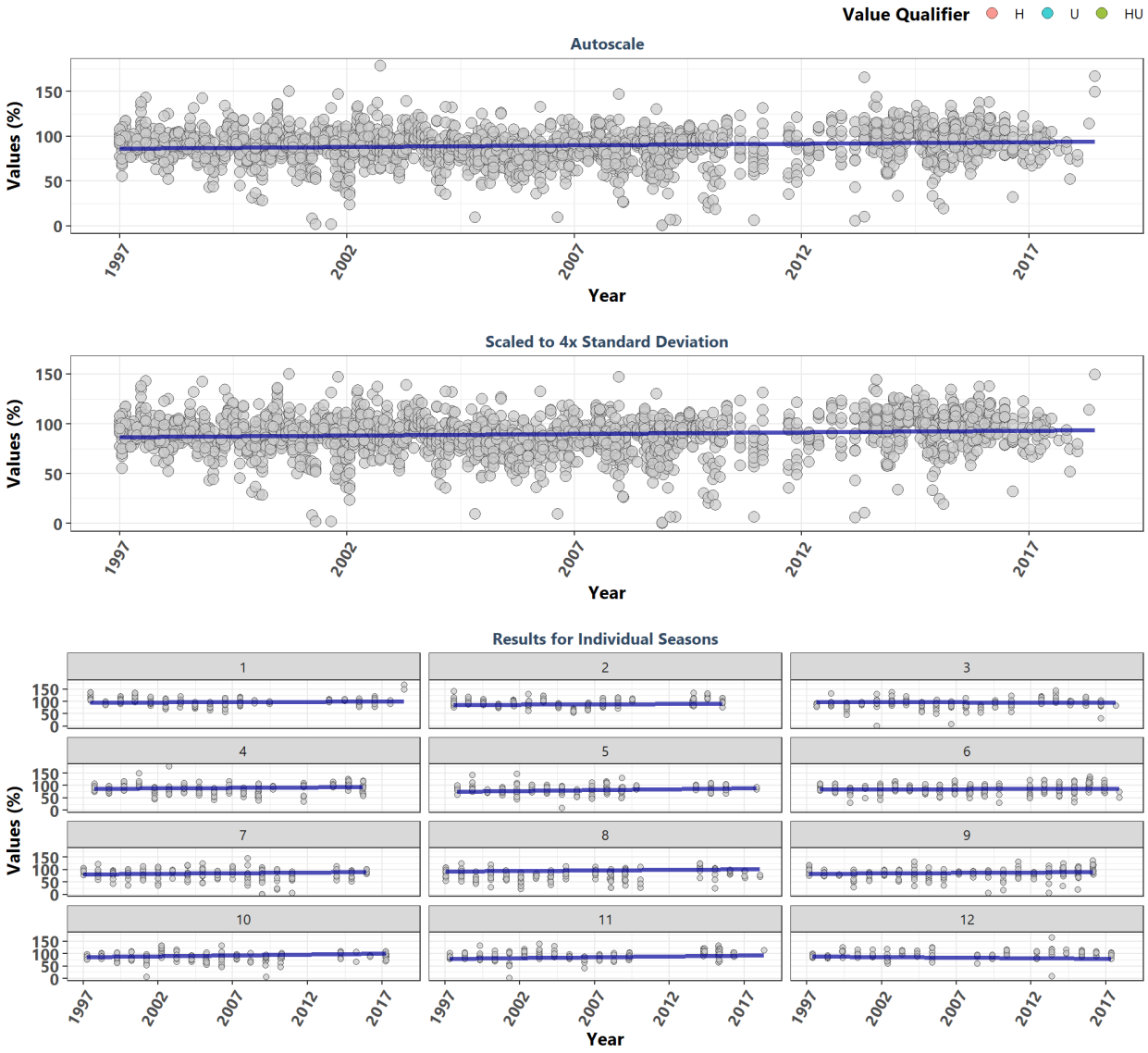
5. Set the plot type as a point plot with the size of the points
6. Add the linear trend
7. Create the title, x-axis, y-axis, and color fill labels
8. Set the y and x limits
9. Make the axis labels bold
10. Plot the arrangement as a set of panels

```
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 = relyear_dd,
                                                                                          y = relyear_dd,
                                                                                          xend = relyear_dd,
                                                                                          yend = relyear_dd),
                 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 = relyear_dd,
                                                                                          y = relyear_dd,
                                                                                          xend = relyear_dd,
                                                                                          yend = relyear_dd),
                 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 = relyear_dd,
                                                                                              y = relyear_dd,
                                                                                              xend = relyear_dd,
                                                                                              yend = relyear_dd,
                      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)
   }
}
```

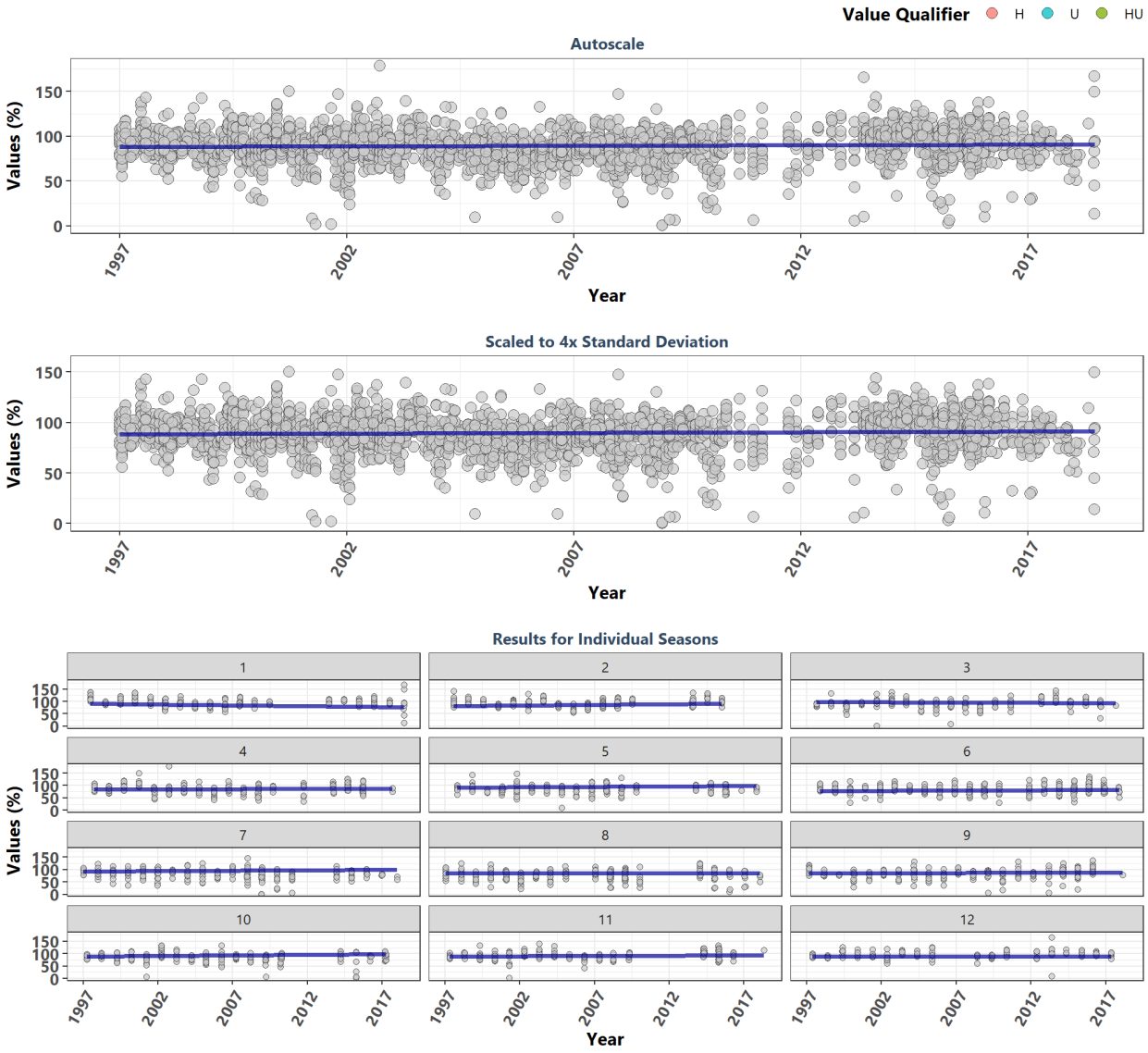
Apalachicola Bay Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	2094	90.00	0.0755	0.3400	86.5378	5.2	0.0000	26.9	0.0047	1
1	158	97.35	0.0795	0.2500	94.5000	-0.7	0.4704	NA	NA	1
2	169	96.40	0.0482	0.2594	86.7356	1.8	0.0692	NA	NA	1
3	175	90.60	-0.0387	-0.1714	98.7214	1.7	0.0805	NA	NA	-1
4	180	89.20	0.0889	0.3177	87.4229	1.0	0.3365	NA	NA	1
5	161	85.50	0.1140	0.6364	75.1091	2.1	0.0368	NA	NA	1
6	210	87.10	0.0421	0.1650	83.4800	1.7	0.0866	NA	NA	1
7	177	85.50	0.1106	0.5000	81.0000	-1.8	0.0779	NA	NA	1
8	165	80.20	0.0940	0.4615	92.2462	2.2	0.0295	NA	NA	1
9	198	85.60	0.0794	0.3793	82.9279	3.3	0.0010	NA	NA	1
10	165	84.80	0.1995	0.7400	86.3400	0.8	0.4224	NA	NA	1
11	183	93.00	0.1567	0.6923	78.6769	4.0	0.0001	NA	NA	1
12	153	96.50	-0.0891	-0.4750	89.7750	1.5	0.1446	NA	NA	-1

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

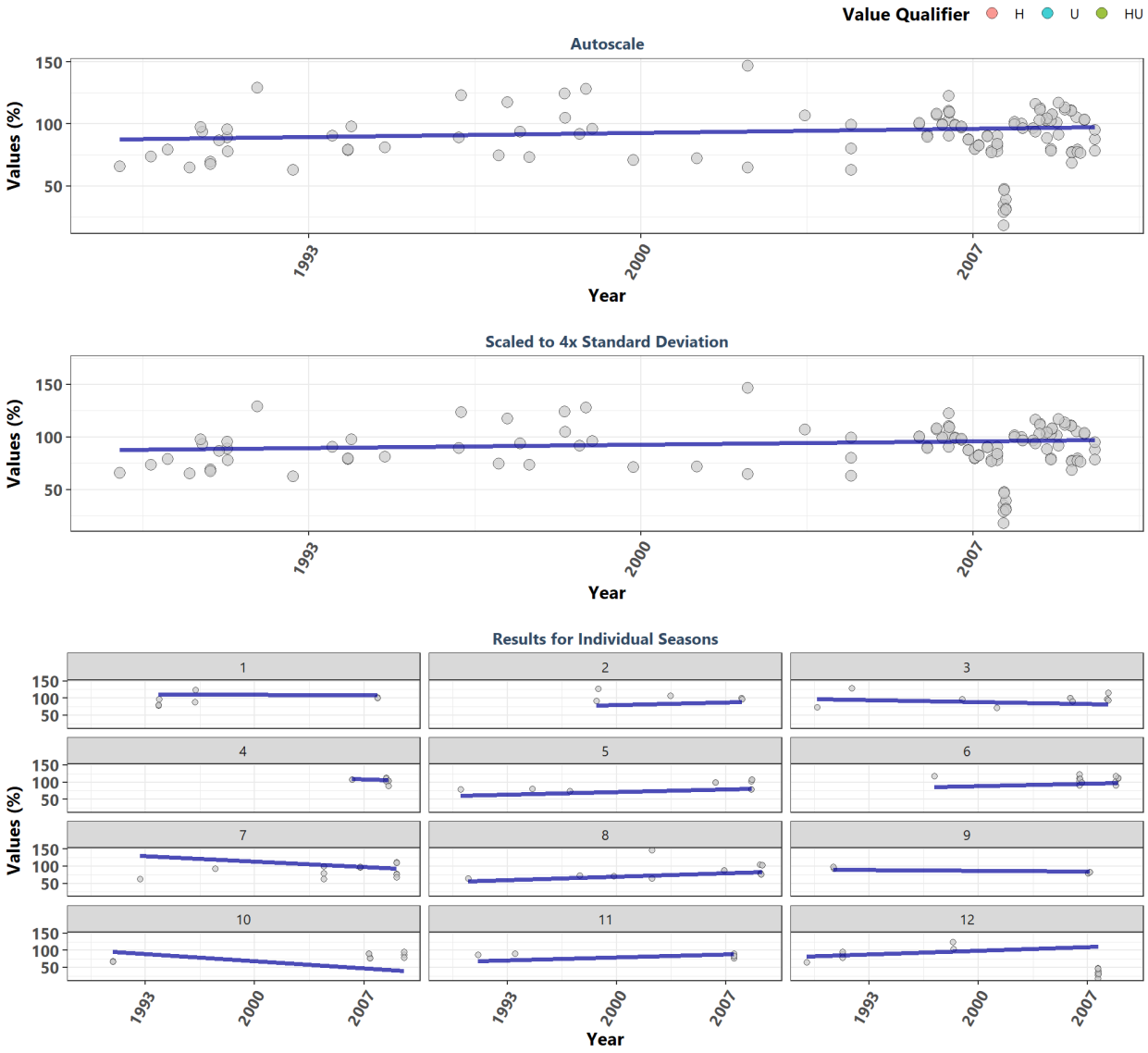
Apalachicola National Estuarine Research Reserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	2571	89.30	0.0302	0.1250	88.3169	2.3	0.0201	35.5	0.0002	1
1	202	95.40	-0.1408	-0.6750	90.6750	-1.6	0.1004	NA	NA	-1
2	205	95.00	0.1276	0.5333	80.2167	1.8	0.0679	NA	NA	1
3	207	90.70	-0.0776	-0.2857	97.9714	1.1	0.2897	NA	NA	-1
4	220	89.40	0.0164	0.0667	85.5833	-0.3	0.7731	NA	NA	1
5	204	86.25	0.0856	0.3933	91.4600	0.3	0.7280	NA	NA	1
6	255	86.20	0.0407	0.2000	77.7500	0.9	0.3876	NA	NA	1
7	221	84.60	0.0826	0.2429	93.4571	-3.1	0.0018	NA	NA	1
8	212	79.35	0.0031	0.0100	85.1750	0.9	0.3775	NA	NA	1
9	232	85.55	0.0363	0.1500	84.5500	2.9	0.0038	NA	NA	1
10	208	85.25	0.1466	0.5185	87.6338	0.1	0.9475	NA	NA	1
11	228	92.30	0.0494	0.1700	89.0000	3.3	0.0009	NA	NA	1
12	177	95.40	-0.0131	-0.0706	90.1059	1.6	0.1021	NA	NA	-1

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

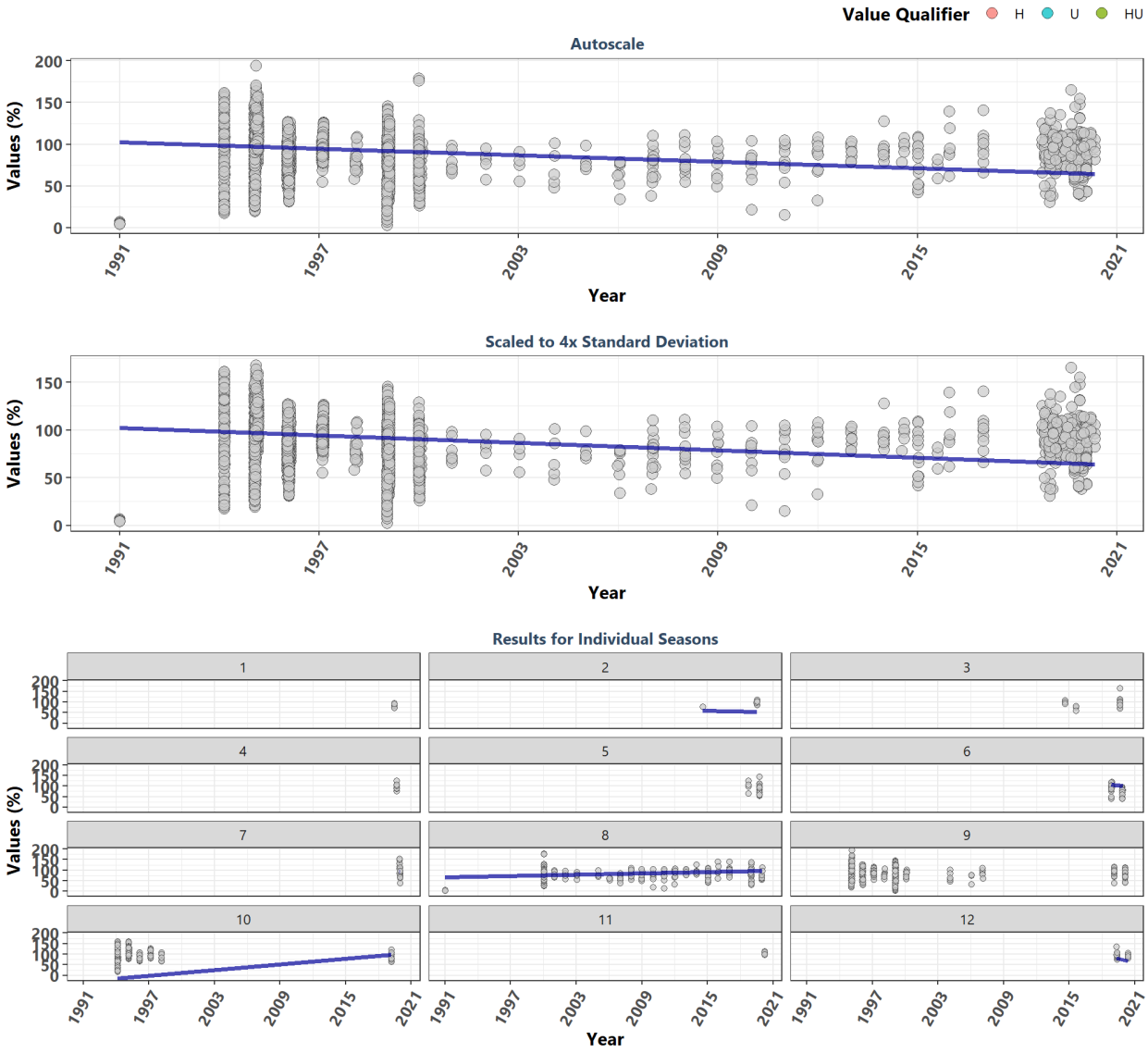
Banana River Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	120	91.10	0.0586	1.1000	64.2424	0.9	0.3521	23.9	0.013	0
1	8	93.45	-0.0110	-0.1200	112.7300	1.8	0.0787	NA	NA	0
2	6	98.20	0.2121	2.4179	16.7821	-0.4	0.6762	NA	NA	0
3	12	96.30	-0.5333	-2.0357	141.6357	0.8	0.4313	NA	NA	0
4	9	NA	-0.1667	-4.2000	230.3000	NA	NA	NA	NA	NA
5	10	90.40	0.4444	2.4062	8.7188	1.4	0.1477	NA	NA	0
6	14	109.25	0.1818	2.4902	24.0848	0.0	1.0000	NA	NA	0
7	12	86.90	-0.2000	-5.2000	249.0000	0.9	0.3473	NA	NA	0
8	13	79.30	0.3205	3.2429	-14.7429	1.6	0.1200	NA	NA	0
9	6	NA	-0.2381	-0.7000	104.4000	NA	NA	NA	NA	NA
10	9	78.50	-0.4066	-7.0286	250.8786	1.7	0.0908	NA	NA	0
11	7	84.10	0.3556	3.0500	0.4250	-0.8	0.4470	NA	NA	0
12	14	47.05	0.5000	3.5833	5.6583	-2.2	0.0266	NA	NA	0

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

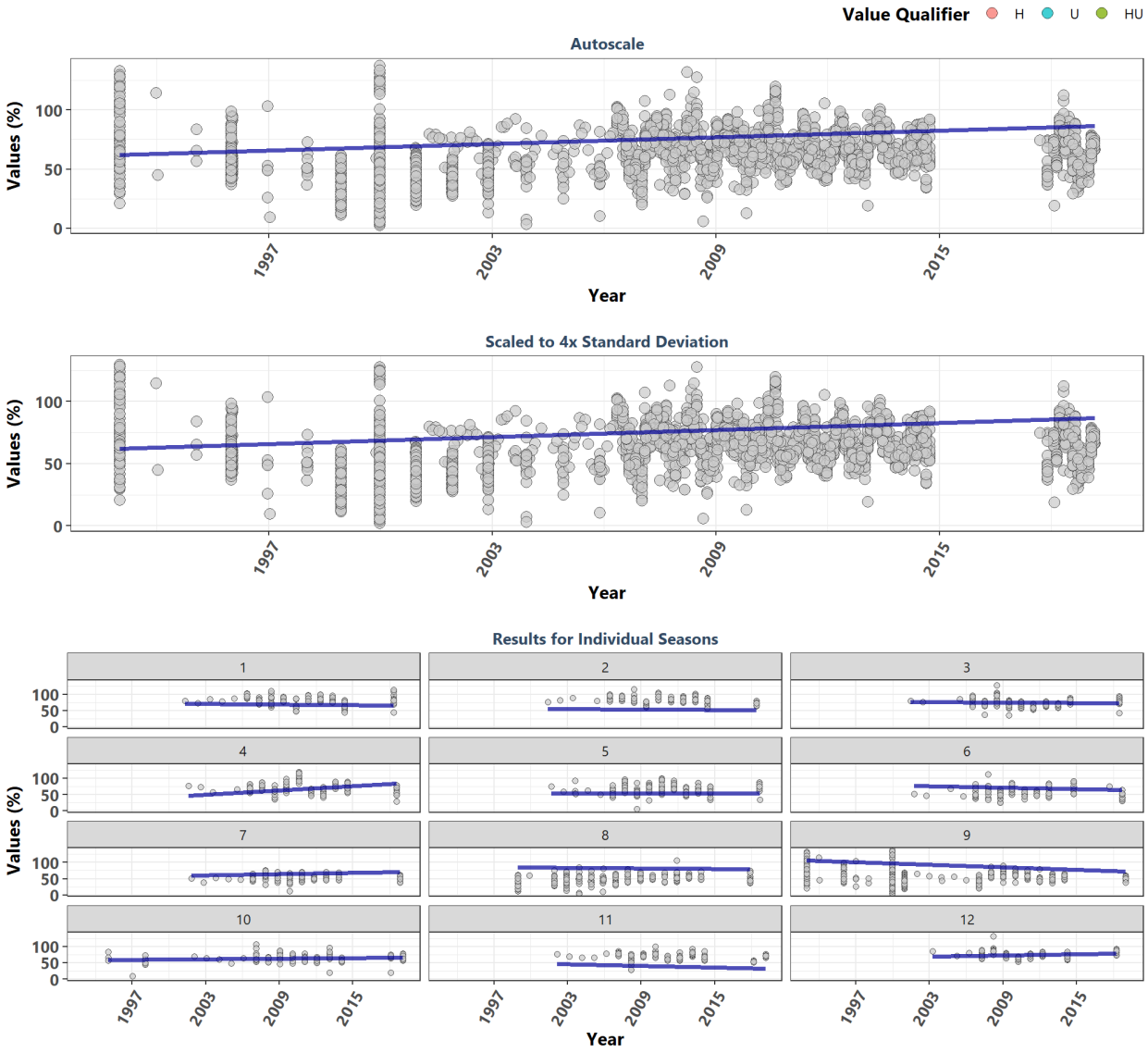
Boca Ciega Bay Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	3756	87.1	-0.0654	-1.3000	102.1500	-7.1	0	107	0	-1
1	8	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	16	NA	-0.0866	-1.4000	92.5000	NA	NA	NA	NA	NA
3	20	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	7	NA	-0.0797	-3.9000	206.0000	NA	NA	NA	NA	NA
5	25	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	39	NA	-0.0667	-6.3500	283.2500	NA	NA	NA	NA	NA
7	21	NA	0.1702	0.9947	65.5474	NA	NA	NA	NA	NA
8	385	74.5	0.0947	1.0600	66.0600	5.6	0	NA	NA	1
9	2641	NA	NA	NA	NA	NA	NA	NA	NA	NA
10	564	104.6	0.1250	4.4200	-28.0300	-4.6	0	NA	NA	1
11	6	NA	-0.1209	-1.8000	111.8000	NA	NA	NA	NA	NA
12	24	NA	-0.3009	-15.3000	515.2000	NA	NA	NA	NA	NA

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

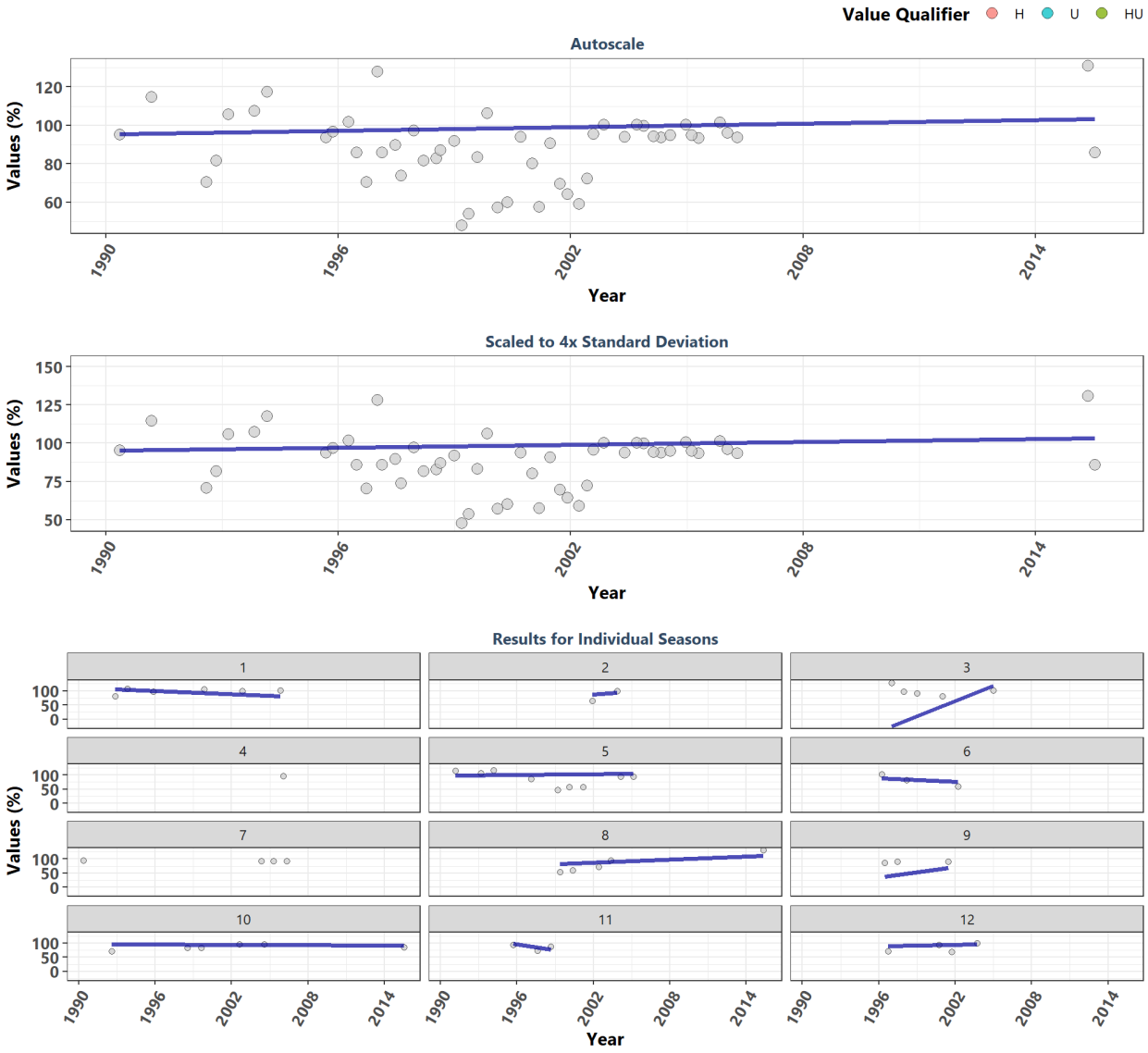
Cockroach Bay Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	3845	53.3	0.0701	0.9375	60.1125	12.6	0.0000	483.6	0	1
1	143	81.3	-0.0597	-0.3429	74.9000	-0.8	0.4245	NA	NA	-1
2	139	81.5	-0.0456	-0.1750	56.4000	-5.4	0.0000	NA	NA	-1
3	147	73.2	-0.0469	-0.2000	77.4000	-0.8	0.3978	NA	NA	-1
4	155	67.7	0.3419	2.1750	23.8750	-1.1	0.2681	NA	NA	1
5	163	65.6	-0.0186	-0.1000	55.7000	1.8	0.0677	NA	NA	-1
6	141	53.7	-0.2099	-0.7437	85.2167	-0.3	0.7427	NA	NA	-1
7	142	52.9	0.0960	0.5400	54.8000	-0.8	0.4183	NA	NA	1
8	1337	47.8	-0.0449	-0.2333	86.2000	19.7	0.0000	NA	NA	-1
9	1024	48.3	-0.3102	-1.3000	108.8000	-6.1	0.0000	NA	NA	-1
10	168	64.2	0.0955	0.2750	58.4250	1.8	0.0652	NA	NA	1
11	166	69.6	-0.1224	-0.8813	56.2312	-4.0	0.0001	NA	NA	-1
12	120	73.8	0.1312	0.6000	61.8000	2.1	0.0322	NA	NA	1

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

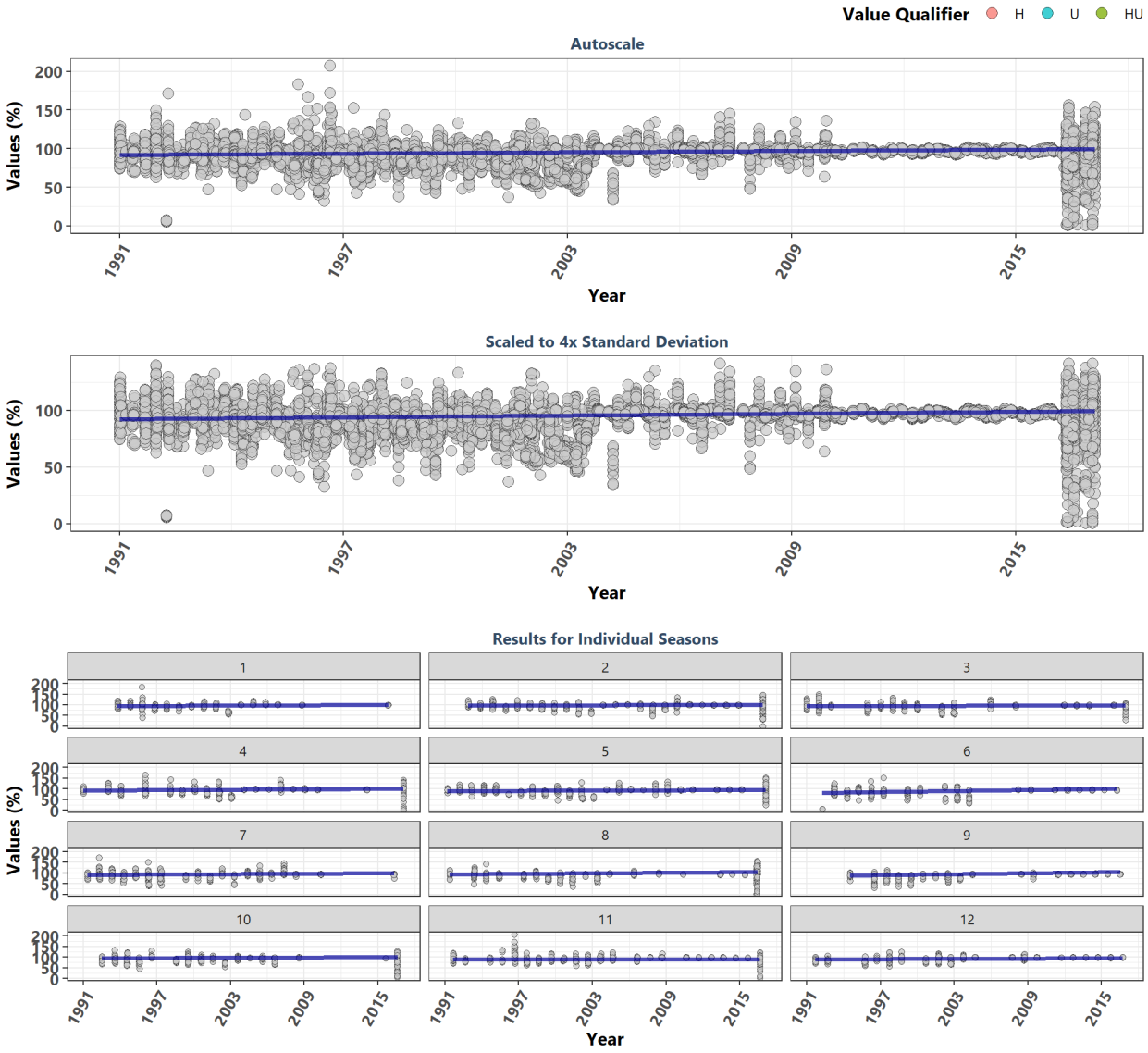
Coupon Bight Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	51	93.50	0.1093	0.3139	95.1853	0.5	0.6301	18.7	0.0447	0
1	6	100.79	-0.2222	-1.8153	110.5219	0.4	0.7071	NA	NA	0
2	2	NA	0.3333	3.1744	48.7727	NA	NA	NA	NA	NA
3	5	97.23	1.0000	17.8006	-149.5254	-0.7	0.4624	NA	NA	0
4	1	NA	-0.4000	-5.5009	146.7413	NA	NA	NA	NA	NA
5	9	94.18	0.2000	0.4826	96.9244	-0.7	0.4655	NA	NA	0
6	3	81.59	-0.3333	-2.2086	102.4541	-1.0	0.2963	NA	NA	0
7	4	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	5	72.09	0.6000	1.8224	65.4527	2.2	0.0275	NA	NA	0
9	3	89.74	1.0000	6.0967	-1.0652	1.0	0.2963	NA	NA	0
10	6	84.59	-0.6667	-0.1099	95.1853	1.5	0.1329	NA	NA	0
11	3	86.99	-1.0000	-7.1548	138.8283	0.0	1.0000	NA	NA	0
12	4	82.10	1.0000	0.9680	82.9590	0.3	0.7341	NA	NA	0

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

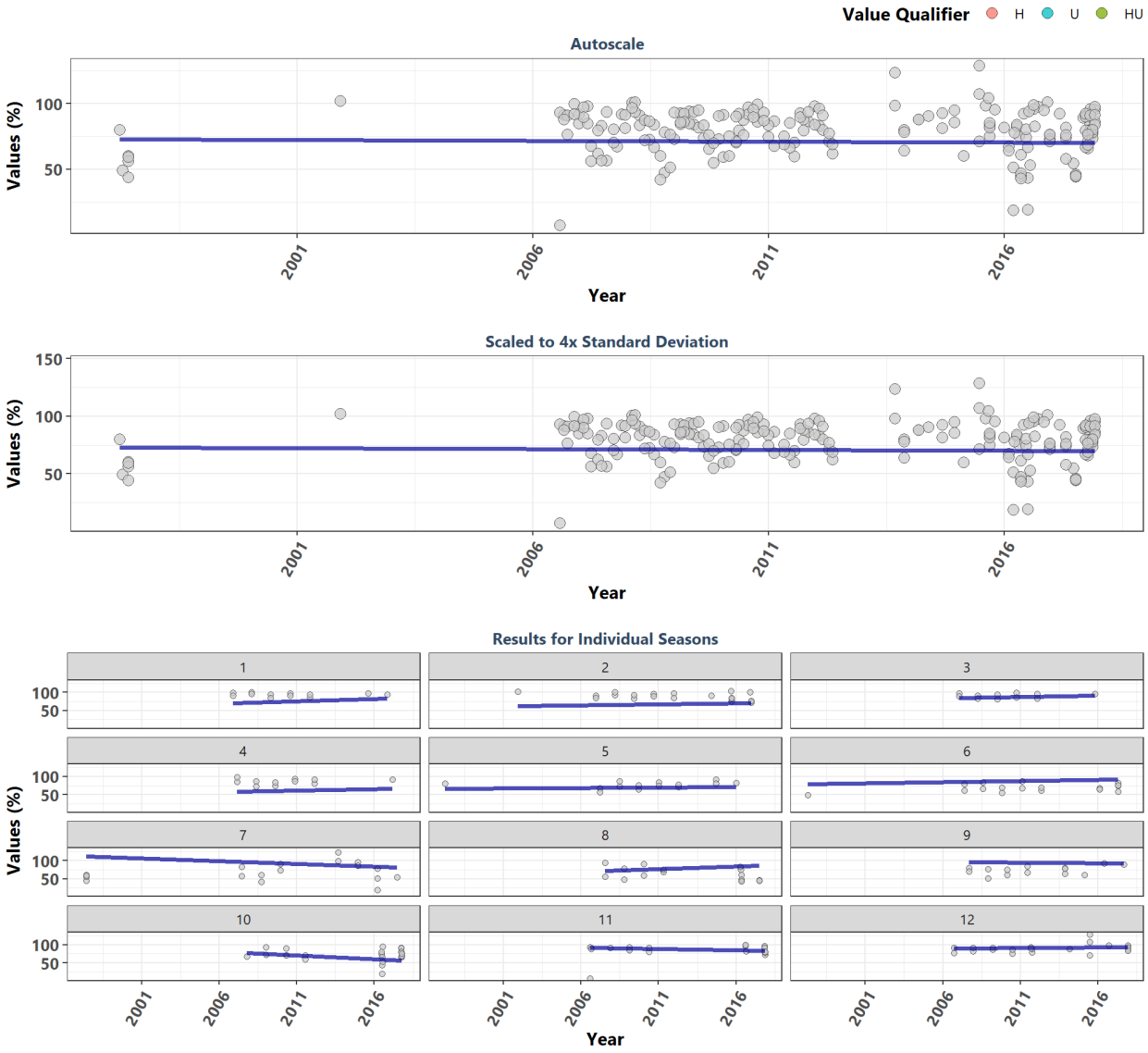
Florida Keys National Marine Sanctuary



Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	13161	94.80	0.1785	0.2719	92.0732	29.6	0.0000	439	0	1
1	989	98.64	0.1851	0.2958	93.4271	12.5	0.0000	NA	NA	1
2	1169	98.50	0.0689	0.0995	96.8464	5.5	0.0000	NA	NA	1
3	1084	98.04	0.1739	0.1948	93.1851	3.4	0.0007	NA	NA	1
4	926	97.27	0.1436	0.2739	93.4620	8.5	0.0000	NA	NA	1
5	1416	95.91	0.1411	0.1973	91.1025	9.8	0.0000	NA	NA	1
6	822	92.83	0.5010	0.8020	79.7275	7.8	0.0000	NA	NA	1
7	1358	93.47	0.1815	0.3261	89.8987	7.8	0.0000	NA	NA	1
8	1079	89.30	0.2656	0.5087	93.0438	0.1	0.9370	NA	NA	1
9	1075	94.16	0.3329	0.6557	87.5387	24.7	0.0000	NA	NA	1
10	1179	92.10	0.1063	0.1343	96.0824	5.3	0.0000	NA	NA	1
11	1153	96.75	0.0016	0.0029	89.2647	7.3	0.0000	NA	NA	1
12	911	98.03	0.1022	0.2171	89.9285	15.1	0.0000	NA	NA	1

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

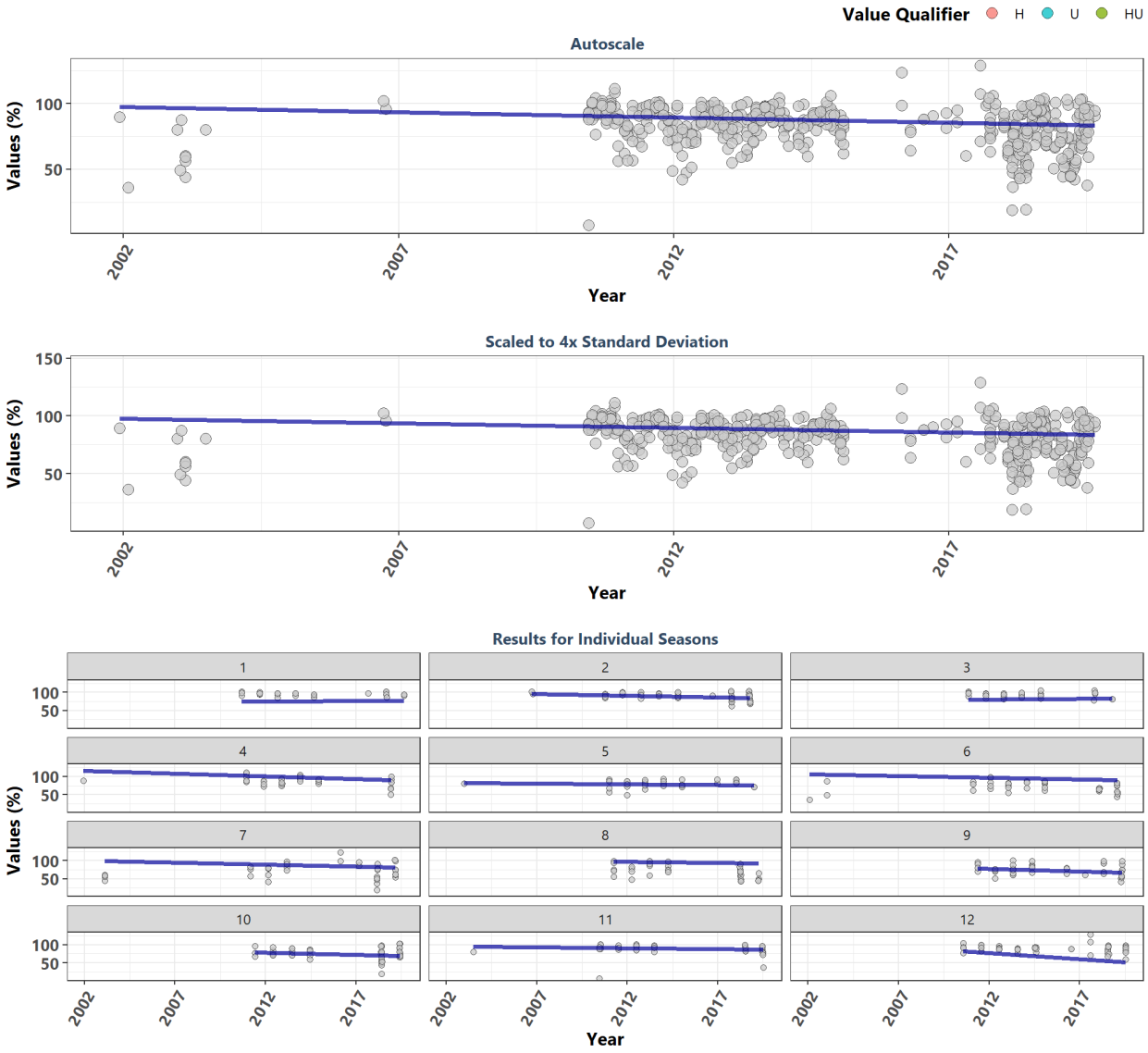
Guana River Marsh Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	211	81.50	-0.0125	-0.1762	73.9688	-0.7	0.4950	23.3	0.016	0
1	12	94.40	0.1978	1.6571	47.5286	-0.3	0.7813	NA	NA	0
2	23	85.00	0.1397	0.7119	55.3976	-2.8	0.0044	NA	NA	0
3	11	90.70	0.2134	0.7625	74.8375	0.1	0.9370	NA	NA	0
4	11	87.00	0.1111	1.1800	41.0700	0.4	0.6926	NA	NA	0
5	14	76.40	0.0500	0.3667	63.4000	1.7	0.0975	NA	NA	0
6	17	67.50	0.1091	0.8688	73.1000	0.8	0.4524	NA	NA	0
7	18	59.95	-0.4229	-1.9500	124.0000	0.6	0.5394	NA	NA	0
8	19	59.30	0.3407	1.7750	47.1125	-2.4	0.0184	NA	NA	0
9	14	75.70	-0.0758	-0.2143	97.9357	0.9	0.3464	NA	NA	0
10	25	71.10	-0.3860	-2.5857	113.6000	0.3	0.7310	NA	NA	0
11	24	85.00	-0.1812	-0.8381	102.6000	-1.3	0.1970	NA	NA	0
12	23	87.80	0.0364	0.2900	86.0600	1.4	0.1560	NA	NA	0

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

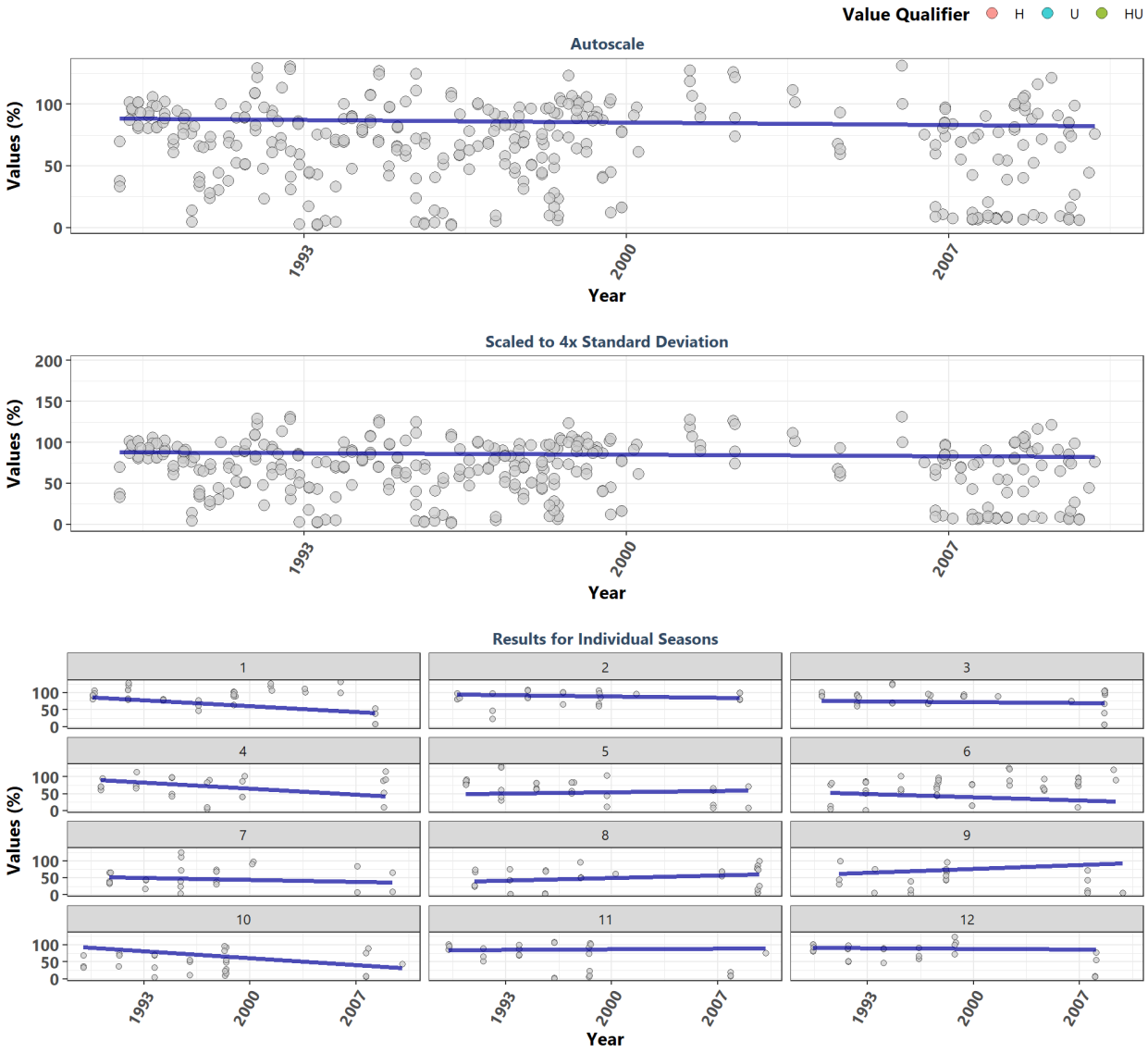
Guana Tolomato Matanzas National Estuarine Research Reserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	501	85.00	-0.1623	-0.7944	101.4083	-5.4	0.0000	18.2	0.0777	-1
1	36	95.75	0.0564	0.2464	71.7179	-2.2	0.0300	NA	NA	1
2	55	93.00	-0.2525	-0.9111	105.2778	-3.3	0.0010	NA	NA	-1
3	31	95.00	0.0471	0.3000	76.1000	-1.2	0.2211	NA	NA	1
4	35	87.00	-0.2990	-1.4600	122.2000	-1.7	0.0832	NA	NA	-1
5	35	81.20	-0.0359	-0.3514	83.9800	0.4	0.6984	NA	NA	-1
6	42	74.60	-0.1548	-0.8700	109.7900	-1.9	0.0624	NA	NA	-1
7	40	76.40	-0.2034	-1.0917	105.5583	0.5	0.6107	NA	NA	-1
8	39	70.00	-0.2508	-0.5333	104.8167	-3.5	0.0005	NA	NA	-1
9	39	75.20	-0.1986	-1.3100	96.8700	-1.0	0.3024	NA	NA	-1
10	50	76.60	-0.1147	-1.0500	93.0500	-0.4	0.7082	NA	NA	-1
11	47	90.70	-0.1267	-0.5000	98.0000	-2.6	0.0106	NA	NA	-1
12	52	89.50	-0.3779	-3.4429	128.5286	-1.3	0.1826	NA	NA	-1

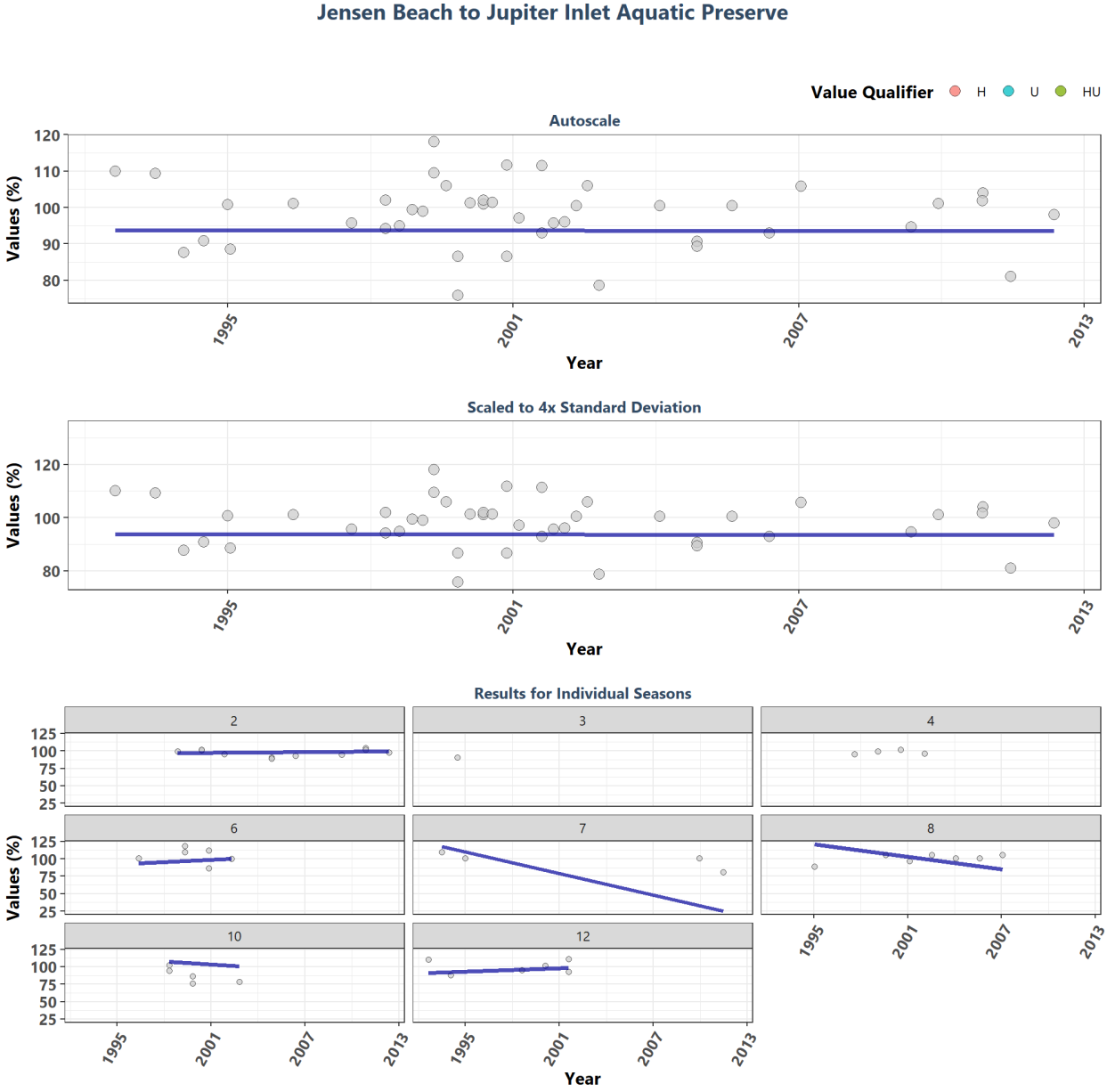
[a] $p < 0.00005$ appear as 0 due to rounding

Indian River-Malabar to Vero Beach Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	350	73.65	-0.0518	-0.6600	102.3718	-1.0	0.2946	23.4	0.0153	0
1	37	90.50	-0.3290	-5.8000	210.7000	-0.6	0.5168	NA	NA	0
2	24	86.80	-0.0751	-1.3000	123.0000	0.6	0.5793	NA	NA	0
3	30	89.13	-0.0367	-0.8250	93.1250	-0.6	0.5366	NA	NA	0
4	25	72.50	-0.2759	-5.8062	215.7437	-0.2	0.8121	NA	NA	0
5	31	71.50	0.1885	1.4521	16.7979	-2.6	0.0087	NA	NA	0
6	41	77.00	-0.2035	-3.2509	124.0223	2.2	0.0269	NA	NA	0
7	25	44.80	-0.0952	-1.9667	95.6167	0.6	0.5517	NA	NA	0
8	30	53.10	0.0867	2.4250	-13.4000	1.5	0.1376	NA	NA	0
9	22	42.75	0.2390	3.8333	-22.6667	-1.3	0.1817	NA	NA	0
10	28	46.45	-0.2249	-6.7500	237.0500	-0.7	0.4780	NA	NA	0
11	28	75.05	0.0833	0.6250	71.1750	-1.7	0.0909	NA	NA	0
12	29	82.20	-0.0805	-0.8000	109.1270	-2.1	0.0340	NA	NA	0

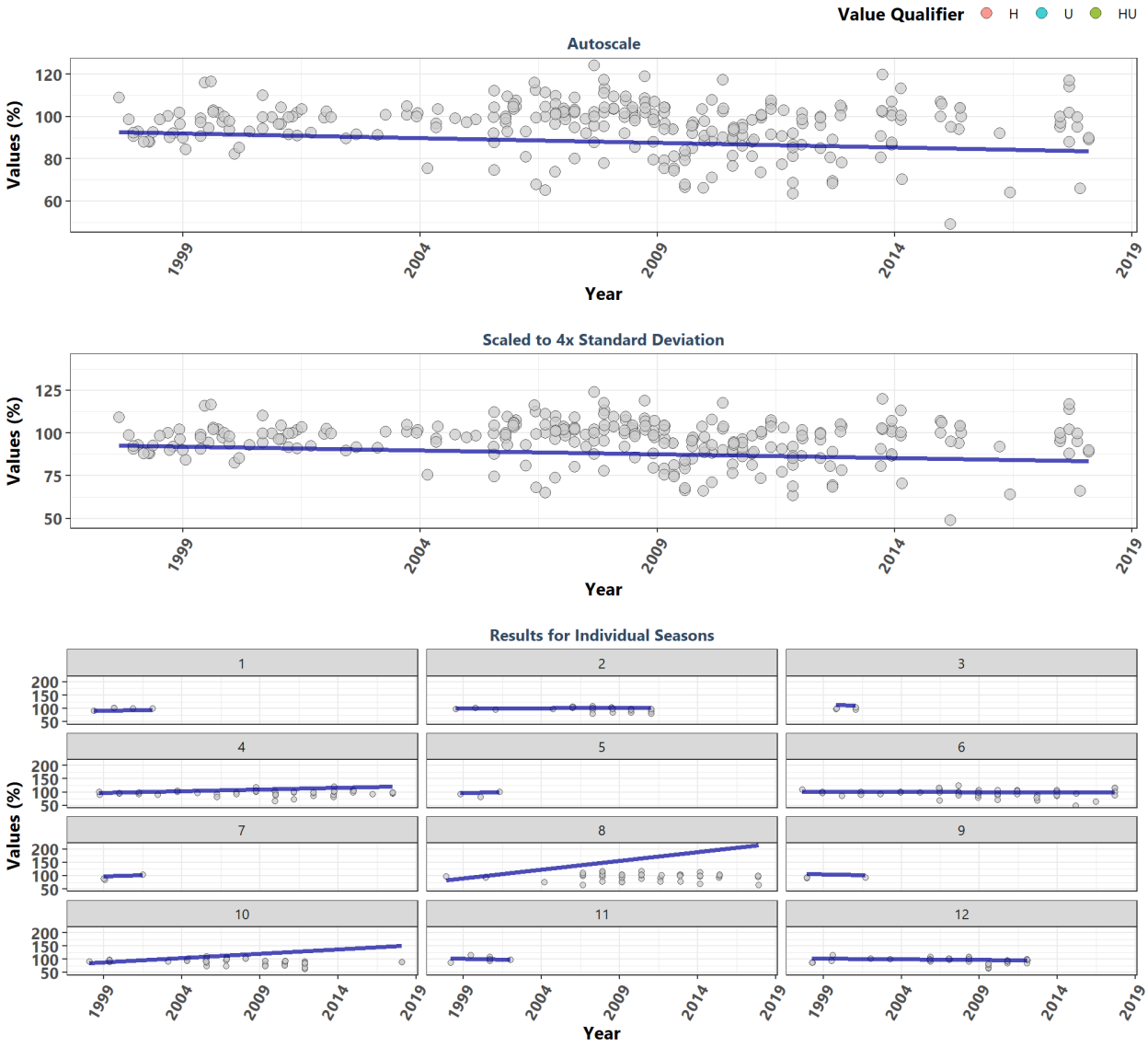
[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	44	99.90	-0.0620	-0.0100	93.8000	-0.1	0.9532	5.5	0.4773	0
2	11	98.00	0.0364	0.2000	93.8000	0.1	0.9374	NA	NA	0
3	1	NA	0.1333	0.6467	87.3800	NA	NA	NA	NA	NA
4	4	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	6	105.30	0.2381	1.7000	68.2000	-0.4	0.6967	NA	NA	0
7	4	100.90	-0.6000	-7.7000	217.5000	-1.0	0.3082	NA	NA	0
8	7	100.50	-0.2000	-4.5500	184.9250	0.6	0.5387	NA	NA	0
10	5	86.60	-0.6667	-2.0746	140.3178	-1.3	0.1917	NA	NA	0
12	6	98.05	0.3333	1.2667	75.3833	0.2	0.8483	NA	NA	0

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

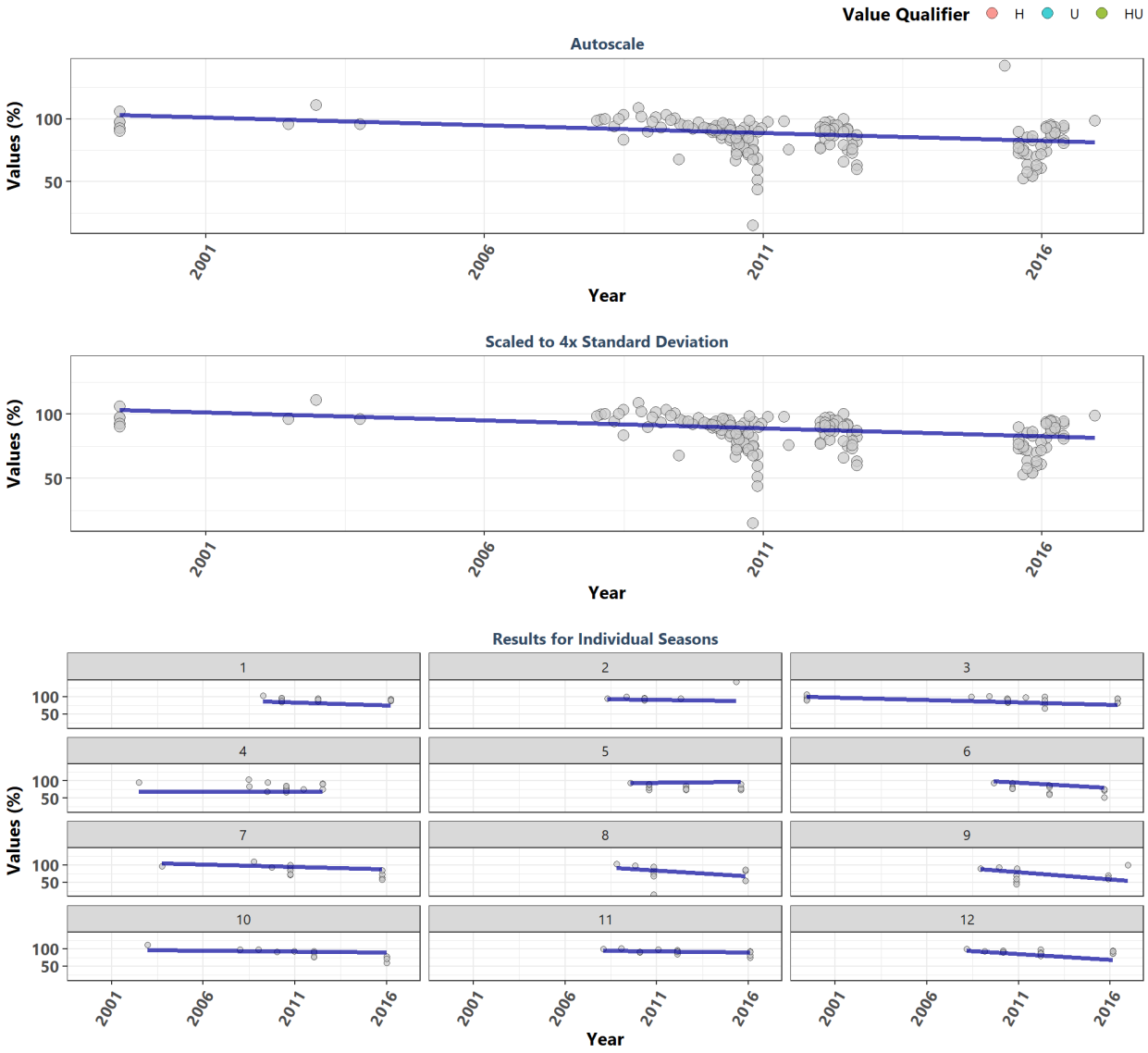
Loxahatchee River-Lake Worth Creek Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	269	98.1	-0.1176	-0.5556	97.0375	-3.0	0.0027	21.6	0.0274	-1
1	5	99.8	0.3333	0.3000	90.0000	0.0	1.0000	NA	NA	1
2	31	99.2	0.1000	0.2000	97.8000	-3.2	0.0016	NA	NA	1
3	4	NA	-0.3957	-2.2750	135.6000	NA	NA	NA	NA	NA
4	44	99.1	0.0000	1.6000	81.9500	0.6	0.5422	NA	NA	1
5	3	92.0	0.0645	0.1569	96.2750	0.0	1.0000	NA	NA	1
6	49	98.5	-0.0642	-0.2000	103.6000	-1.0	0.3280	NA	NA	-1
7	3	NA	0.0000	1.3500	86.1000	NA	NA	NA	NA	NA
8	43	100.0	0.6667	8.2000	16.0000	-0.6	0.5477	NA	NA	1
9	3	NA	-0.1933	-0.9111	113.0778	NA	NA	NA	NA	NA
10	43	92.9	0.3333	4.0500	51.5000	-2.7	0.0060	NA	NA	1
11	6	99.6	-0.2890	-1.3000	113.7000	0.0	1.0000	NA	NA	-1
12	35	98.5	-0.0969	-0.4900	106.8300	-1.6	0.1020	NA	NA	-1

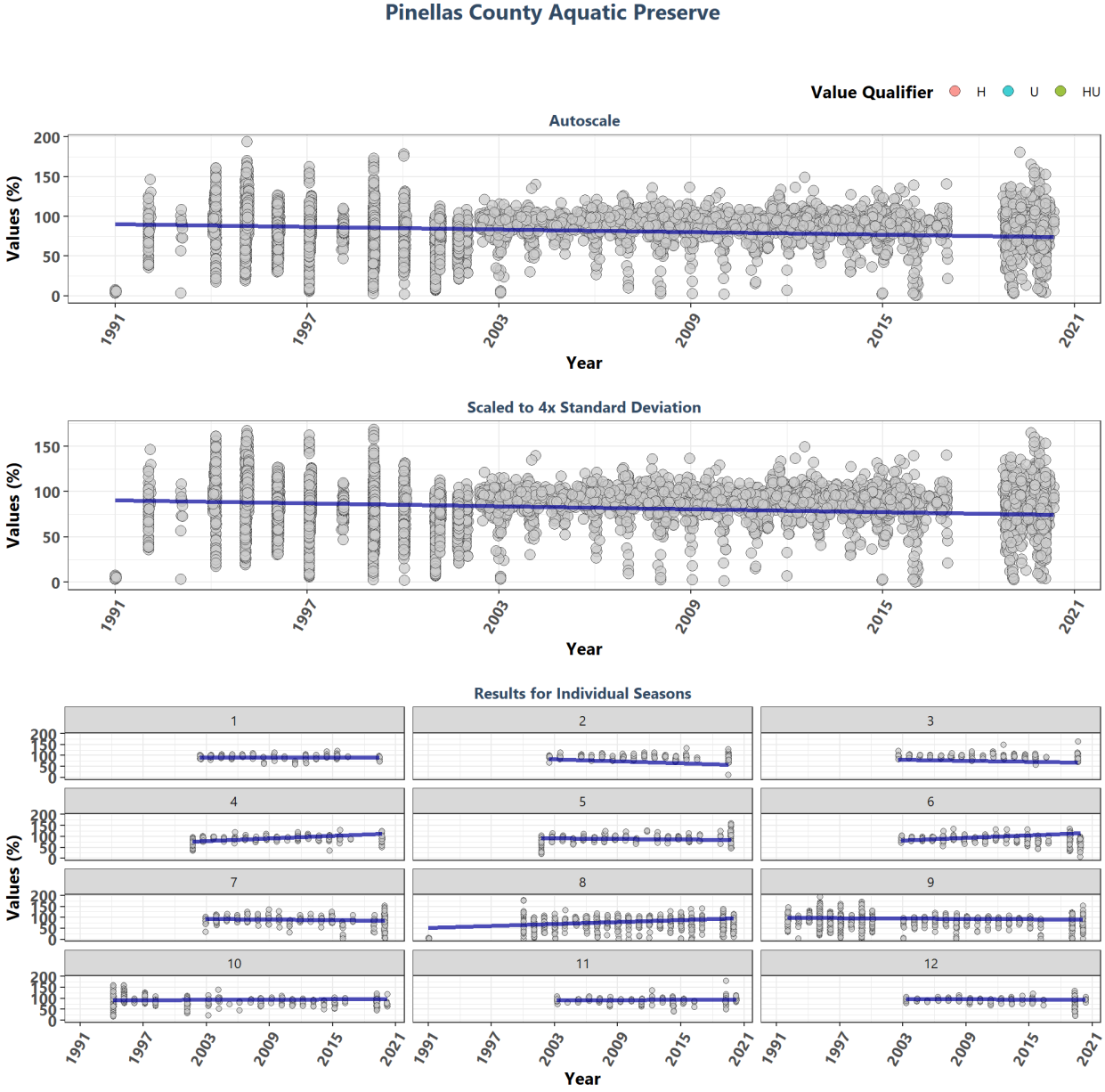
[a] $p < 0.00005$ appear as 0 due to rounding

Nassau River-St. Johns River Marshes Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	163	89.60	-0.2928	-1.2400	107.5825	-5.1	0.0000	12.7	0.3162	-1
1	13	91.20	-0.3048	-1.7200	109.6400	-0.8	0.4383	NA	NA	-1
2	9	94.90	-0.2190	-0.6000	100.2000	0.3	0.7296	NA	NA	-1
3	22	92.55	-0.1429	-1.4750	105.5250	-1.9	0.0641	NA	NA	-1
4	15	83.40	-0.0182	-0.0500	69.2500	-0.7	0.4693	NA	NA	-1
5	15	80.40	0.1111	0.4250	88.5250	-1.6	0.1036	NA	NA	1
6	13	77.40	-0.7121	-3.0154	139.0462	-2.4	0.0175	NA	NA	-1
7	12	84.85	-0.3516	-1.4167	114.8667	-2.6	0.0106	NA	NA	-1
8	12	78.65	-0.4872	-3.3750	134.7750	-1.5	0.1313	NA	NA	-1
9	11	68.50	-0.3182	-4.1400	140.7500	0.0	1.0000	NA	NA	-1
10	12	90.80	-0.1667	-0.4667	99.1333	-3.2	0.0012	NA	NA	-1
11	14	92.20	-0.2814	-0.6227	101.8909	-1.8	0.0796	NA	NA	-1
12	15	90.60	-0.5455	-3.3500	135.1000	-1.1	0.2545	NA	NA	-1

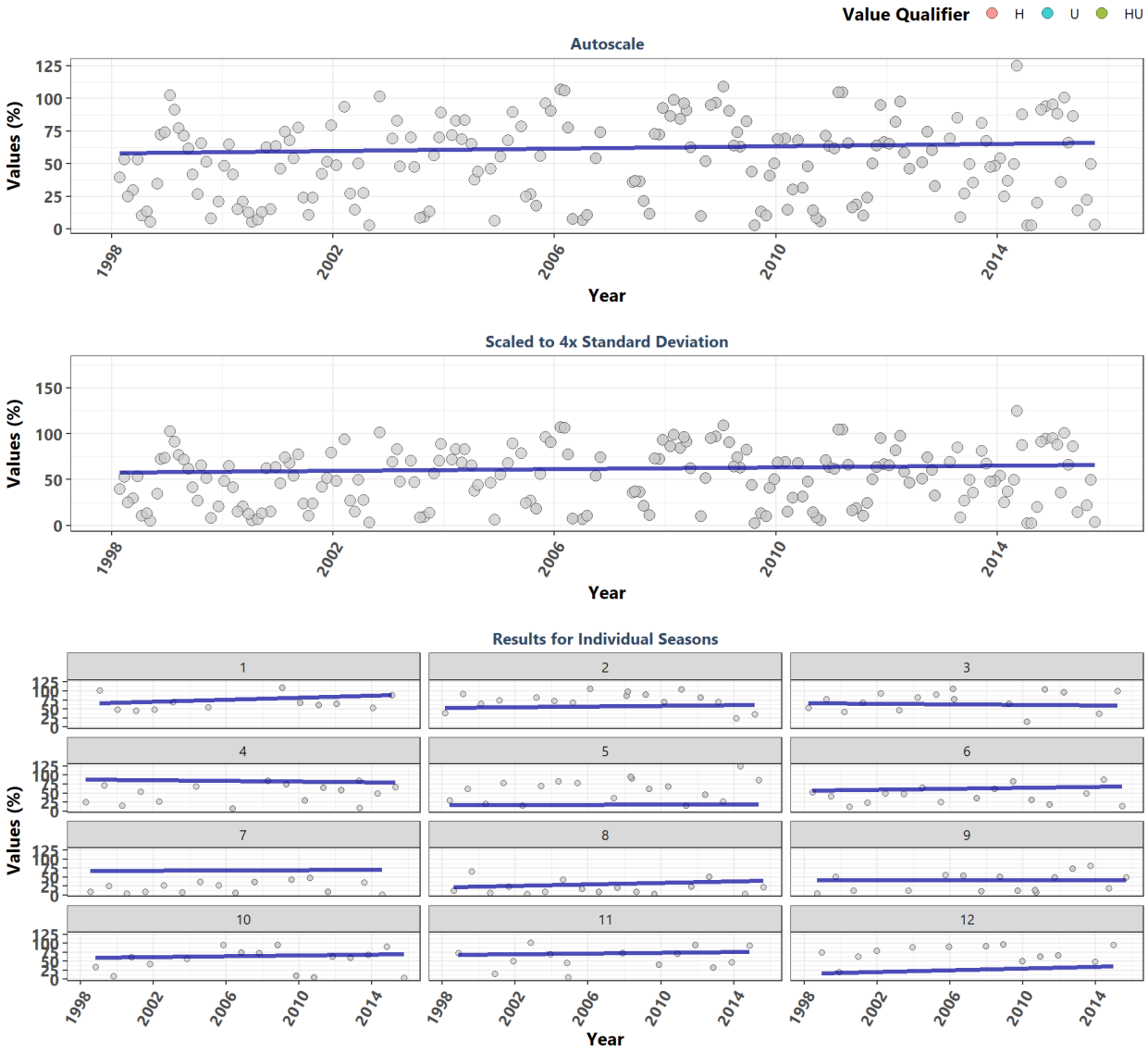
[a] $p < 0.00005$ appear as 0 due to rounding



Season	N	Median	tau	Slope	Int.	z	p__z	chi__sq	p__chi__sq	Trend
All	10496	85.40	-0.0166	-0.5542	90.5048	-8.9	0.0000	1085.4	0	-1
1	180	93.10	0.0240	0.0700	89.7290	-2.2	0.0264	NA	NA	1
2	220	95.90	-0.3362	-1.4833	100.1167	2.1	0.0336	NA	NA	-1
3	198	95.25	-0.0874	-0.7500	88.6000	-2.1	0.0319	NA	NA	-1
4	521	79.90	0.4394	1.8909	57.1273	15.7	0.0000	NA	NA	1
5	449	83.60	-0.0888	-0.4493	97.8335	14.5	0.0000	NA	NA	-1
6	323	87.00	0.4092	1.9000	59.0000	-3.0	0.0024	NA	NA	1
7	263	87.50	-0.1118	-0.4417	98.0417	-2.2	0.0293	NA	NA	-1
8	1429	68.60	0.2351	1.5000	52.1000	13.5	0.0000	NA	NA	1
9	5241	84.10	-0.1112	-0.2500	98.1000	-9.7	0.0000	NA	NA	-1
10	1226	92.70	0.0942	0.2008	91.2805	-18.1	0.0000	NA	NA	1
11	195	91.20	0.0955	0.2500	87.6000	0.5	0.6174	NA	NA	1
12	251	93.60	-0.1017	-0.2500	100.7500	2.4	0.0183	NA	NA	-1

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

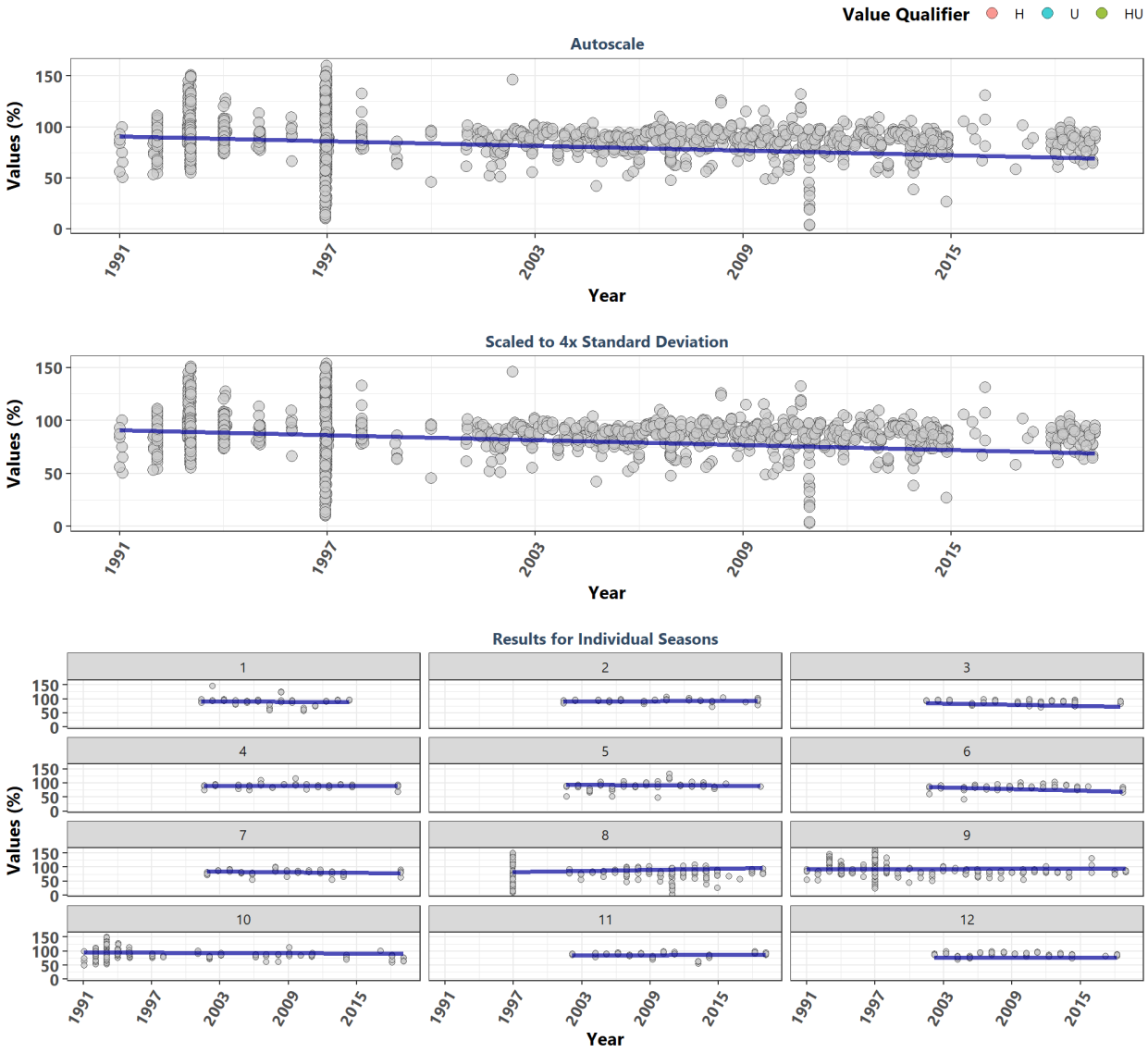
Rocky Bayou State Park Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	257	54.10	0.0633	0.4500	57.8250	1.4	0.1722	3.6	0.9799	0
1	16	65.20	0.1342	1.4688	64.3812	0.4	0.6829	NA	NA	0
2	24	82.35	0.1126	0.4333	53.4000	-0.5	0.5829	NA	NA	0
3	22	77.60	-0.0474	-0.4000	66.9000	0.9	0.3941	NA	NA	0
4	22	58.60	-0.0833	-0.4600	87.4100	0.7	0.4786	NA	NA	0
5	24	65.35	0.0395	0.1100	16.8000	0.4	0.7083	NA	NA	0
6	21	41.70	0.0833	0.6800	56.7000	0.0	1.0000	NA	NA	0
7	20	26.65	0.0409	0.1714	68.0429	1.2	0.2152	NA	NA	0
8	23	17.90	0.1884	1.0000	22.7000	0.2	0.8111	NA	NA	0
9	24	34.70	-0.0048	-0.0273	42.0000	1.3	0.2027	NA	NA	0
10	23	62.50	0.0580	0.5818	58.9500	-0.3	0.7703	NA	NA	0
11	19	70.10	0.0526	0.5333	67.9333	0.2	0.8324	NA	NA	0
12	19	73.80	0.2053	1.2125	15.1313	0.3	0.7780	NA	NA	0

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

Terra Ceia Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	2024	89.30	-0.1049	-0.7750	90.7451	-8.9	0.0000	78.1	0	-1
1	55	92.90	-0.0865	-0.2000	92.8000	-0.9	0.3704	NA	NA	-1
2	52	94.55	0.0260	0.0667	90.3167	0.6	0.5737	NA	NA	1
3	59	91.60	-0.1567	-0.7118	91.1735	-1.7	0.0973	NA	NA	-1
4	56	91.55	0.0181	0.0500	88.8500	0.3	0.7822	NA	NA	1
5	61	89.40	-0.1479	-0.1833	95.2667	2.0	0.0507	NA	NA	-1
6	57	85.60	-0.1819	-1.0000	96.3000	0.7	0.4558	NA	NA	-1
7	53	82.60	-0.1495	-0.3500	88.9000	-1.6	0.1145	NA	NA	-1
8	328	77.90	0.1716	0.7000	77.5000	-0.2	0.8759	NA	NA	1
9	673	90.30	0.0543	0.0750	93.1250	-7.3	0.0000	NA	NA	1
10	520	89.75	-0.0835	-0.1500	95.4500	-5.5	0.0000	NA	NA	-1
11	52	89.75	0.0683	0.1472	82.8028	0.2	0.8555	NA	NA	1
12	58	89.20	-0.0051	-0.0594	78.2565	-1.0	0.3393	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
```

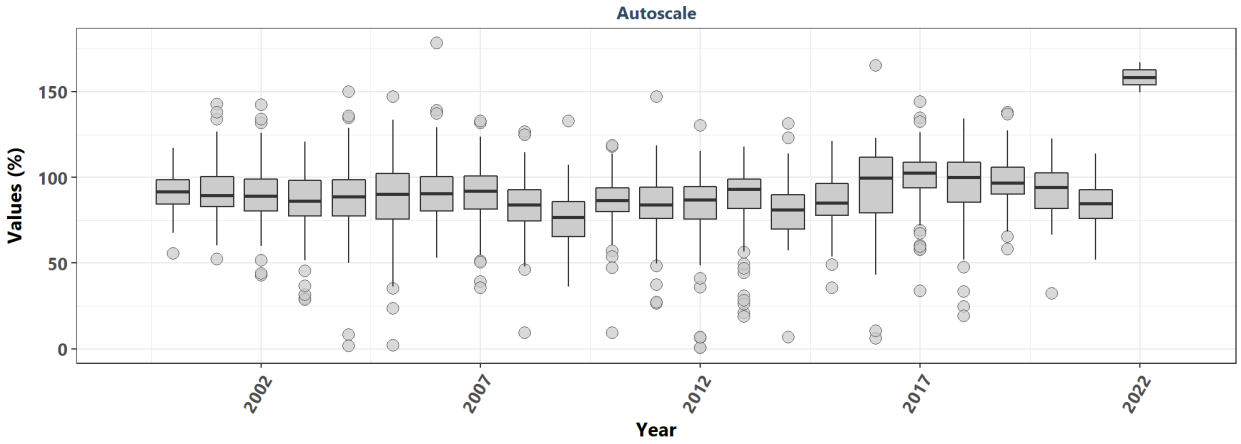
```
p2 <- ggplot(data=plot_data,
             aes(x=Year, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Scaled to 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") +
```

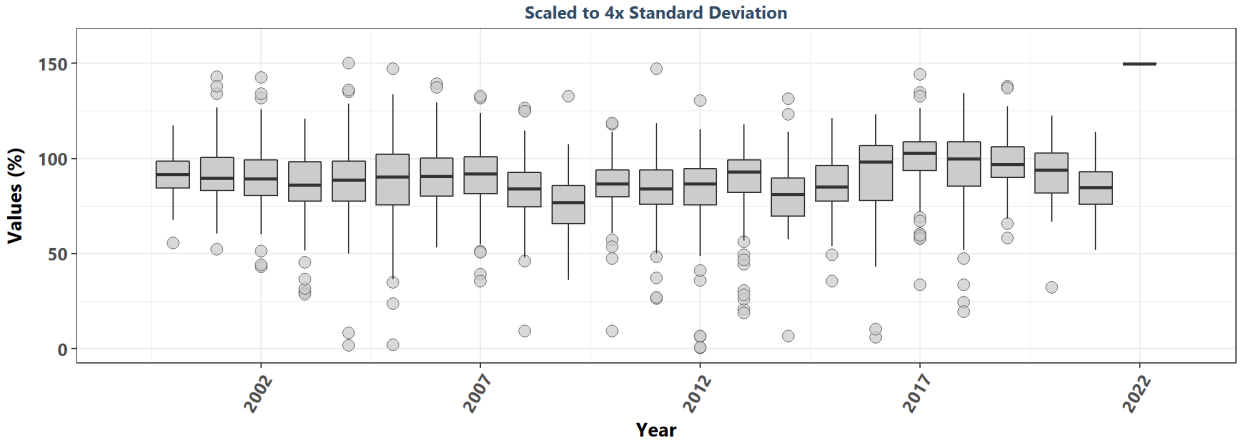
```
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits=c(year_lower - 1, year_upper + 1),
                      breaks=rev(seq(year_upper,
                                     year_lower, -x_scale))) +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal") +
   guides(color=guide_legend(nrow=1))

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") +
```

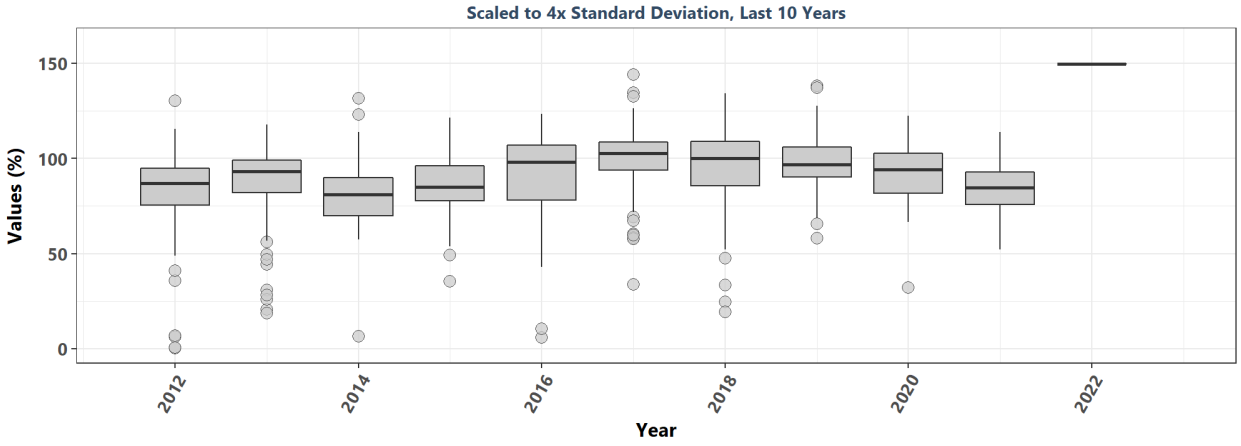
```
        guides(fill=guide_legend(nrow=1))

    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)
  }
}
```

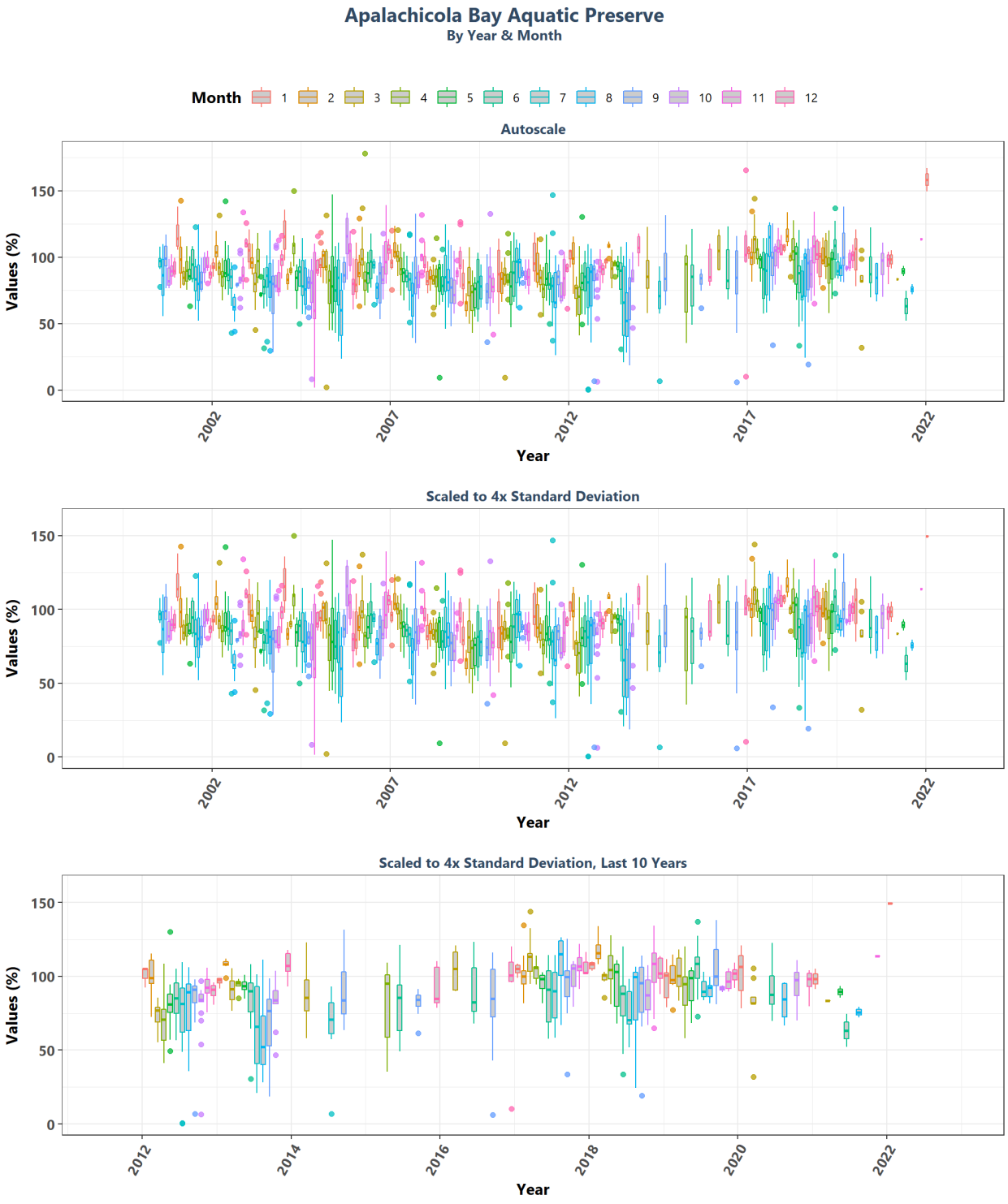
Apalachicola Bay Aquatic Preserve

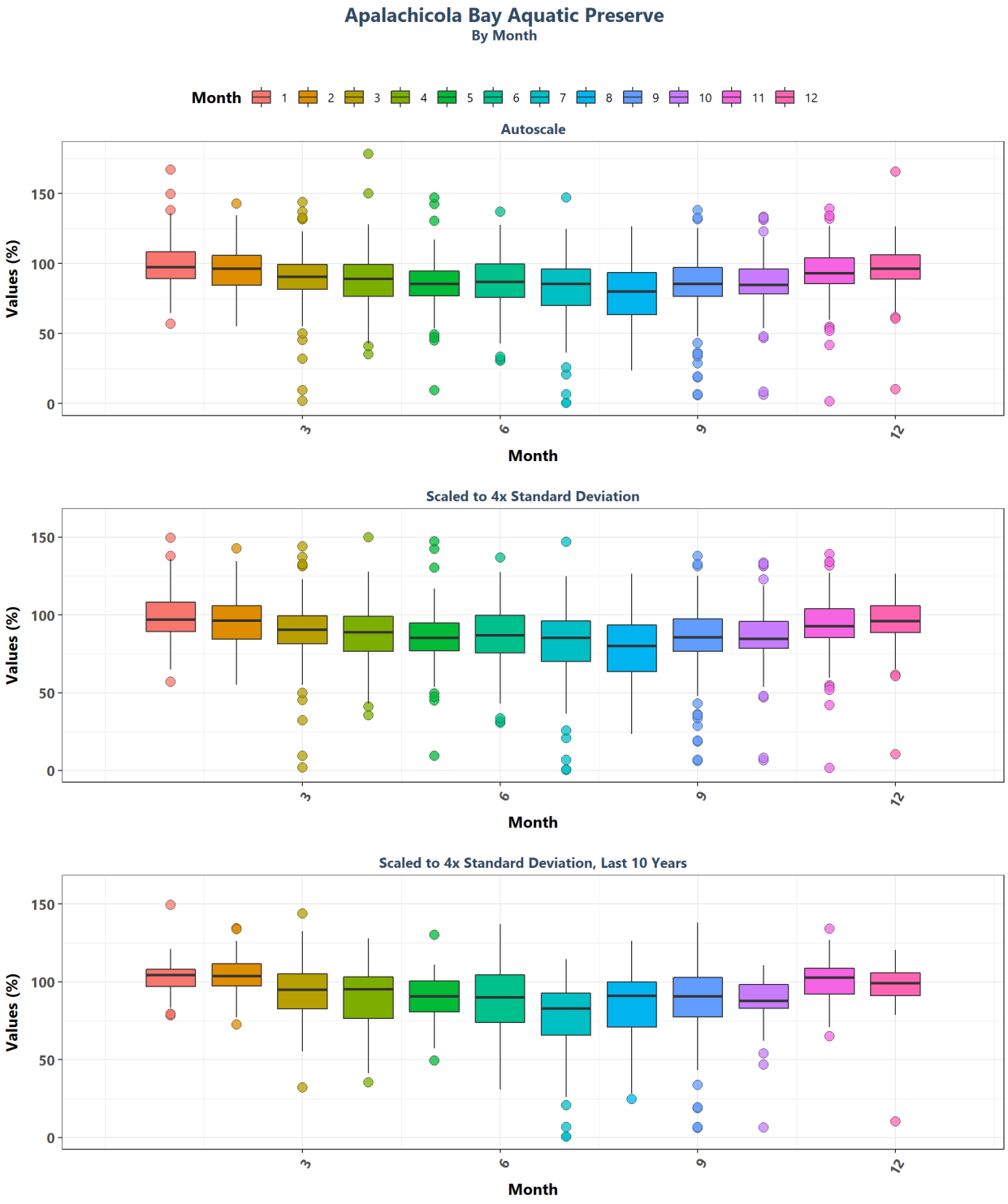
By Year





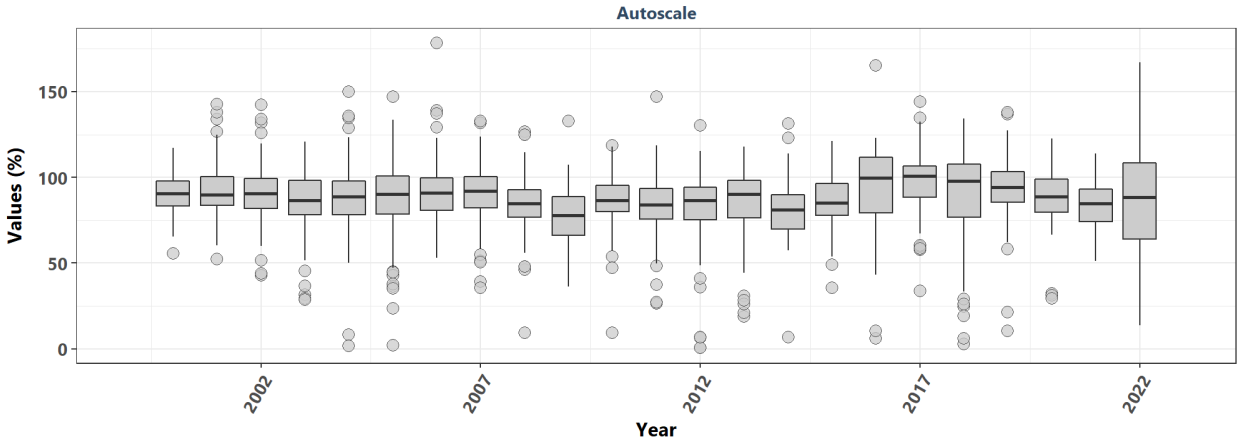


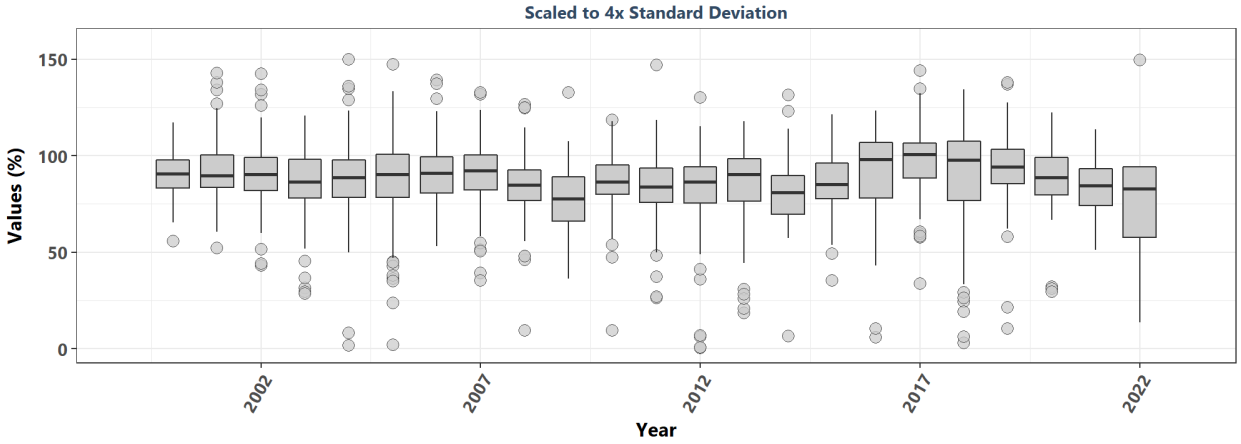

Apalachicola Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Year
Scaled to 4x Standard Deviation
Values (%)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Year

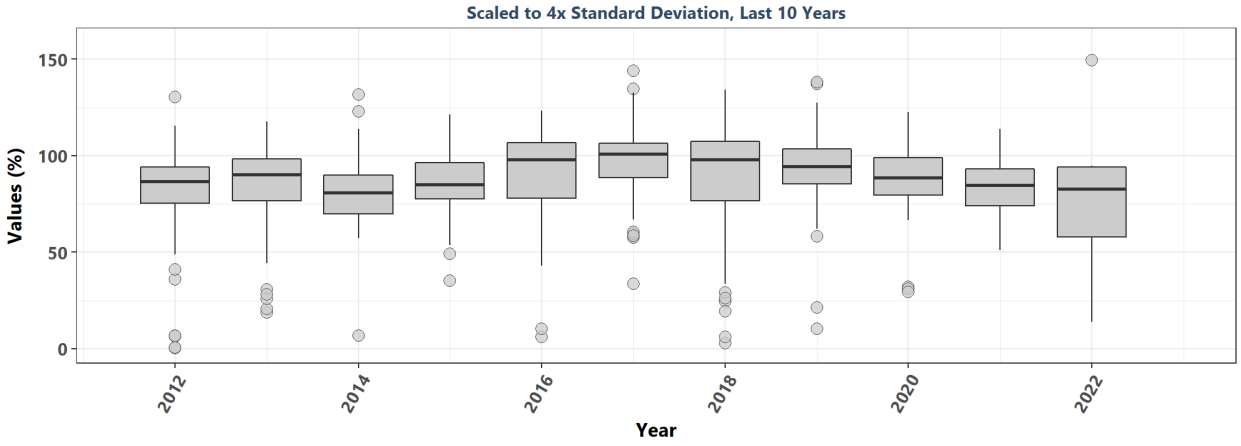

Apalachicola Bay Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
3
6
9
12
Month

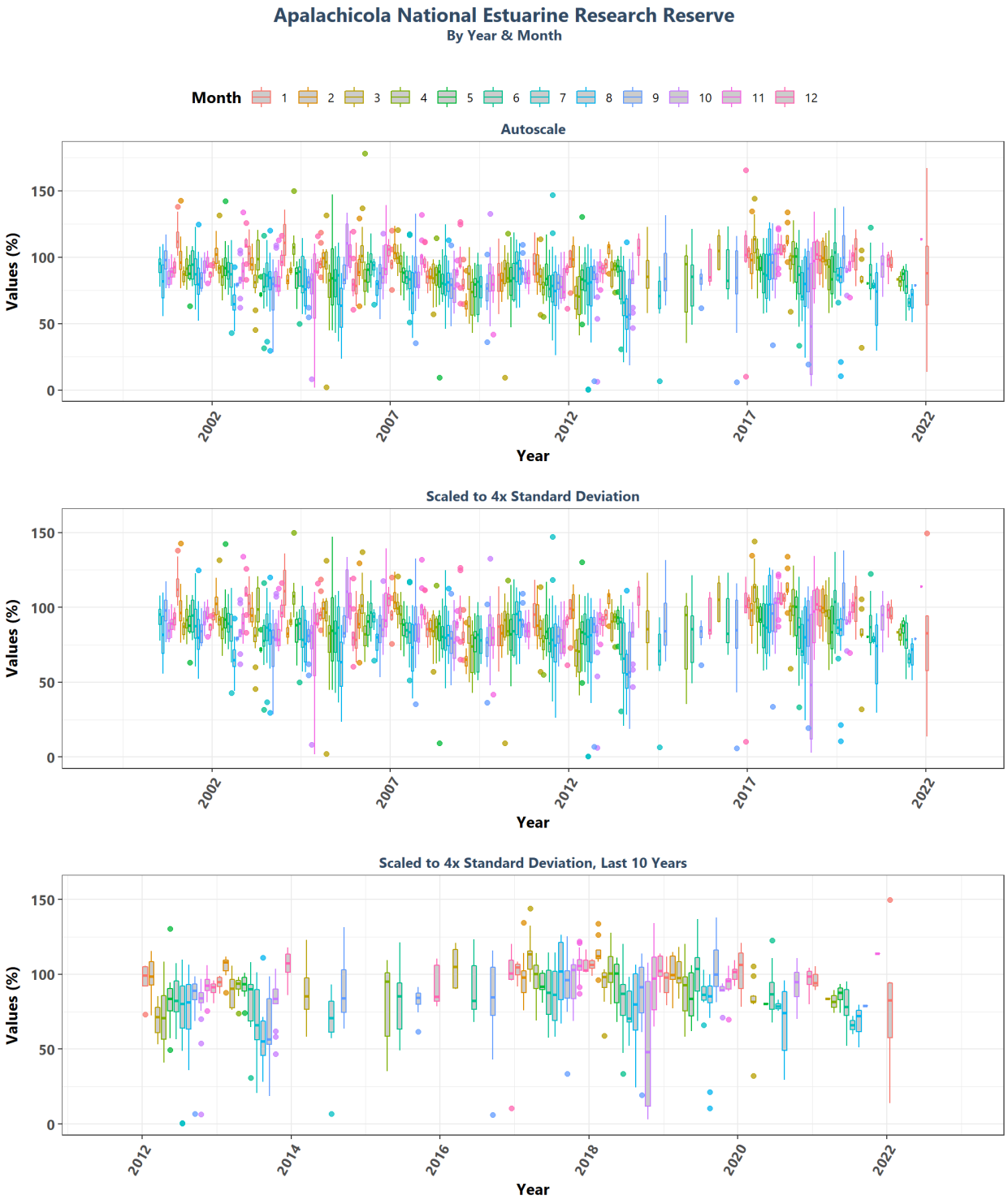
Apalachicola National Estuarine Research Reserve

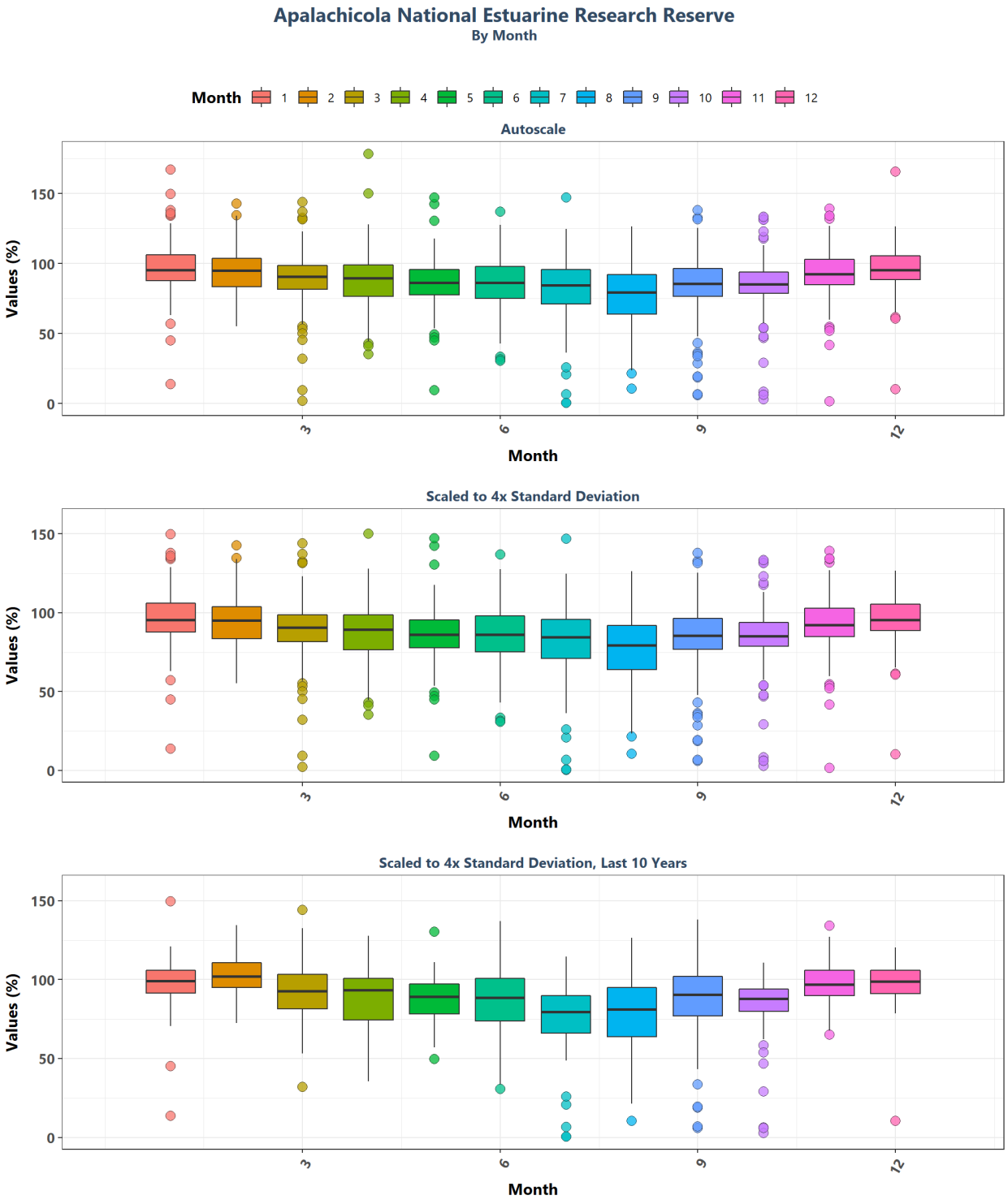
By Year





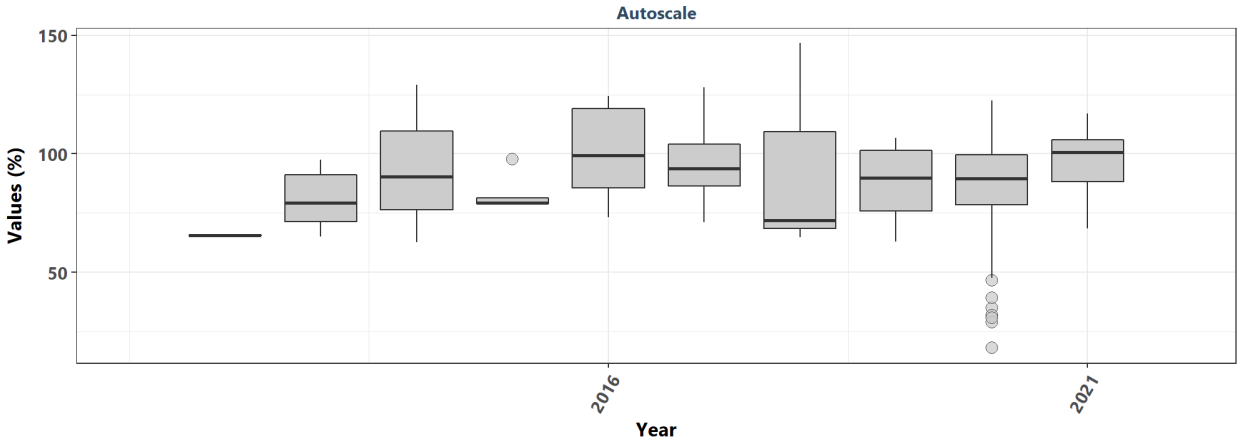


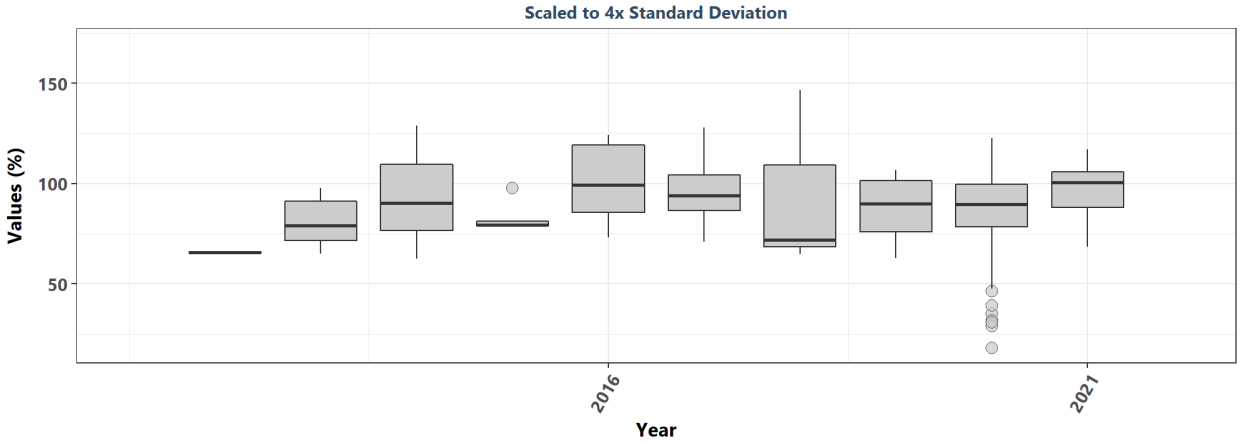

Apalachicola National Estuarine Research Reserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
150
100
50
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
2002
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
2012
2014
2016
2018
2020
2022
Year

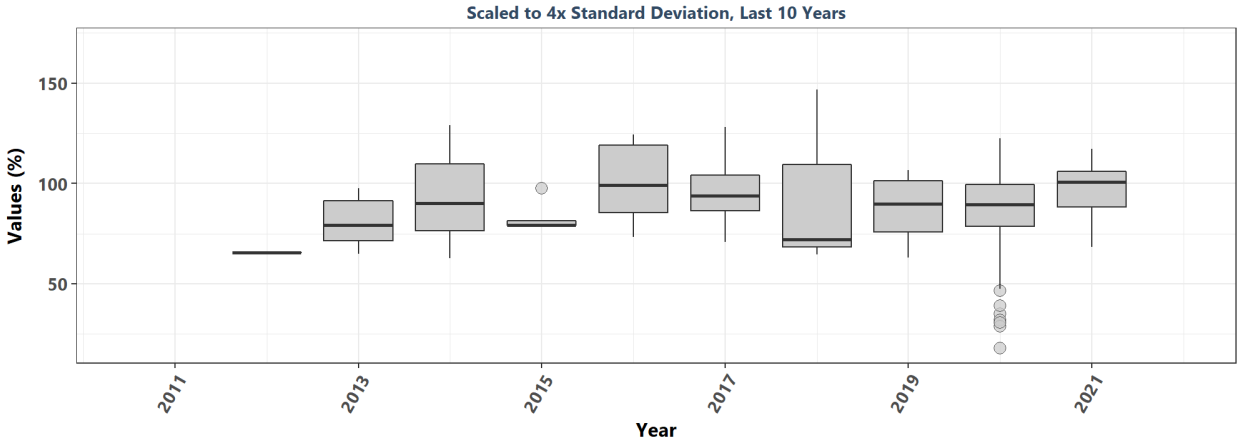

Apalachicola National Estuarine Research Reserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
3
6
9
12
Month

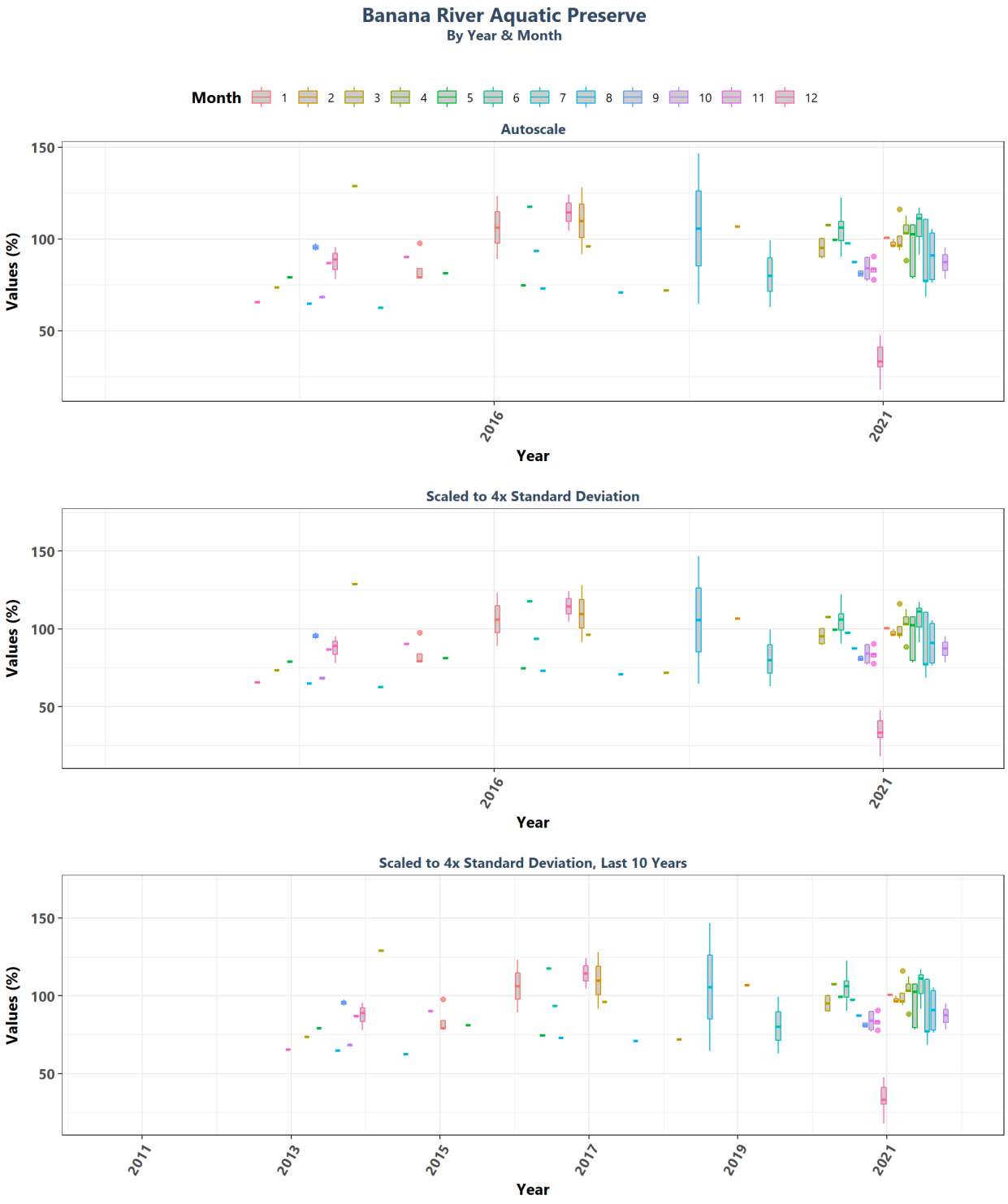
Banana River Aquatic Preserve

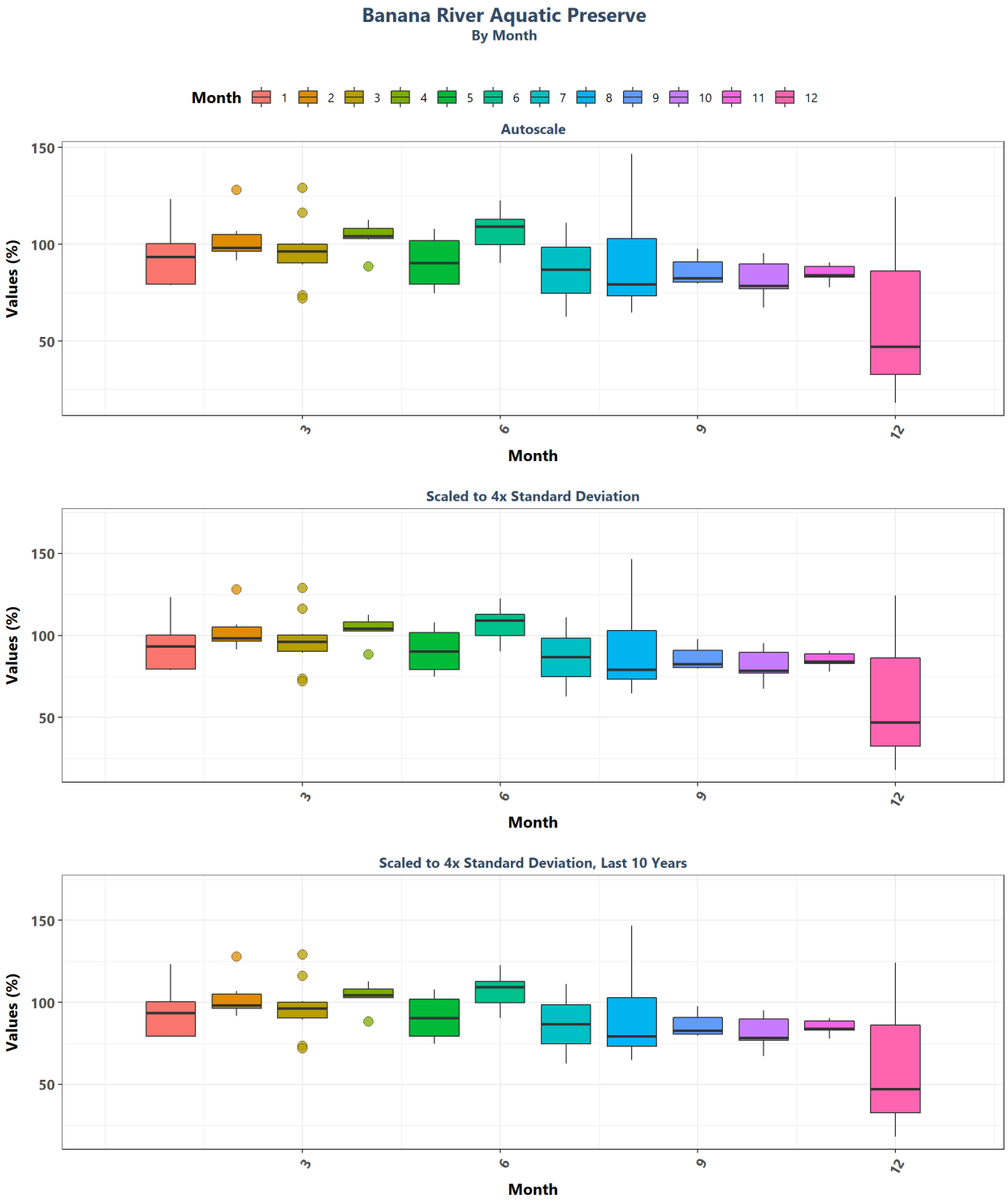
By Year





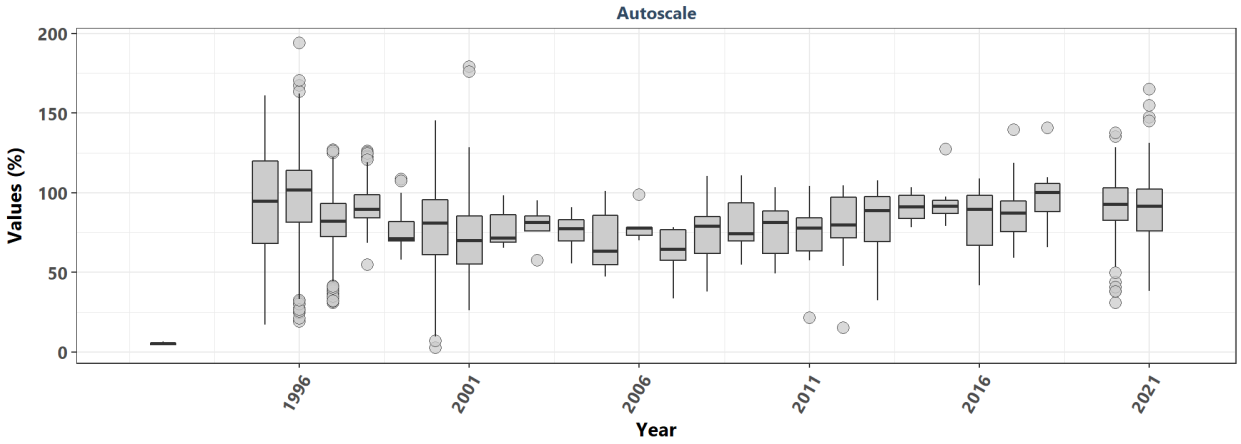


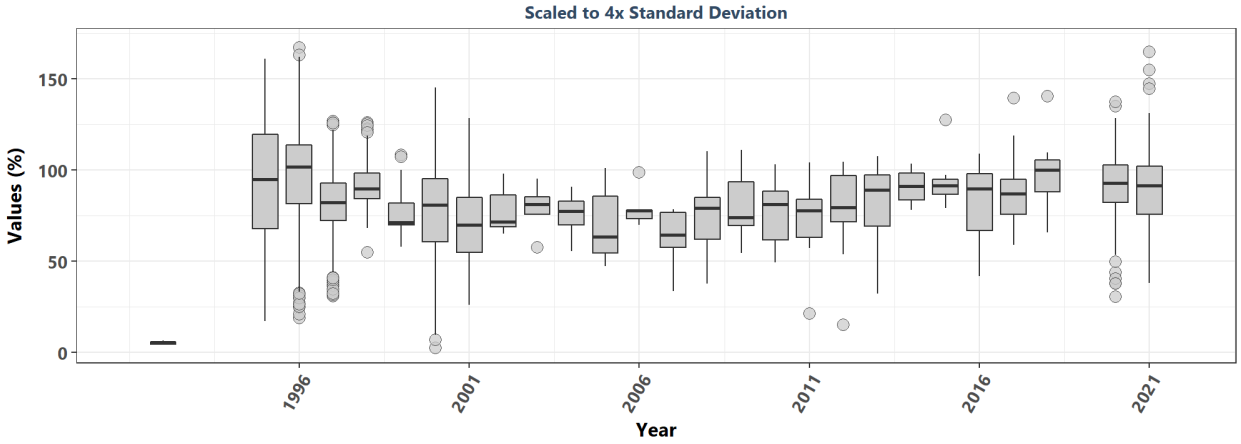

Banana River Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
150
100
50
2016
2021
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
2011
2013
2015
2017
2019
2021
Year

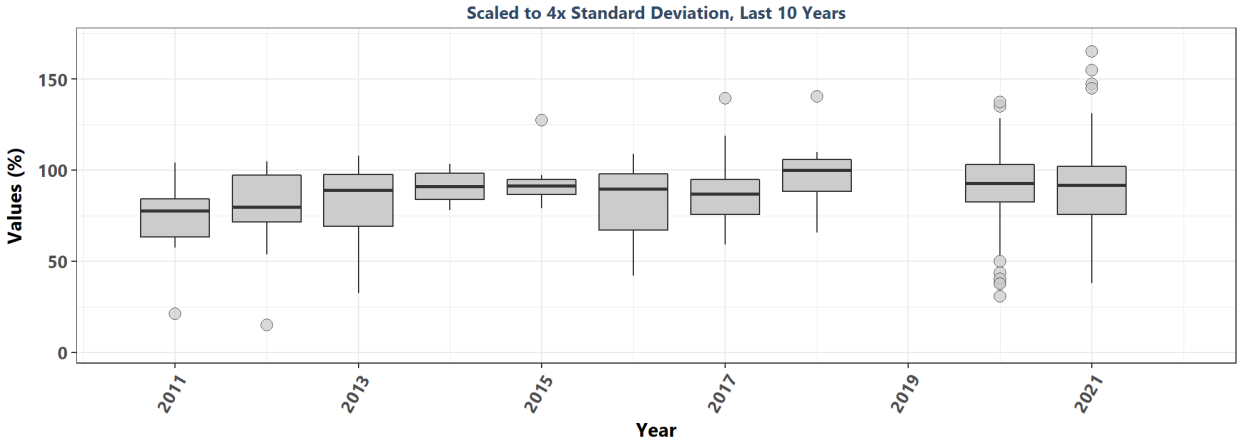

Banana River Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
150
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
150
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
3
6
9
12
Month

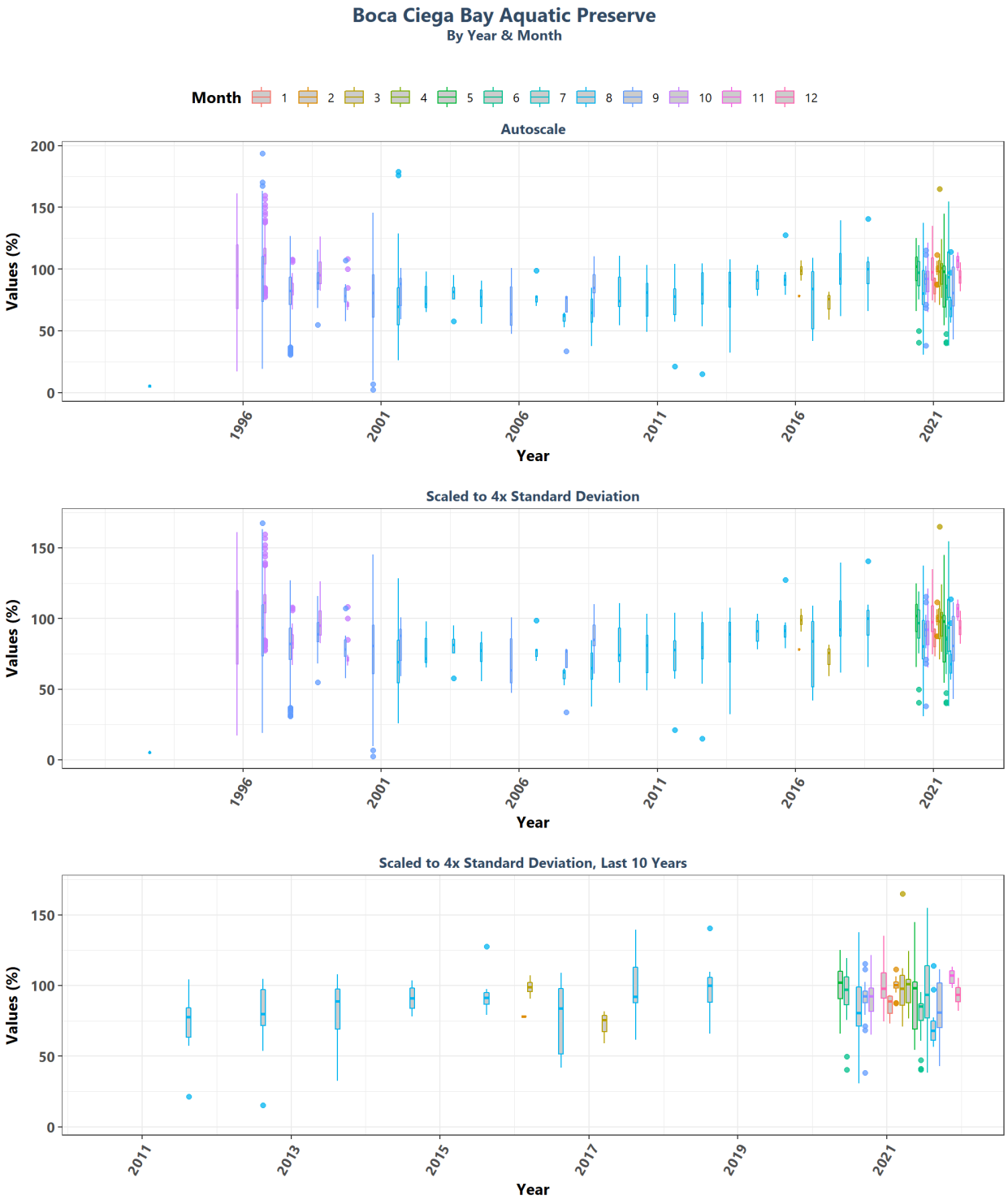
Boca Ciega Bay Aquatic Preserve

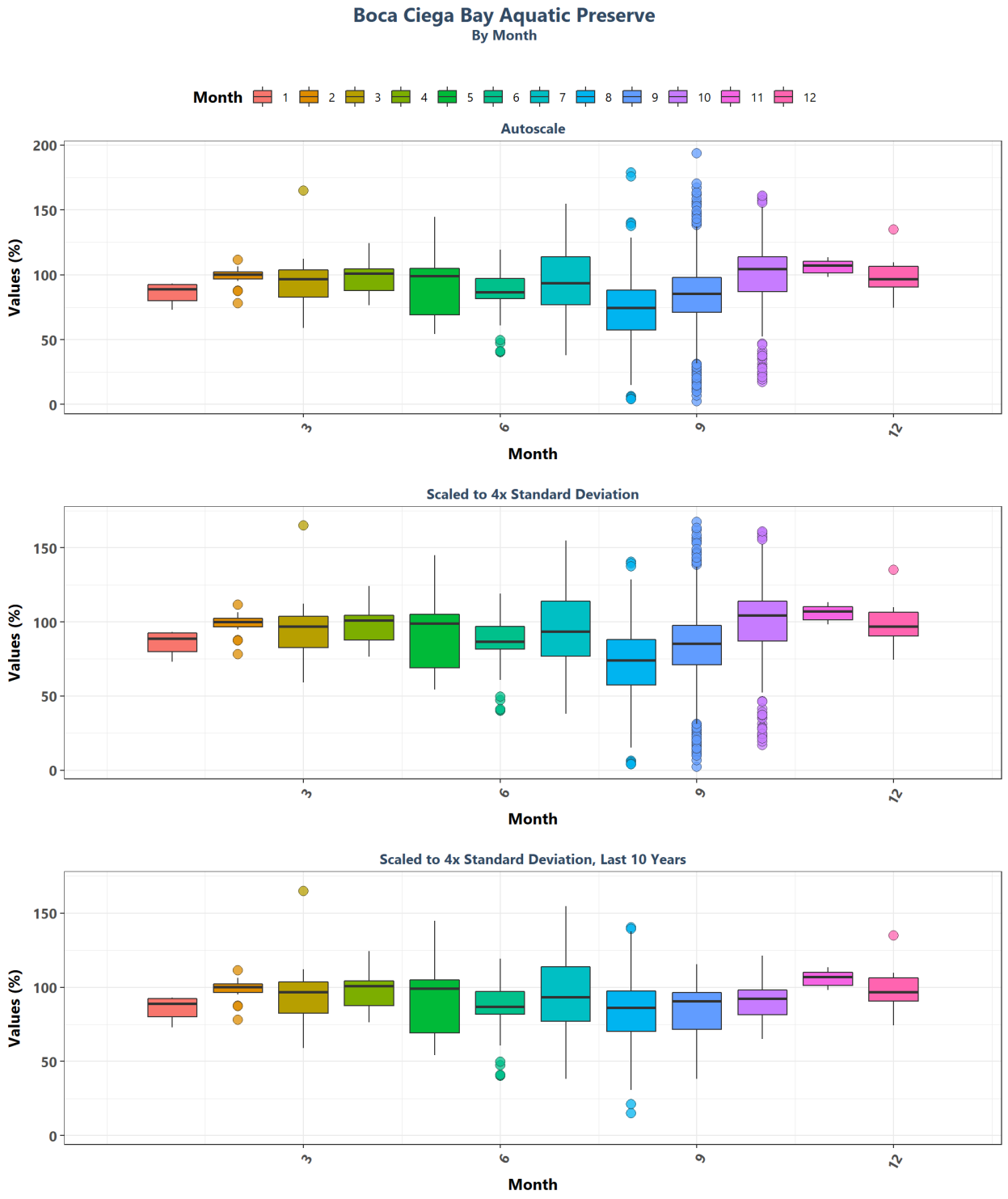
By Year





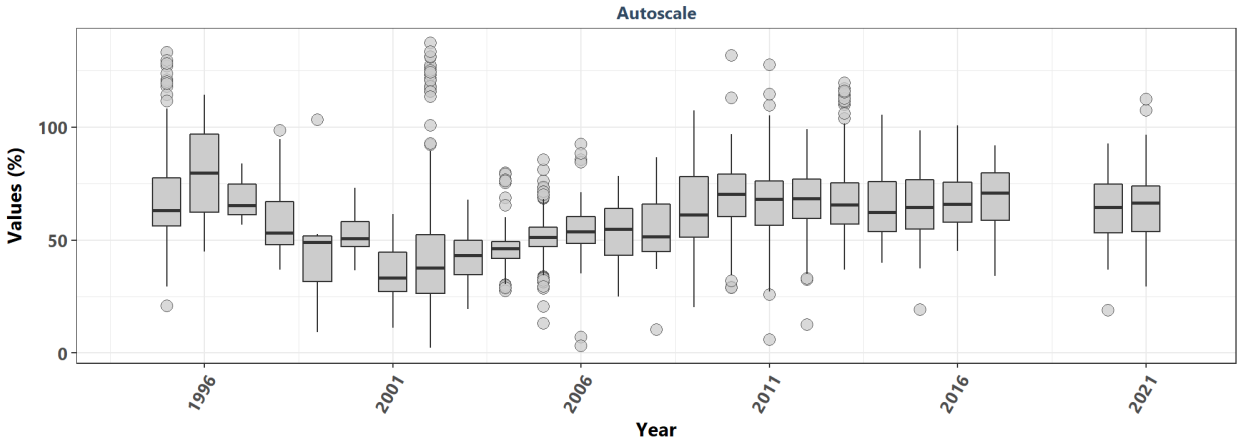


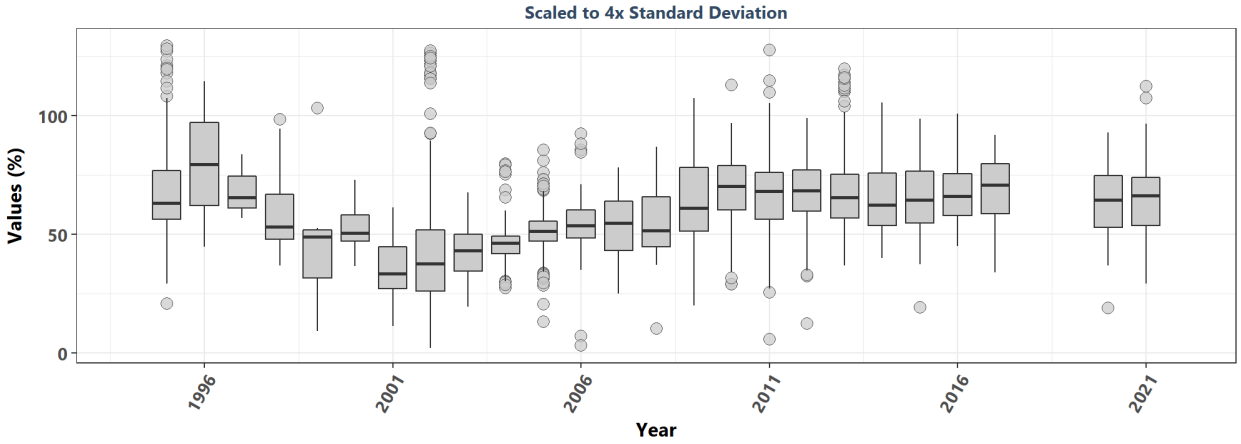

Boca Ciega Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Year
Scaled to 4x Standard Deviation
Values (%)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Year

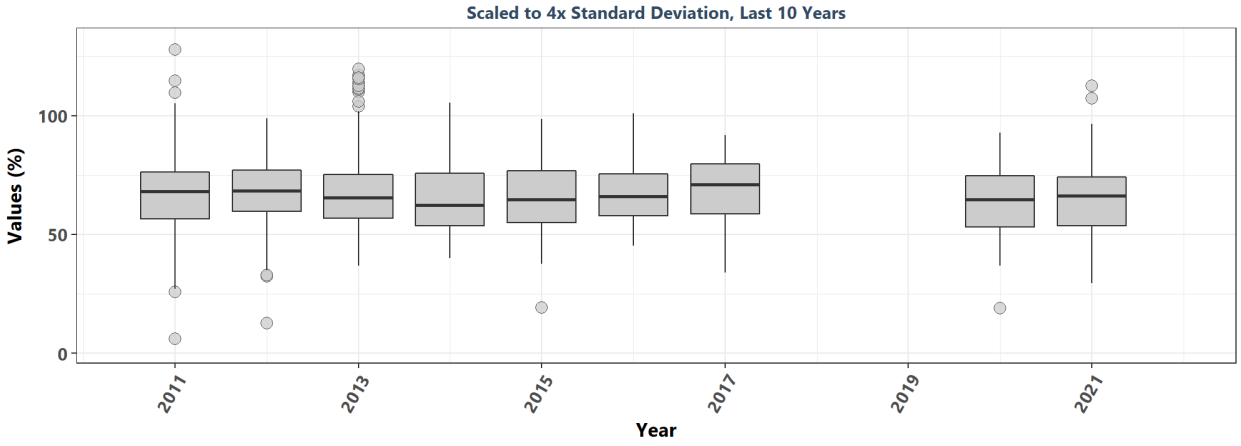

Boca Ciega Bay Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
200
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
3
6
9
12
Month

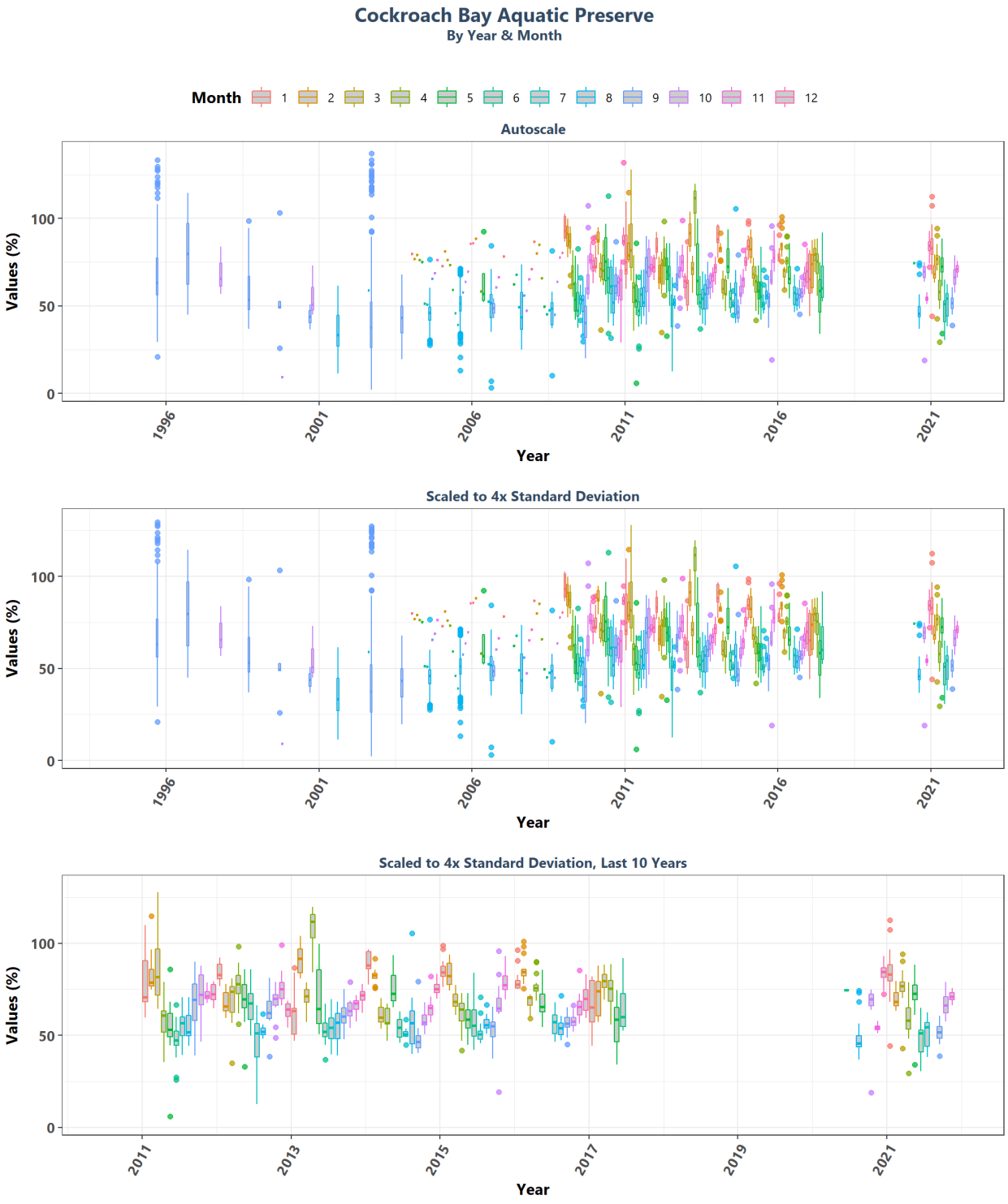
Cockroach Bay Aquatic Preserve

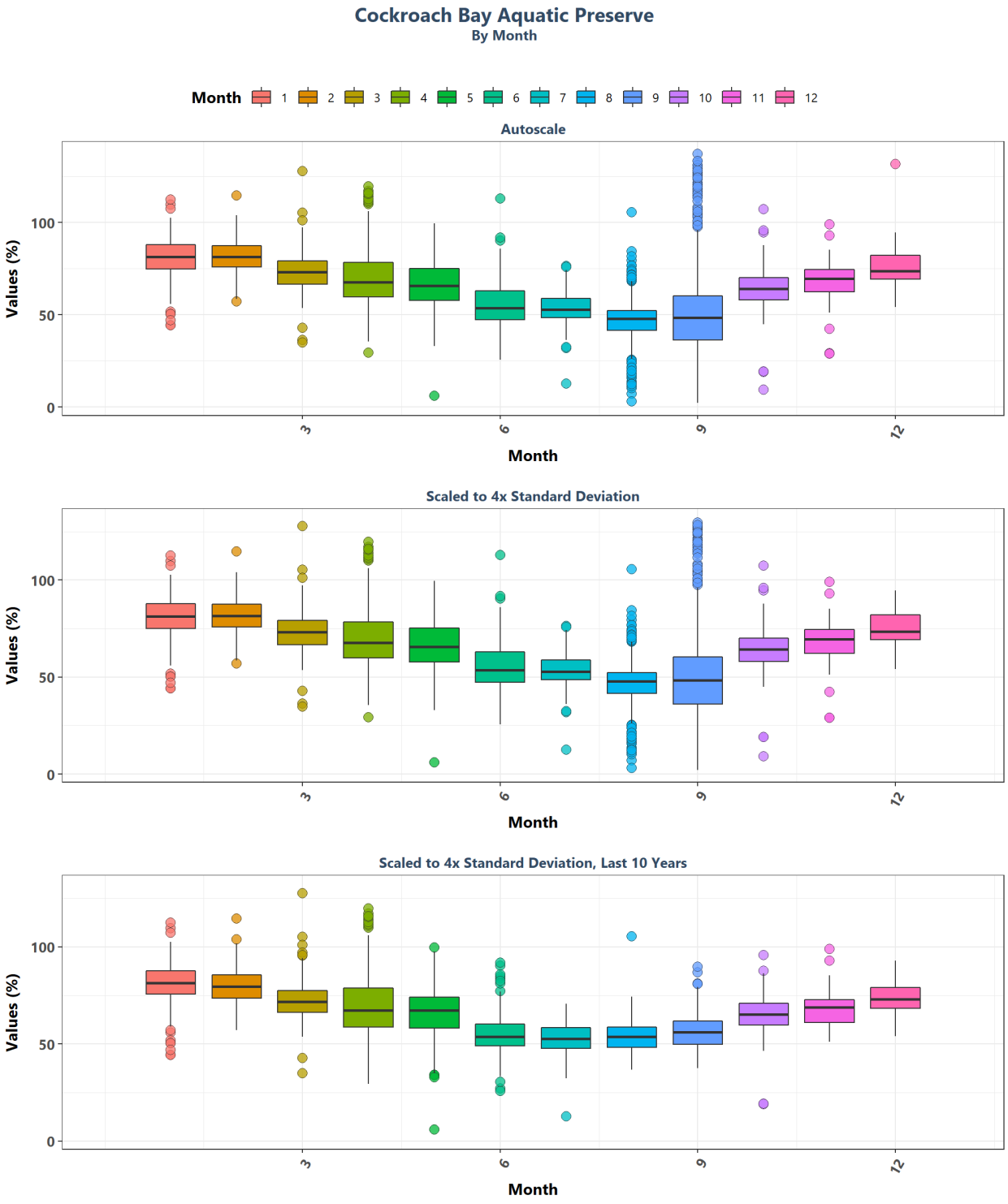
By Year





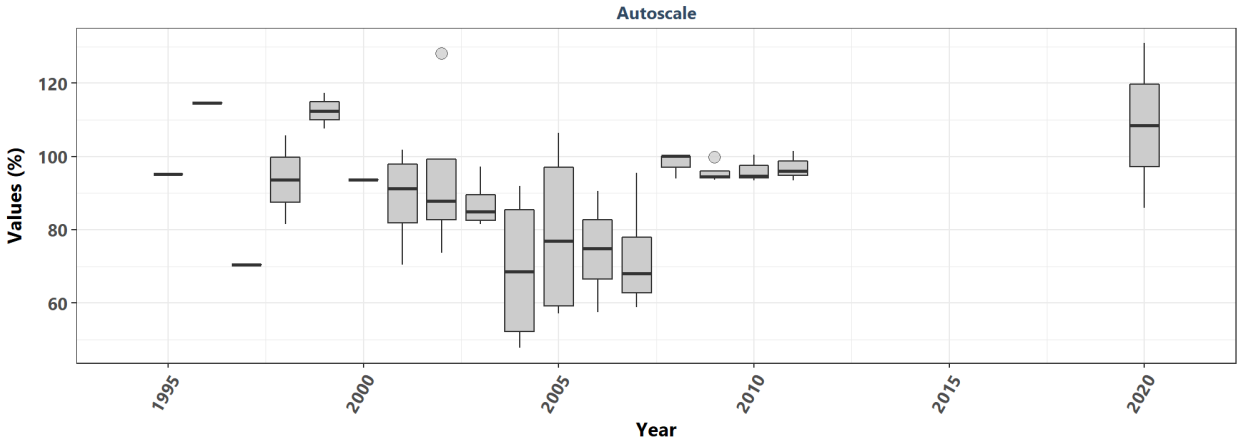


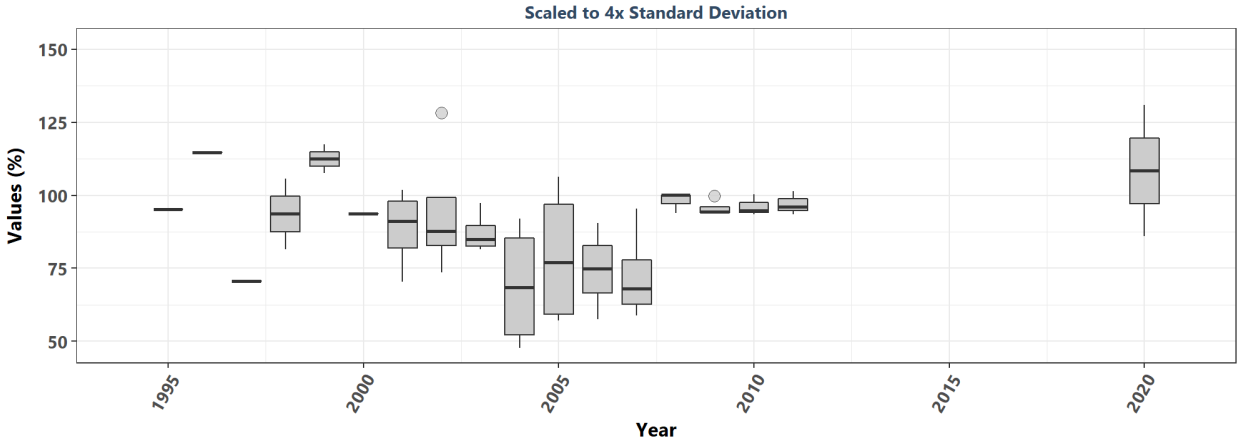

Cockroach Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
0
1996
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (%)
100
50
0
1996
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
100
50
0
2011
2013
2015
2017
2019
2021
Year

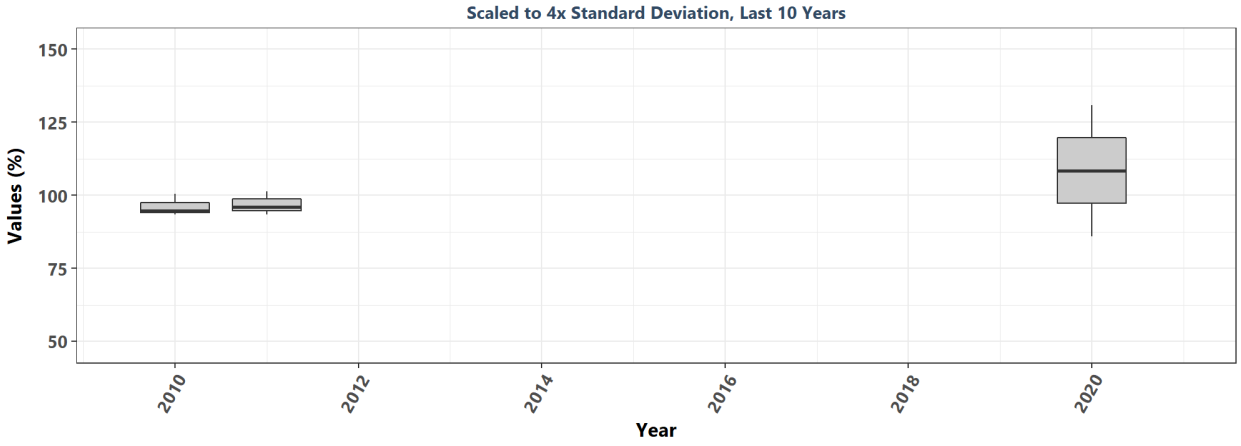

Cockroach Bay Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
100
50
0
3
6
9
12
Month

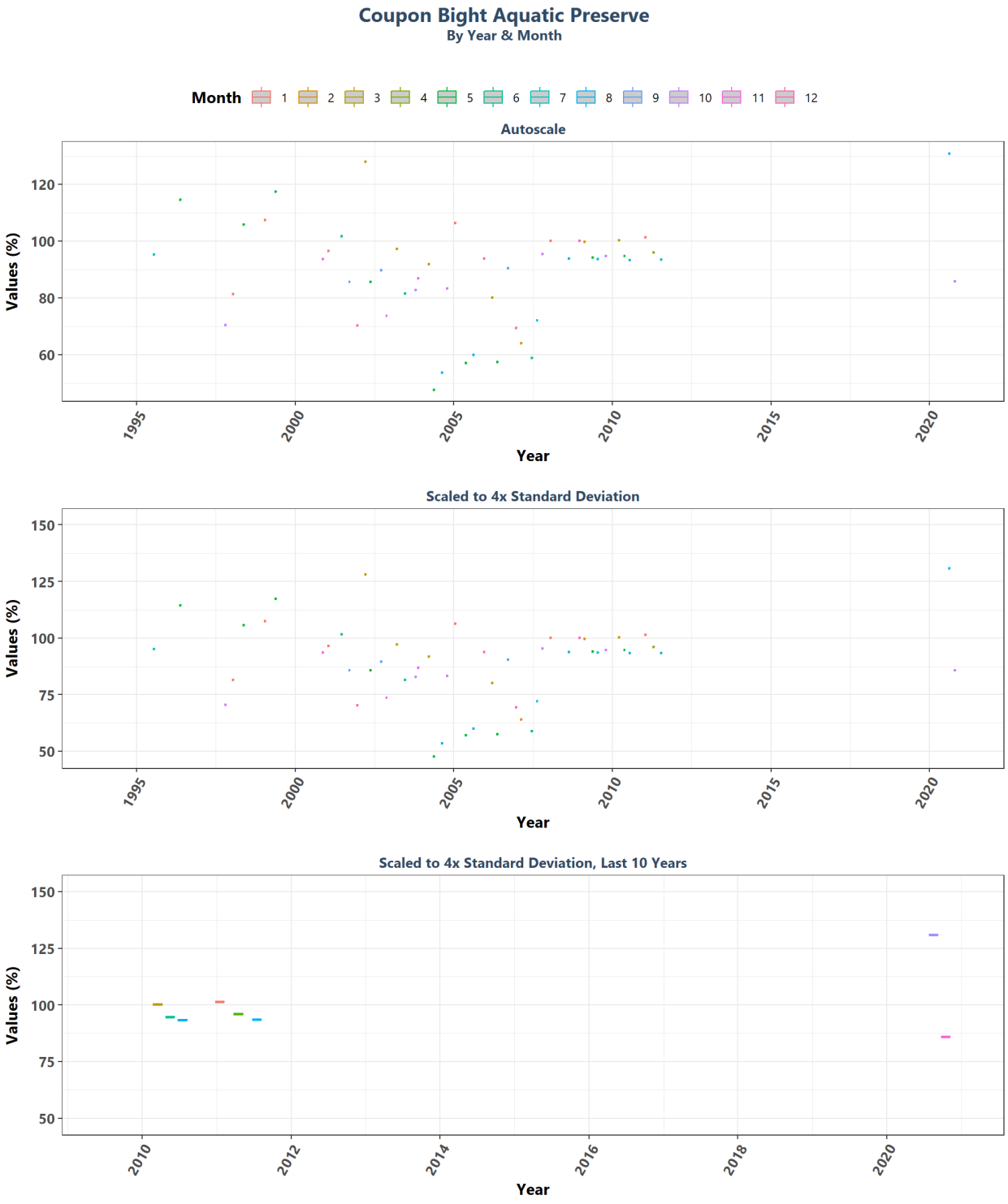
Coupon Bight Aquatic Preserve

By Year

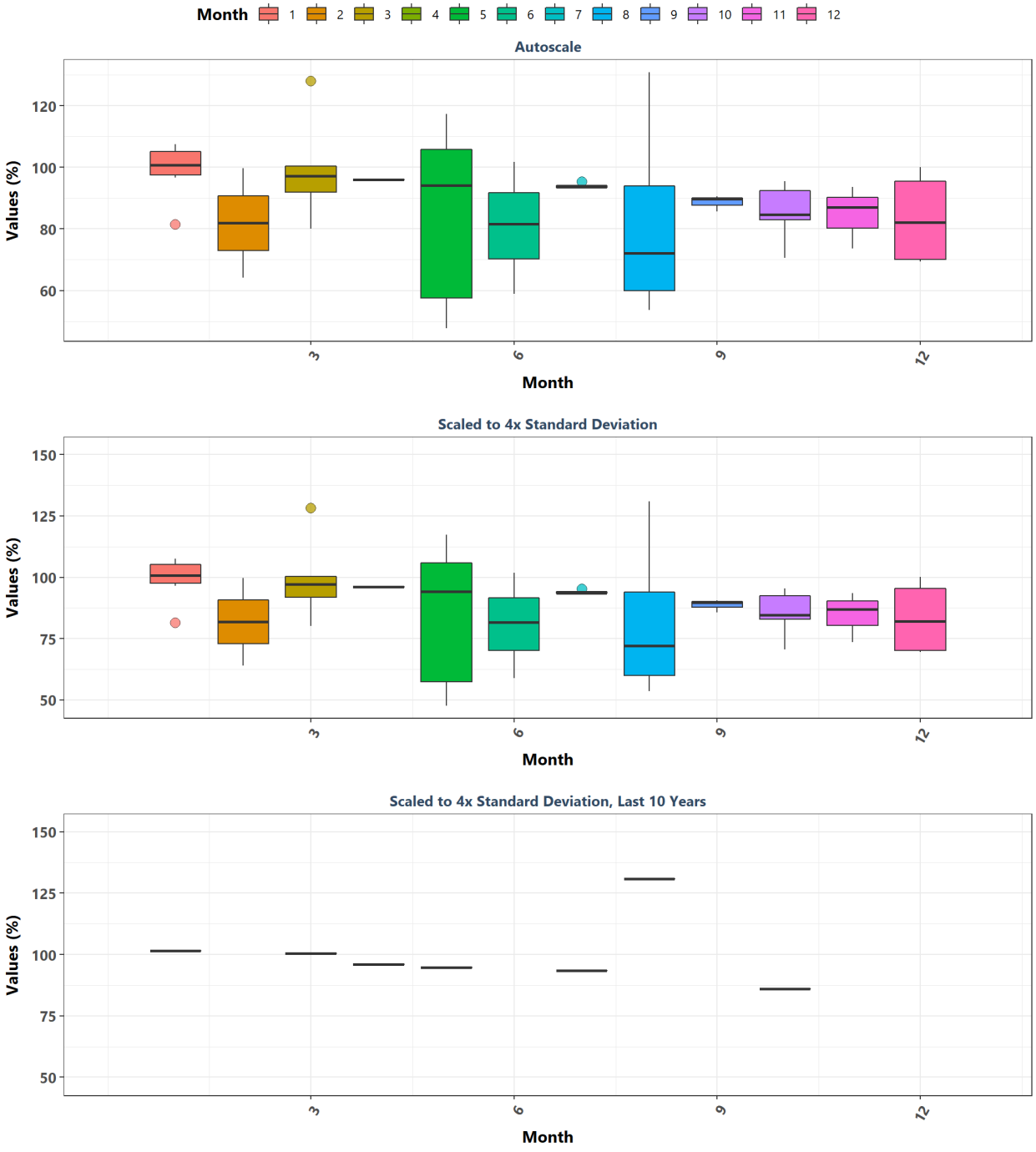






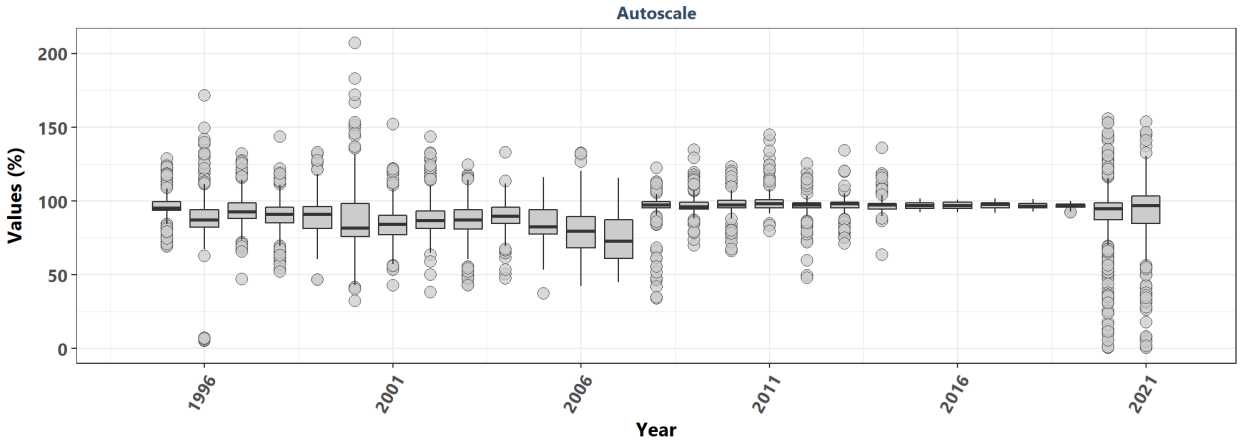

Coupon Bight Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
120
100
80
60
1995
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation
Values (%)
150
125
100
75
50
1995
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
125
100
75
50
2010
2012
2014
2016
2018
2020
Year

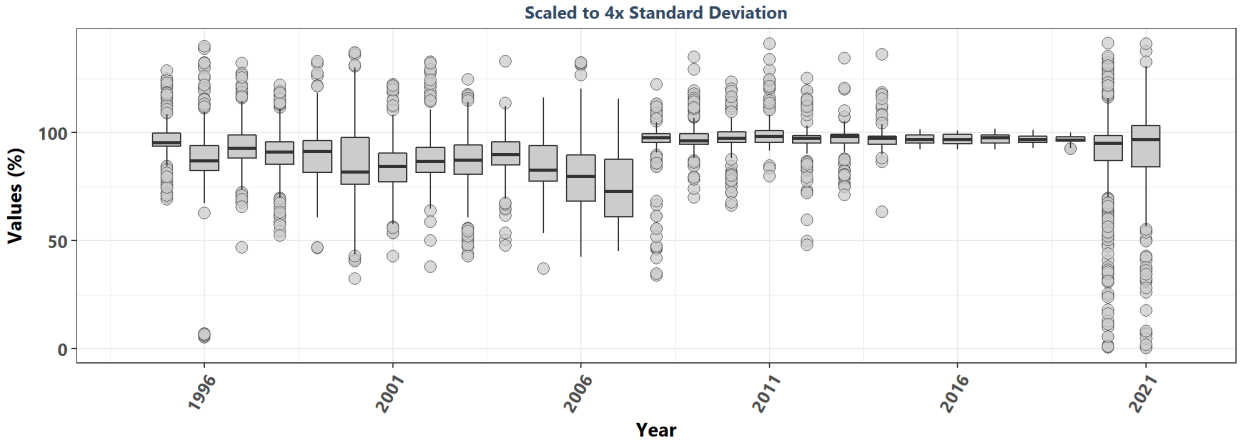
Coupon Bight Aquatic Preserve
By Month

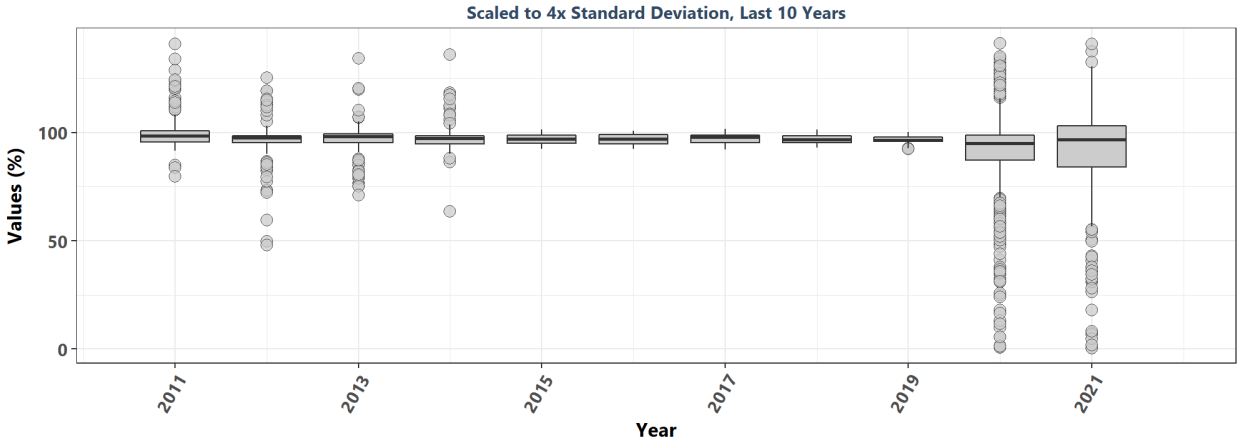


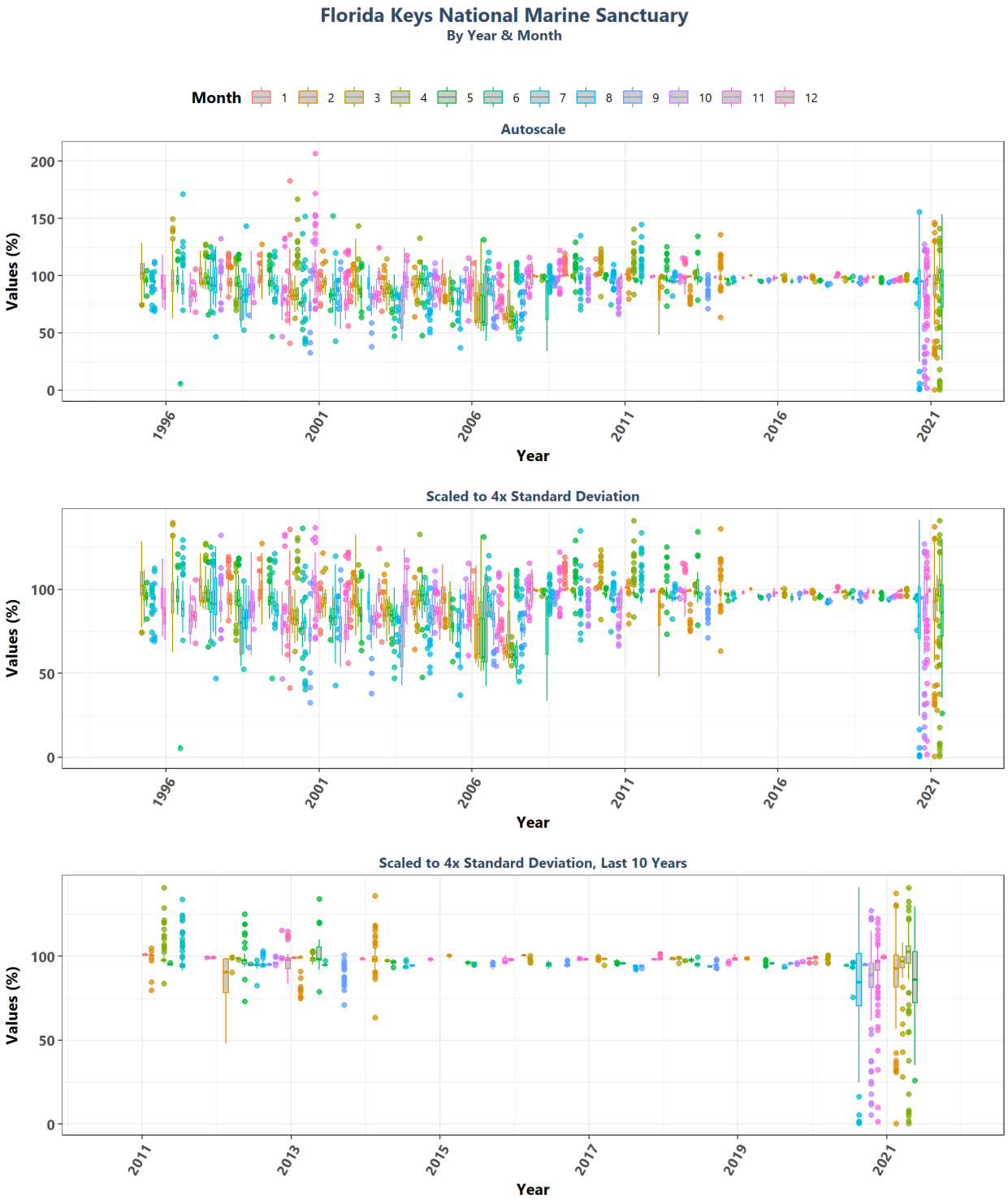
Florida Keys National Marine Sanctuary

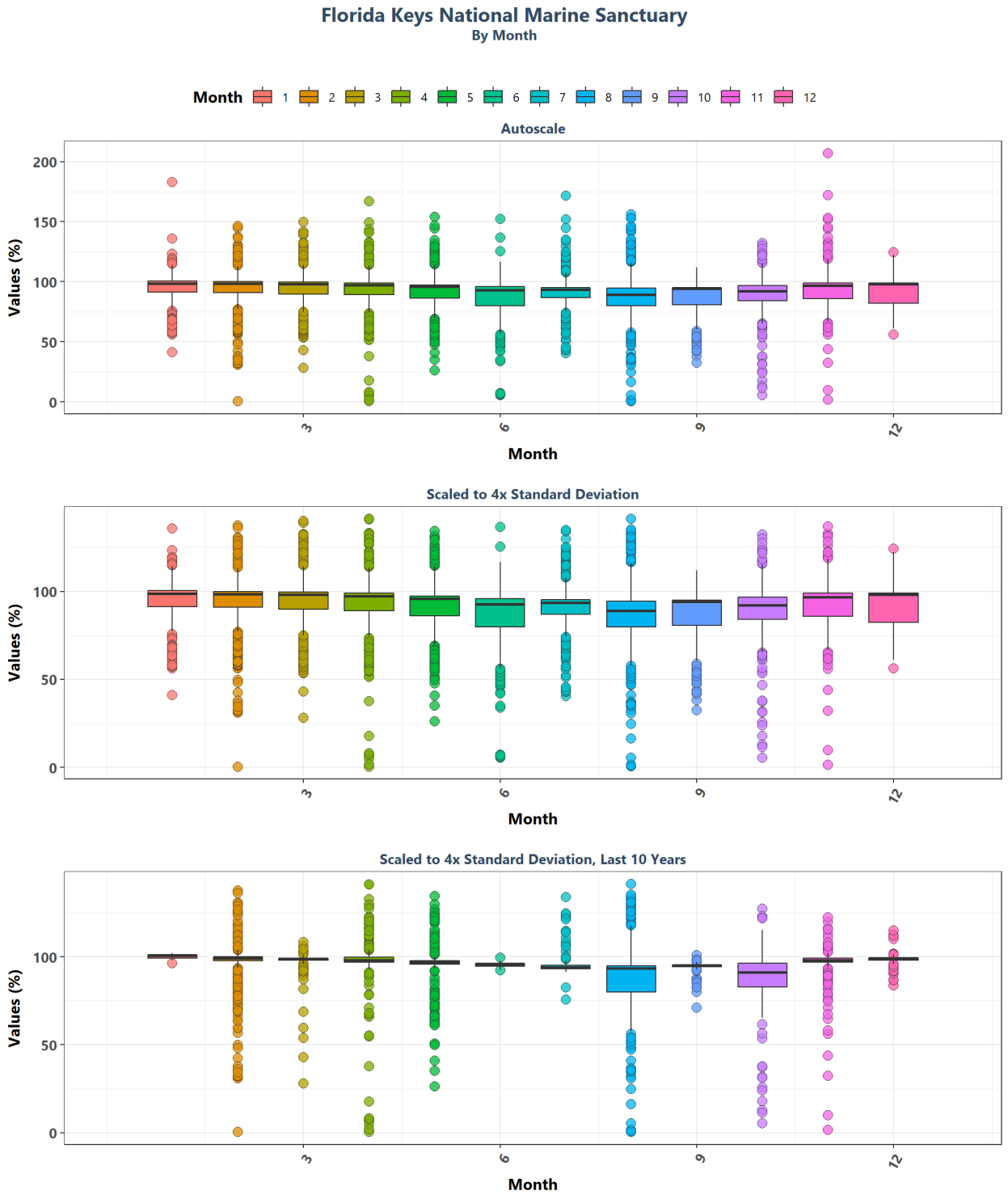
By Year



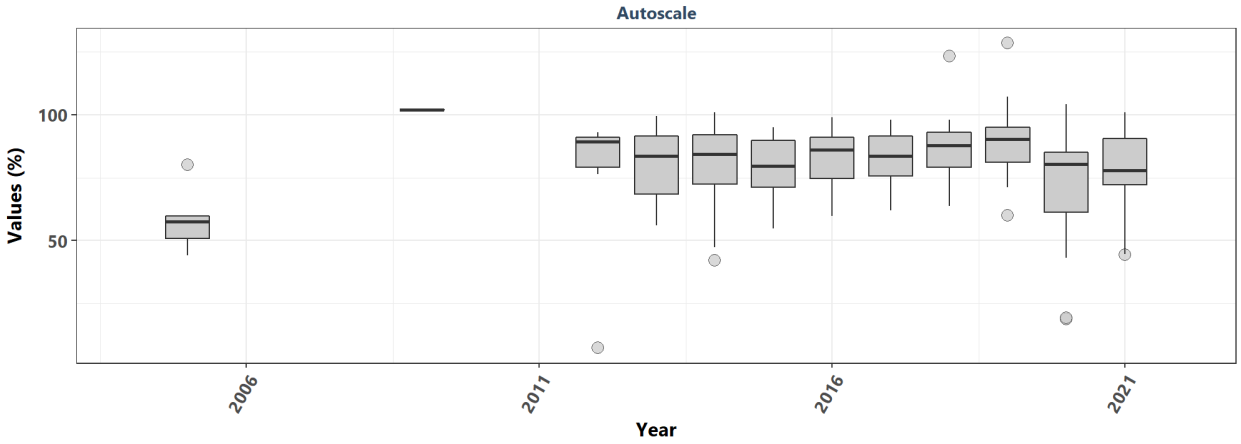


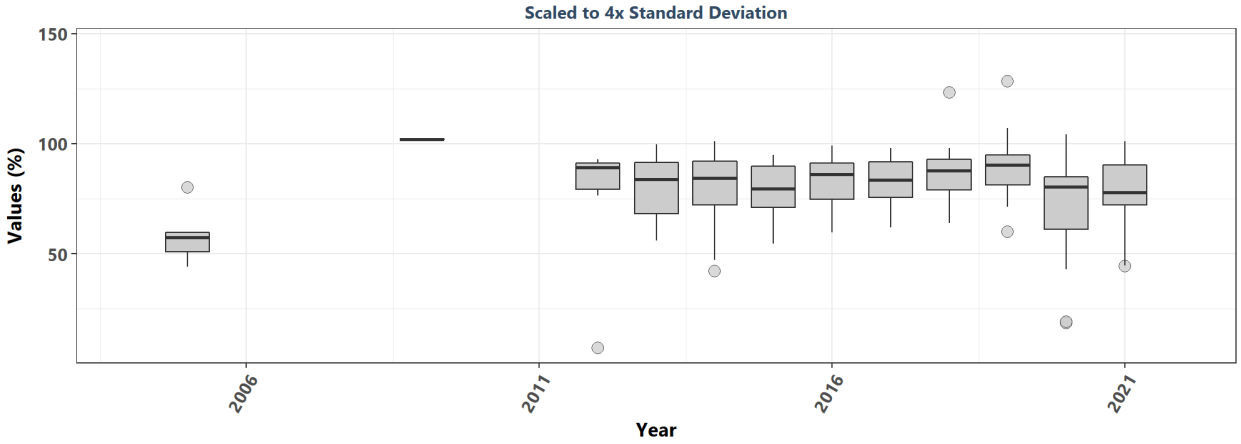


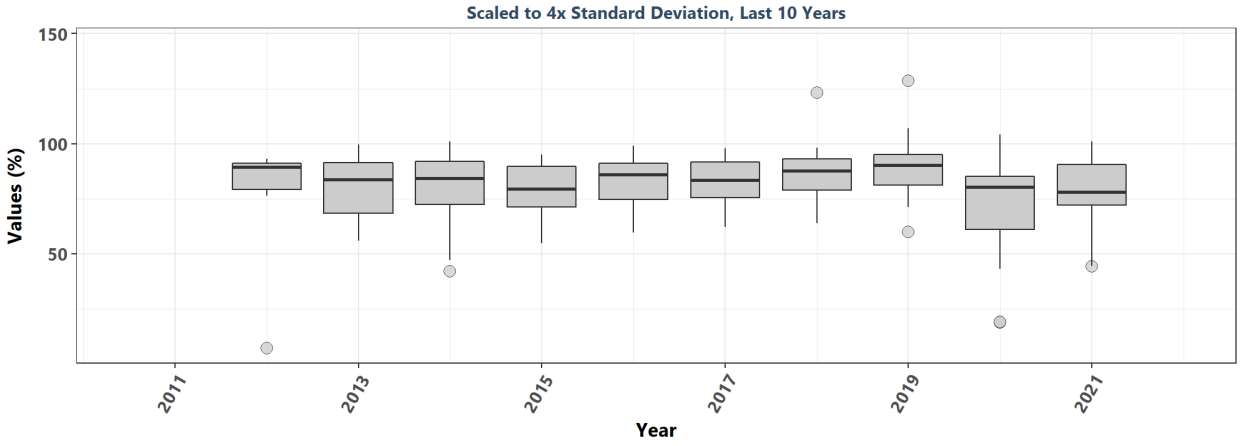

Florida Keys National Marine Sanctuary
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Year
Scaled to 4x Standard Deviation
Values (%)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Year

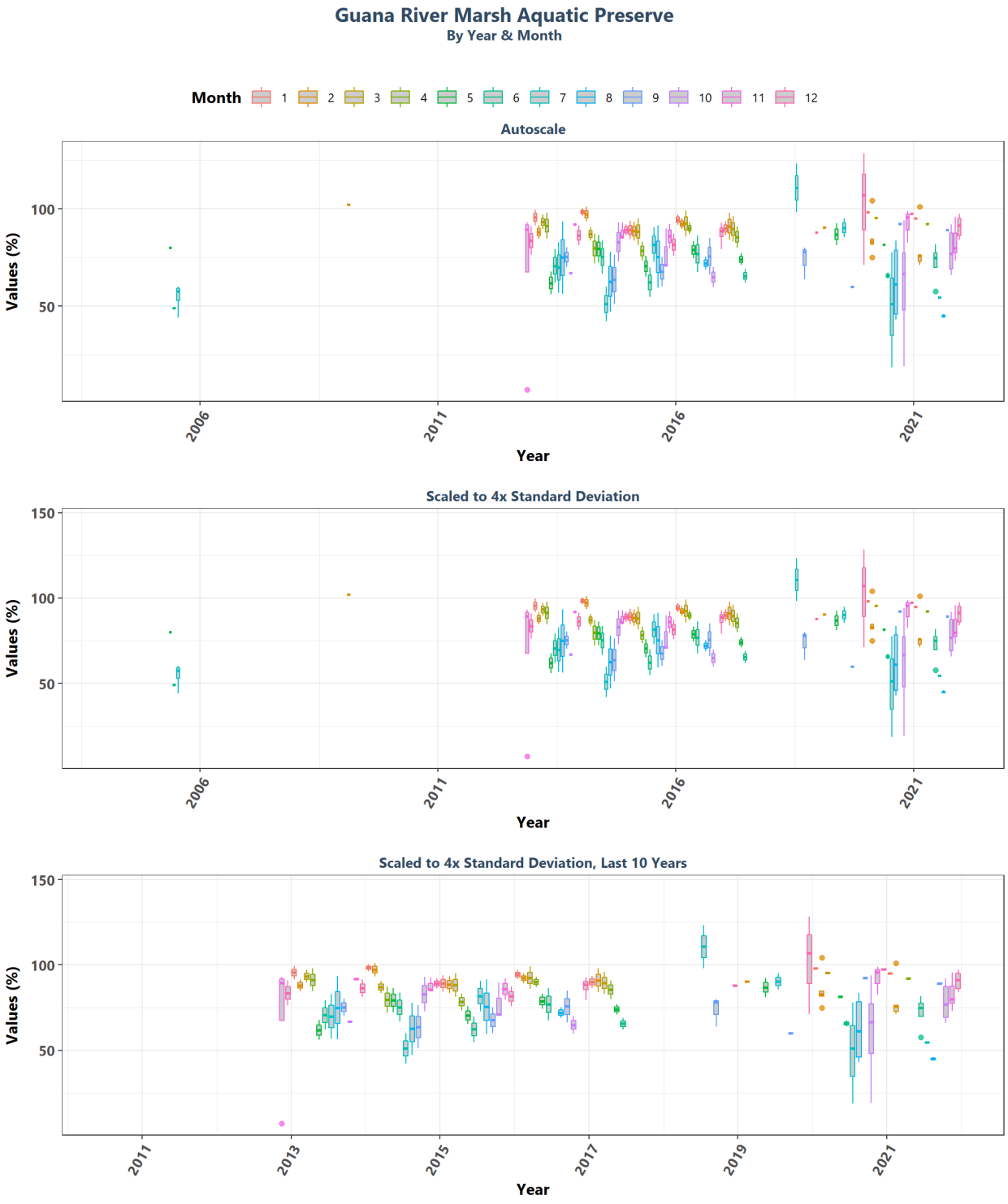

Florida Keys National Marine Sanctuary
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Month
Scaled to 4x Standard Deviation
Values (%)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Month

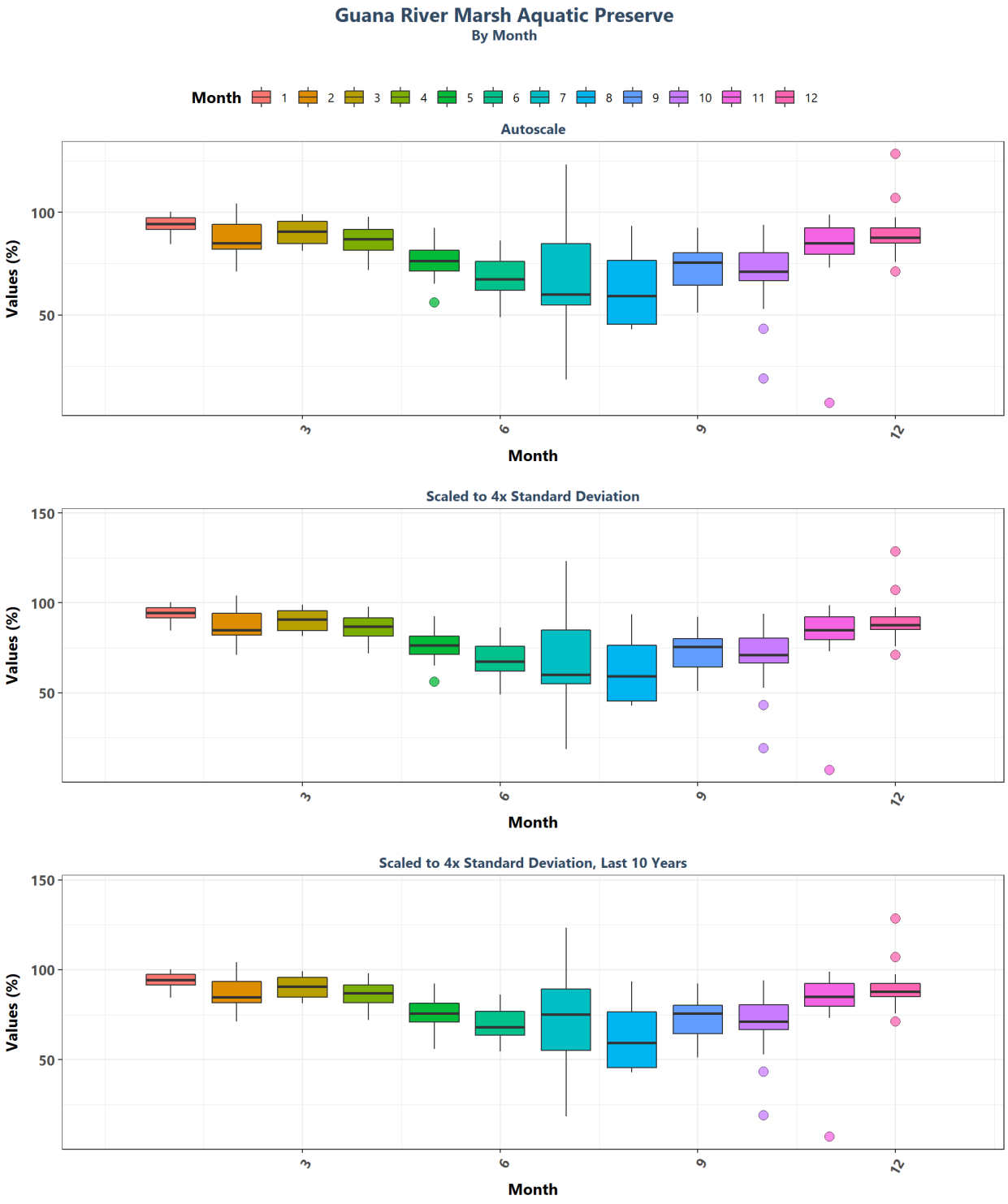
Guana River Marsh Aquatic Preserve
By Year





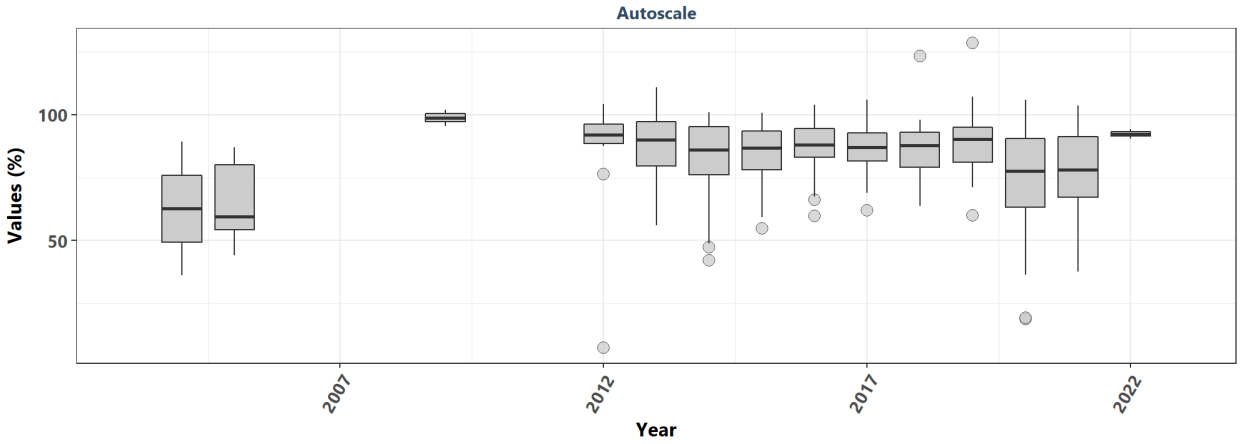


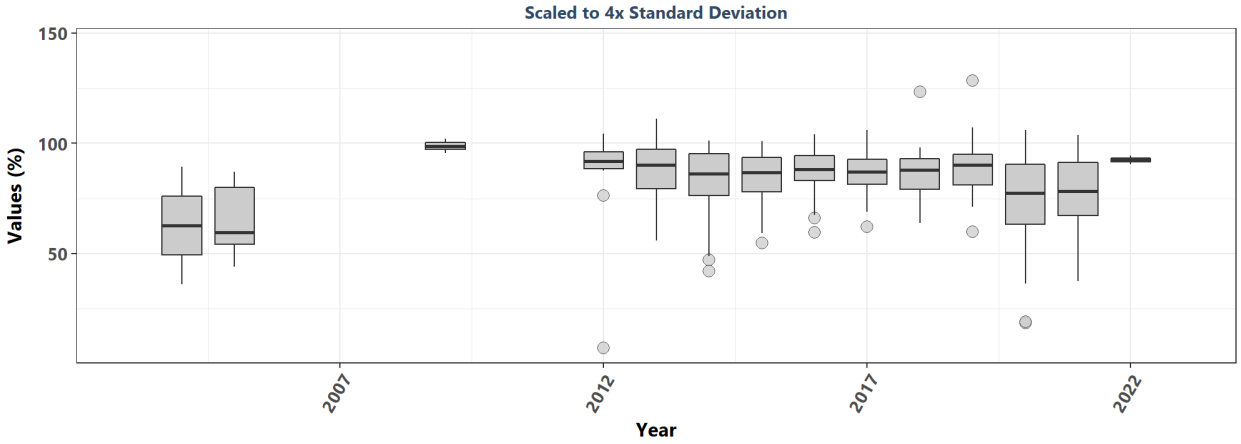

Guana River Marsh Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
2011
2013
2015
2017
2019
2021
Year

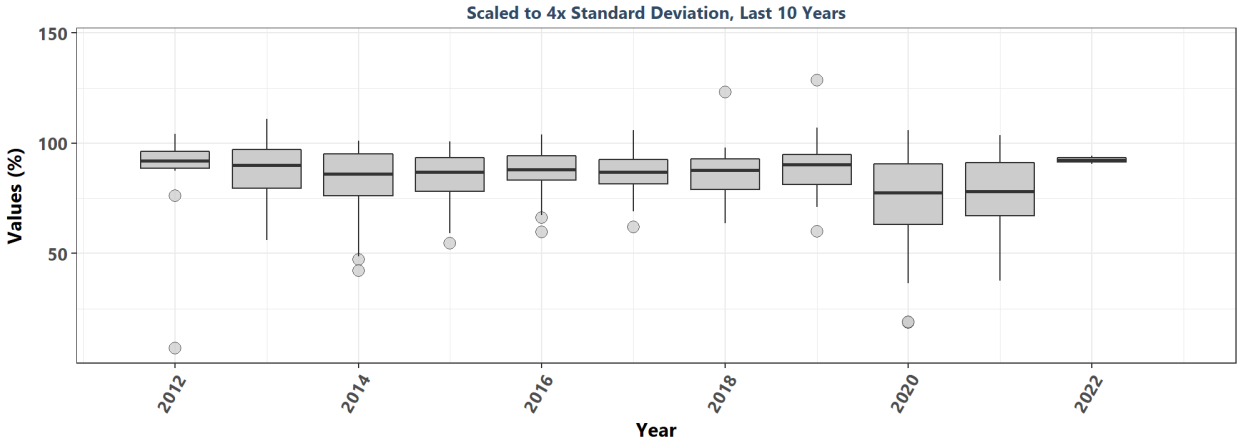

Guana River Marsh Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
150
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
3
6
9
12
Month

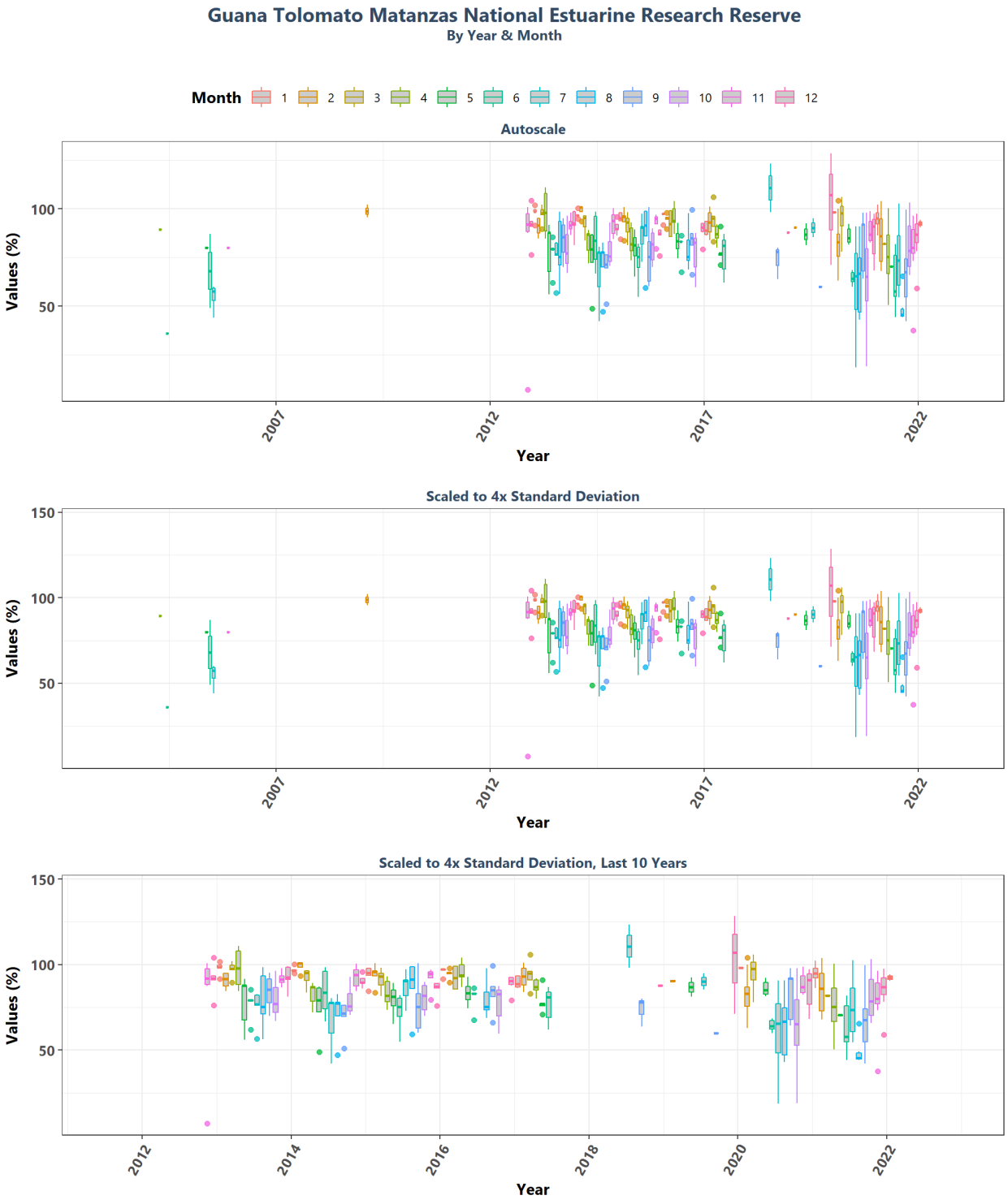
Guana Tolomato Matanzas National Estuarine Research Reserve

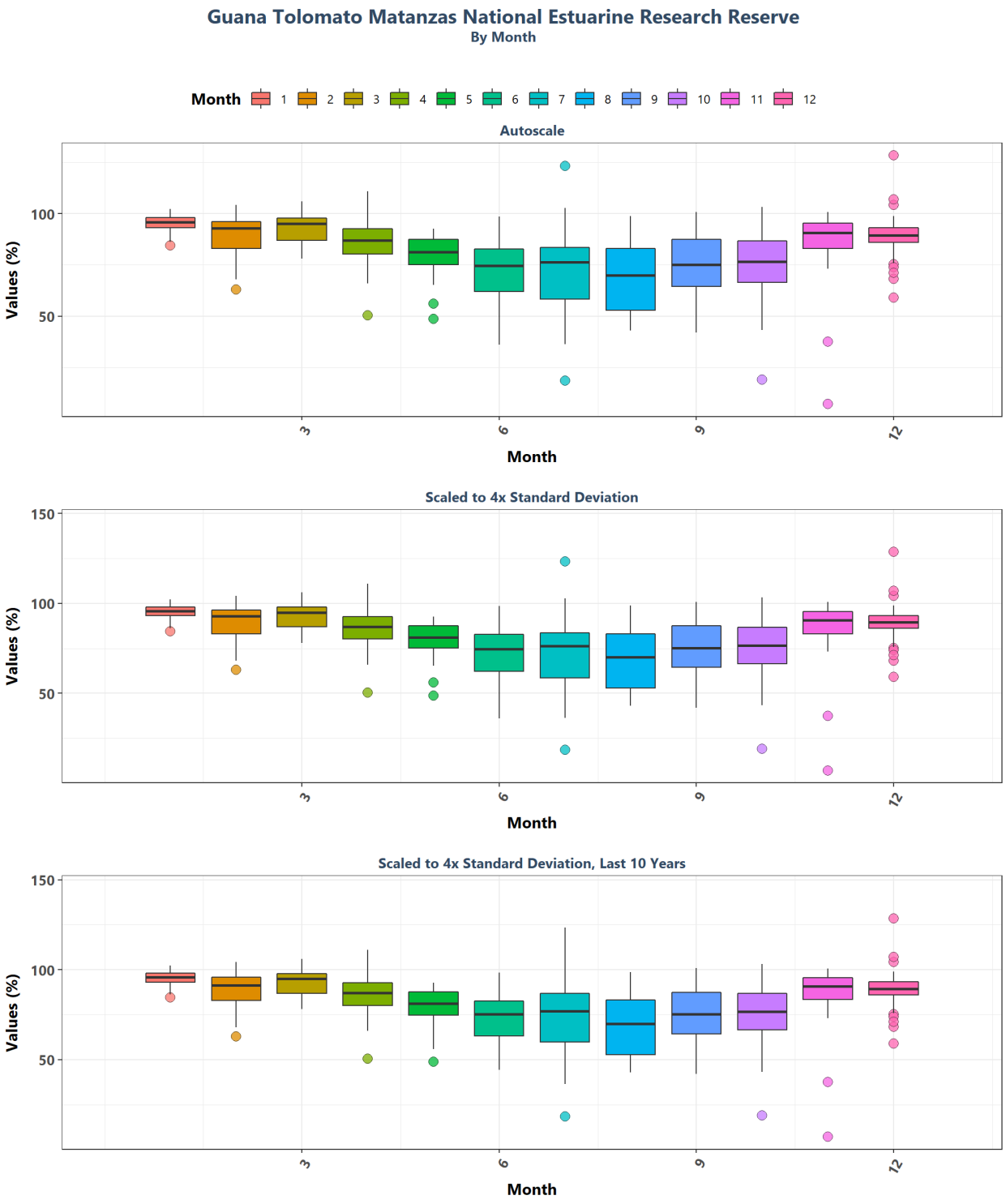
By Year





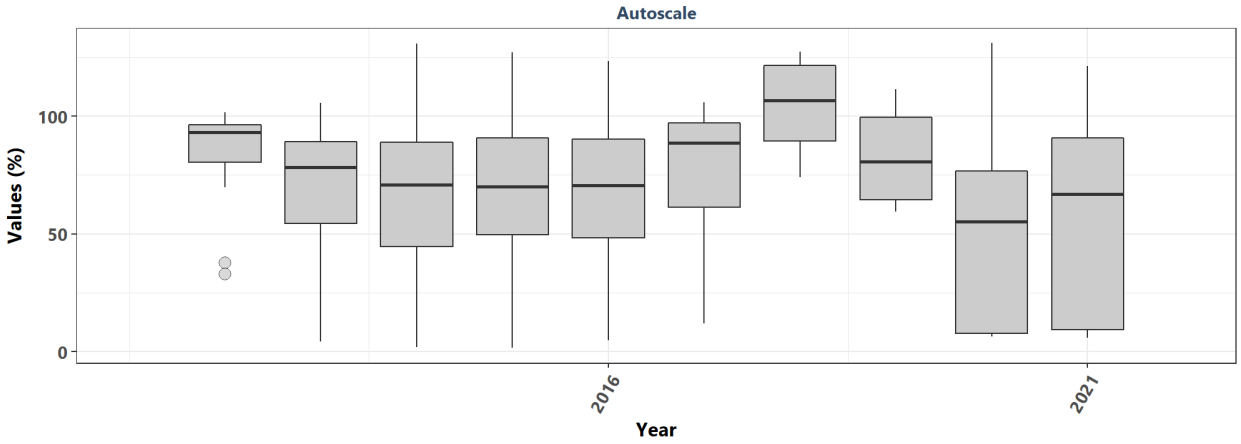


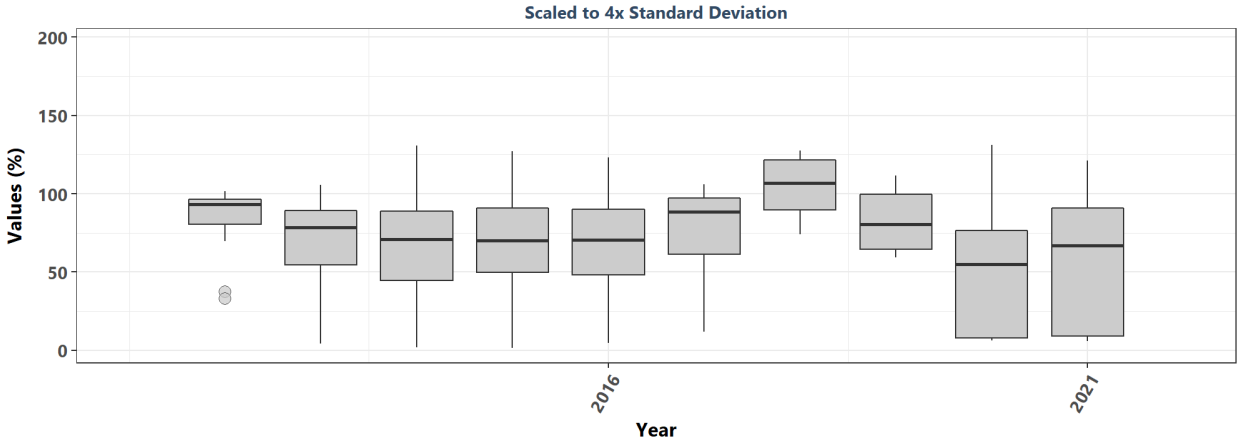

Guana Tolomato Matanzas National Estuarine Research Reserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
2007
2012
2017
2022
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
2012
2014
2016
2018
2020
2022
Year

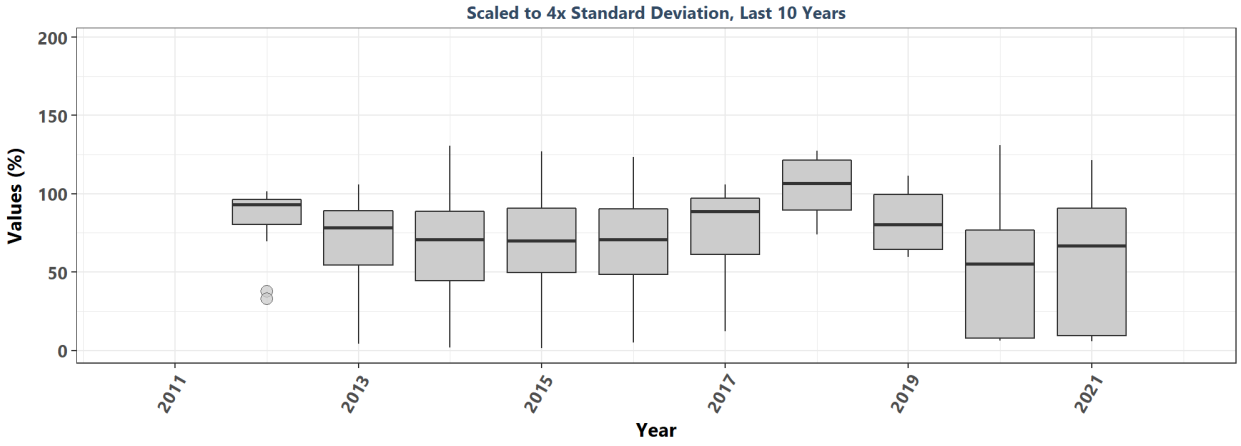

Guana Tolomato Matanzas National Estuarine Research Reserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
150
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
3
6
9
12
Month

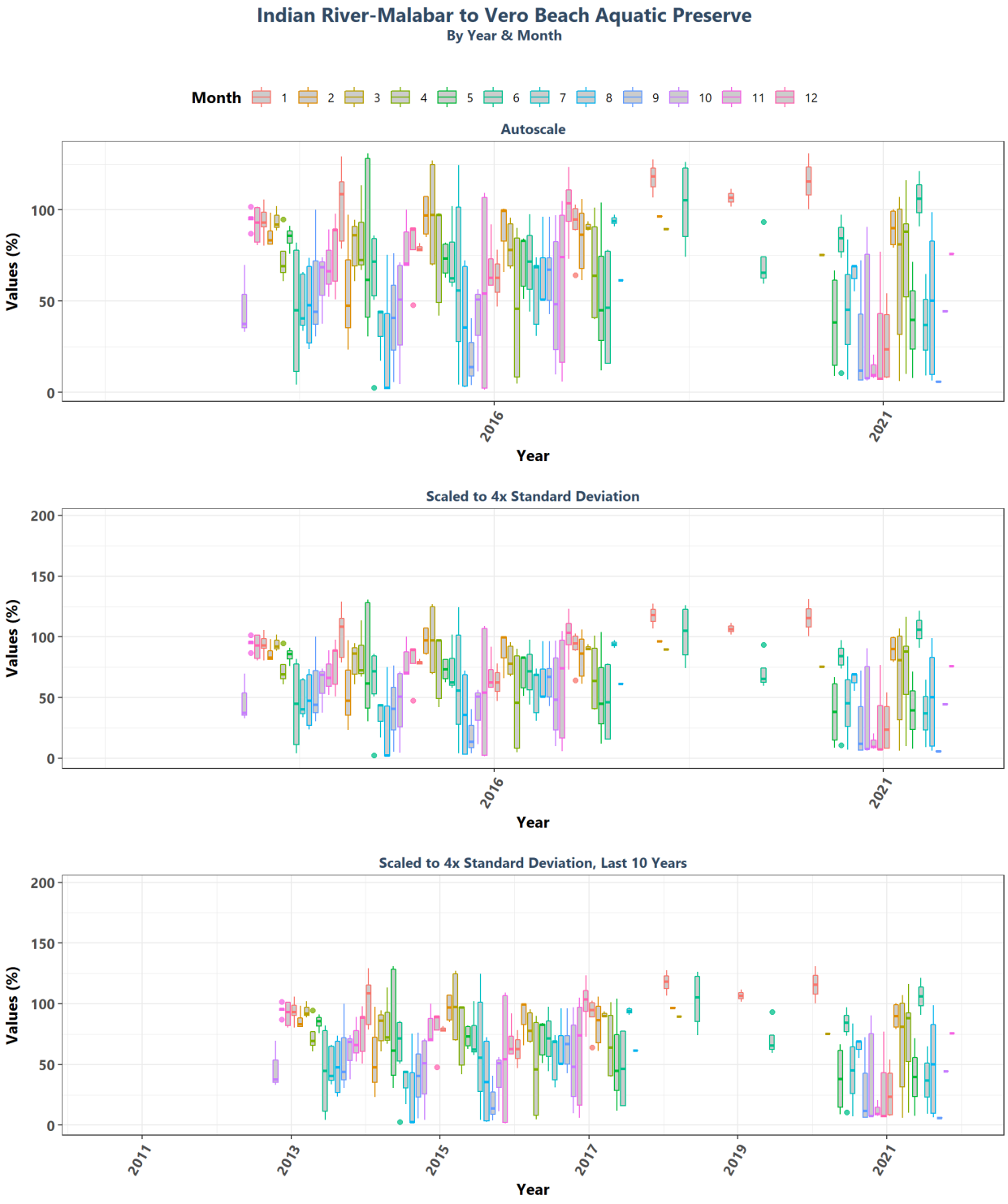
Indian River-Malabar to Vero Beach Aquatic Preserve

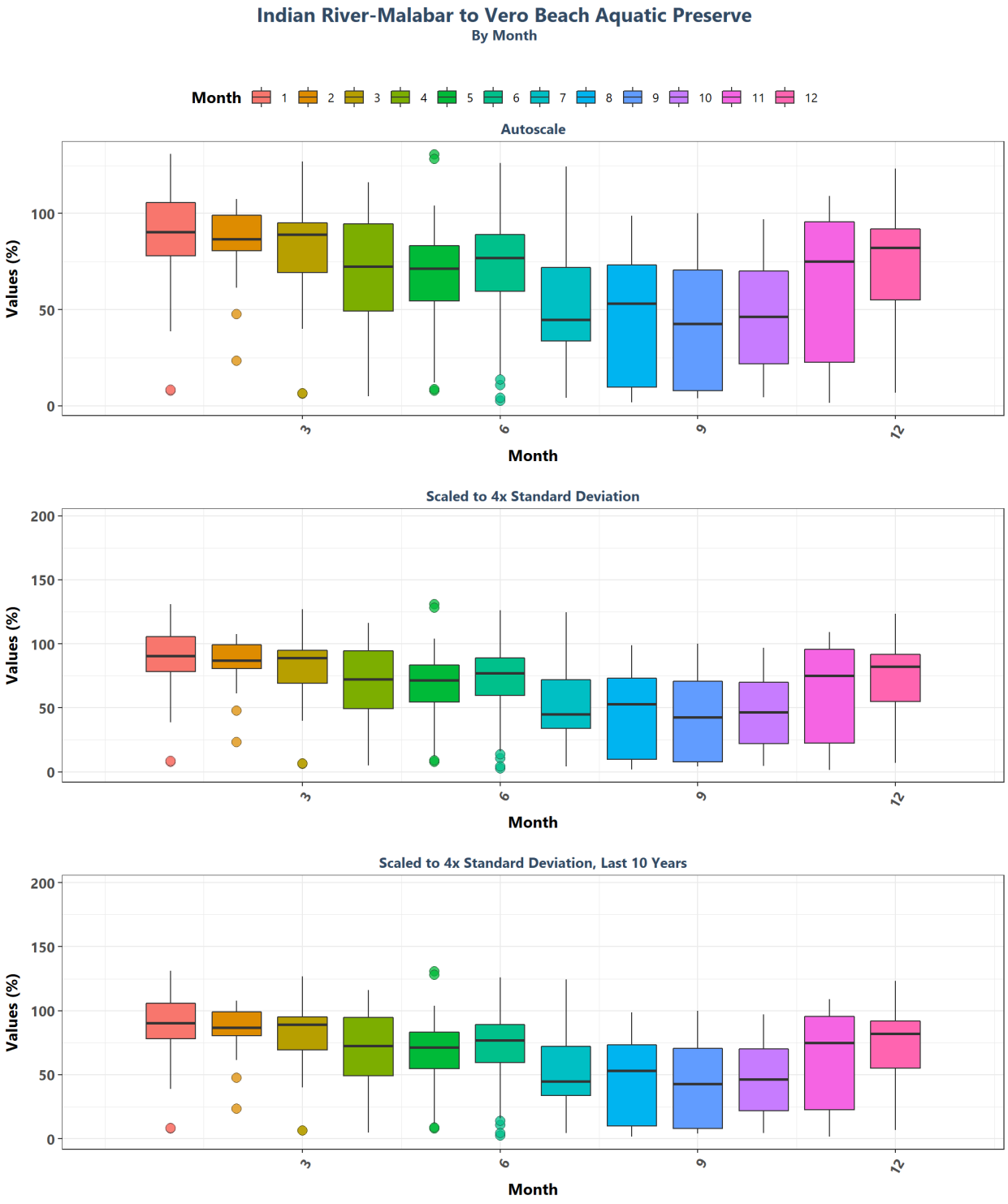
By Year





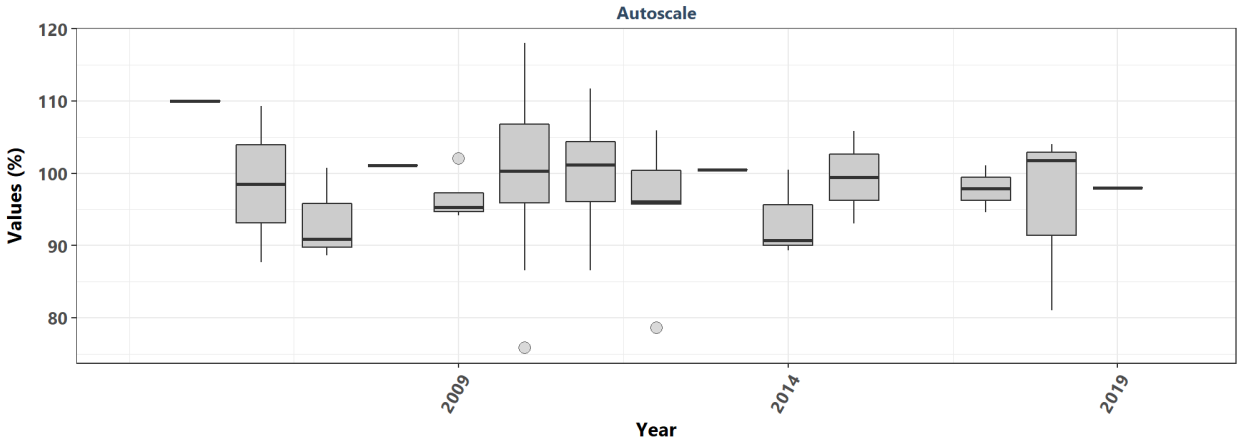


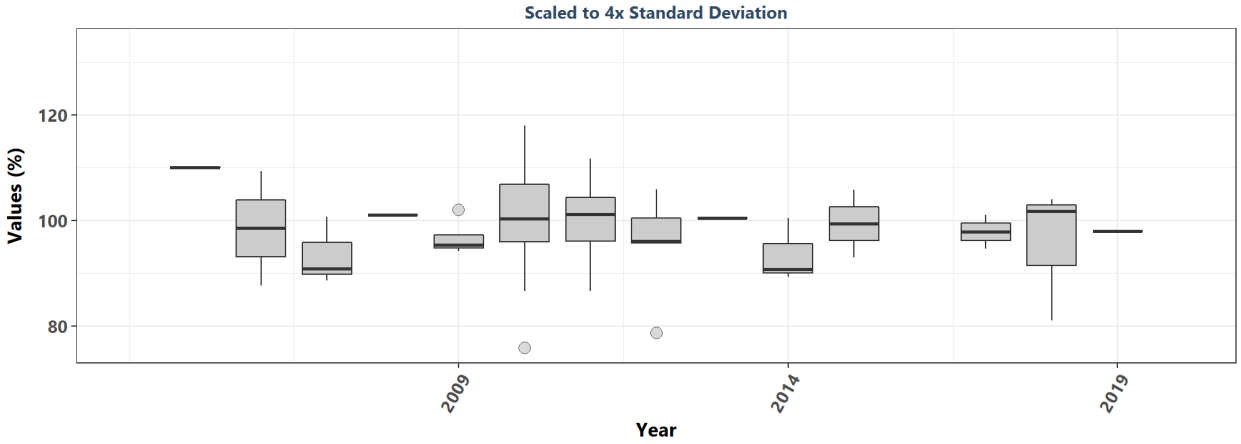

Indian River-Malabar to Vero Beach Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Year
Scaled to 4x Standard Deviation
Scaled to 4x Standard Deviation, Last 10 Years

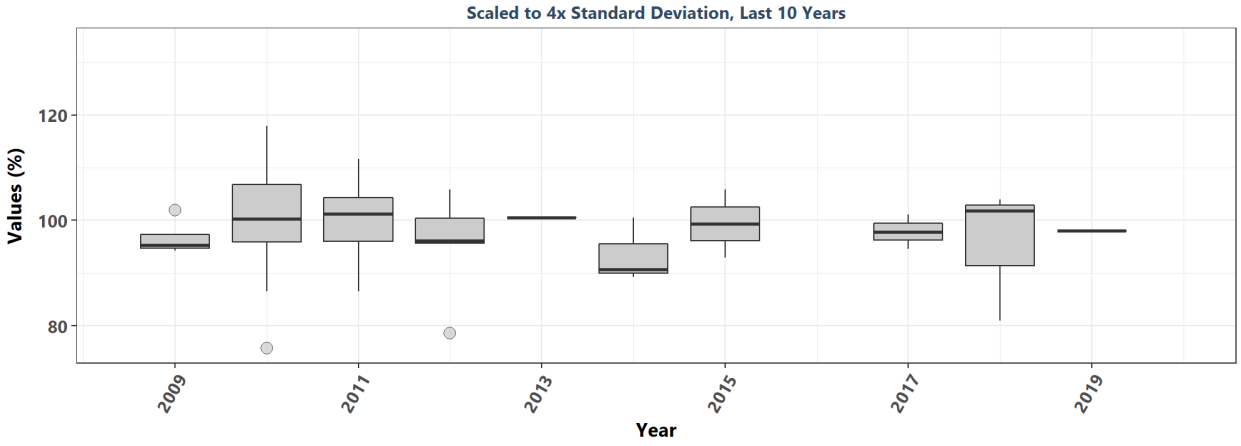

Indian River-Malabar to Vero Beach Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Month
Scaled to 4x Standard Deviation
Values (%)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Month

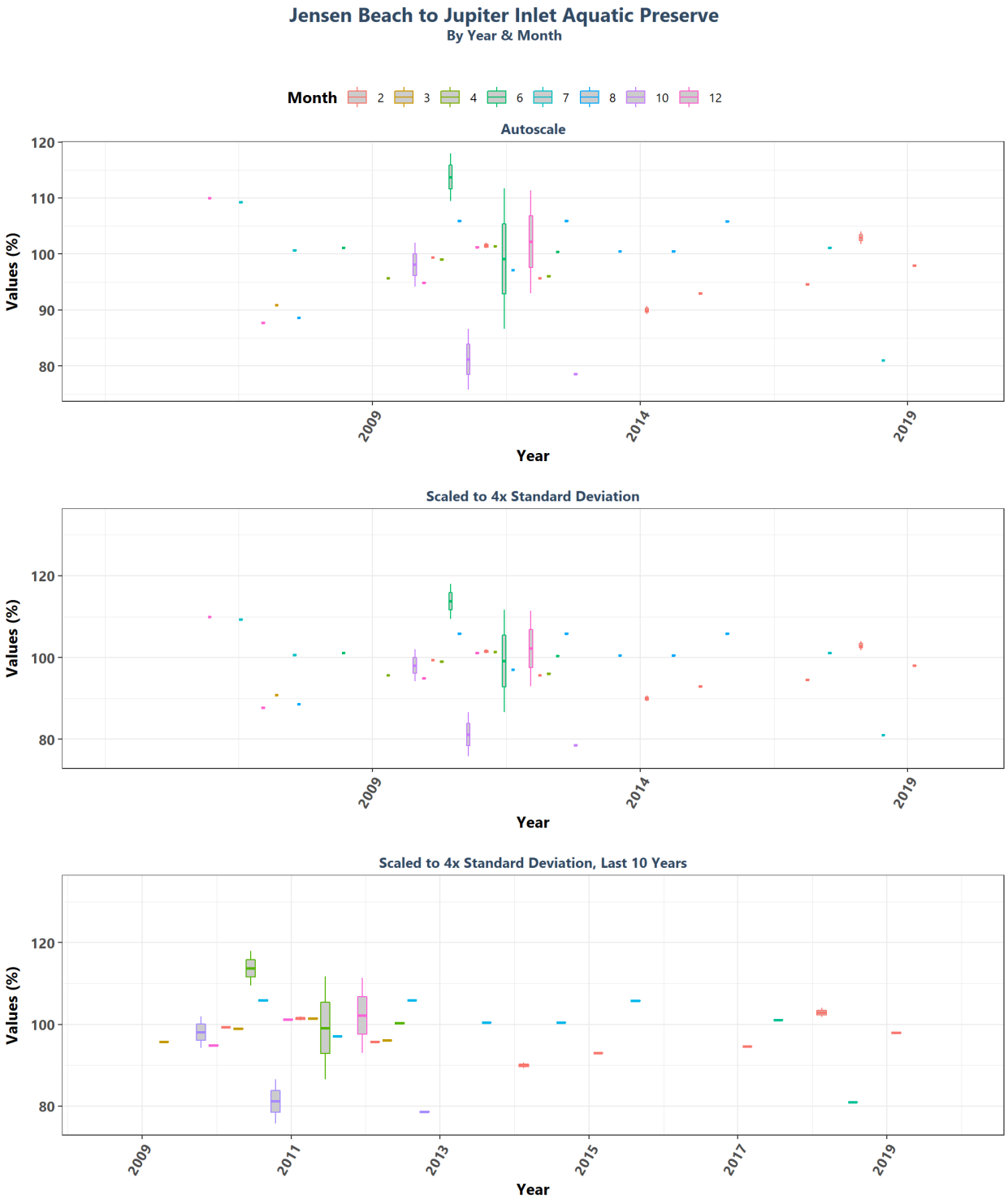
Jensen Beach to Jupiter Inlet Aquatic Preserve

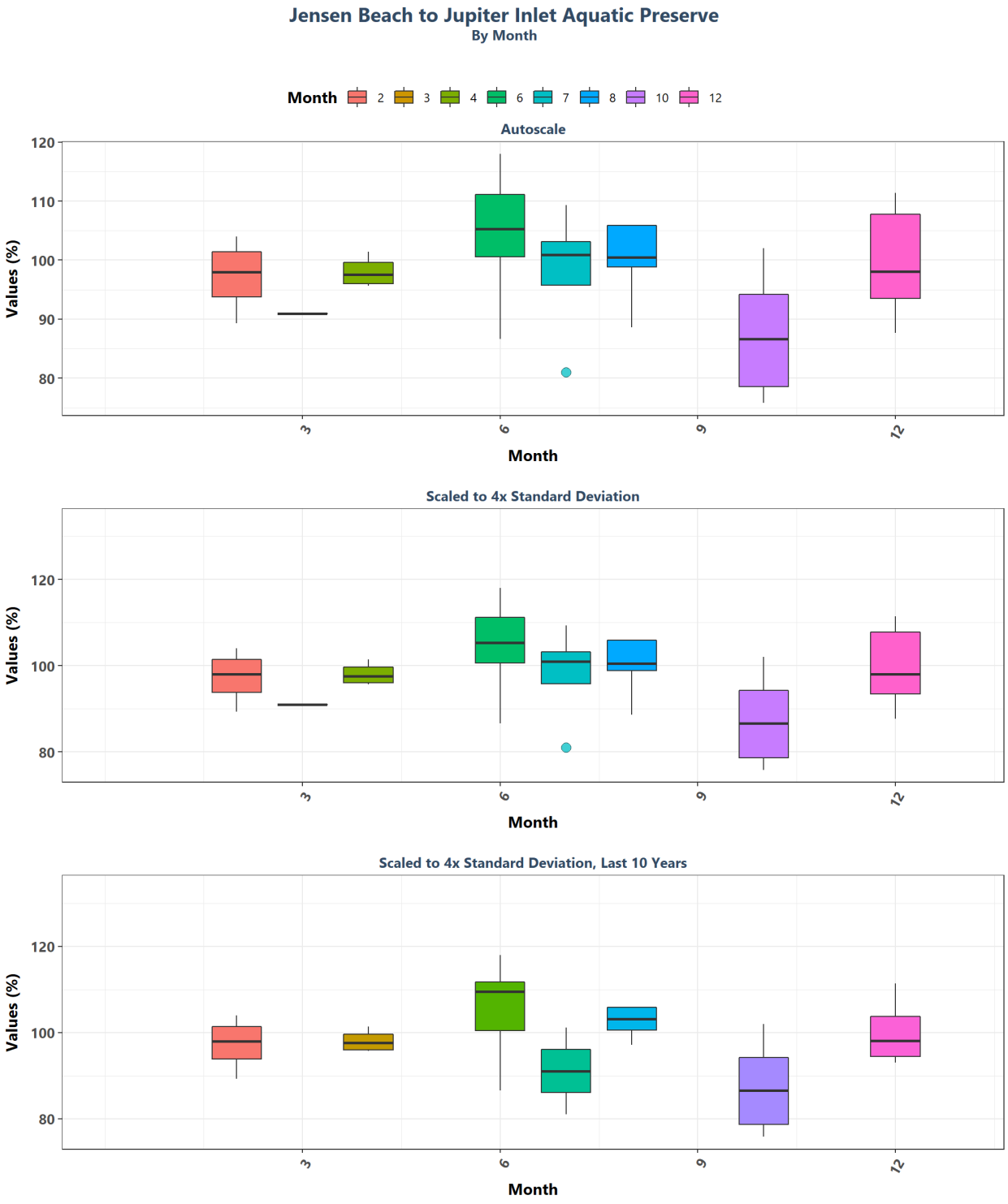
By Year





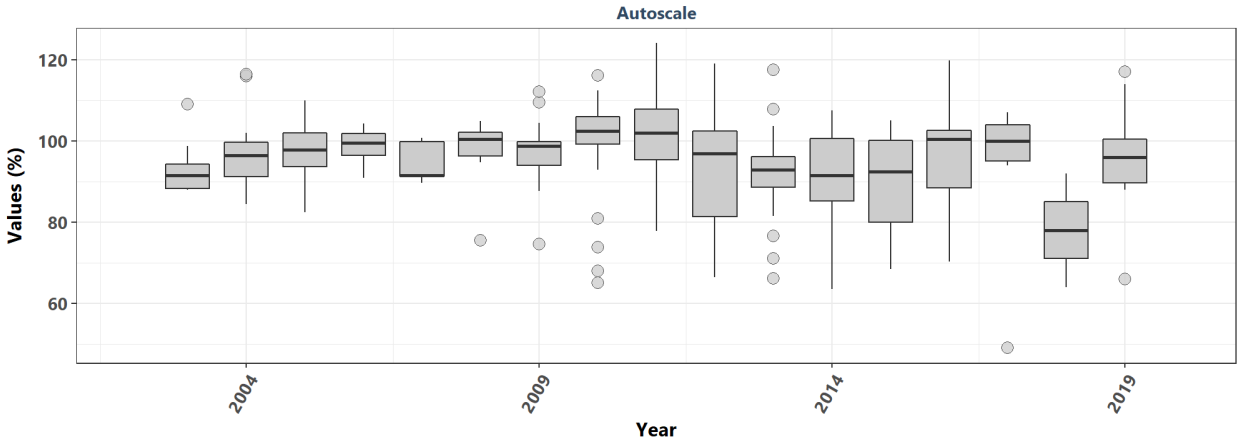


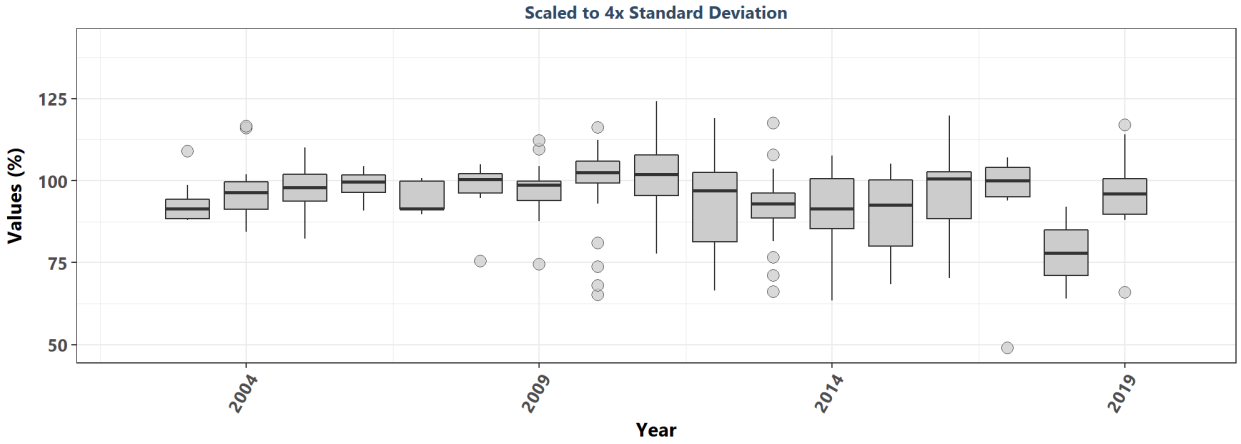

Jensen Beach to Jupiter Inlet Aquatic Preserve
By Year & Month
Month 2 3 4 6 7 8 10 12
Autoscale
Values (%)
120
110
100
90
80
2009
2014
2019
Year
Scaled to 4x Standard Deviation
Values (%)
120
100
80
2009
2014
2019
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
120
100
80
2009
2011
2013
2015
2017
2019
Year

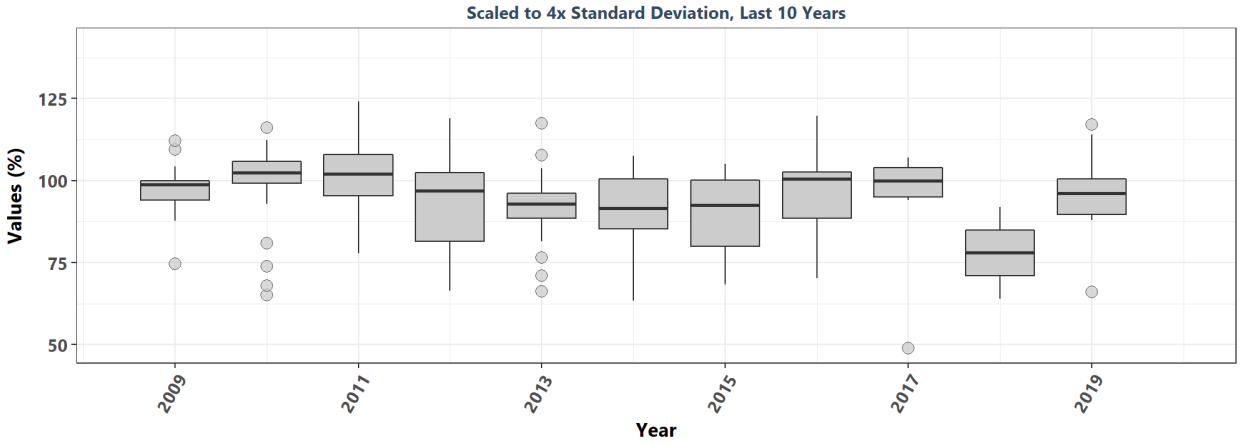

Jensen Beach to Jupiter Inlet Aquatic Preserve
By Month
Month 2 3 4 6 7 8 10 12
Autoscale
Values (%)
120
110
100
90
80
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
120
100
80
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
120
100
80
3
6
9
12
Month

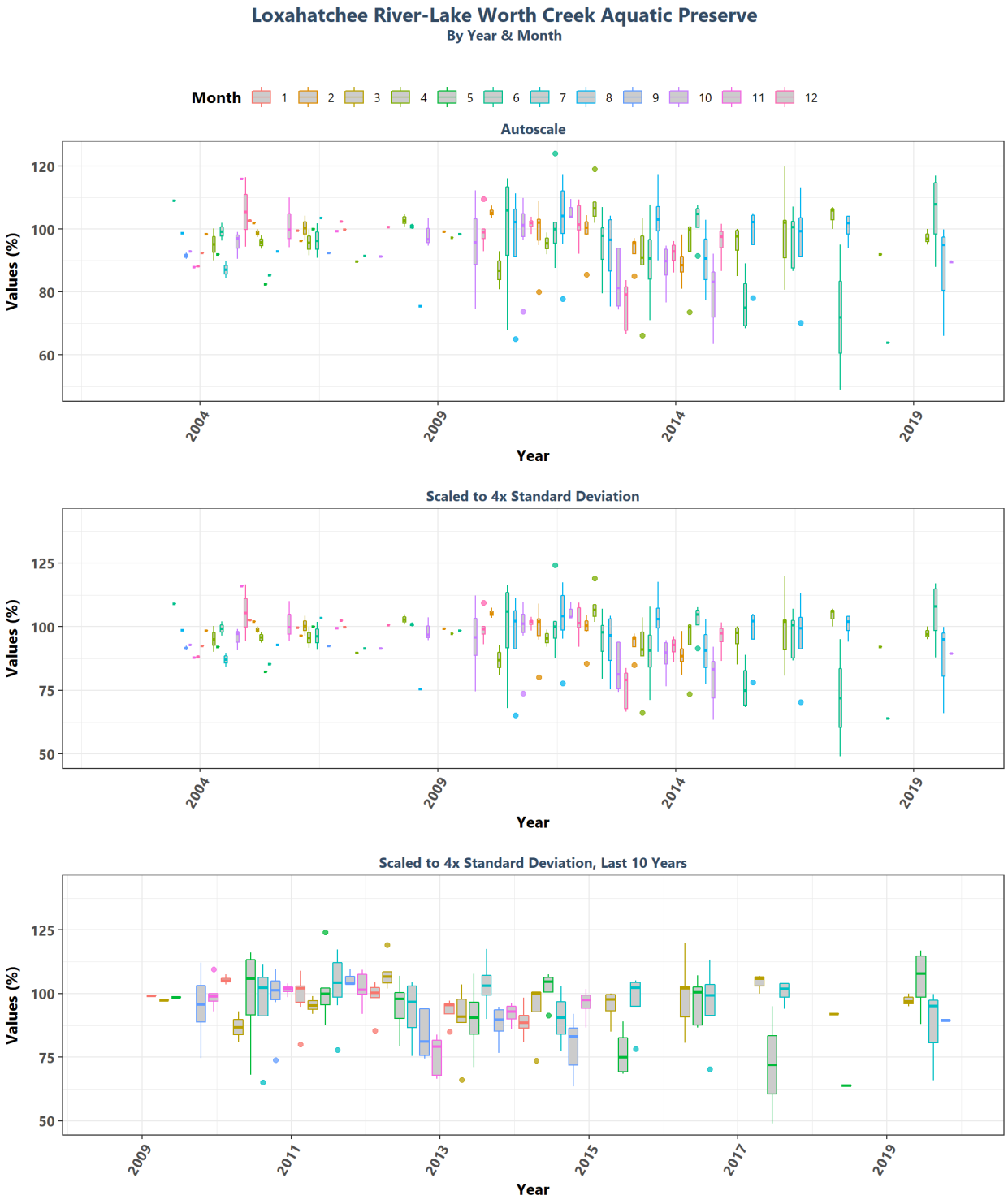
Loxahatchee River-Lake Worth Creek Aquatic Preserve

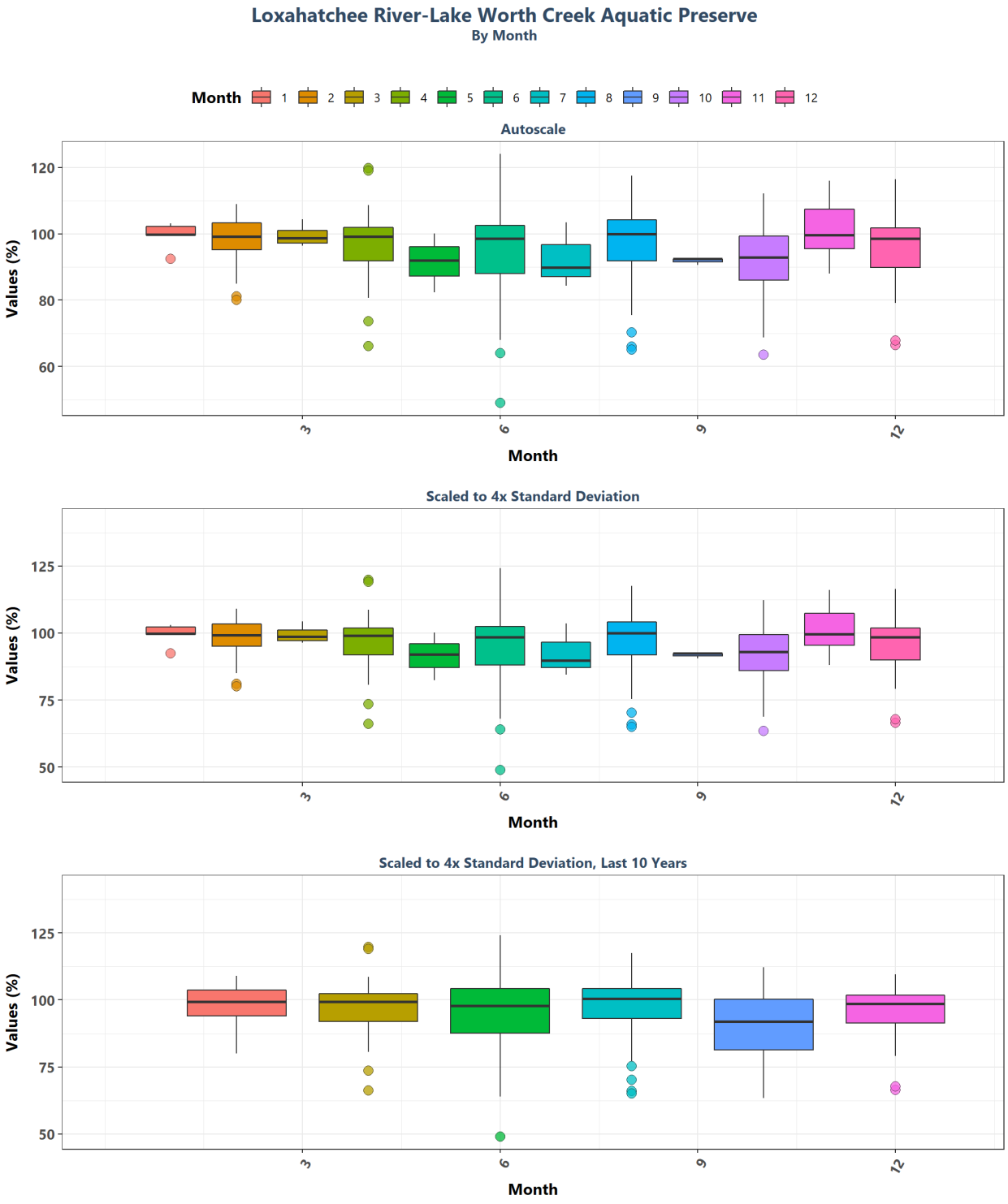
By Year





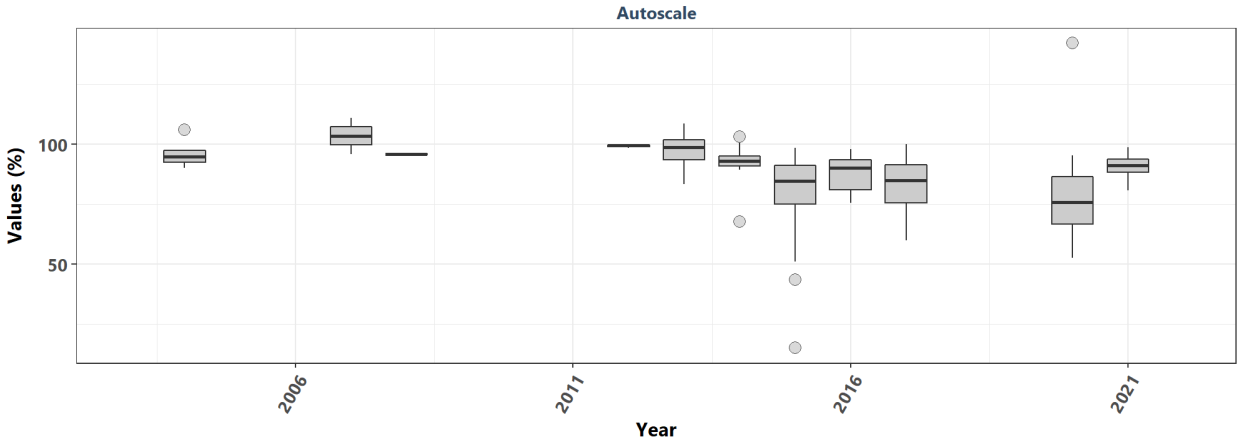


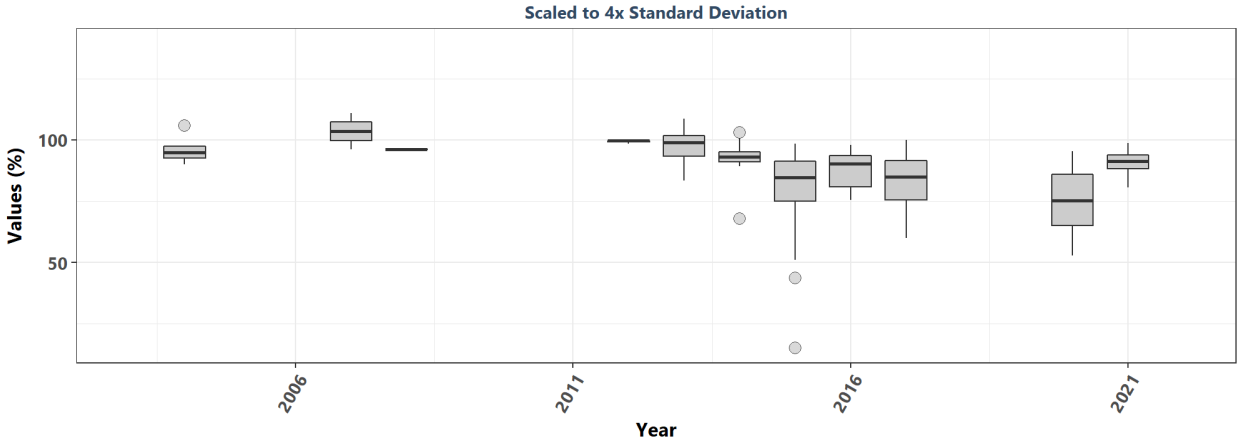

Loxahatchee River-Lake Worth Creek Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
120
100
80
60
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation
Values (%)
125
100
75
50
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
125
100
75
50
2009
2011
2013
2015
2017
2019
Year

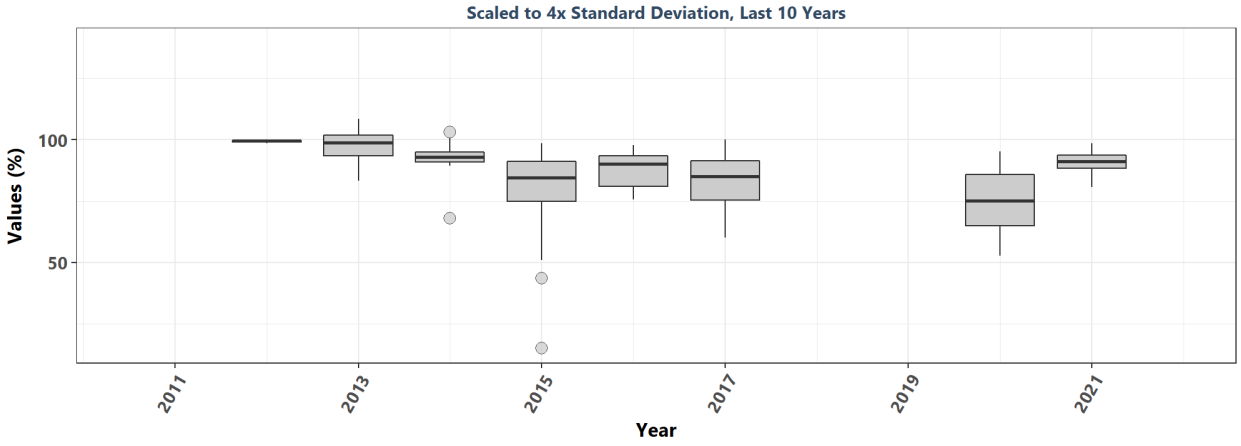

Loxahatchee River-Lake Worth Creek Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
120
100
80
60
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
125
100
75
50
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
125
100
75
50
3
6
9
12
Month

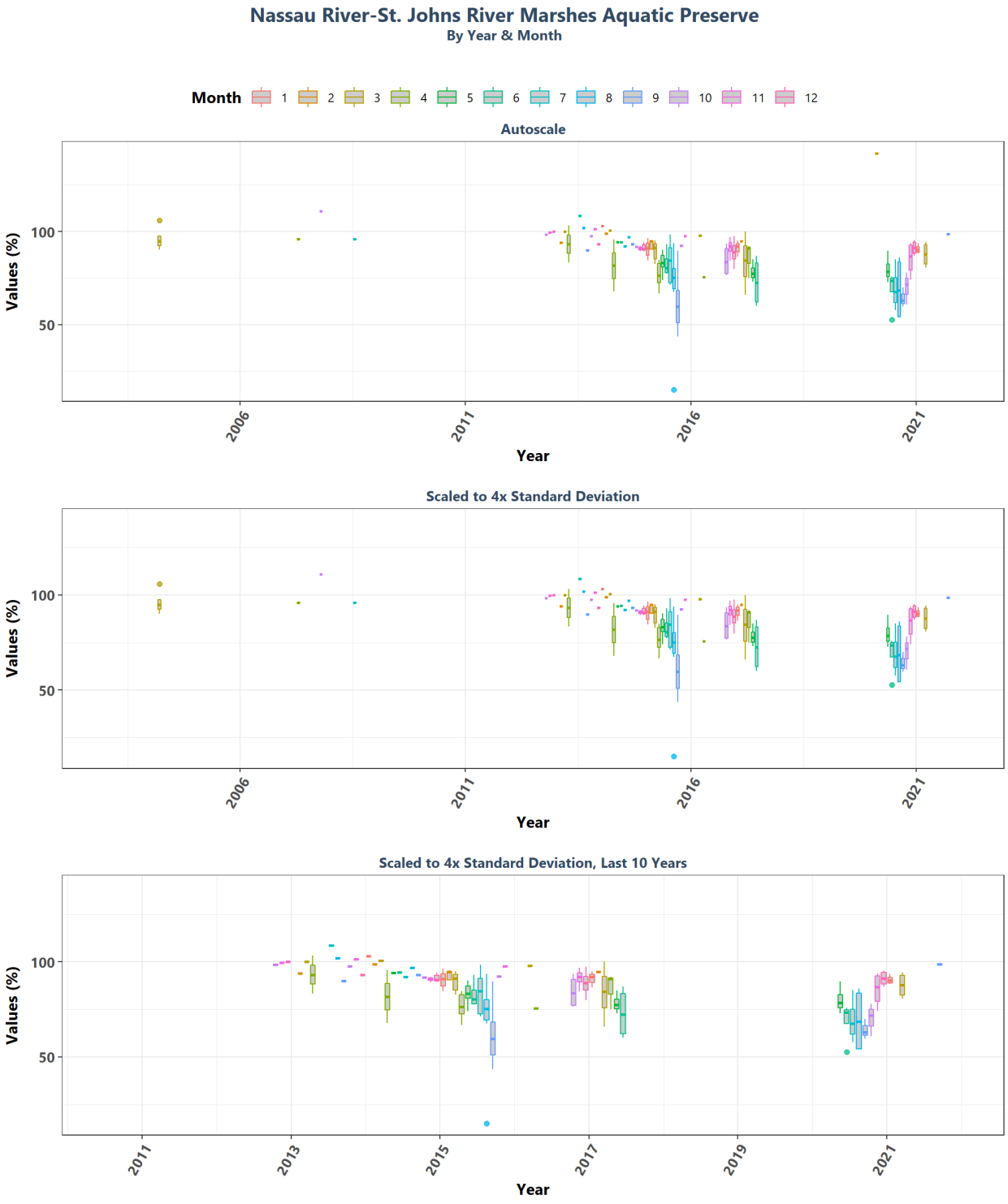
Nassau River-St. Johns River Marshes Aquatic Preserve

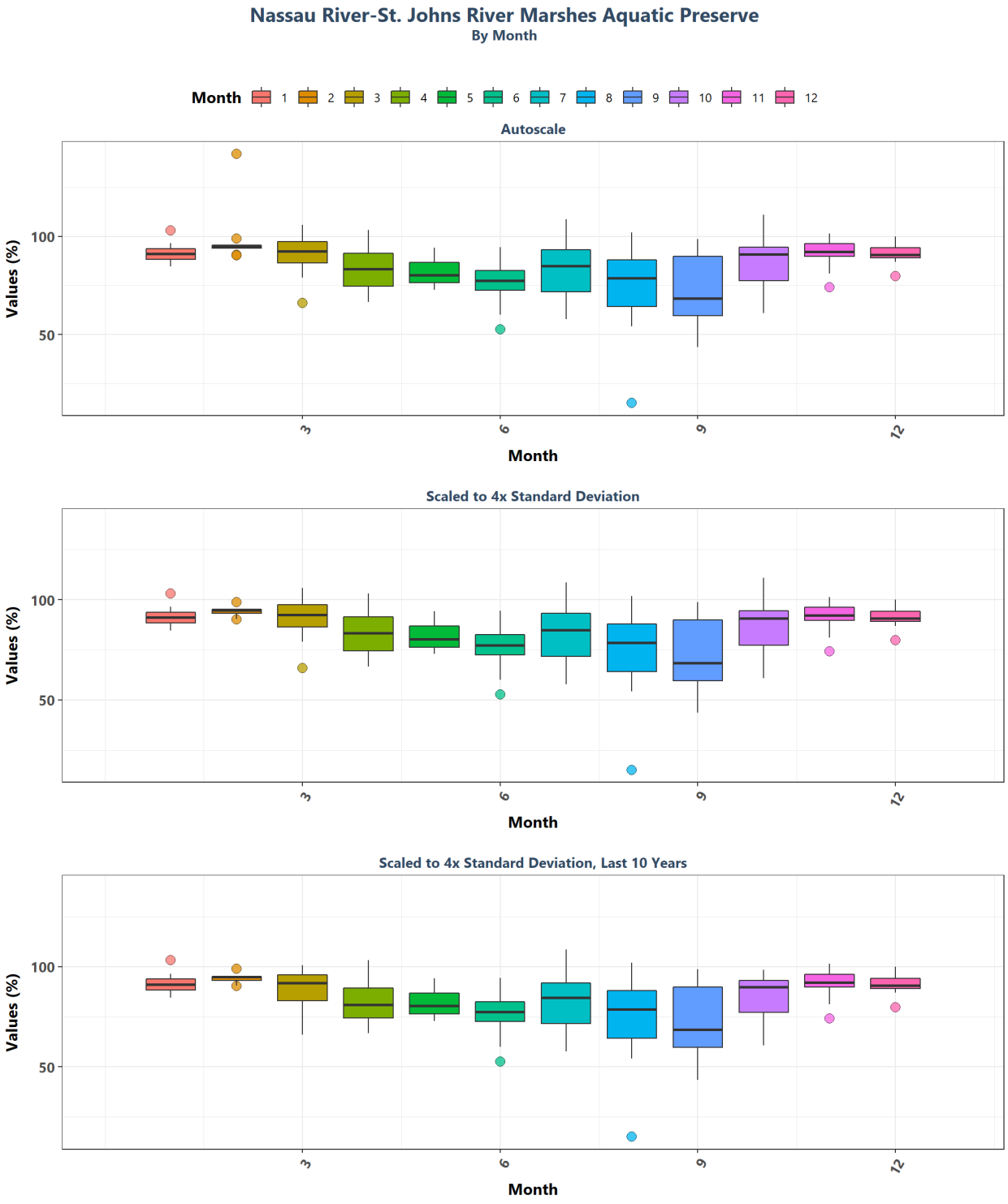
By Year





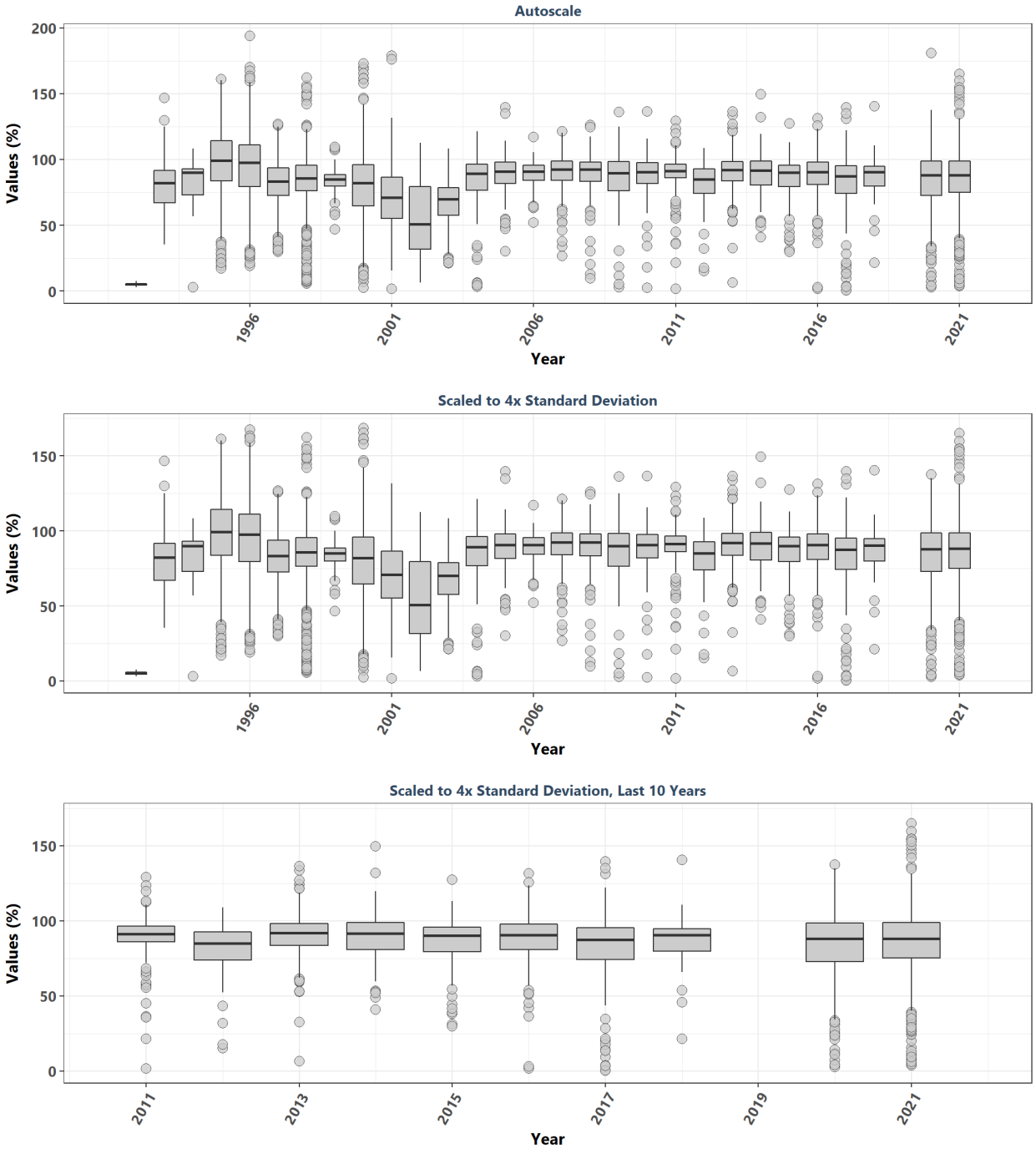


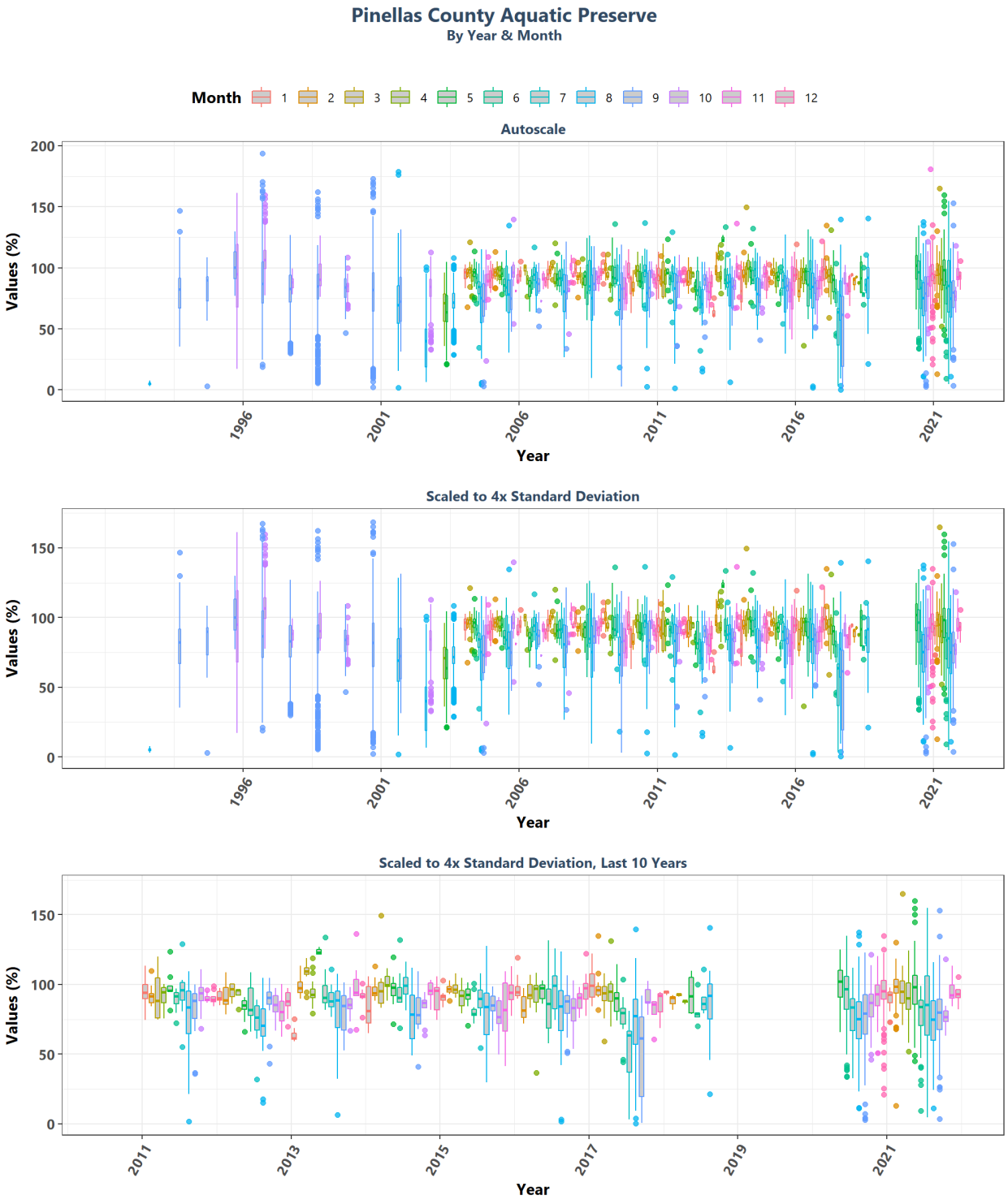

Nassau River-St. Johns River Marshes Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (%)
100
50
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
100
50
2011
2013
2015
2017
2019
2021
Year

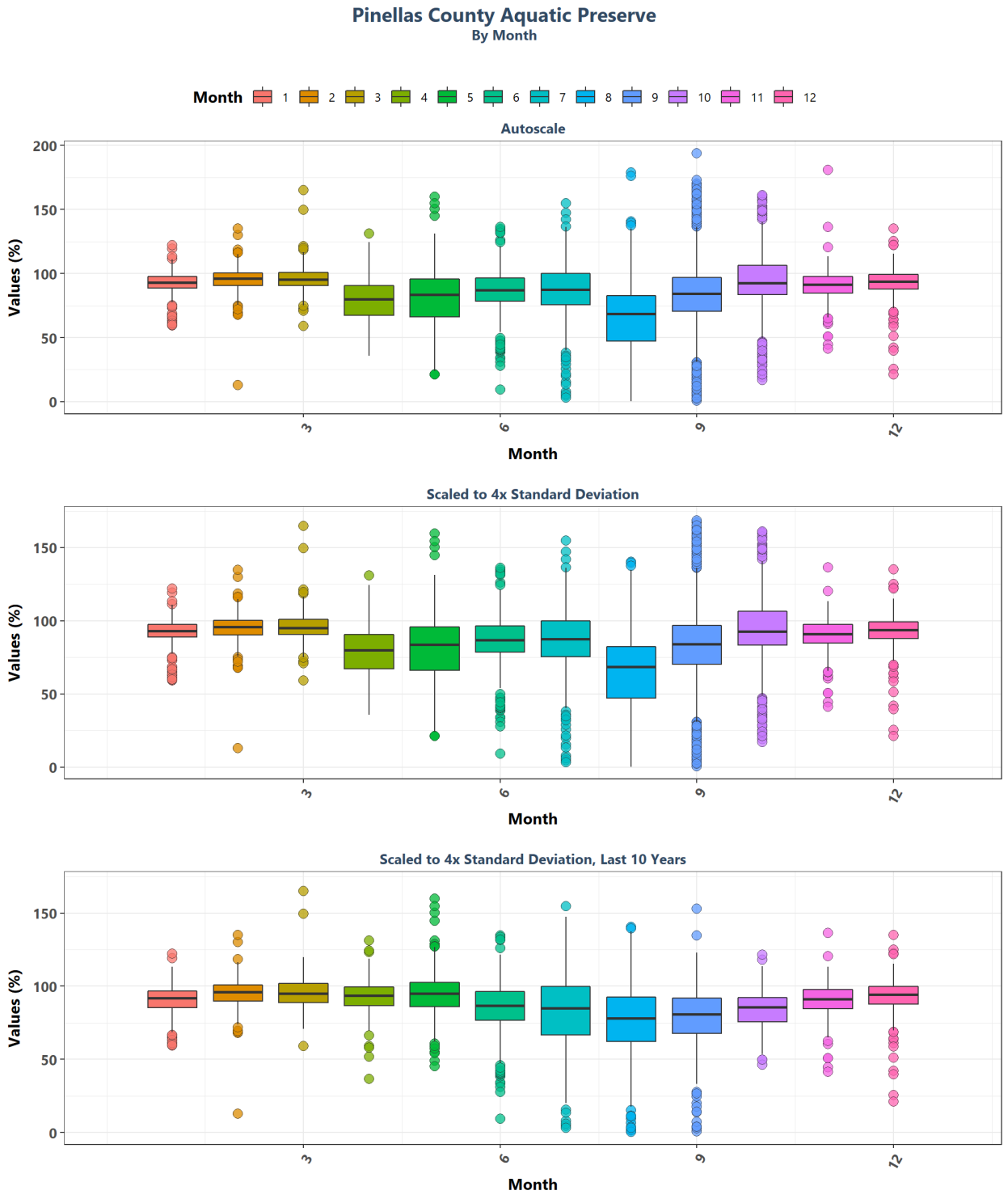

Nassau River-St. Johns River Marshes Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
100
50
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
100
50
3
6
9
12
Month

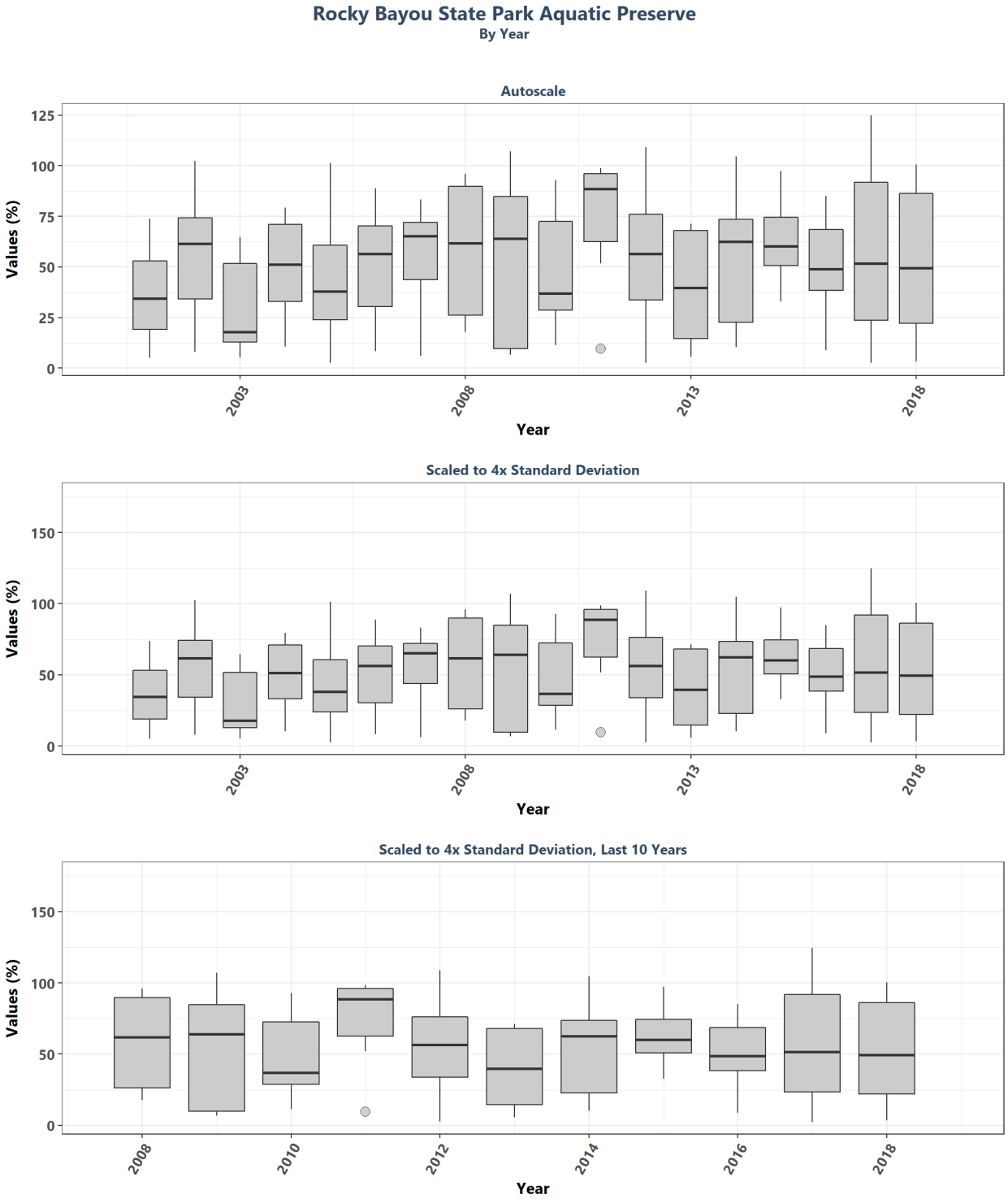
Pinellas County Aquatic Preserve

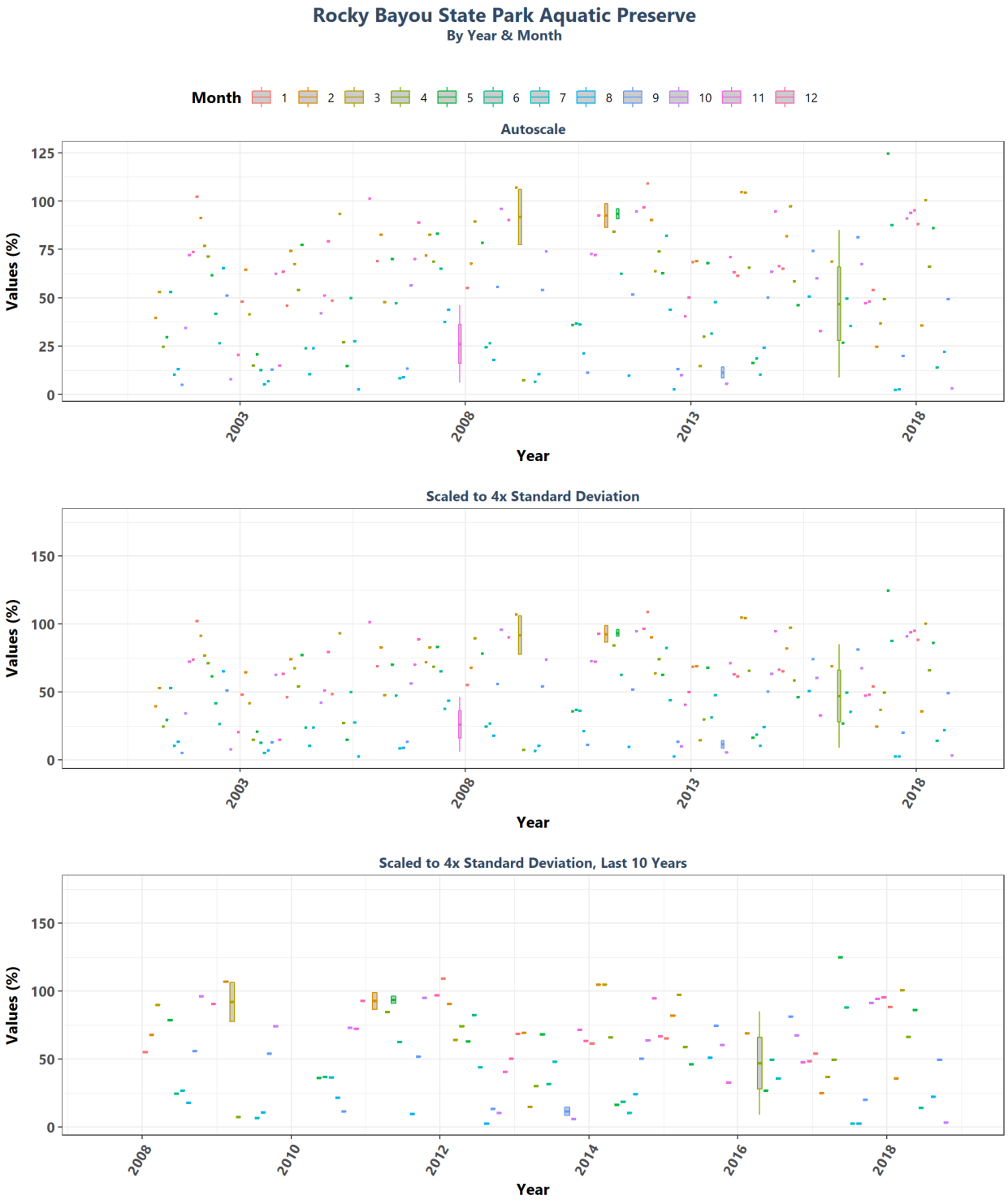
By Year

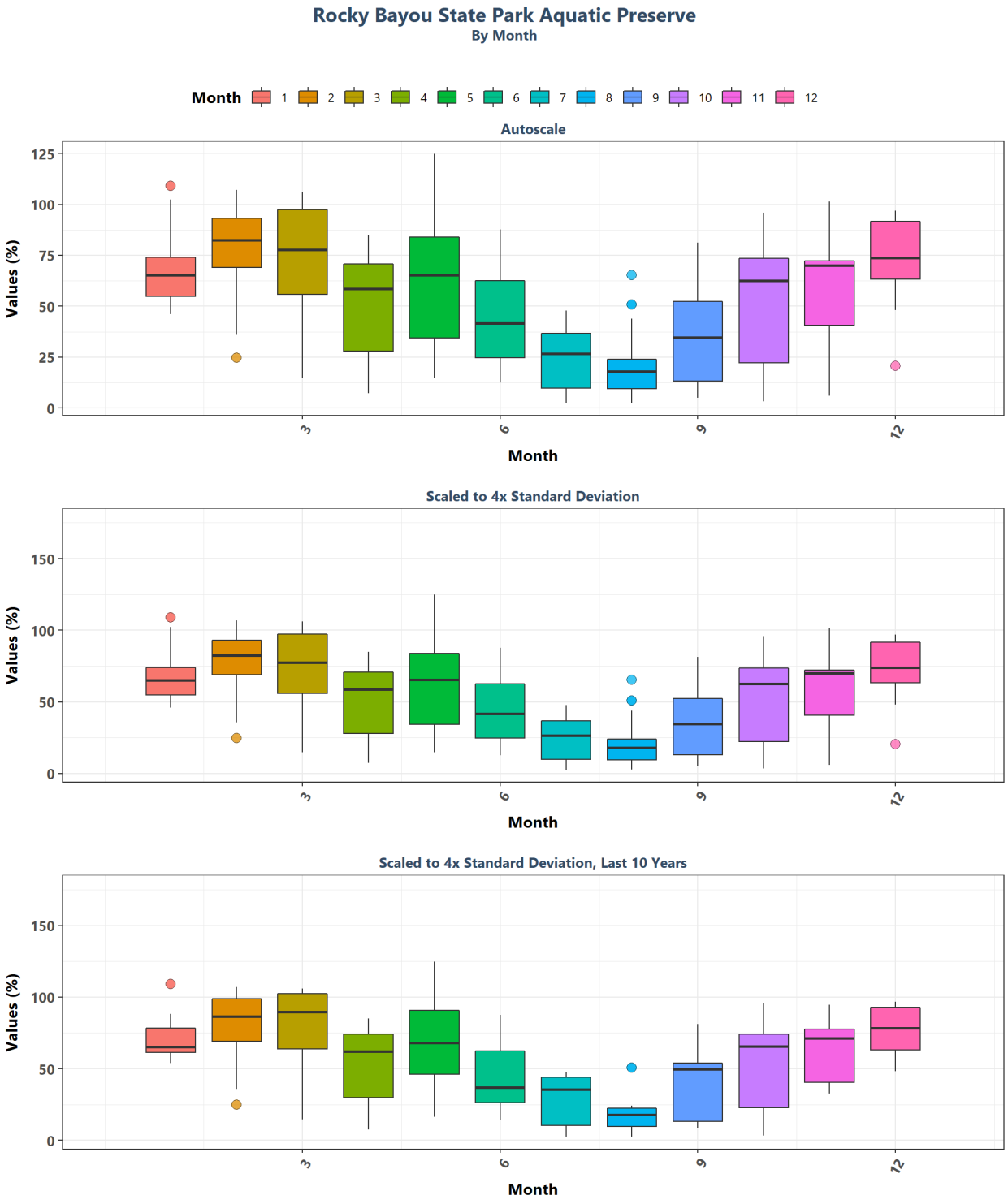


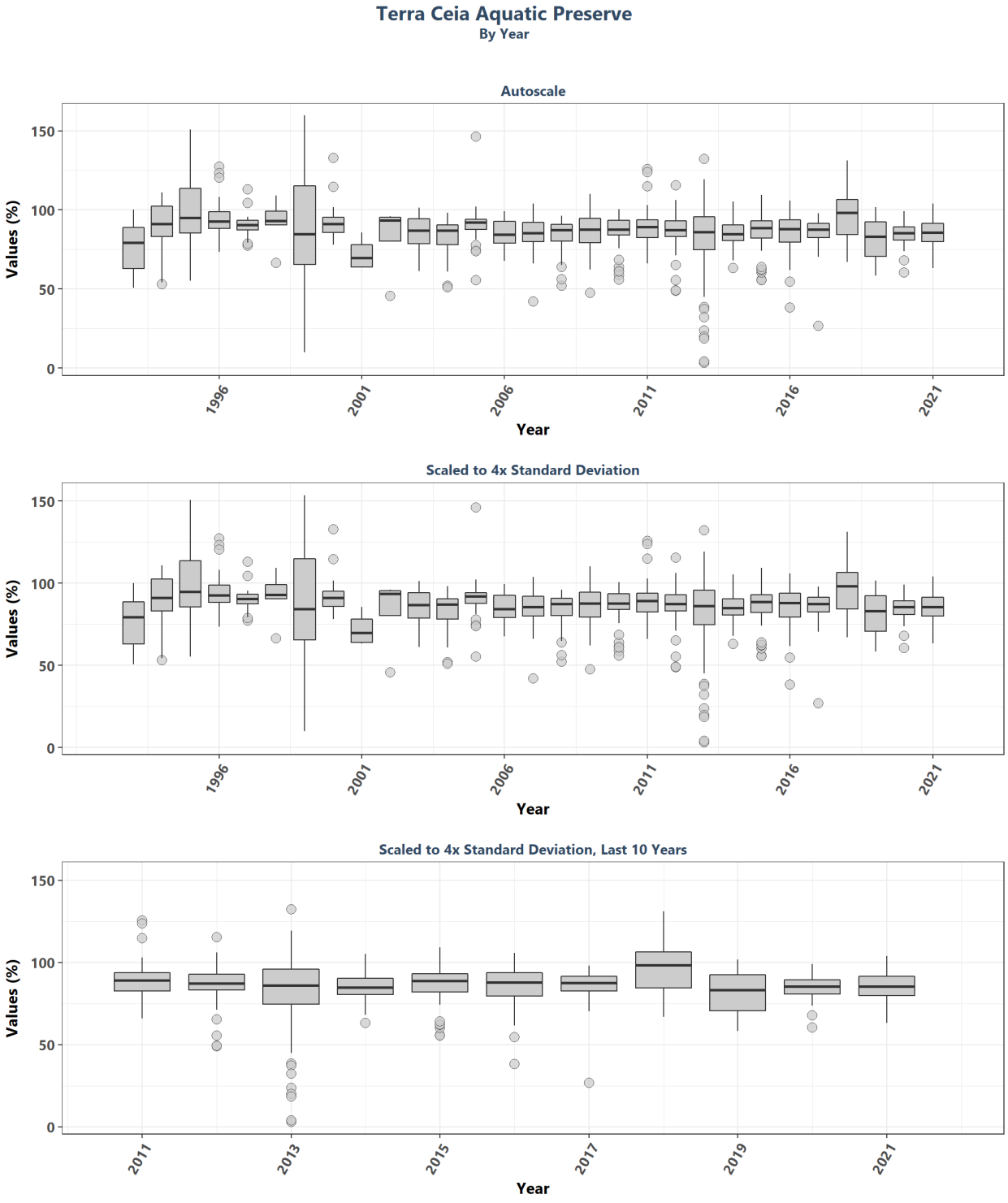

Pinellas County Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Year
Scaled to 4x Standard Deviation
Values (%)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Year

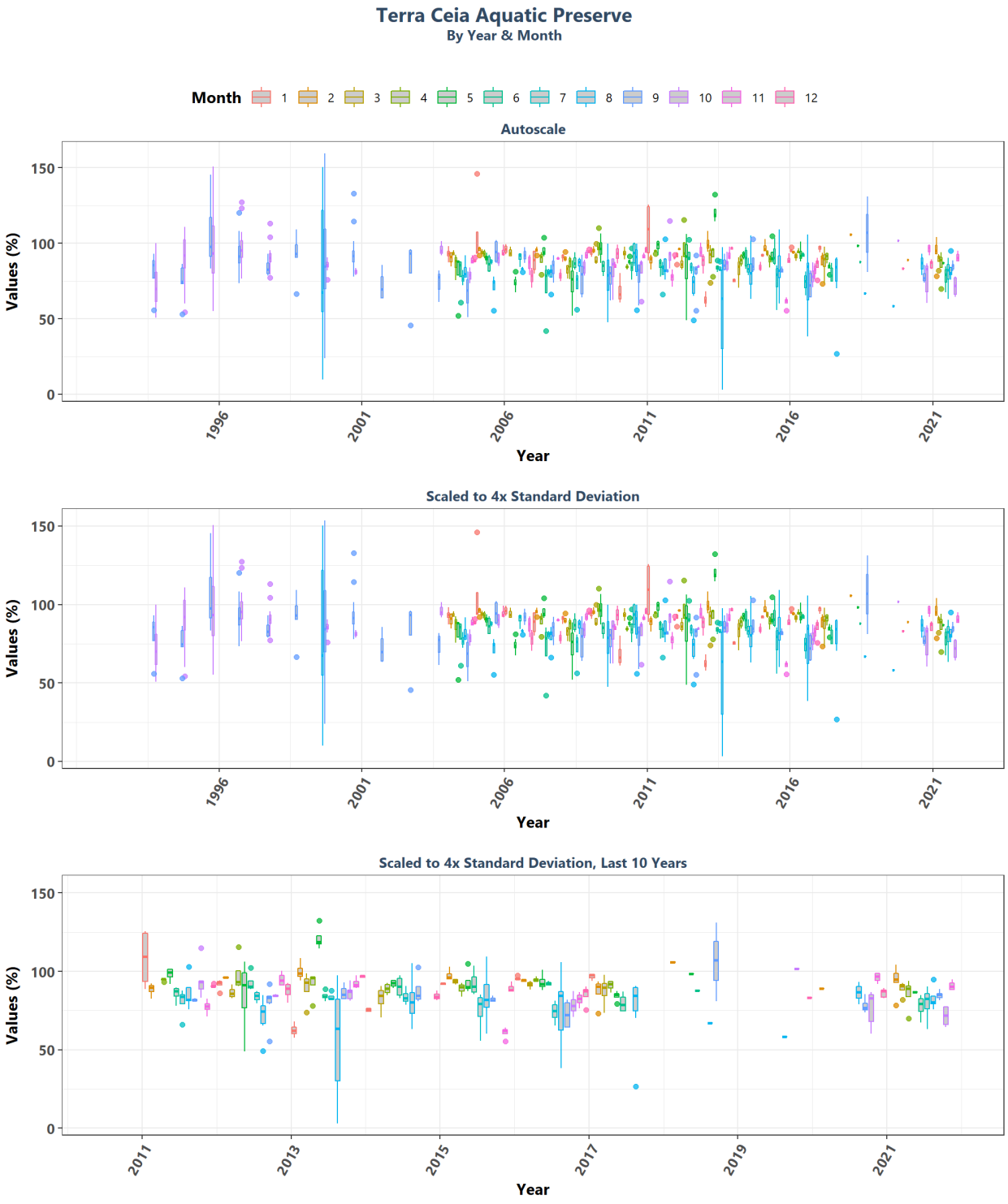

Pinellas County Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Month
Scaled to 4x Standard Deviation
Values (%)
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Month

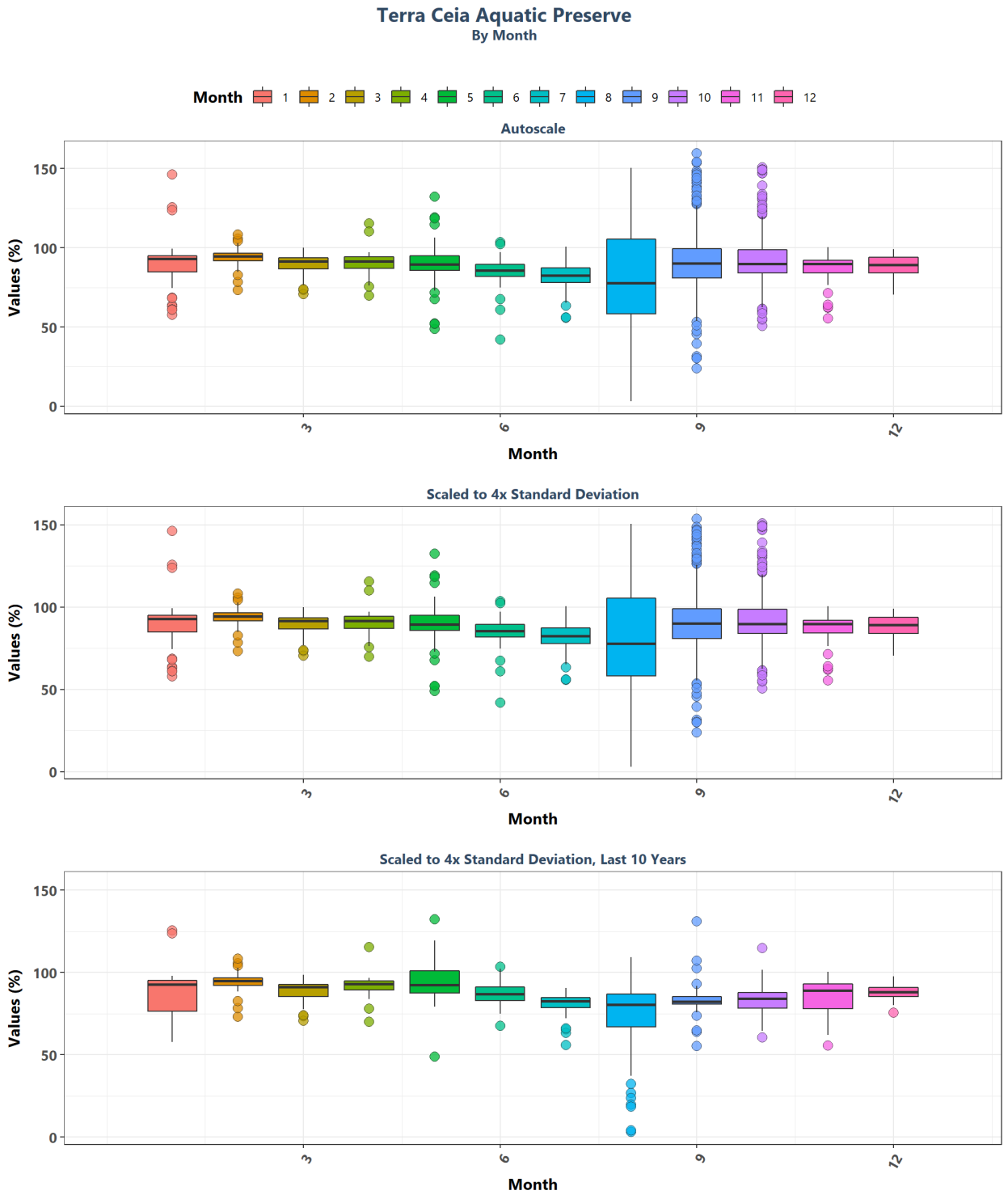

Rocky Bayou State Park Aquatic Preserve
By Year
Autoscale
Values (%)
125
100
75
50
25
0
2003
2008
2013
2018
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
2003
2008
2013
2018
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
2008
2010
2012
2014
2016
2018
Year


Rocky Bayou State Park Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
125
100
75
50
25
0
2003
2008
2013
2018
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
2003
2008
2013
2018
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
2008
2010
2012
2014
2016
2018
Year


Rocky Bayou State Park Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
125
100
75
50
25
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
3
6
9
12
Month


Terra Ceia Aquatic Preserve
By Year
Autoscale
Values (%)
150
100
50
0
1996
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
1996
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
2011
2013
2015
2017
2019
2021
Year


Terra Ceia Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
150
100
50
0
1996
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation
Values (%)
150
100
50
0
1996
2001
2006
2011
2016
2021
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
150
100
50
0
2011
2013
2015
2017
2019
2021
Year




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