SEACAR Discrete Water Quality Analysis: Field Bottom Turbidity

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: 9547, Number Passed Filter: 9547
## 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),]
```

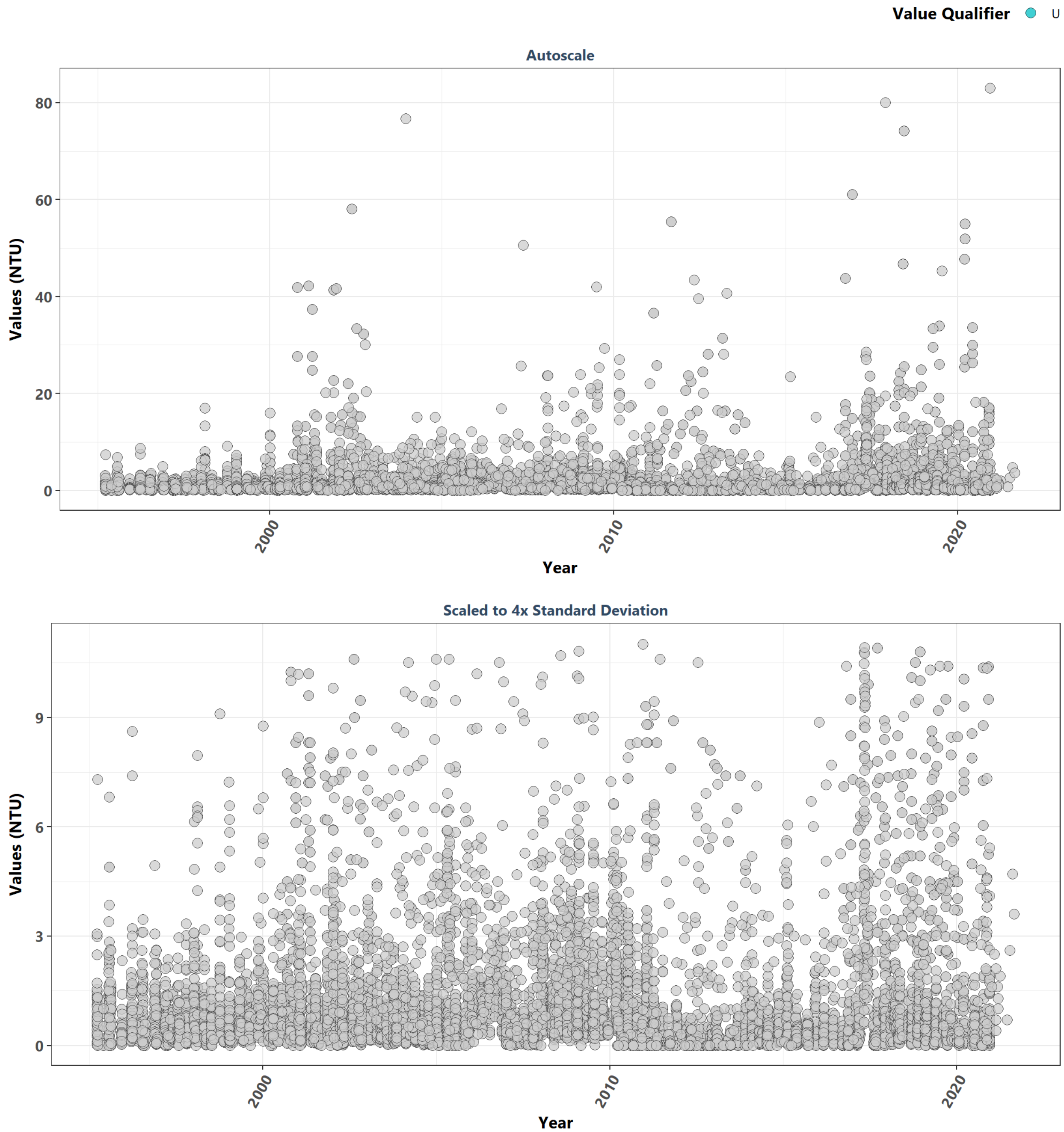
Appendix I: Scatter Plot of Entire Dataset

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))
```

Scatter Plot for Entire Dataset



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.

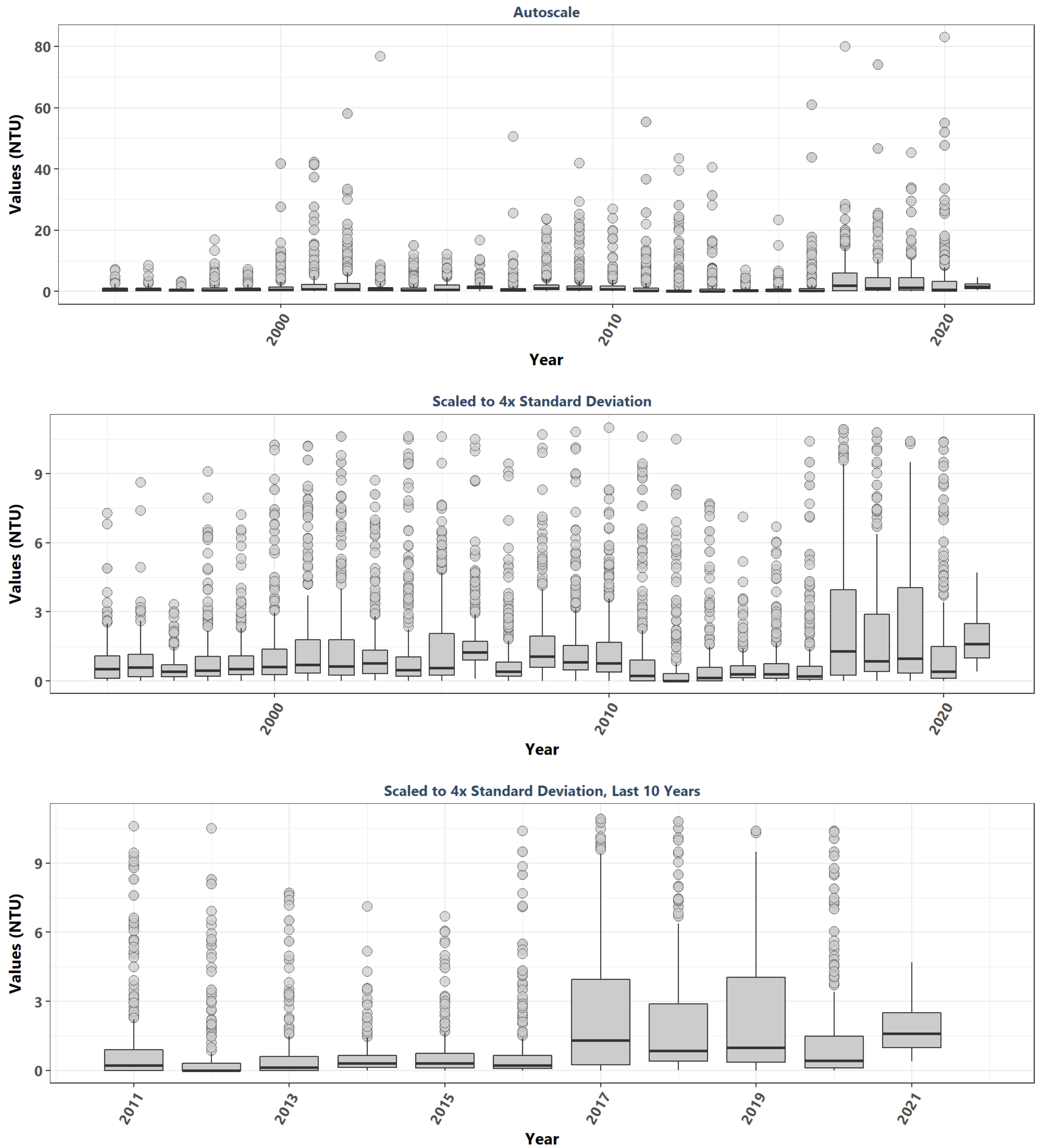
```
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=YearMonthDec, y=ResultValue,
                 group=YearMonth, color=as.factor(Month))) +
   geom_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +
   labs(subtitle="Autoscale", x="Year",
        y=paste0("Values (", unit,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

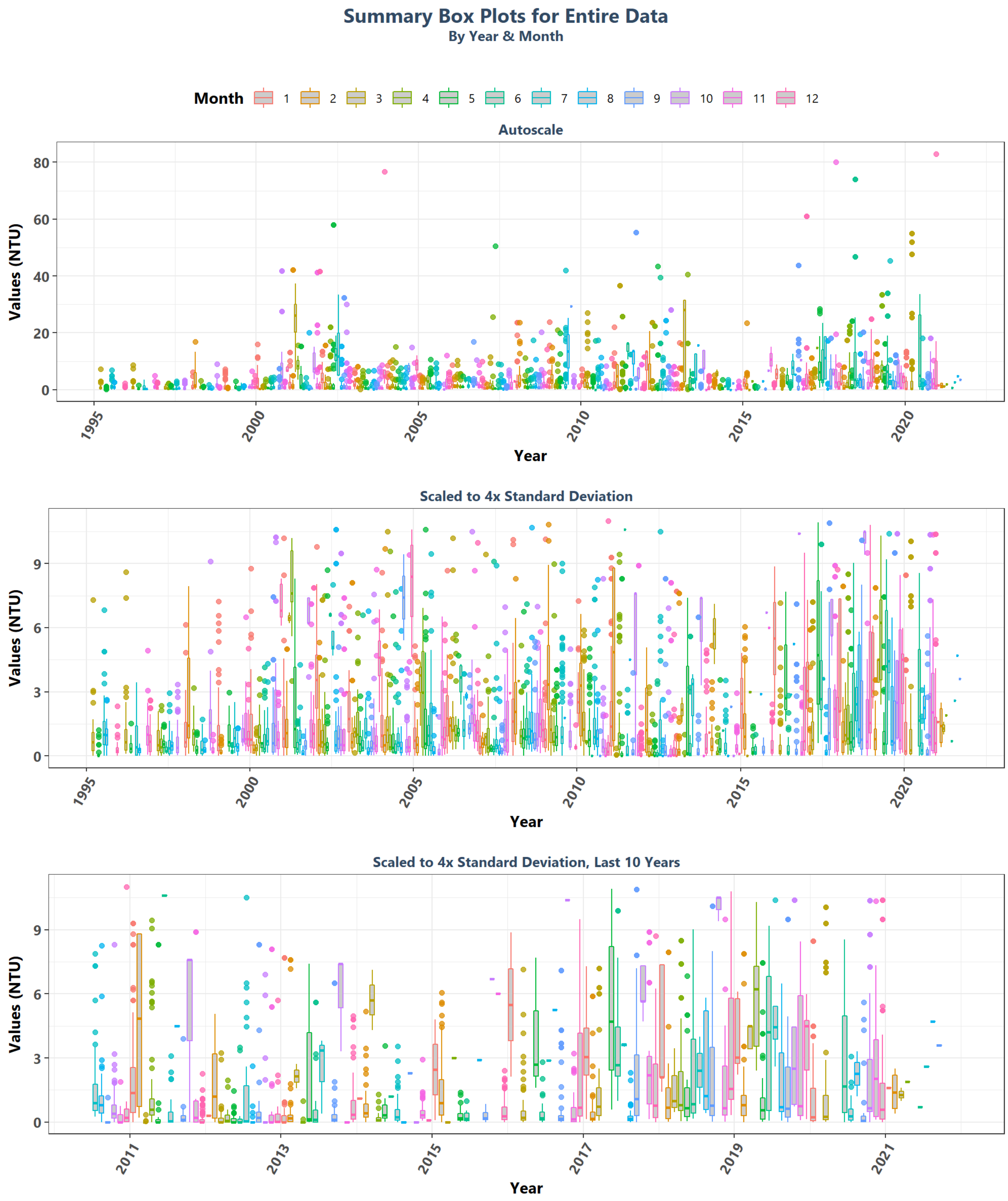
The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

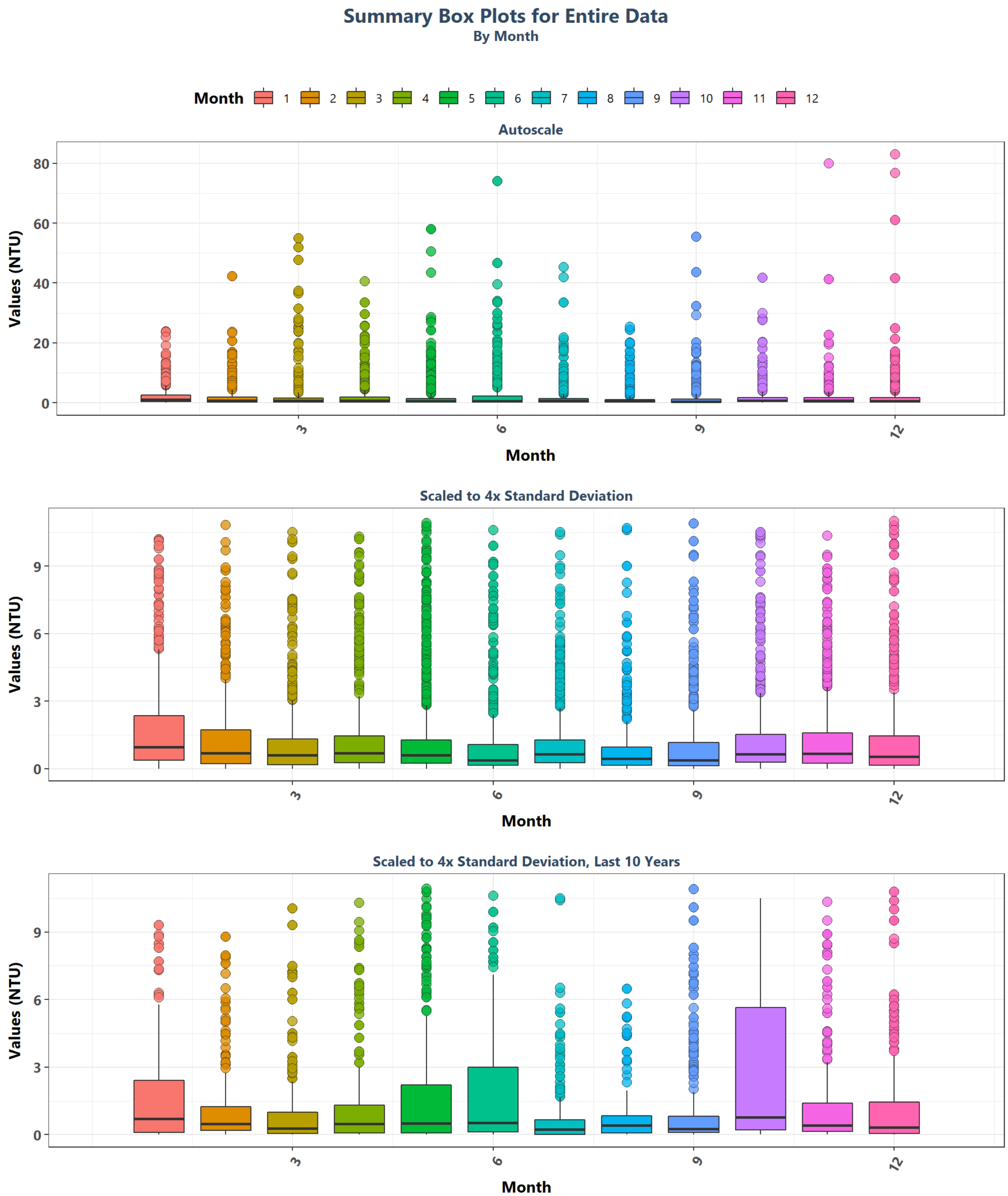
```r
p1 <- ggplot(data=data[data$Include==TRUE, ],
             aes(x=Month, y=ResultValue,
                 group=Month, fill=as.factor(Month))) +
   geom_boxplot(color="#333333", outlier.shape=21, outlier.size=3,
                outlier.color="#333333", outlier.alpha=0.75) +
   labs(subtitle="Autoscale",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



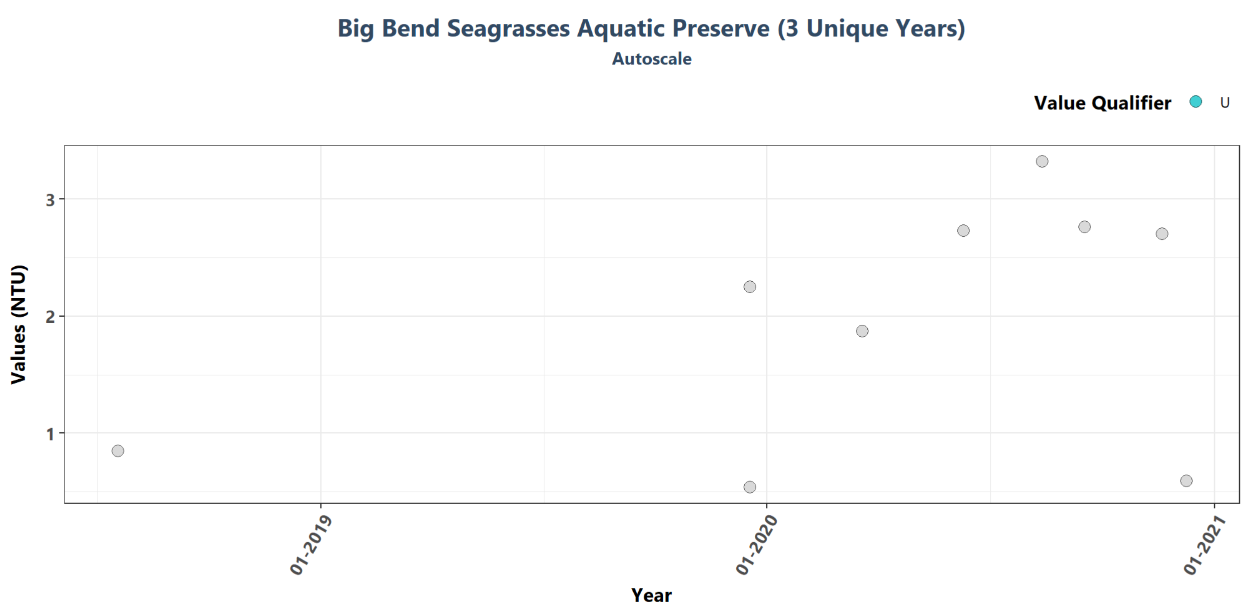

Summary Box Plots for Entire Data
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
Year
Scaled to 4x Standard Deviation
Values (NTU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
Year



Appendix III: Excluded Managed Areas

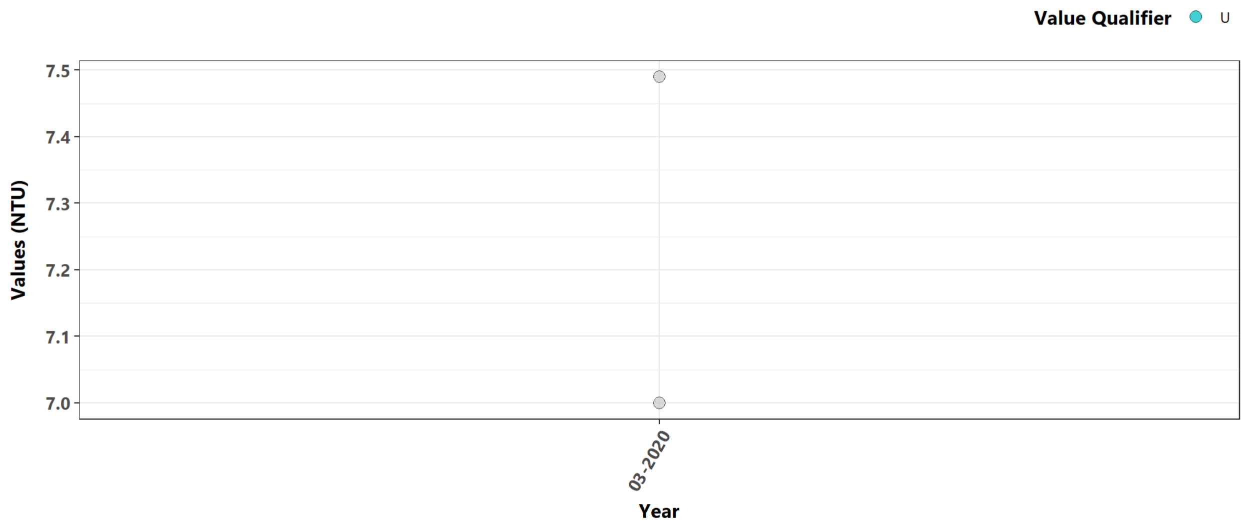
Scatter plots of data values are created for managed areas that have fewer than 10 separate years of data entries. Data points are colored based on specific value qualifiers of interest.

```
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)
   }
}
```



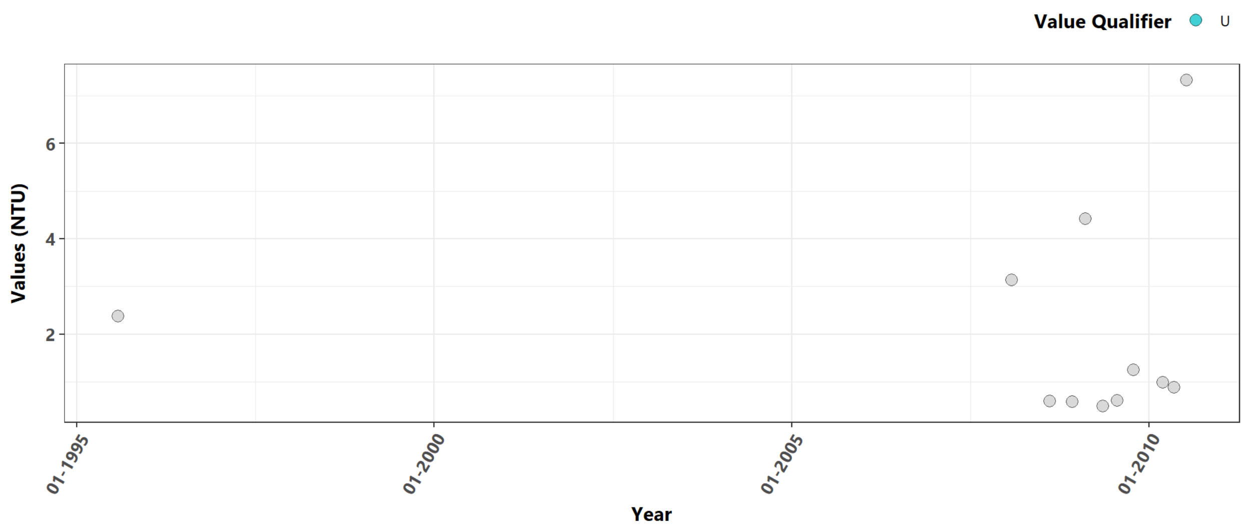
Boca Ciega Bay Aquatic Preserve (1 Unique Years)

Autoscale



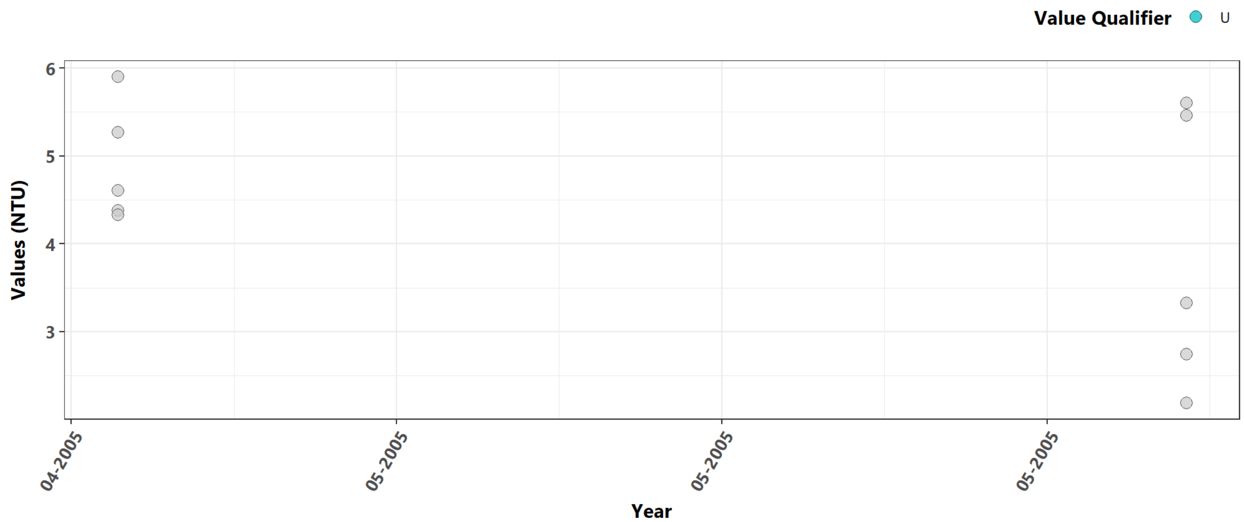
Coupon Bight Aquatic Preserve (4 Unique Years)

Autoscale



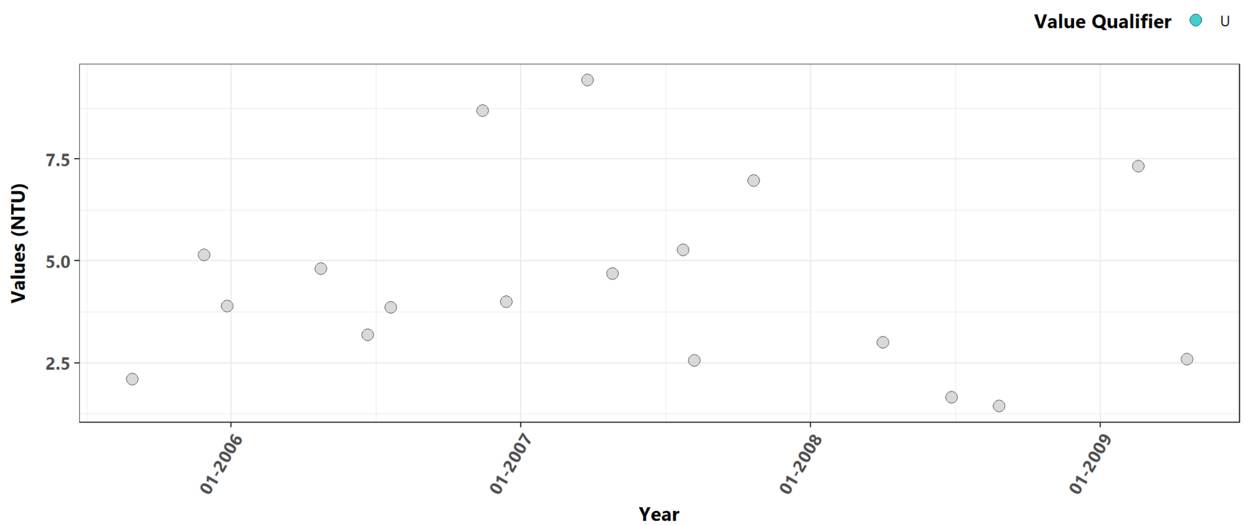
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve (1 Unique Years)

Autoscale



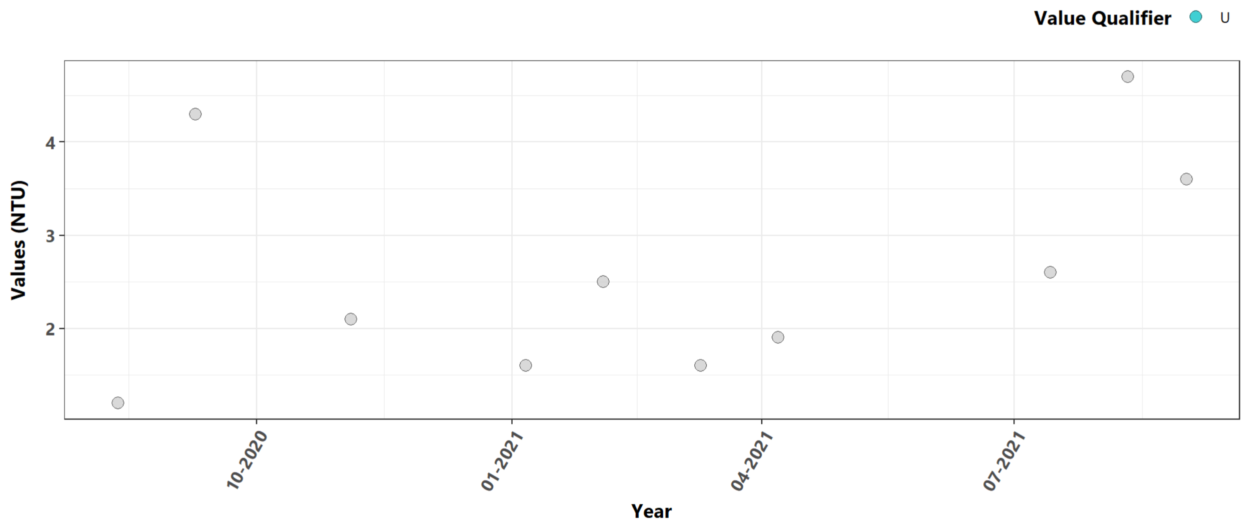
Jensen Beach to Jupiter Inlet Aquatic Preserve (5 Unique Years)

Autoscale



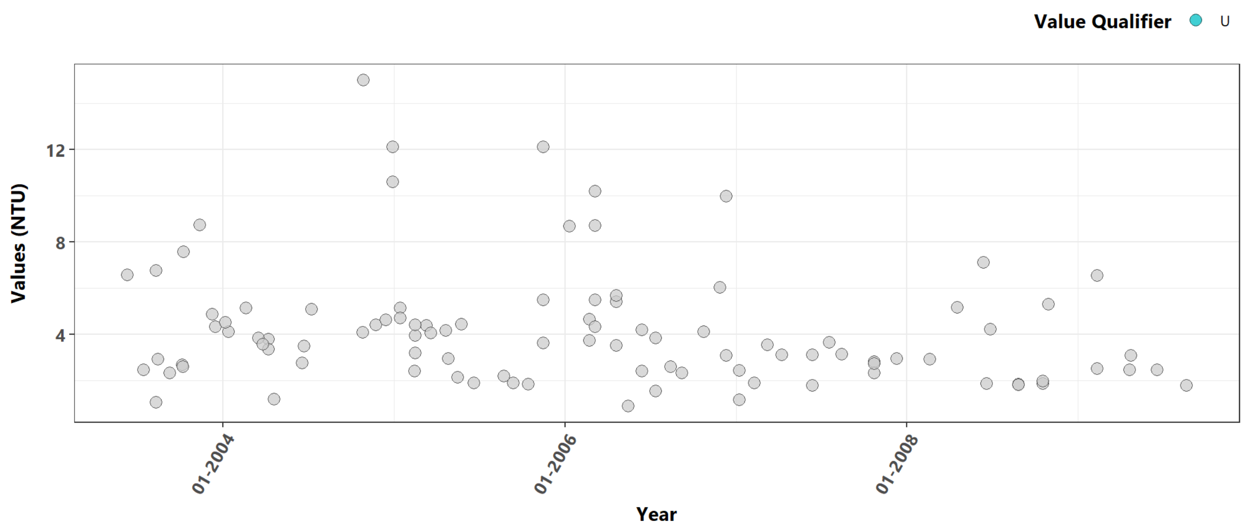
Lemon Bay Aquatic Preserve (2 Unique Years)

Autoscale



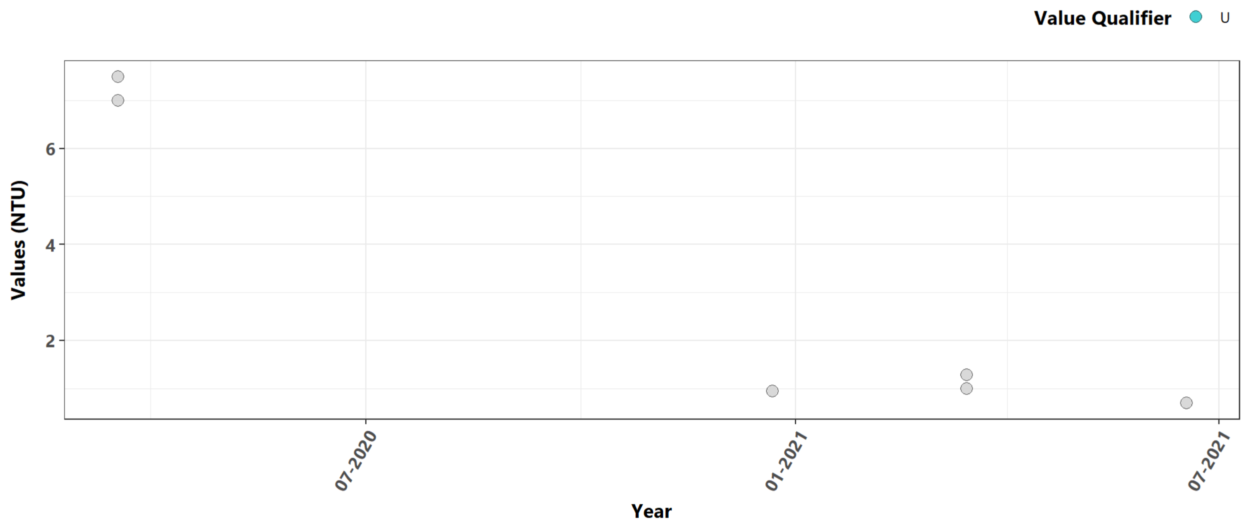
Loxahatchee River-Lake Worth Creek Aquatic Preserve (7 Unique Years)

Autoscale



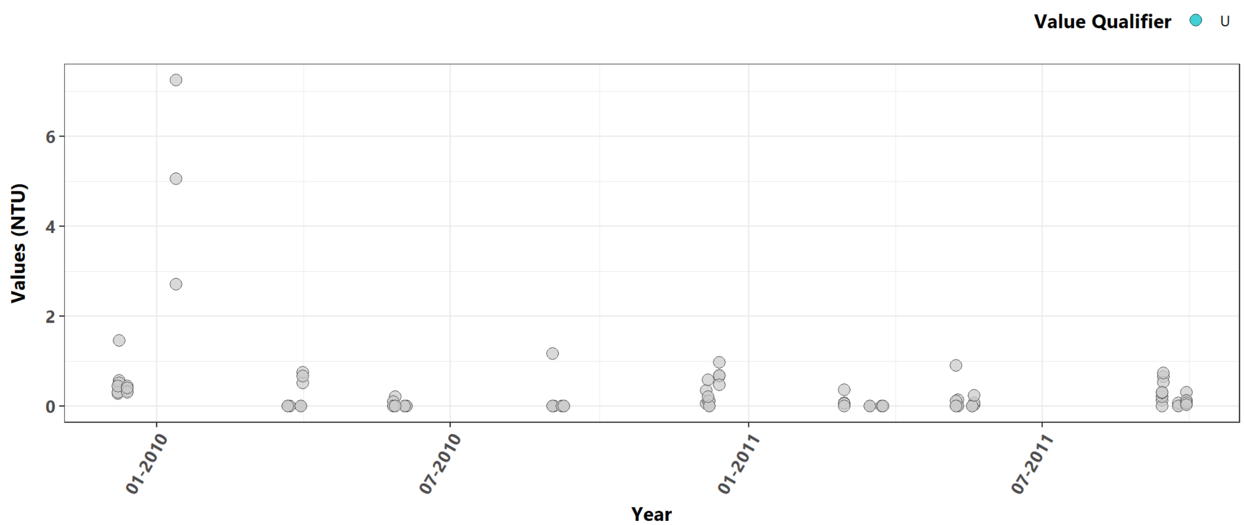
Pinellas County Aquatic Preserve (2 Unique Years)

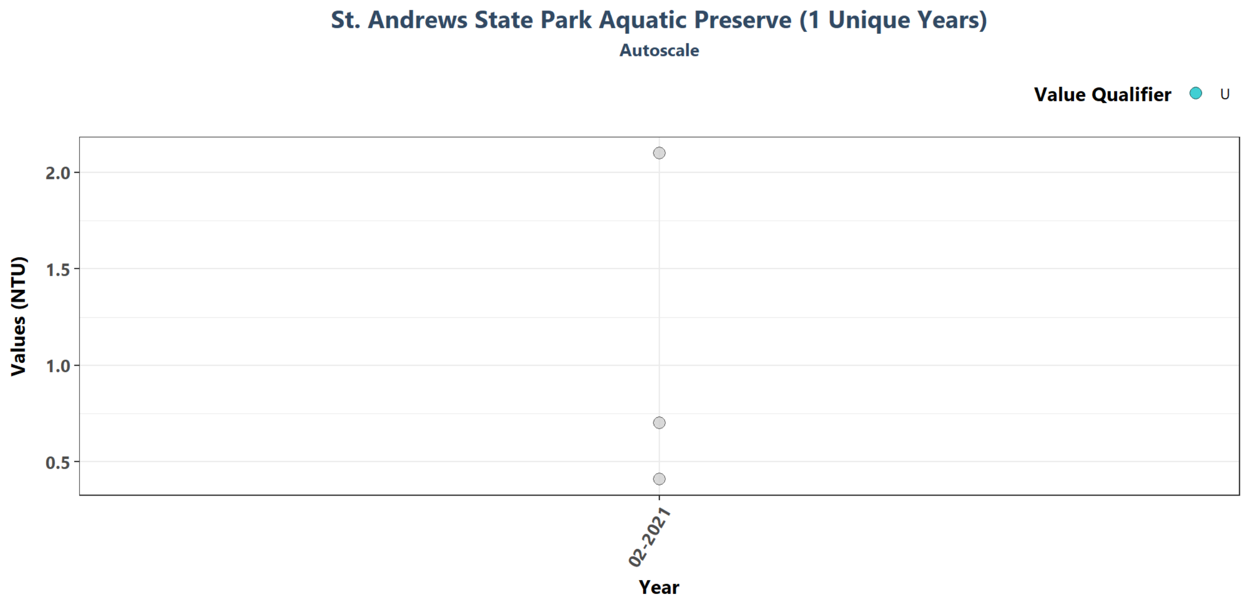
Autoscale



Southeast Florida Coral Reef Ecosystem Conservation Area (3 Unique Years)

Autoscale





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

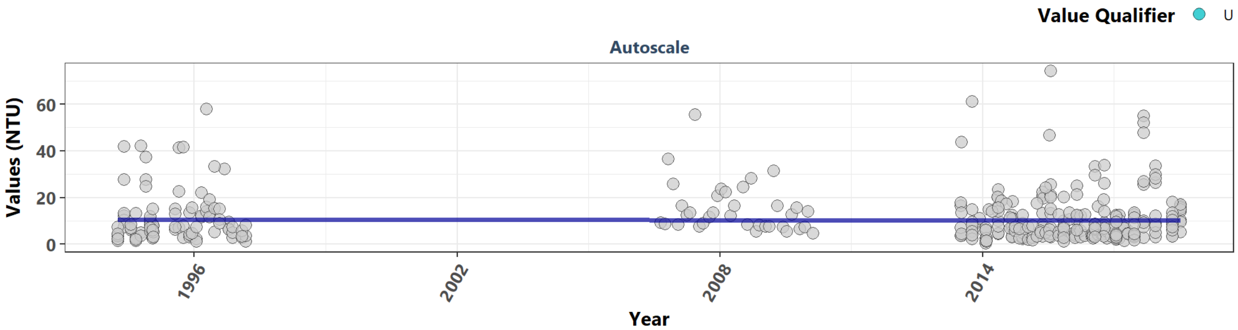
```
if(n==0){
   print("There are no managed areas that qualify.")
} else {
   for (i in 1:n) {
      plot_data <- data[data$SufficientData==TRUE &
                           data$ManagedAreaName==MA_Include[i],]
      plot_data$Season <- factor(plot_data$Month, levels = c("All", seq(1, 12)), ordered = 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 <
```

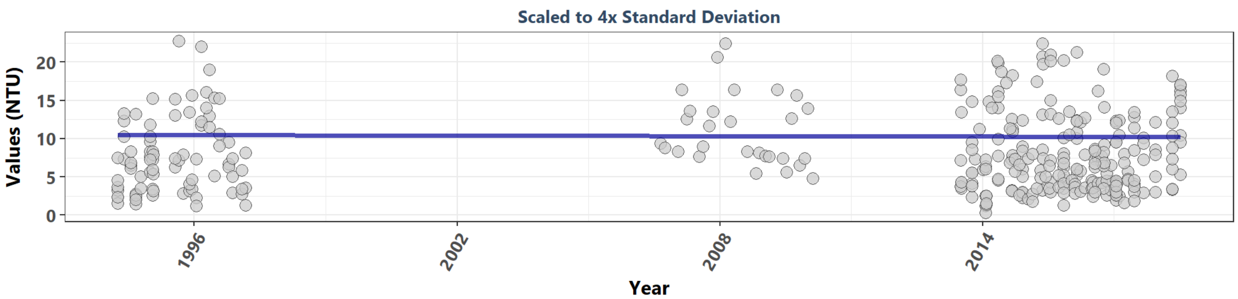
```
                                       quantile(data$ResultValue, 0.98)])
sd_RV <- sd(plot_data$ResultValue[plot_data$ResultValue <
                                     quantile(data$ResultValue, 0.98)])
x_scale <- ifelse(year_upper - year_lower > 30, 10, 5)
y_scale <- mn_RV + 4 * sd_RV

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$relyear_dd), 5, ceiling) - round_any(min(plot_data$rel

xlabs <- seq(max(plot_data$Year) - round_any(max(plot_data$relyear_dd), 5, ceiling),
             max(plot_data$Year),
             by = (max(plot_data$Year) - (max(plot_data$Year) - round_any(max(plot_data$relyear_d

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

# plot_data[is.na(VQ_Plot), VQ_Plot := "None"]
p1 <- ggplot(data=plot_data,
             aes(x=relyear_dd, y=ResultValue, fill = VQ_Plot)) +
   geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
   # geom_abline(aes(slope=s_slope, intercept=s_int),
   #             color="#000099", size=1.2, alpha=0.7) +
   geom_segment(data = KT.Stats[ManagedAreaName==MA_Include[i] & Season == "All", ], aes(x = rely
                                                                                            y = relyear_
                                                                                            xend = relyea
                                                                                            yend = relyea
                color="#000099", size=1.2, alpha=0.7, inherit.aes = FALSE) +
   labs(subtitle="Autoscale",
        x="Year", y=paste0("Values (", unit, ")"),
        fill="Value Qualifier") +
   plot_theme +
   theme(legend.position="top", legend.box="horizontal",
         legend.justification="right") +
   {if(inc_H==TRUE){
      scale_fill_manual(values=c("H"= "#F8766D", "U"= "#00BFC4",
                                 "HU"="#7CAE00"), na.value="#cccccc")
```

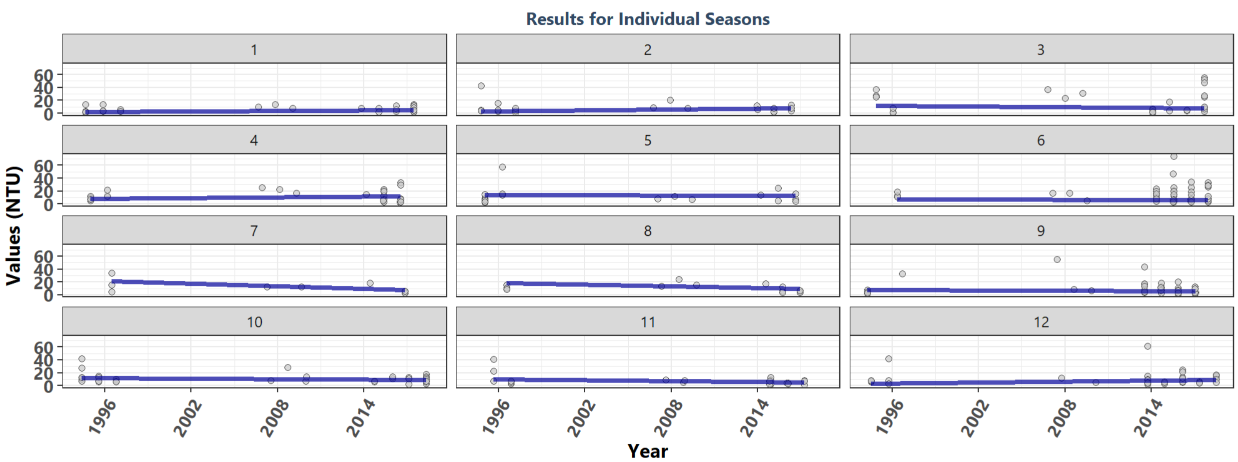
```
    } else if(param_name=="Secchi_Depth"){
       scale_fill_manual(values=c("S"= "#F8766D", "U"= "#00BFC4",
                                  "SU"="#7CAE00"), na.value="#cccccc")
    } else {
       scale_fill_manual(values=c("U"= "#00BFC4"), na.value="#cccccc")
    }} +
    scale_x_continuous(breaks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

 p2 <- ggplot(data=plot_data,
              aes(x=relyear_dd, y=ResultValue, fill=VQ_Plot)) +
    geom_point(shape=21, size=3, color="#333333", alpha=0.75) +
    # geom_abline(aes(slope=s_slope, intercept=s_int),
    #             color="#000099", size=1.2, alpha=0.7) +
    geom_segment(data = KT.Stats[ManagedAreaName==MA_Include[i] & Season == "All", ], aes(x = rely
                                                                                              y = relyear_
                                                                                              xend = relye
                                                                                              yend = relye

                 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 = rely
                                                                                              y = relyear_
                                                                                              xend = relye
                                                                                              yend = relye

                 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



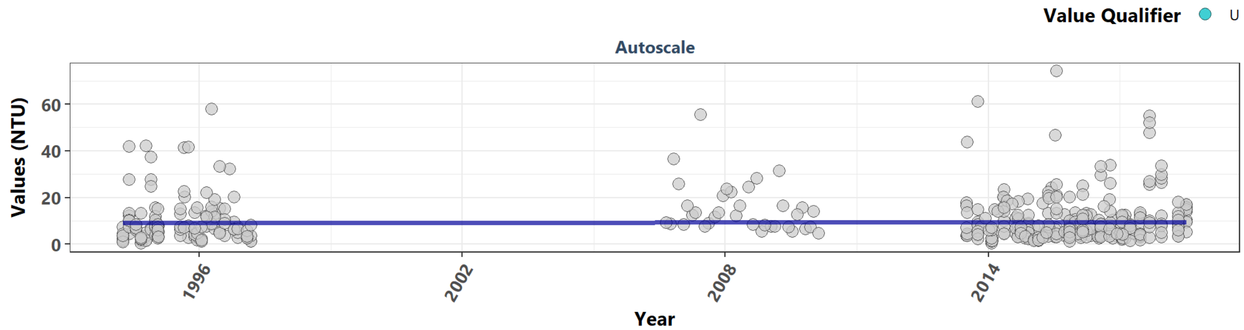


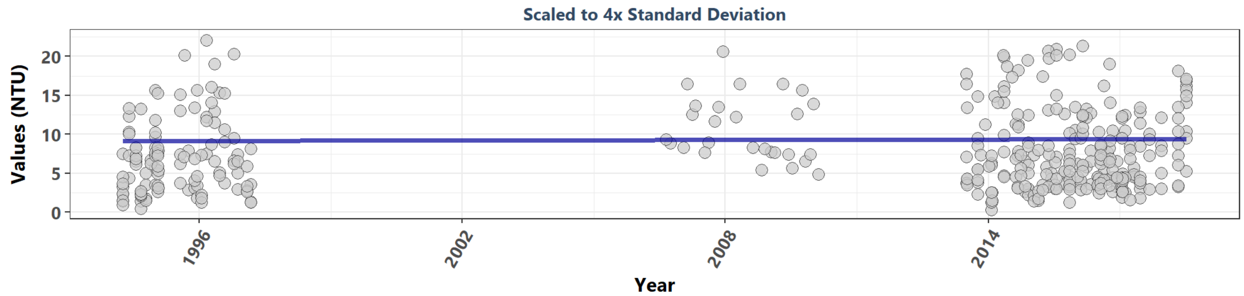


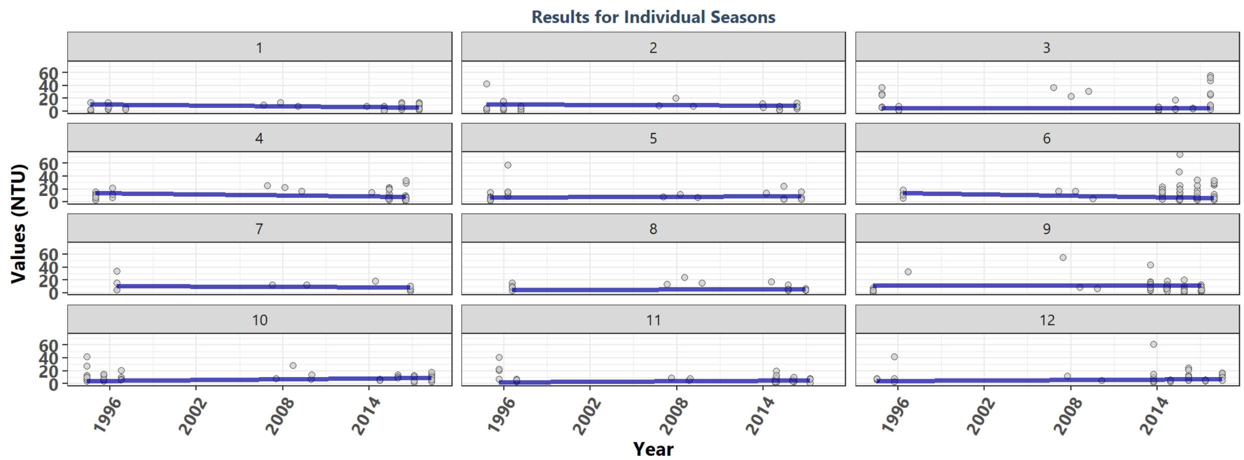
Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	363	7.60	-0.0224	-0.0112	10.6006	-0.3	0.7941	22.7	0.0197	0
1	31	4.02	0.2409	0.1446	0.9840	1.9	0.0538	NA	NA	0
2	20	6.75	0.1075	0.2800	0.2400	0.0	0.9739	NA	NA	0
3	31	7.24	-0.1580	-0.1944	13.5556	0.9	0.3913	NA	NA	0
4	35	8.50	0.2048	0.2417	5.4833	-1.4	0.1705	NA	NA	0
5	21	7.90	-0.0119	-0.0743	14.8314	1.3	0.1830	NA	NA	0
6	49	12.90	-0.0105	-0.0119	6.9816	-0.1	0.9086	NA	NA	0
7	9	12.50	-0.3889	-0.8154	29.6235	-1.4	0.1577	NA	NA	0
8	13	10.60	-0.3718	-0.5129	23.4235	-1.7	0.0811	NA	NA	0
9	55	6.20	-0.0721	-0.1000	8.7000	-0.8	0.4326	NA	NA	0
10	33	10.24	-0.1742	-0.1511	13.4121	-1.4	0.1514	NA	NA	0
11	24	5.97	-0.2754	-0.2612	12.5012	-1.9	0.0558	NA	NA	0
12	42	8.09	0.1847	0.2758	1.0562	1.7	0.0813	NA	NA	0

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

Apalachicola National Estuarine Research Reserve



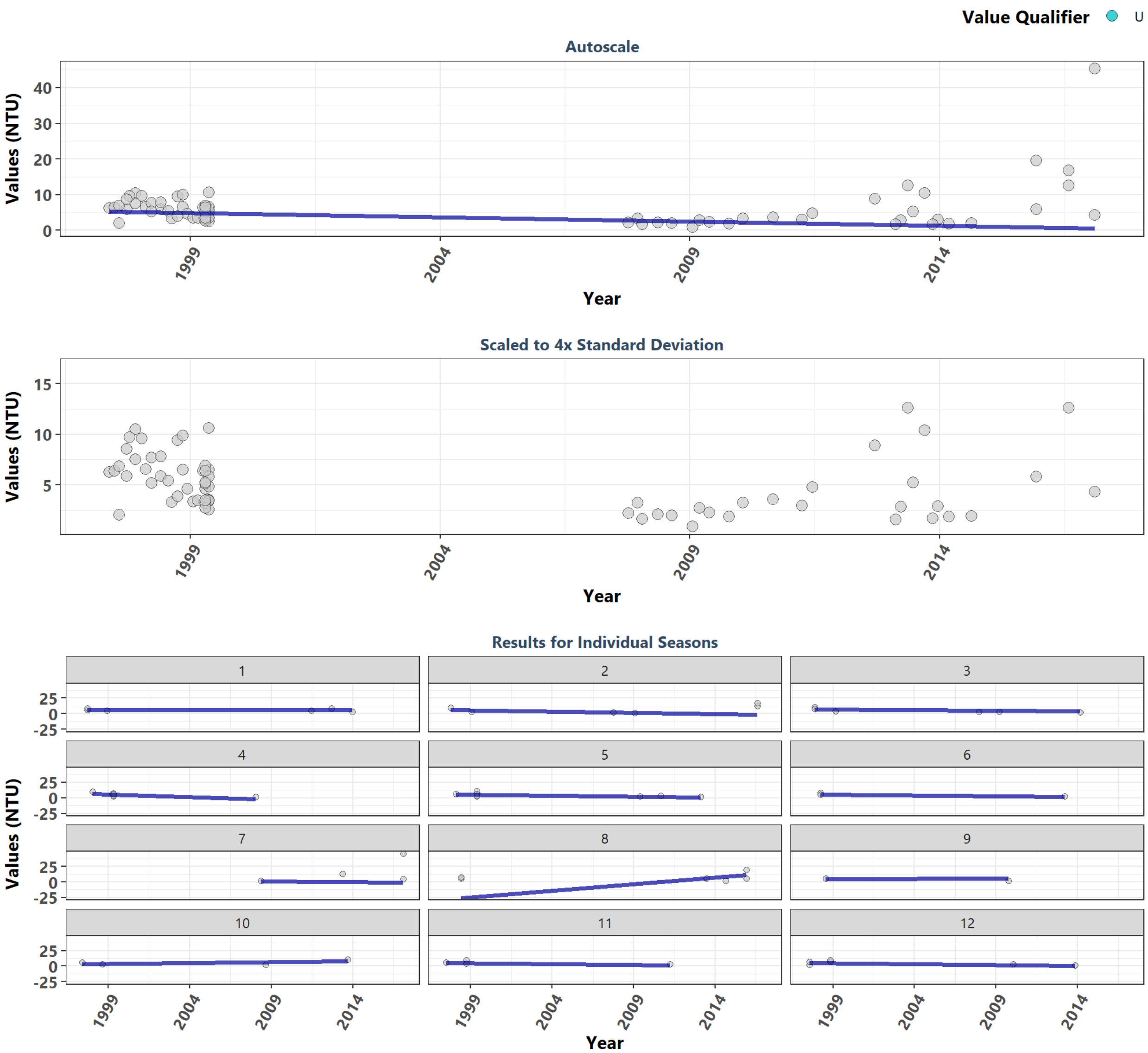




Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	418	7.30	0.0048	0.0125	9.0340	0.4	0.6961	20.1	0.0443	0
1	36	4.01	-0.2065	-0.2333	12.3733	2.2	0.0256	NA	NA	0
2	26	4.80	-0.1134	-0.1412	12.1706	0.8	0.4477	NA	NA	0
3	36	6.50	-0.0153	-0.0149	5.1413	1.1	0.2906	NA	NA	0
4	39	8.50	-0.2727	-0.3111	16.9333	-1.0	0.2984	NA	NA	0
5	26	7.47	0.1323	0.1139	6.3361	1.0	0.3315	NA	NA	0
6	52	11.79	-0.3167	-0.4331	18.7828	0.0	0.9871	NA	NA	0
7	11	10.40	-0.1295	-0.1094	11.7319	-1.2	0.2493	NA	NA	0
8	16	7.74	0.1077	0.0847	3.5292	-1.7	0.0867	NA	NA	0
9	60	4.77	0.0023	0.0012	11.7594	-0.2	0.8657	NA	NA	0
10	40	9.44	0.1545	0.2225	2.3175	-1.2	0.2375	NA	NA	0
11	31	6.54	0.2571	0.1840	0.2380	-1.7	0.0971	NA	NA	0
12	45	7.88	0.1222	0.1618	2.4539	1.5	0.1309	NA	NA	0

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

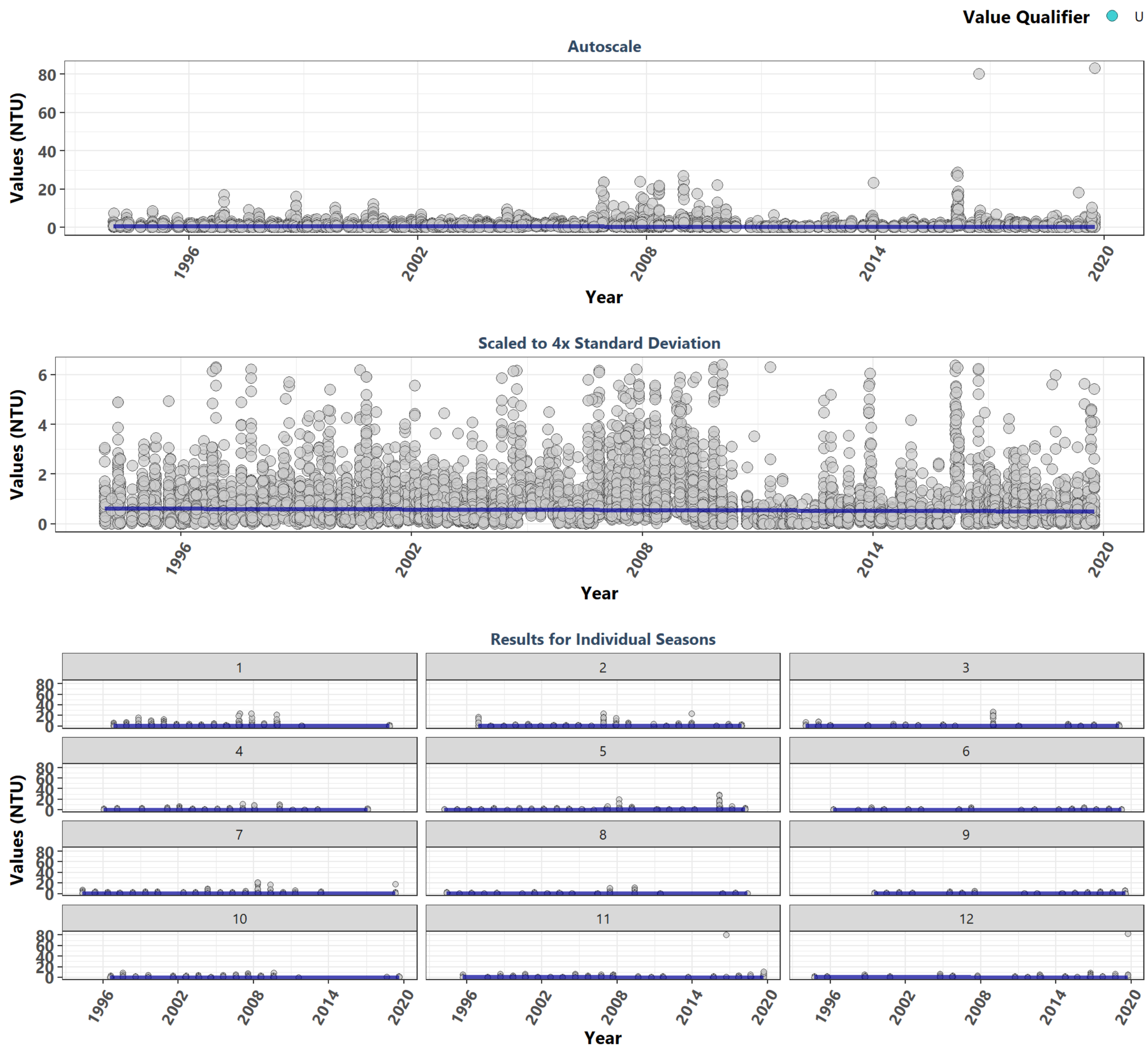
Banana River Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	70	4.98	-0.3047	-0.3041	7.6178	-3.5	0.0004	13.5	0.261	-1
1	6	5.34	-0.0667	-0.0029	5.8986	-0.6	0.5661	NA	NA	-1
2	6	6.54	-0.9333	-0.5172	9.8303	0.2	0.8483	NA	NA	-1
3	6	3.37	-0.2667	-0.1830	7.9076	-2.5	0.0129	NA	NA	-1
4	11	5.12	-0.3455	-0.9912	14.0413	-2.1	0.0351	NA	NA	-2
5	12	3.55	-1.0000	-0.3900	8.5200	-2.1	0.0394	NA	NA	-2
6	3	NA	-0.6667	-0.3017	7.6233	NA	NA	NA	NA	NA
7	4	8.46	-0.3939	-0.2953	6.2129	0.7	0.4701	NA	NA	-1
8	6	5.84	0.5000	2.6225	-47.9288	0.0	1.0000	NA	NA	2
9	2	NA	0.0000	0.0760	3.8835	NA	NA	NA	NA	NA
10	4	4.80	0.1333	0.3333	1.0400	0.0	1.0000	NA	NA	1
11	4	5.12	-0.5000	-0.3109	7.6123	-0.7	0.4701	NA	NA	-1
12	6	4.88	-0.3333	-0.3610	7.7630	-0.8	0.4357	NA	NA	-1

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

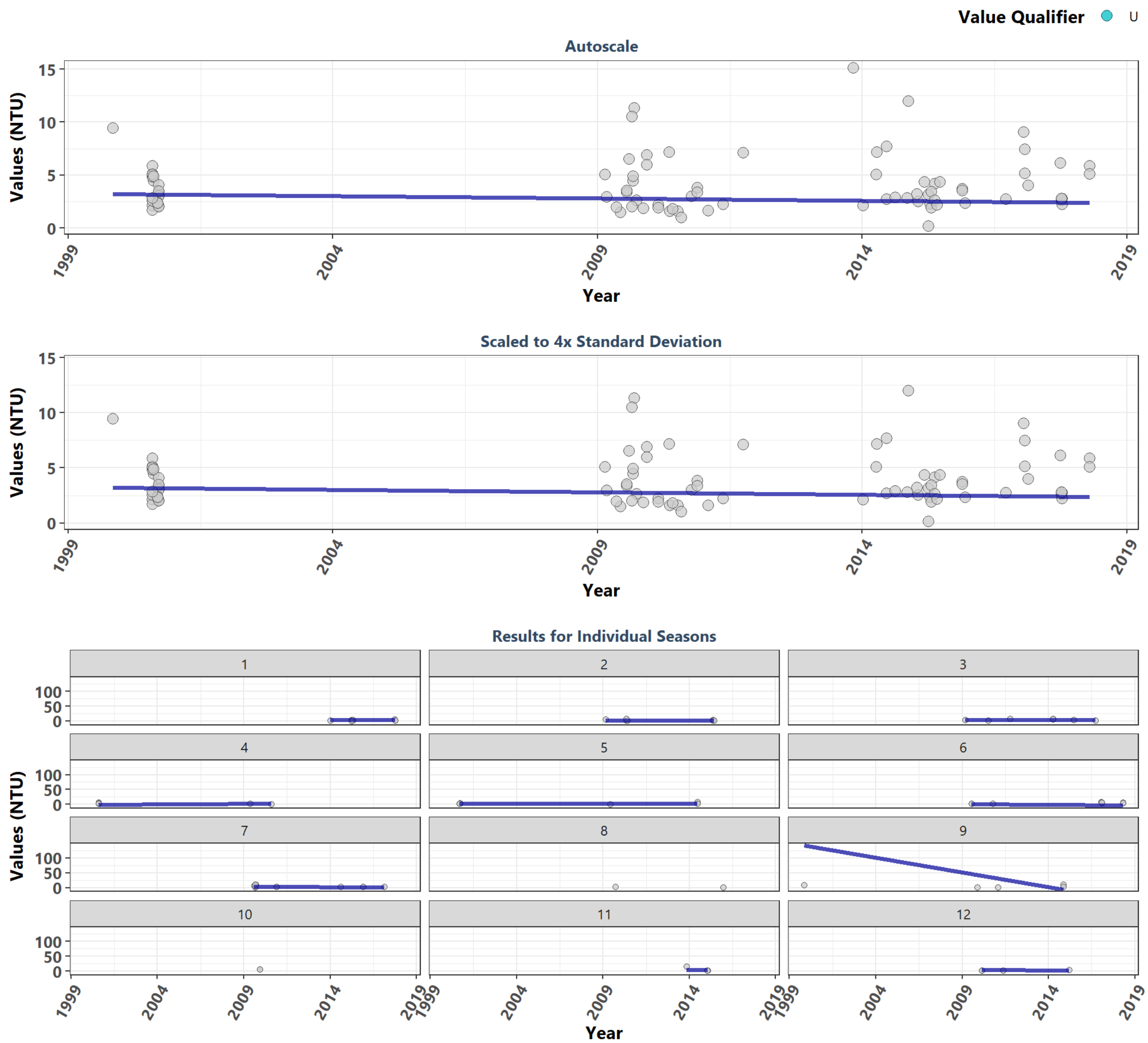
Florida Keys National Marine Sanctuary



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	8137	0.53	-0.0462	-0.0042	0.6350	-5.2	0.0000	160.2	0	-1
1	779	0.80	-0.0957	-0.0126	0.8259	-3.6	0.0003	NA	NA	-1
2	610	0.58	-0.0923	-0.0100	0.7050	-0.7	0.5133	NA	NA	-1
3	633	0.56	-0.1186	-0.0109	0.7623	-4.5	0.0000	NA	NA	-1
4	558	0.55	-0.0509	-0.0041	0.4635	-4.9	0.0000	NA	NA	-1
5	977	0.58	0.1521	0.0211	0.2220	7.1	0.0000	NA	NA	1
6	478	0.30	-0.0356	-0.0050	0.6350	2.3	0.0226	NA	NA	-1
7	1006	0.61	-0.0853	-0.0173	1.0723	-4.6	0.0000	NA	NA	-1
8	519	0.41	-0.0177	-0.0020	0.6155	-1.7	0.0819	NA	NA	-1
9	625	0.30	-0.0041	0.0000	0.4400	-7.2	0.0000	NA	NA	-1
10	679	0.56	0.0695	0.0050	0.2150	-1.4	0.1632	NA	NA	1
11	634	0.54	-0.1915	-0.0150	0.6350	-3.5	0.0005	NA	NA	-1
12	639	0.44	-0.1387	-0.0183	0.8384	-0.2	0.8766	NA	NA	-1

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

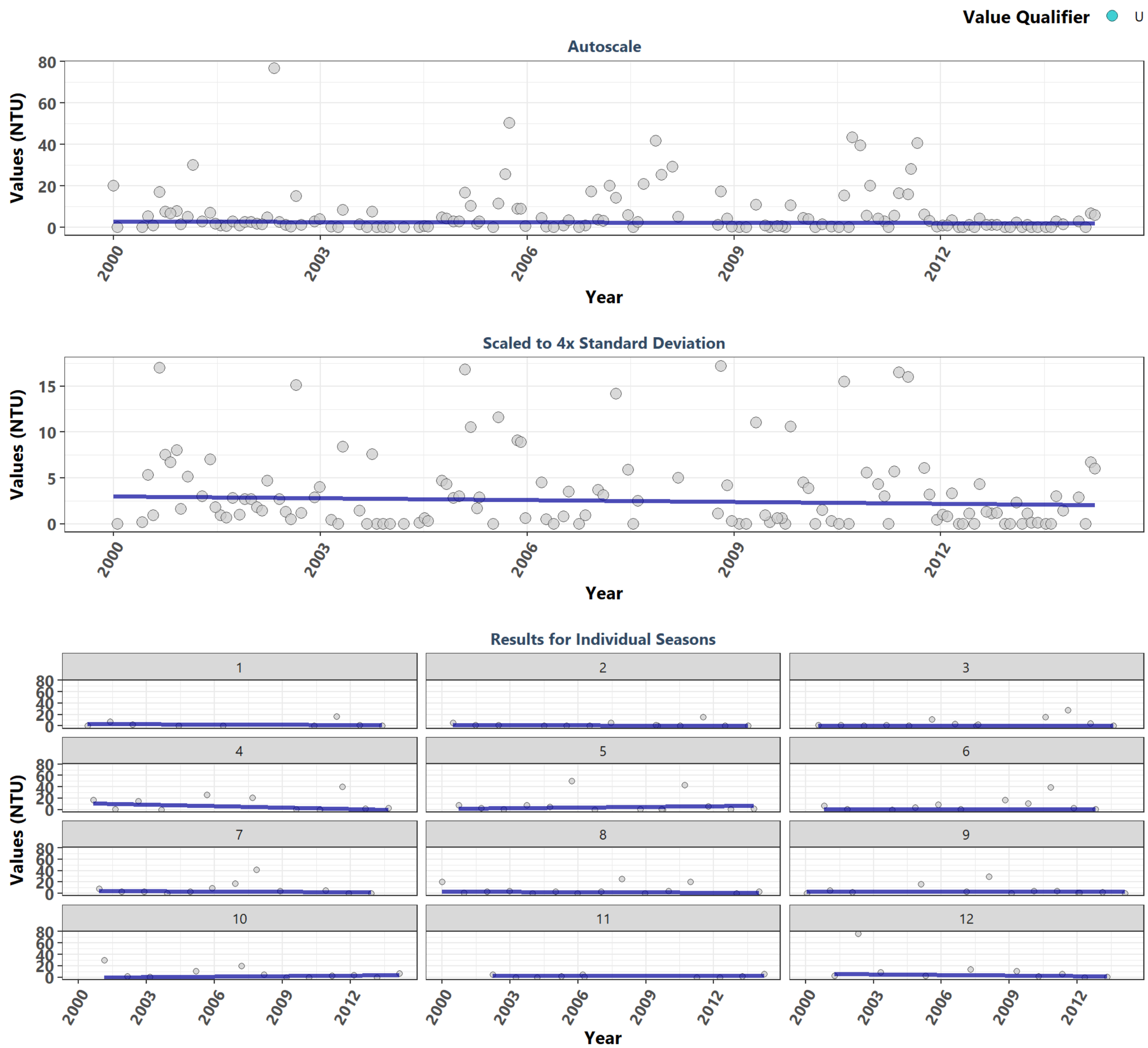
Indian River-Malabar to Vero Beach Aquatic Preserve



Season	N	Median	tau	Slope	Int.	z	p_z	chi_sq	p_chi_sq	Trend
All	86	3.23	-0.0266	-0.0550	3.7180	-0.5	0.6260	14.4	0.1557	0
1	11	2.58	0.0000	-0.0092	4.5325	1.0	0.2957	NA	NA	0
2	7	2.62	0.2364	0.2167	-2.1867	0.0	1.0000	NA	NA	0
3	7	4.34	0.0000	0.0014	2.9764	0.0	1.0000	NA	NA	0
4	12	3.64	0.8333	0.5350	-7.1675	-1.9	0.0545	NA	NA	0
5	13	2.99	-1.0000	-0.0640	3.7180	0.0	1.0000	NA	NA	0
6	8	5.11	-0.2879	-0.4186	7.8307	1.0	0.3004	NA	NA	0
7	12	3.91	-0.0476	-0.1820	5.8960	-1.3	0.1825	NA	NA	0
8	2	NA	NA	NA	NA	NA	NA	NA	NA	NA
9	5	2.81	-0.6667	-12.2400	260.2500	0.0	1.0000	NA	NA	0
10	2	NA	0.3214	0.3329	-2.5457	NA	NA	NA	NA	NA
11	3	NA	0.1000	0.2142	-1.0450	NA	NA	NA	NA	NA
12	4	2.20	-0.2879	-0.3133	9.3883	1.4	0.1486	NA	NA	0

[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	142	2.70	-0.0873	-0.0667	3.0045	-1.4	0.1767	8.8	0.6422	0
1	9	0.30	-0.1795	-0.1500	3.6500	-0.2	0.8339	NA	NA	0
2	13	0.80	-0.3462	-0.1250	1.8000	-1.6	0.1062	NA	NA	0
3	13	1.40	-0.0833	-0.0163	0.4144	0.9	0.3904	NA	NA	0
4	11	3.00	-0.4667	-0.8300	10.9900	0.0	1.0000	NA	NA	0
5	14	2.10	0.2000	0.3750	1.6750	-0.7	0.4747	NA	NA	0
6	11	4.30	0.0889	0.0000	0.8000	0.8	0.4363	NA	NA	0
7	12	3.55	-0.0455	-0.1400	4.4600	-0.1	0.8907	NA	NA	0
8	14	2.95	-0.1538	-0.1630	3.4039	-0.1	0.9563	NA	NA	0
9	13	2.30	-0.0220	-0.0091	3.0091	-0.8	0.4237	NA	NA	0
10	12	3.15	0.1923	0.2833	-0.5833	-1.0	0.3328	NA	NA	0
11	10	0.80	0.0000	0.0000	3.0000	0.3	0.7801	NA	NA	0
12	10	4.35	-0.2273	-0.3111	5.7944	-1.8	0.0736	NA	NA	0

[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

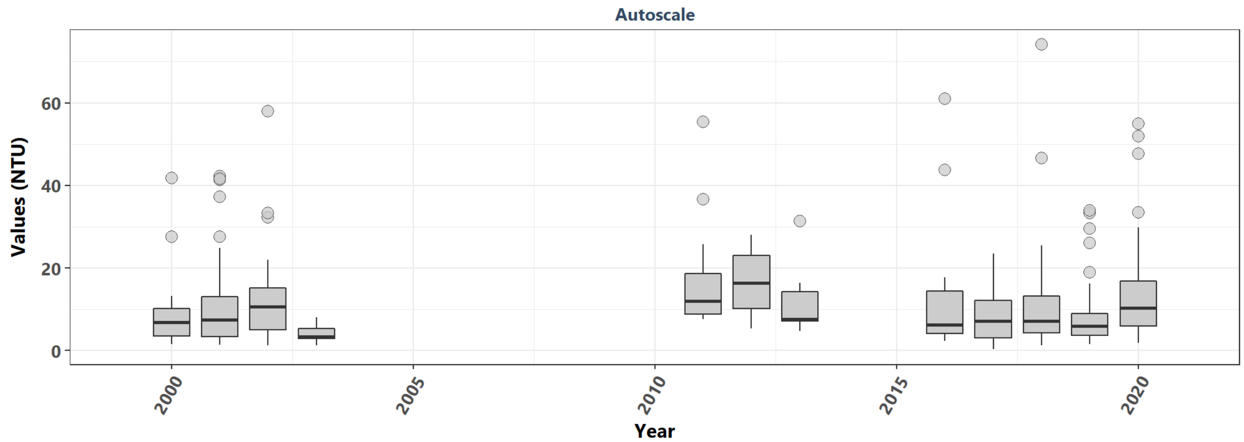
```
if(n==0){
   print("There are no managed areas that qualify.")
} else {
   for (i in 1:n) {
      plot_data <- data[data$SufficientData==TRUE &
                           data$ManagedAreaName==MA_Include[i],]
      year_lower <- min(plot_data$Year)
      year_upper <- max(plot_data$Year)
      min_RV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

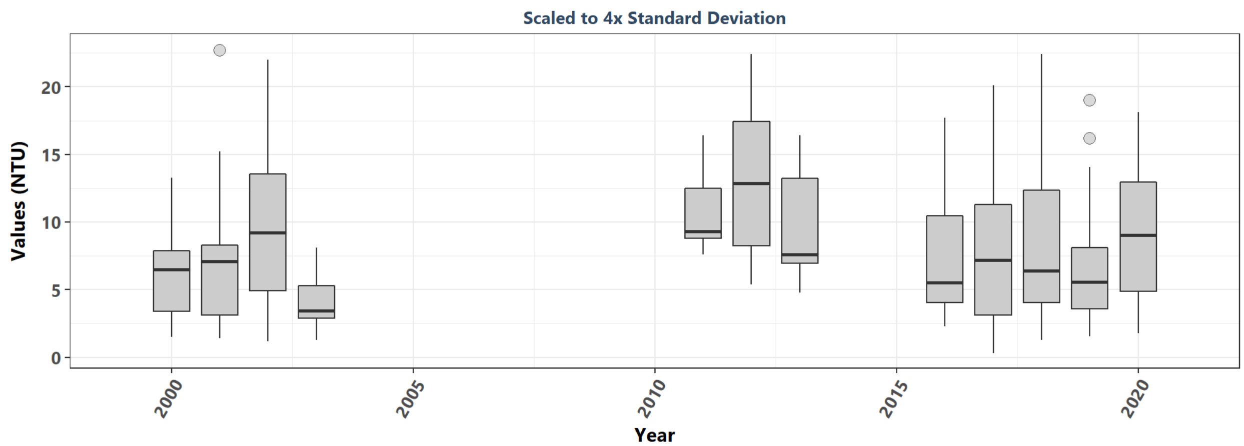
```r
p2 <- ggplot(data=plot_data,
             aes(x=Year, y=ResultValue, group=Year)) +
   geom_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,
                outlier.size=3, outlier.color="#333333",
                outlier.fill="#cccccc", outlier.alpha=0.75) +
   labs(subtitle="Scaled to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

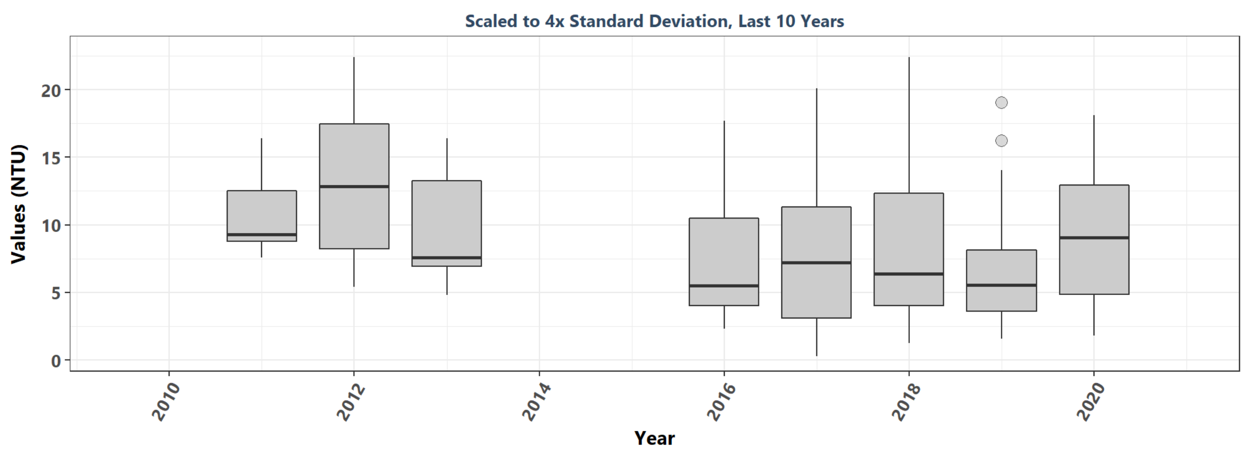
```
    ylim(min_RV, y_scale) +
    scale_x_continuous(limits=c(year_lower - 1, year_upper + 1),
                       breaks=rev(seq(year_upper,
                                      year_lower, -x_scale))) +
    plot_theme +
    theme(legend.position="top", legend.box="horizontal") +
    guides(color=guide_legend(nrow=1))

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") +
```

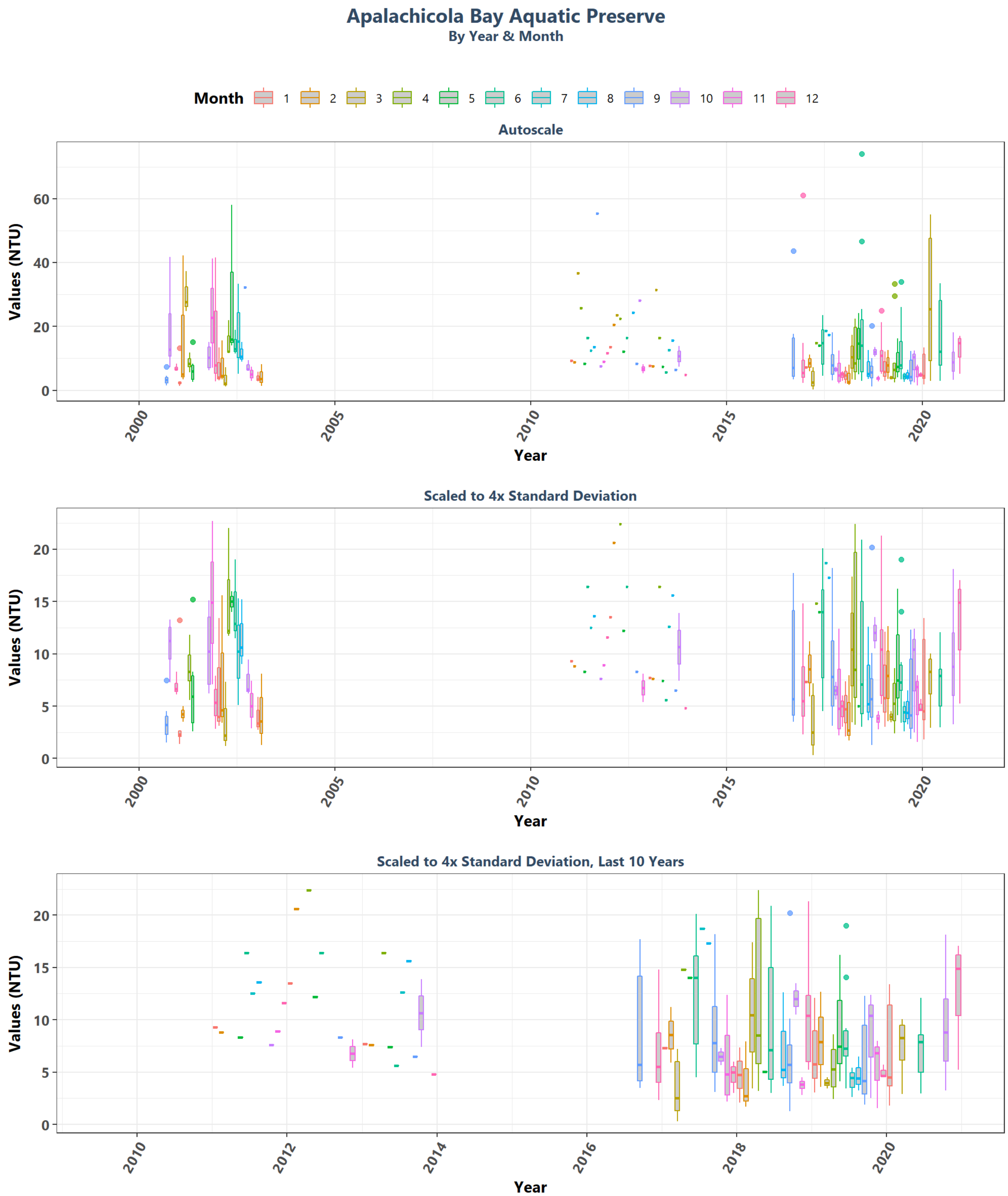
```
      guides(fill=guide_legend(nrow=1))

    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



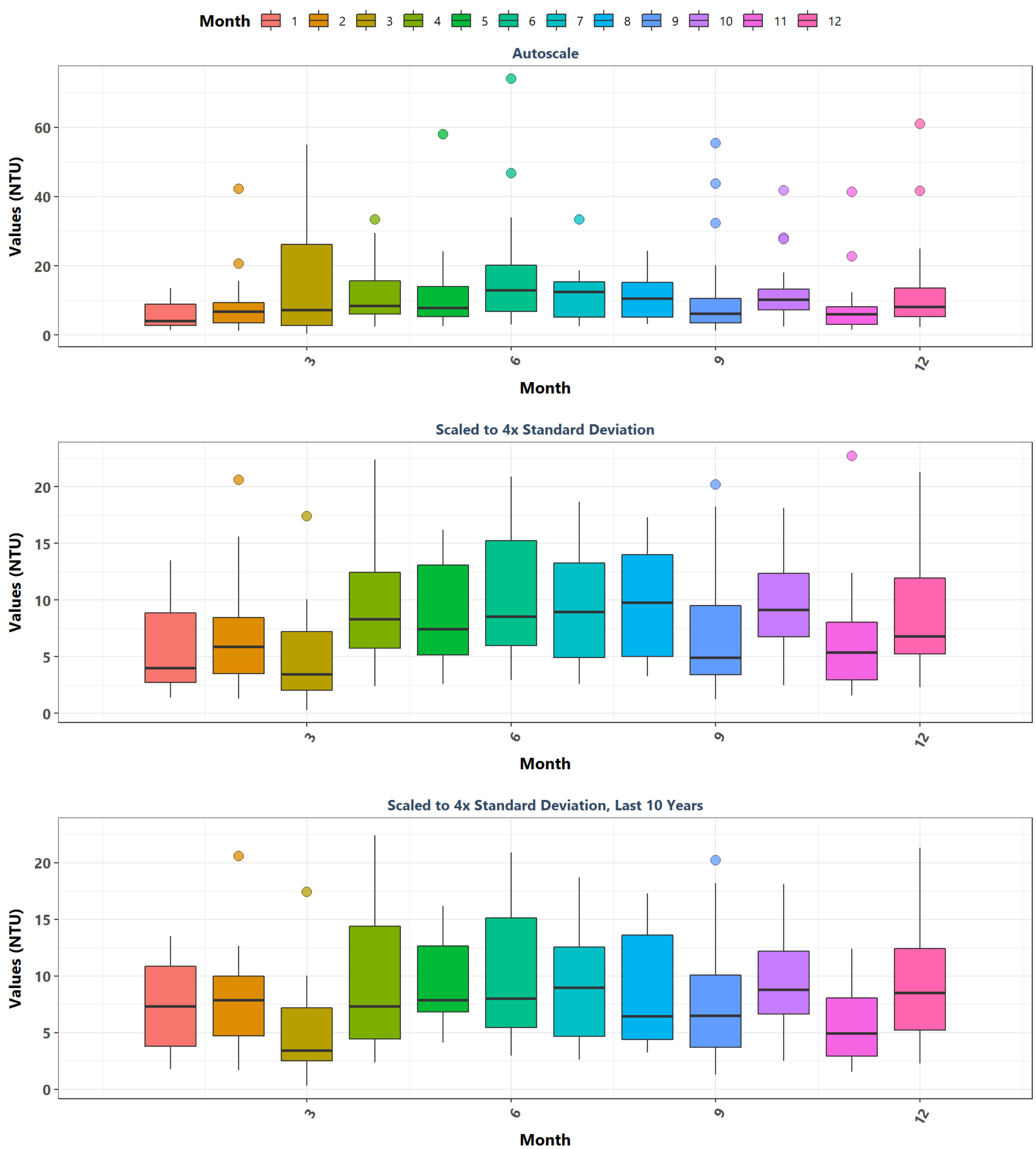






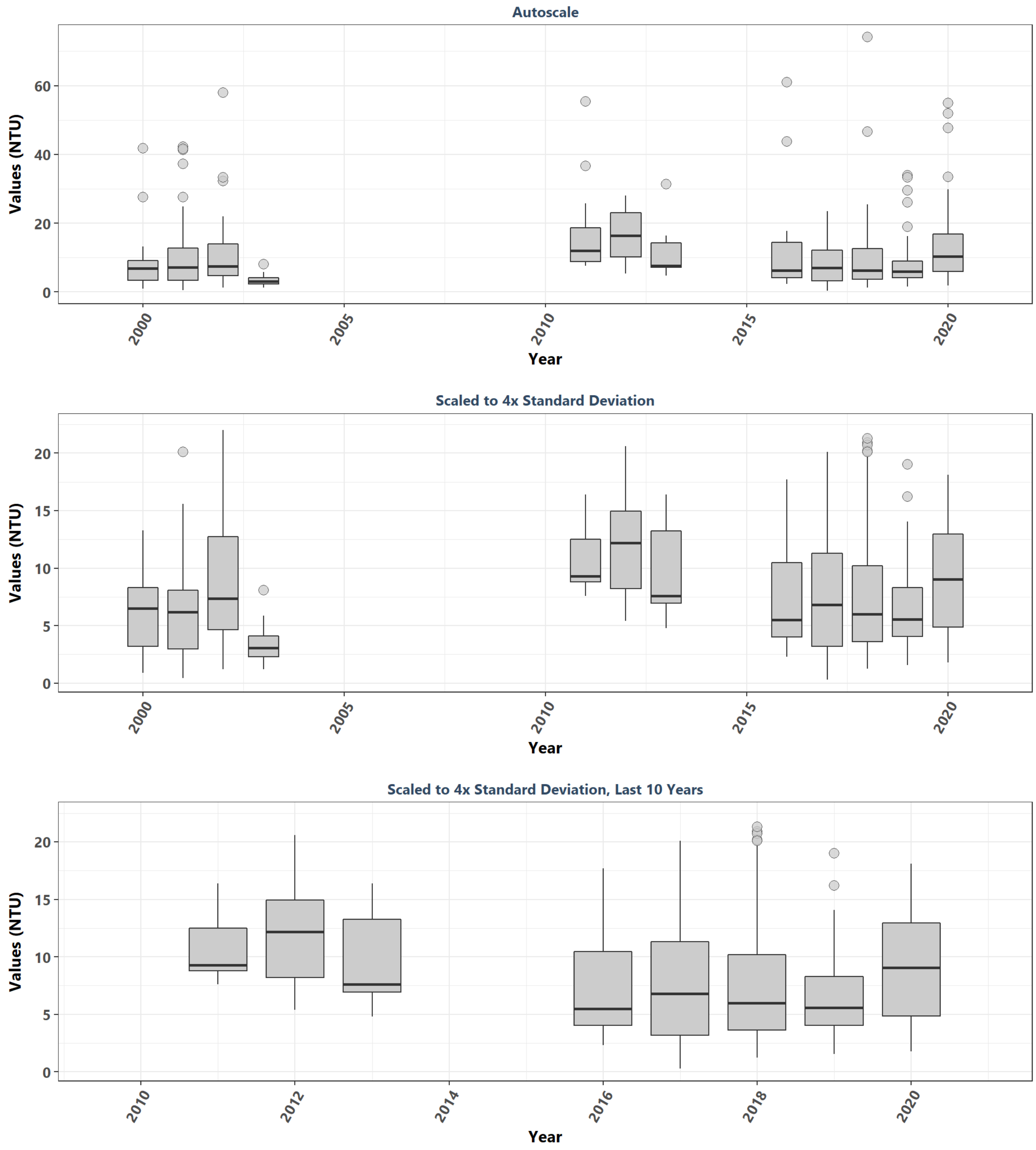
Apalachicola Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
Year
Scaled to 4x Standard Deviation
Values (NTU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
Year

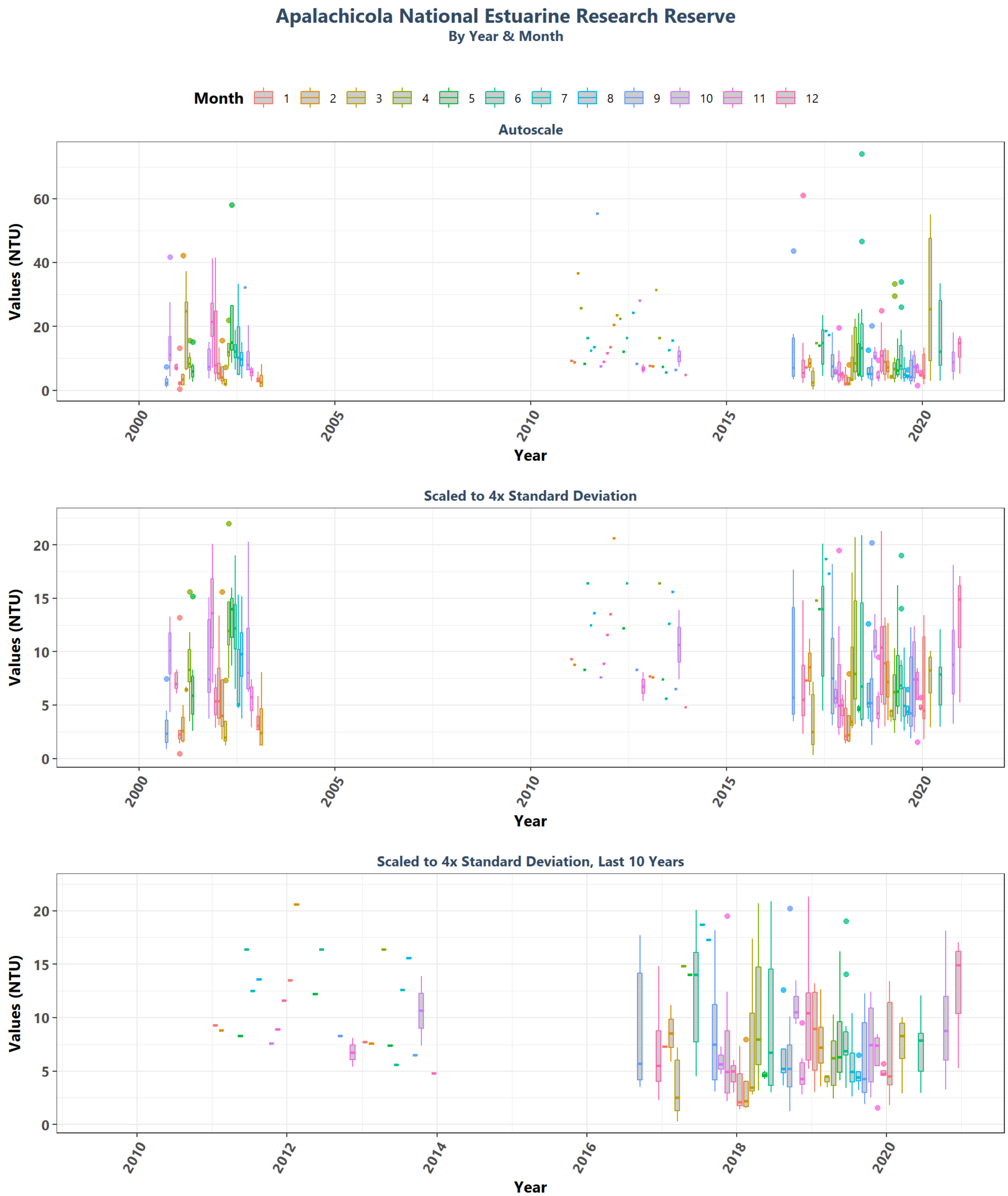
Apalachicola Bay Aquatic Preserve
By Month

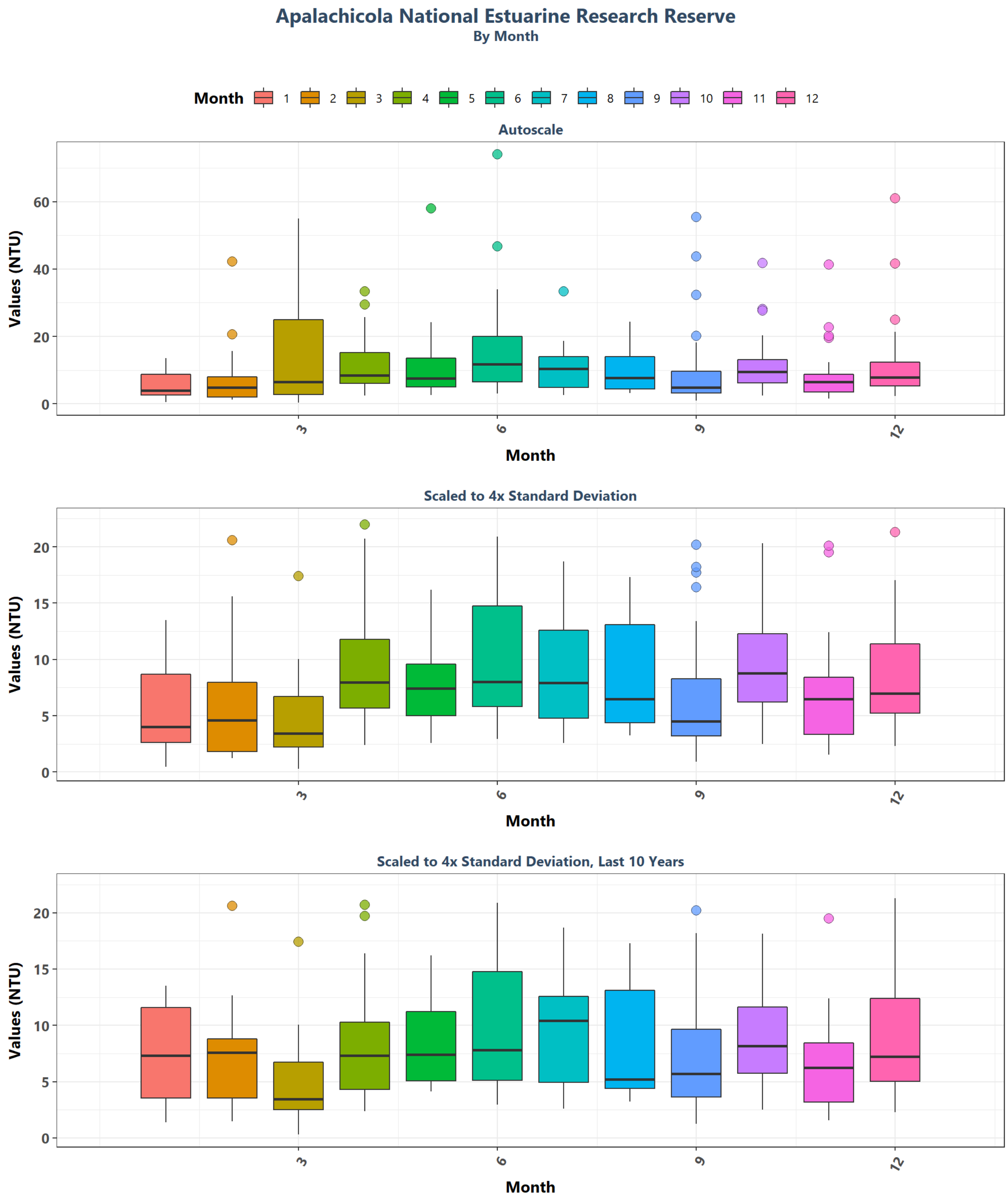


Apalachicola National Estuarine Research Reserve

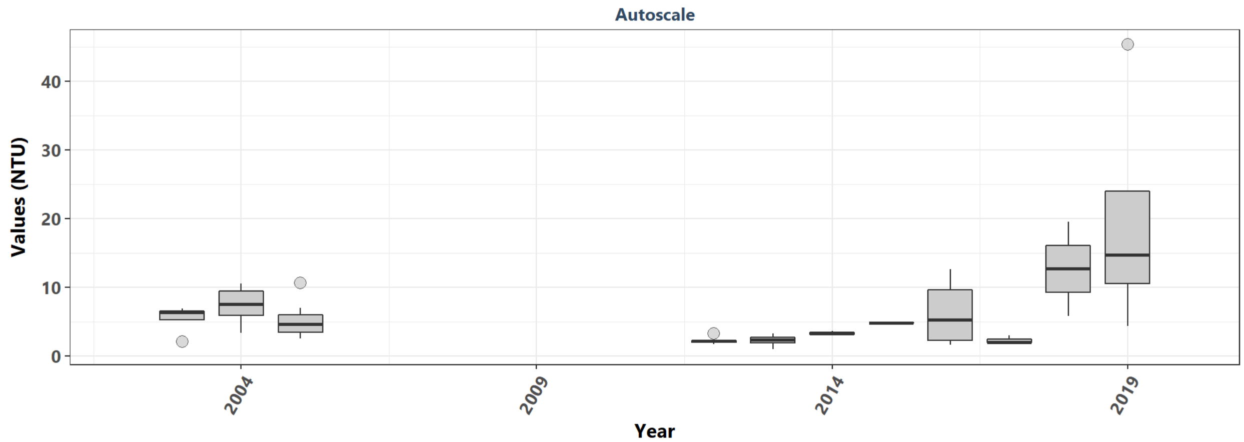
By Year

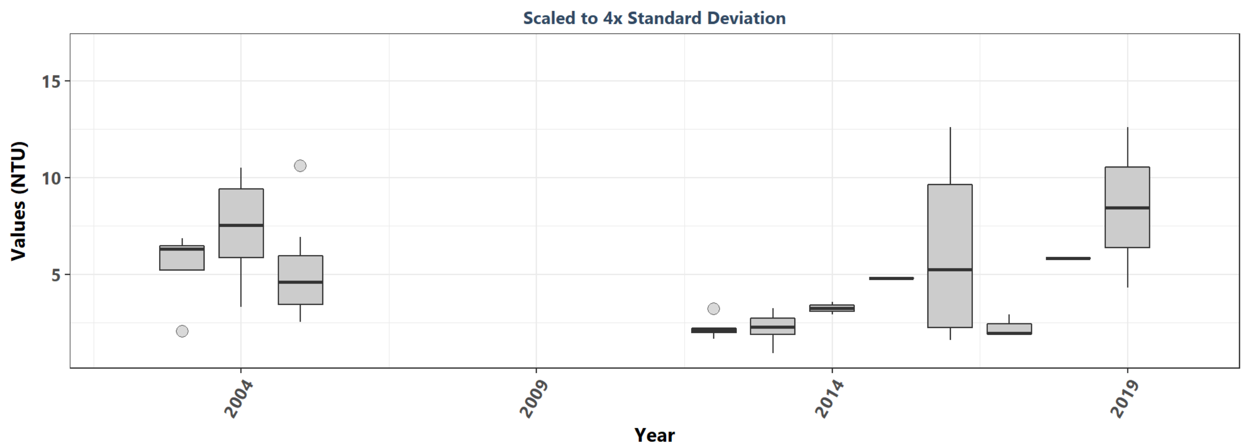


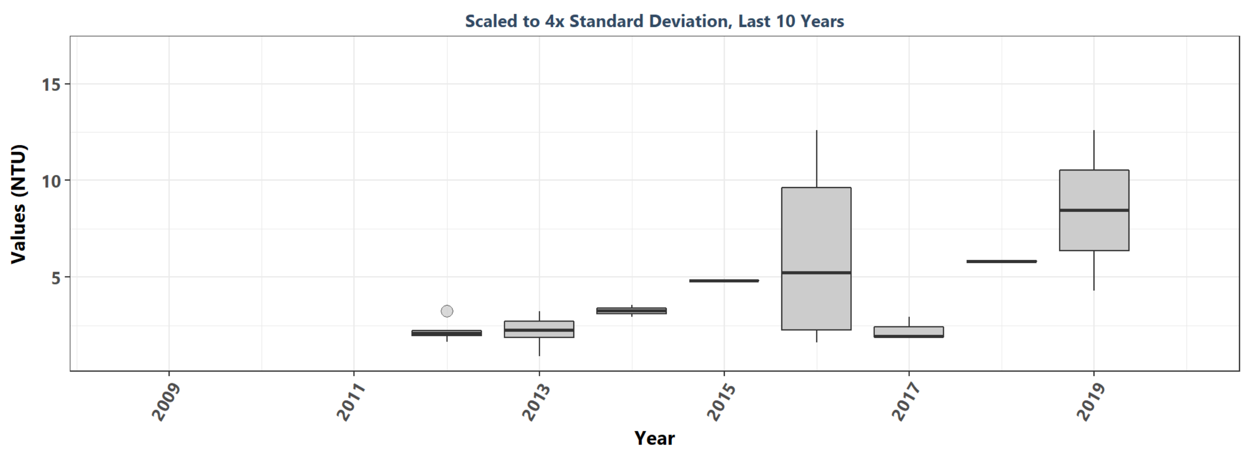

Apalachicola National Estuarine Research Reserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
Year
Scaled to 4x Standard Deviation
Values (NTU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
Year

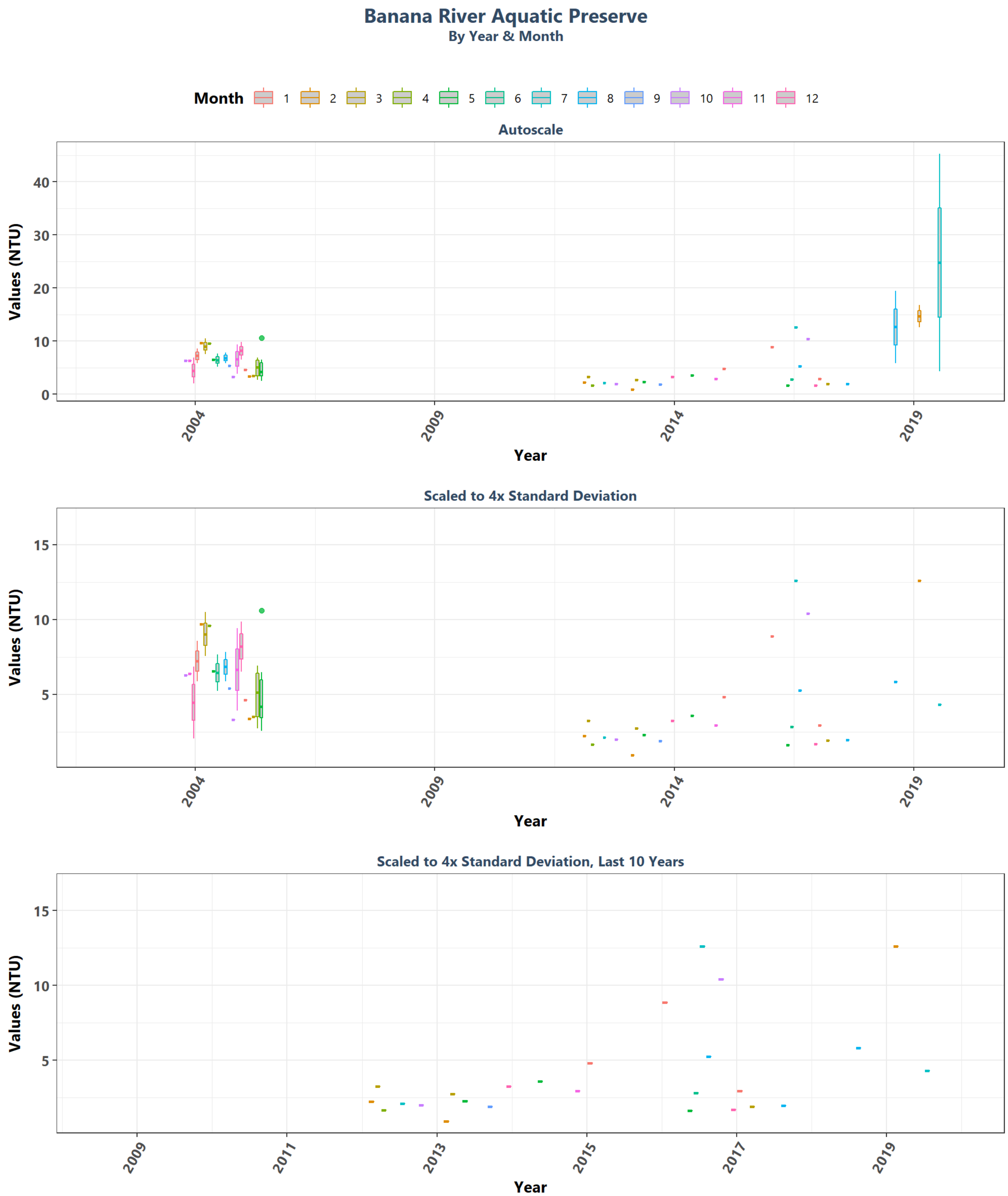

Apalachicola National Estuarine Research Reserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
0
20
40
60
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (NTU)
0
5
10
15
20
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
0
5
10
15
20
3
6
9
12
Month

Banana River Aquatic Preserve
By Year



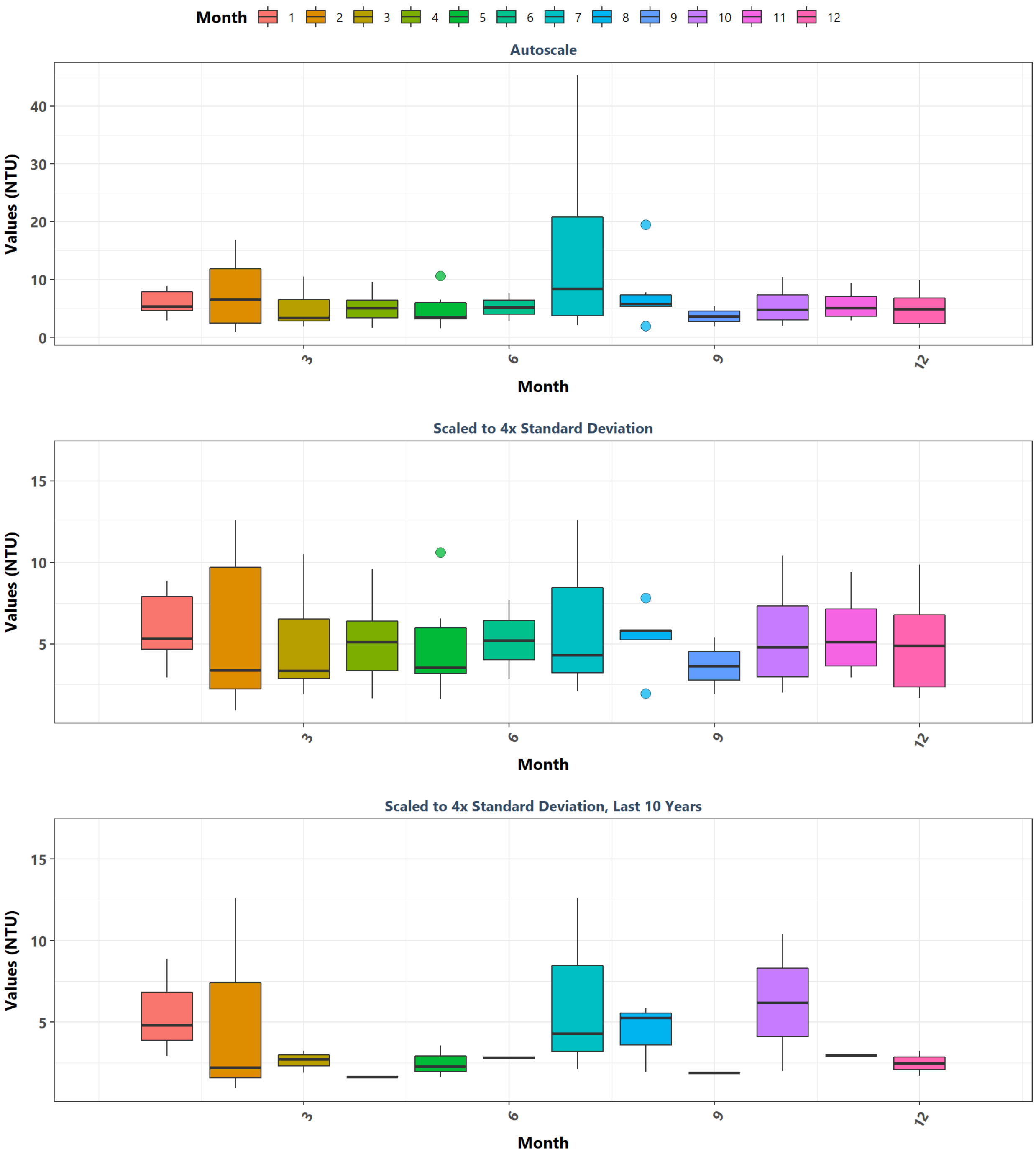


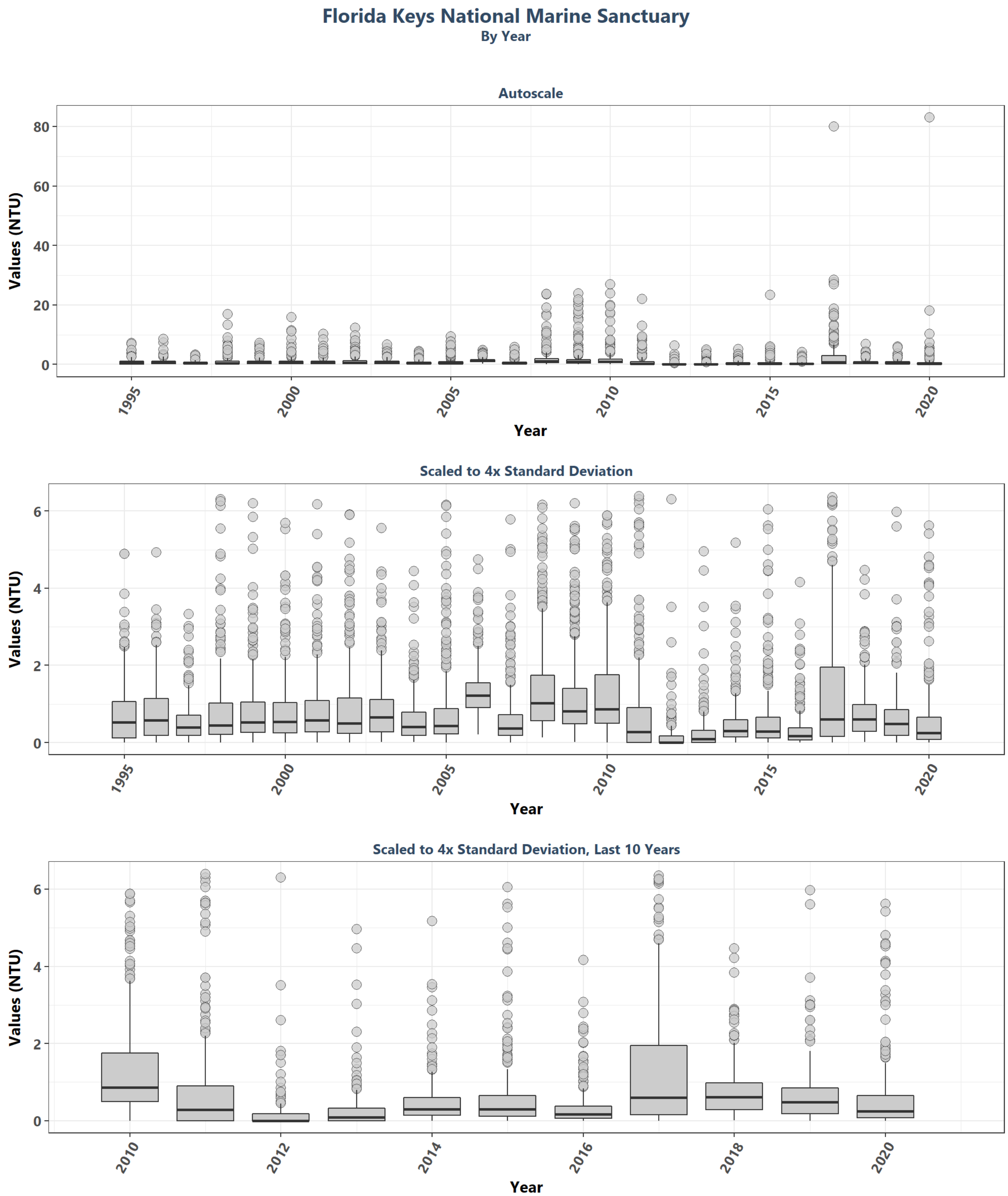



Banana River Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
0
10
20
30
40
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation
Values (NTU)
5
10
15
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
5
10
15
2009
2011
2013
2015
2017
2019
Year

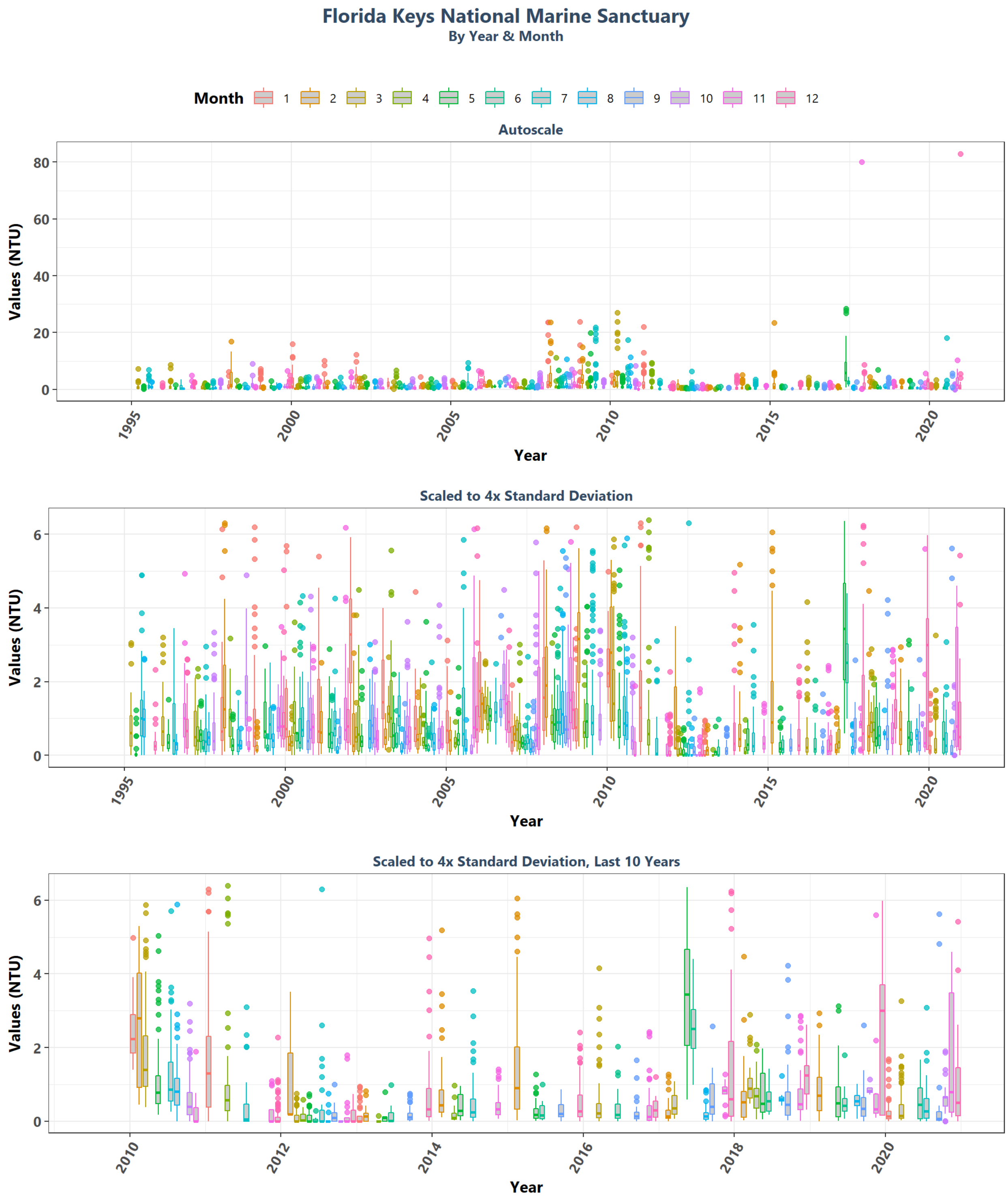
Banana River Aquatic Preserve

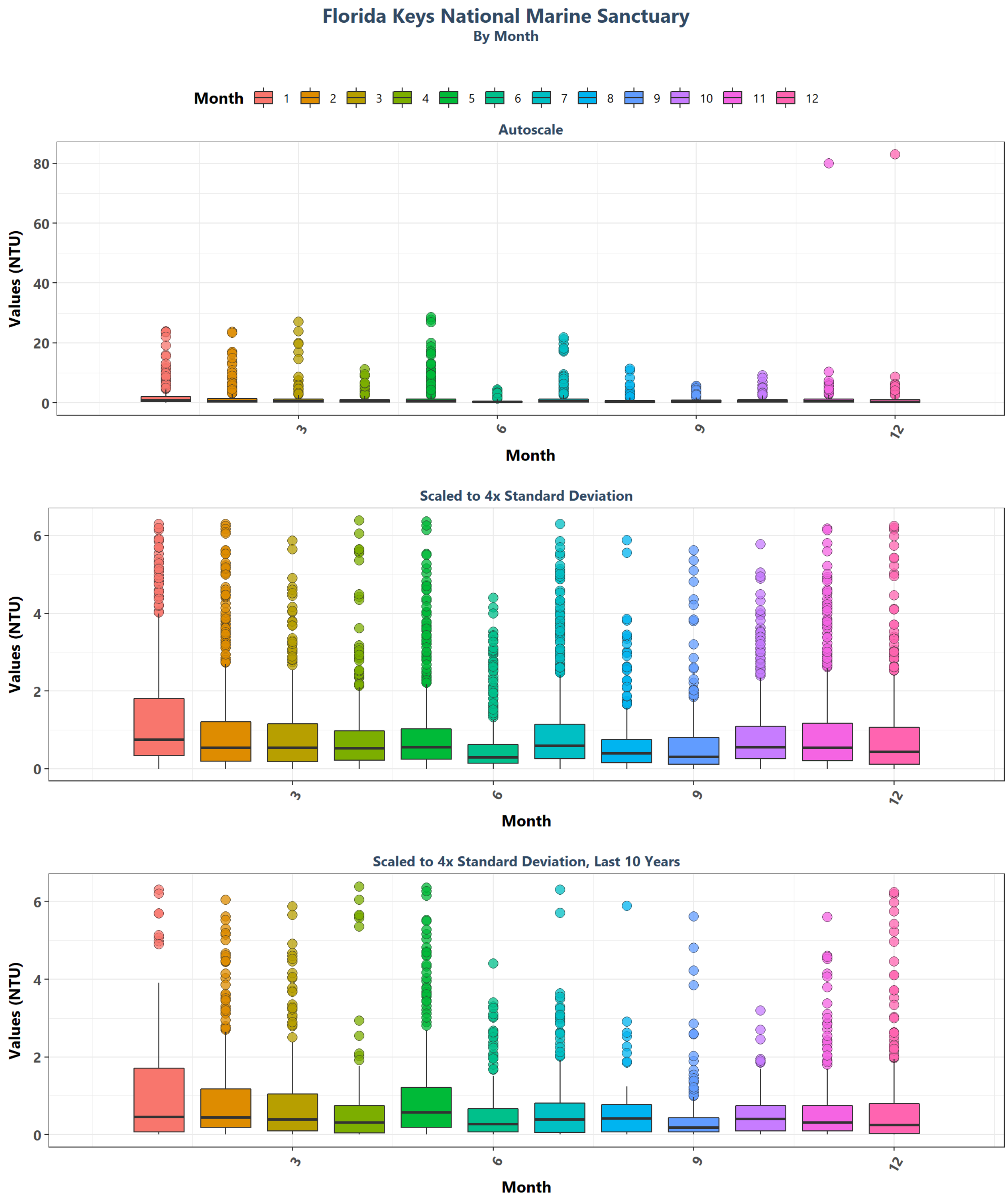
By Month



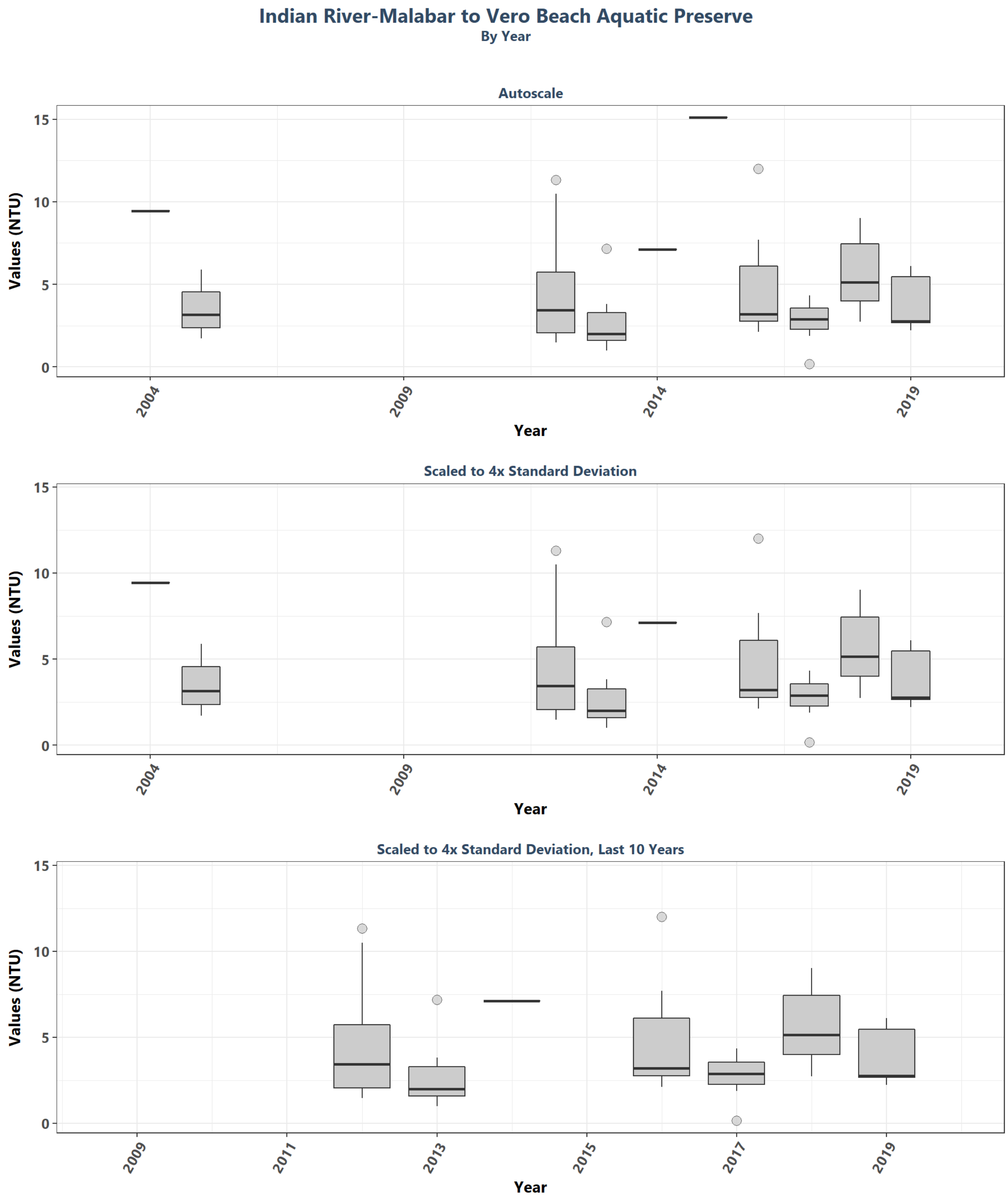


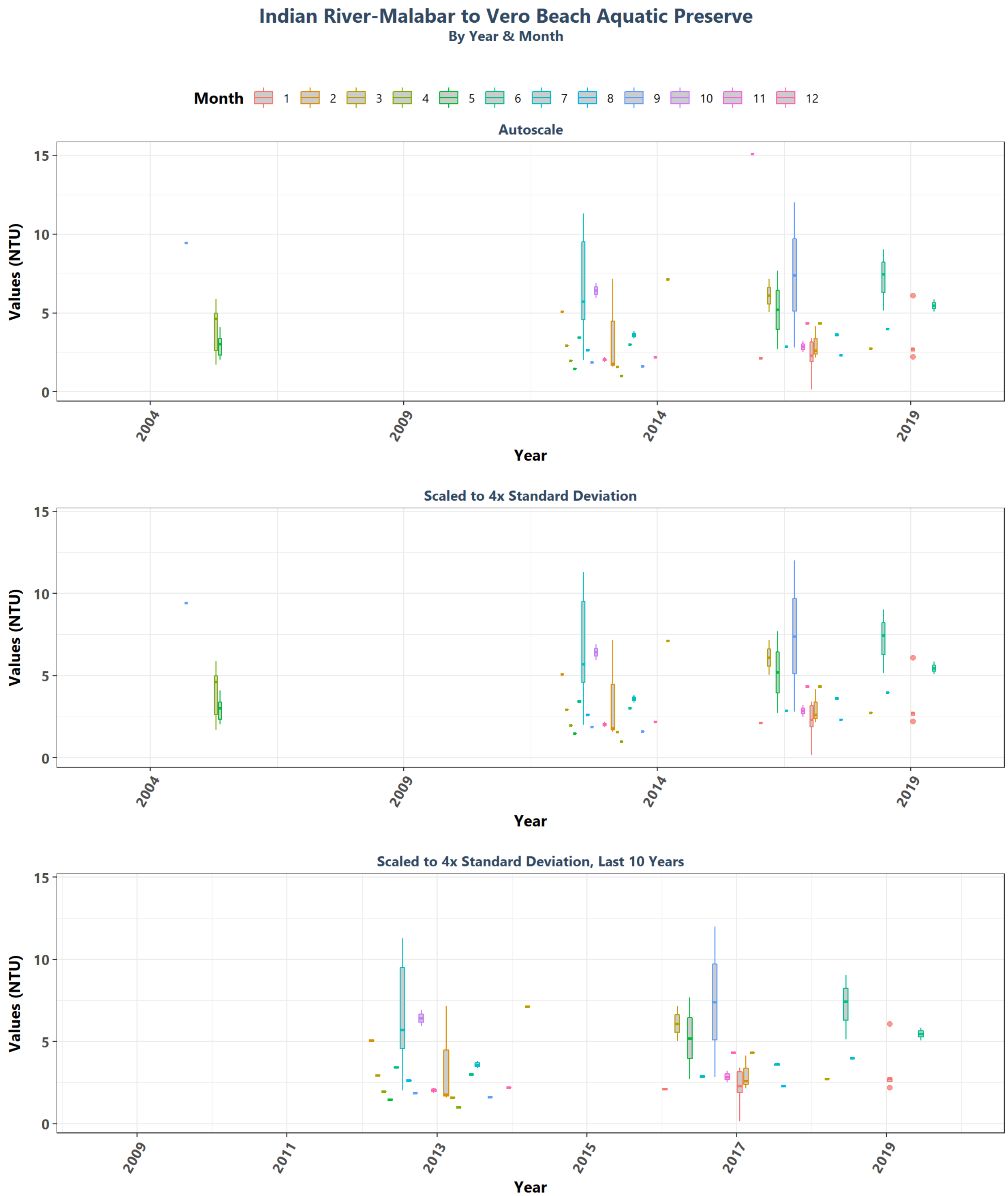
Florida Keys National Marine Sanctuary
By Year
Autoscale
Values (NTU)
80
60
40
20
0
1995
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation
Values (NTU)
6
4
2
0
1995
2000
2005
2010
2015
2020
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
6
4
2
0
2010
2012
2014
2016
2018
2020
Year

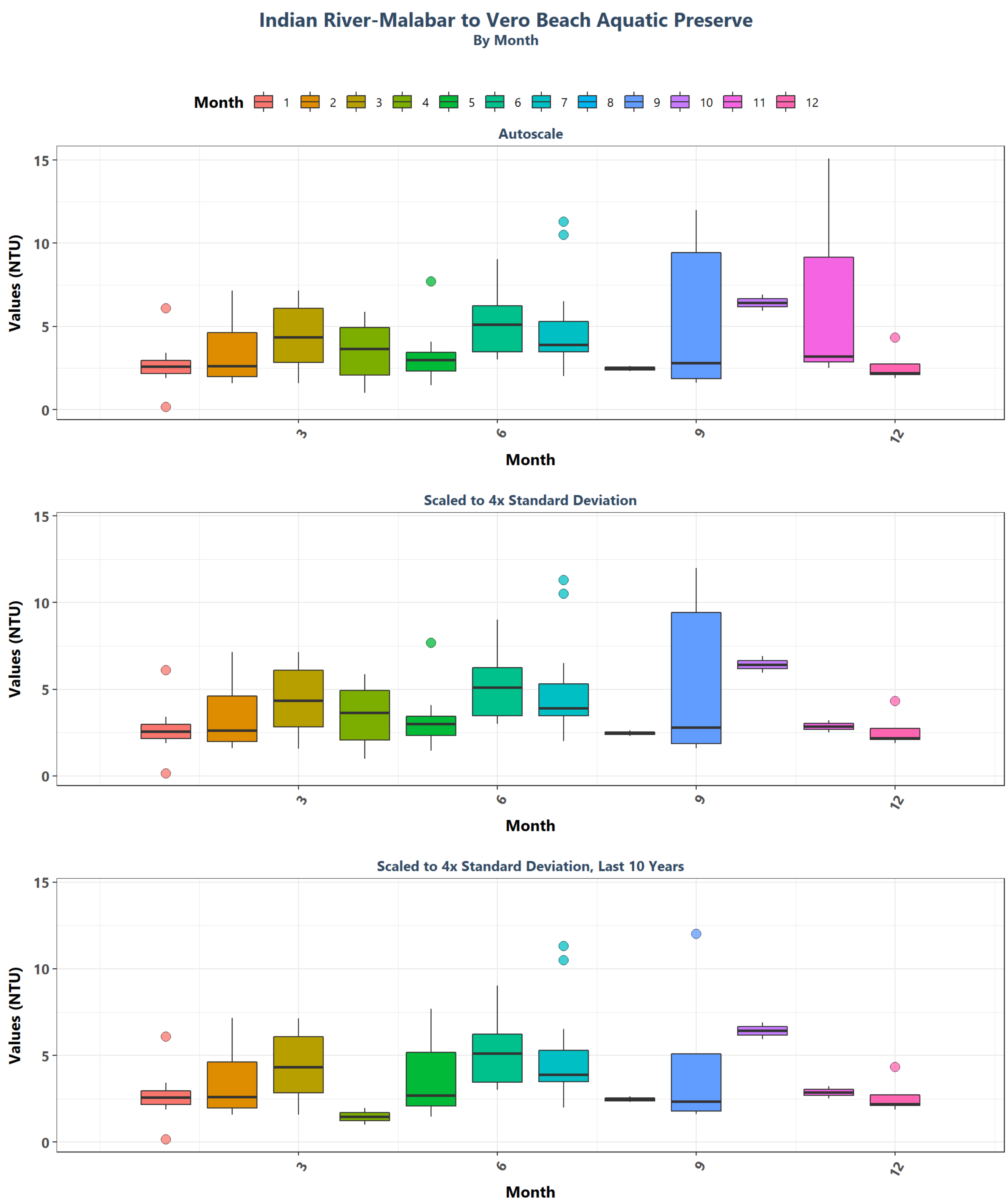

Florida Keys National Marine Sanctuary
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
Year
Scaled to 4x Standard Deviation
Values (NTU)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
Year



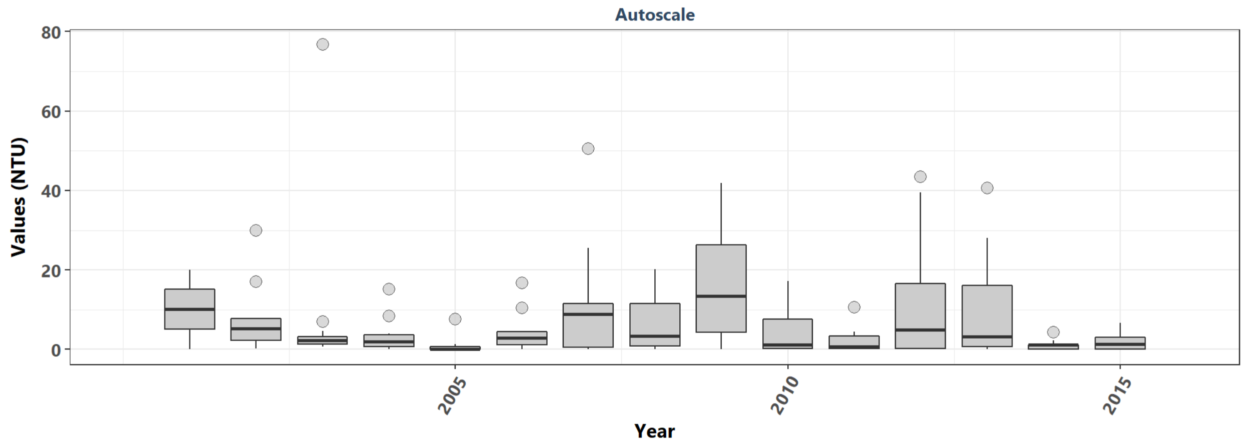
Florida Keys National Marine Sanctuary
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
80
60
40
20
0
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (NTU)
6
4
2
0
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
6
4
2
0
3
6
9
12
Month

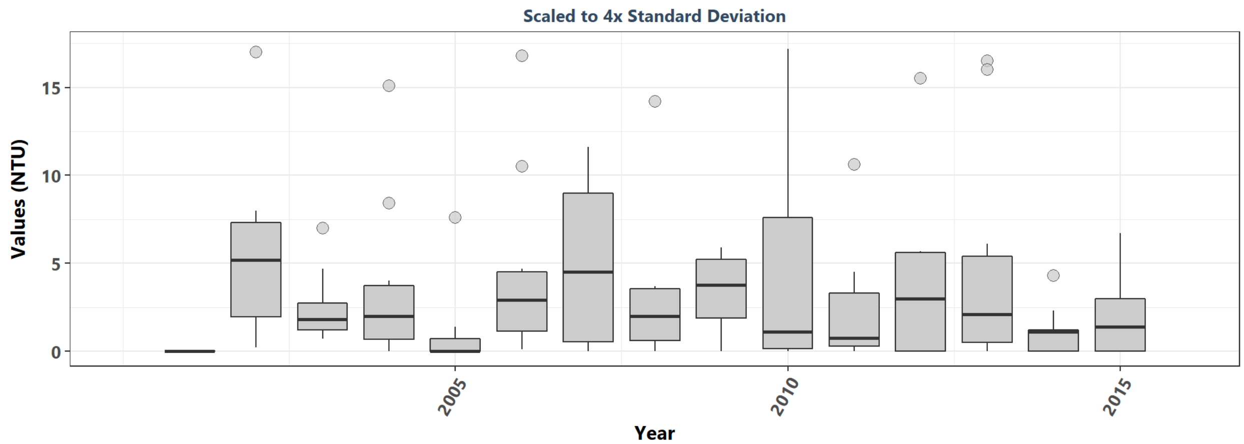

Indian River-Malabar to Vero Beach Aquatic Preserve
By Year
Autoscale
Values (NTU)
15
10
5
0
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation
Values (NTU)
15
10
5
0
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
15
10
5
0
2009
2011
2013
2015
2017
2019
Year

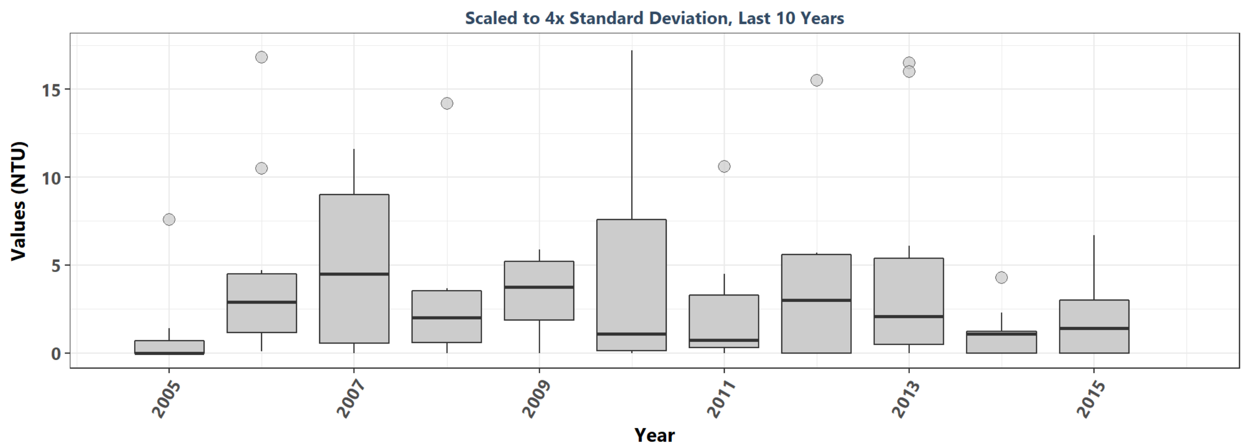

Indian River-Malabar to Vero Beach Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
15
10
5
0
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation
Values (NTU)
15
10
5
0
2004
2009
2014
2019
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
15
10
5
0
2009
2011
2013
2015
2017
2019
Year

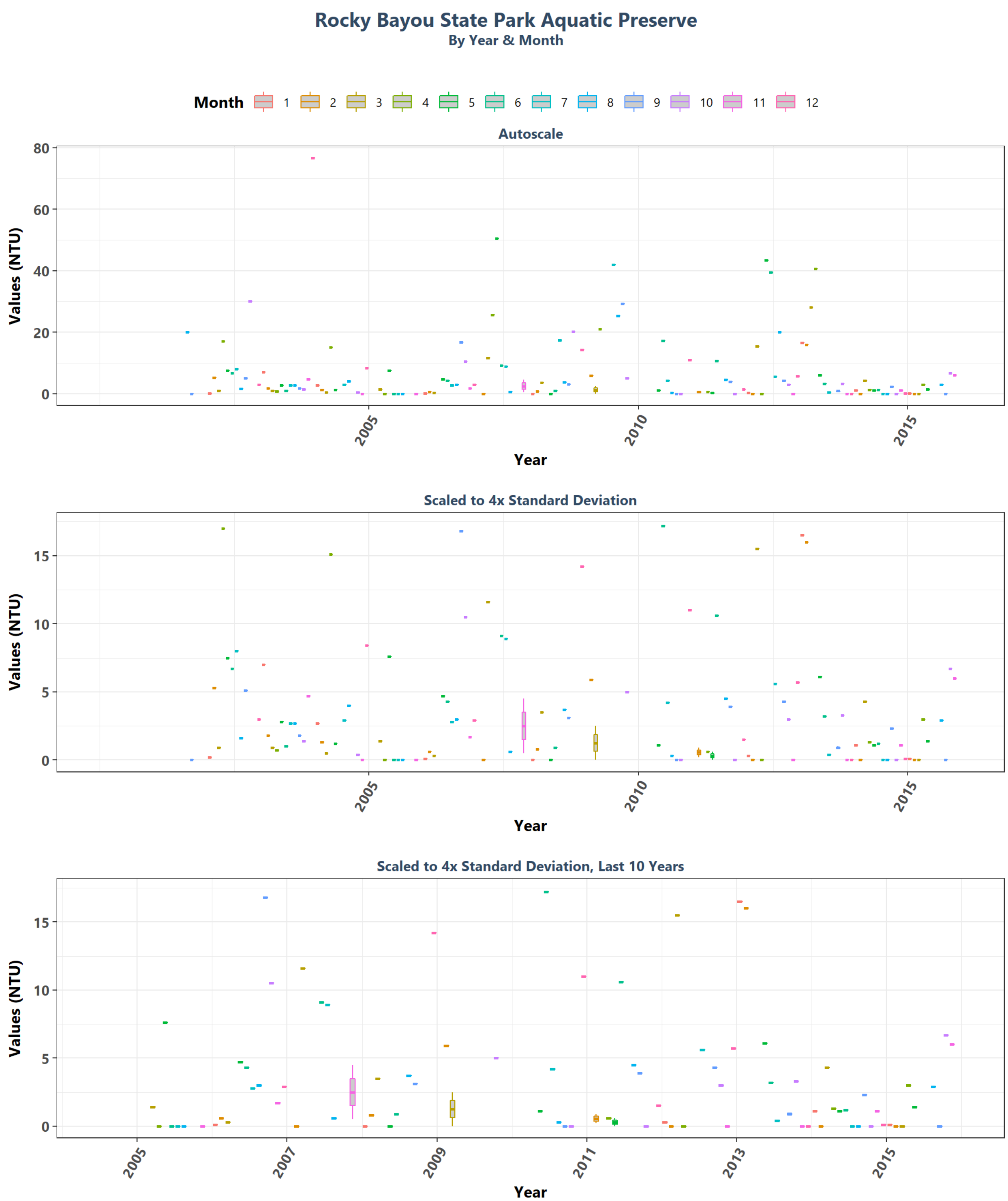

Indian River-Malabar to Vero Beach Aquatic Preserve
By Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
0
5
10
15
3
6
9
12
Month
Scaled to 4x Standard Deviation
Values (NTU)
0
5
10
15
3
6
9
12
Month
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
0
5
10
15
3
6
9
12
Month

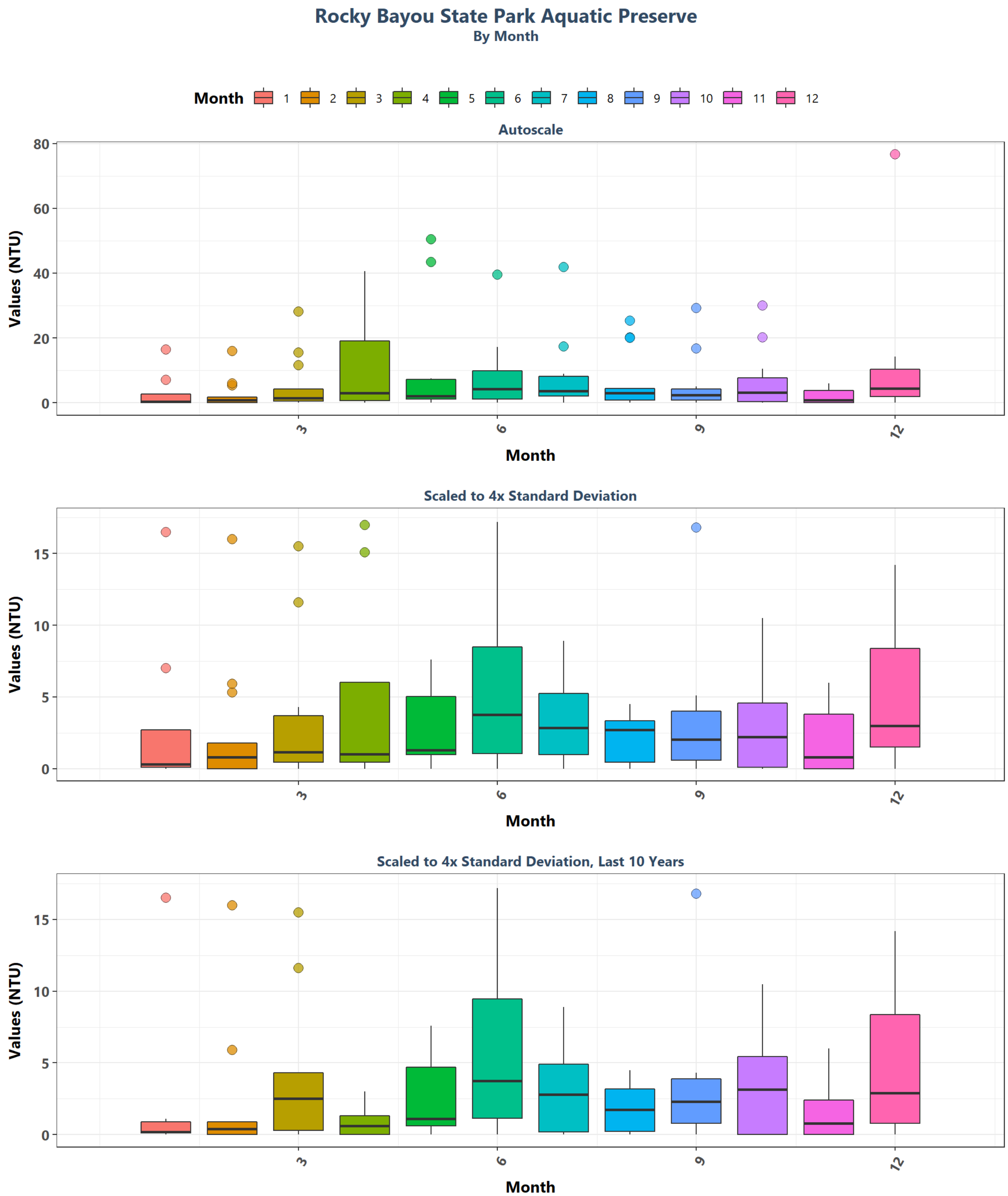
Rocky Bayou State Park Aquatic Preserve
By Year








Rocky Bayou State Park Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (NTU)
80
60
40
20
0
2005
2010
2015
Year
Scaled to 4x Standard Deviation
Values (NTU)
15
10
5
0
2005
2010
2015
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (NTU)
15
10
5
0
2005
2007
2009
2011
2013
2015
Year




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