SEACAR Discrete Water Quality Analysis: Field Bottom Dissolved Oxygen Saturation

Last compiled on 19 May, 2022

Contents

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(dplyr)
library(lubridate)
library(ggplot2)
```

```
library(ggpubr)
library(scales)
library(EnvStats)
library(tidyr)
options(scipen = 999)
```

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.

```
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 = "")

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

parameter <- unique(data$ParameterName)
unit <- unique(data$ParameterUnits)
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 Summary Before Filtering

This part is to create a data summary file that looks at the total amount of data, how many pass the initial database filtering, and how many are impacted by various ValueQualifiers. The ValueQualifiers of interest are H, I, Q, S, and U.

```
data_summ <- data %>%
   group_by(AreaID, ManagedAreaName) %>%
   summarize(ParameterName = parameter,
             RelativeDepth = depth,
             ActivityType = activity,
             N_Total = length(ResultValue),
             N_Include = length(ResultValue[Include==1]),
             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))
```

```
## `summarise()` has grouped output by 'AreaID'. You can override using the `.groups` argument.
```

```
data_summ <- as.data.table(data_summ[order(data_summ$ManagedAreaName), ])
fwrite(data_summ, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                           "_DataSummary_BeforeFilter.csv"), sep = ",")
```

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.

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

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

MA_Years <- data[data$Include == TRUE, ] %>%
   group_by(ManagedAreaName) %>%
   summarize(N = length(unique(Year)))
MA_Years <- as.data.table(MA_Years[order(MA_Years$ManagedAreaName), ])
MA_Years$Enough_Time <- ifelse(MA_Years$N < 10, FALSE, TRUE)
data <- merge.data.frame(data, MA_Years[,c("ManagedAreaName", "Enough_Time")],
                         by = "ManagedAreaName")

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

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$VQ_Plot[data$ProgramID!=476] <- gsub("[^U]+", "",
                                             data$VQ_Plot[data$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: 45855, Number Passed Filter: 45524
## 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) %>%
   summarize(ParameterName = parameter,
             RelativeDepth = depth,
             ActivityType = activity,
             N_Total = length(ResultValue),
             N_AnalysisUse = length(ResultValue[Use_In_Analysis==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))
```

```
## `summarise()` has grouped output by 'AreaID'. You can override using the `.groups` argument.
```

```
data_summ <- as.data.table(data_summ[order(data_summ$ManagedAreaName), ])
fwrite(data_summ, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                         "_DataSummary_AfterFilter.csv"), sep = ",")
```

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 `Use_In_Analysis` 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_names <- unique(data$ManagedAreaName[data$Use_In_Analysis == TRUE])
MA_names <- MA_names[order(MA_names)]
n <- length(MA_names)

MA_YM_Stats <- data[data$Use_In_Analysis == TRUE, ] %>%
   group_by(AreaID, ManagedAreaName, Year, Month) %>%
   summarize(ParameterName = parameter,
             RelativeDepth = depth,
             ActivityType = activity,
             N = length(ResultValue),
             Min = min(ResultValue),
             Max = max(ResultValue),
             Median = median(ResultValue),
             Mean = mean(ResultValue),
             StandardDeviation = sd(ResultValue),
             ProgramIDs = paste(sort(unique(ProgramID), decreasing = FALSE),
                                collapse = ', '))
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$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 = "|")
```

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

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

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 `Use_In_Analysis` 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) %>%
   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 = "|")
```

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 `Use_In_Analysis` 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) {
```

```
   tau <- NULL
   tryCatch({
      ken <-
         kendallSeasonalTrendTest(
            y = data$ResultValue,
            season = data$Month,
            year = data$Year,
            independent.obs = independent
         )
      tau <- ken$estimate[1]
      p <- ken$p.value[2]
      slope <- ken$estimate[2]
      intercept <- ken$estimate[3]
      trend <- trend_calculator(slope, stats.median, p)
   }, warning = function(w) {
      print(w)
   }, error = function(e) {
      print(e)
   }, finally = {
      if (!exists("tau")) {
         tau <- NULL
      }
      if (!exists("p")) {
         p <- NULL
      }
      if (!exists("slope")) {
         slope <- NULL
      }
      if (!exists("intercept")) {
         intercept <- NULL
      }
      if (!exists("trend")) {
         trend <- NULL
      }
   })
   KT <-c(unique(data$AreaID),
          unique(data$ManagedAreaName),
          parameter,
          depth,
          activity,
          independent,
          stats.median,
          nrow(data),
          stats.minYear,
          stats.maxYear,
          tau,
          p,
          slope,
          intercept,
          trend)
   return(KT)
}
runStats <- function(data) {
```

```
   data$Index <- as.Date(data$SampleDate) # , "%Y-%m-%d")
   data$ResultValue <- as.numeric(data$ResultValue)
   # Calculate basic stats
   stats.median <- median(data$ResultValue, na.rm = TRUE)
   stats.minYear <- min(data$Year, na.rm = TRUE)
   stats.maxYear <- max(data$Year, na.rm = TRUE)
   # Calculate Kendall Tau and Slope stats, then update appropriate columns and table
   KT <- tauSeasonal(data, TRUE, stats.median,
                     stats.minYear, stats.maxYear)
   if (is.null(KT[11])) {
      KT <- tauSeasonal(data, FALSE, stats.median,
                        stats.minYear, stats.maxYear)
   }
   if (is.null(KT.Stats) == TRUE) {
      KT.Stats <- KT
   } else{
      KT.Stats <- rbind(KT.Stats, KT)
   }
   return(KT.Stats)
}
trend_calculator <- function(slope, median_value, p) {
   trend <-
      if (p < .05 & abs(slope) > abs(median_value) / 10.) {
         if (slope > 0) {
            2
         }
         else {
            -2
         }
      }
   else if (p < .05 & abs(slope) < abs(median_value) / 10.) {
      if (slope > 0) {
         1
      }
      else {
         -1
      }
   }
   else
      0
   return(trend)
}
KT.Stats <- NULL
# Loop that goes through each managed area. List of managed areas stored in MA_Years$ManagedAreaName
c_names <- c("AreaID", "ManagedAreaName", "ParameterName", "RelativeDepth",
             "ActivityType", "Independent", "Median", "N", "EarliestYear",
             "LatestYear", "tau", "p", "SennSlope", "SennIntercept", "Trend")
if(n==0){
   KT.Stats <- data.frame(matrix(ncol=15, nrow=0))
   colnames(KT.Stats) <- c_names
   fwrite(KT.Stats, paste0(out_dir,"/", param_name, "_", file_date, "_",
                           activity, "_", depth,
                           "_KendallTau_Stats.txt"), sep = "|")
```

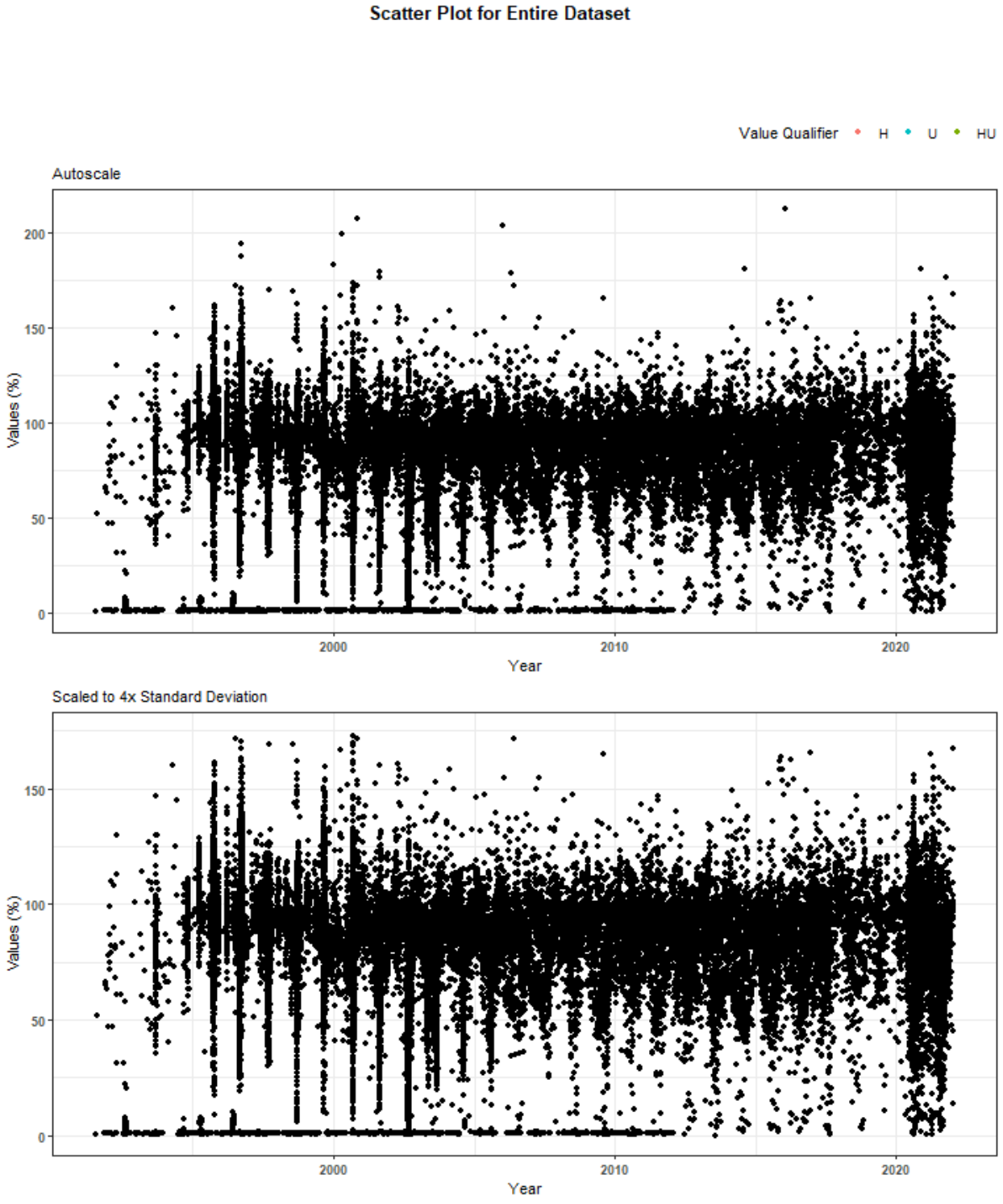
```
} else{
   for (i in 1:n) {
      values <- data[data$Use_In_Analysis == TRUE &
                        data$ManagedAreaName == MA_names[i], ]
      if (nrow(values) > 0) {
         KT.Stats <- runStats(values)
      }
   }
   KT.Stats <- as.data.frame(KT.Stats)
   c_names <- c("AreaID", "ManagedAreaName", "ParameterName", "RelativeDepth",
                "ActivityType", "Independent", "Median", "N", "EarliestYear",
                "LatestYear", "tau", "p", "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$Median <- as.numeric(KT.Stats$Median)
   KT.Stats$N <- as.integer(KT.Stats$N)
   KT.Stats$EarliestYear <- as.integer(KT.Stats$EarliestYear)
   KT.Stats$LatestYear <- as.integer(KT.Stats$LatestYear)
   KT.Stats$tau <- round(as.numeric(KT.Stats$tau), digits=4)
   KT.Stats$p <- round(as.numeric(KT.Stats$p), 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)
   fwrite(KT.Stats, paste0(out_dir,"/", param_name, "_", activity, "_", depth,
                           "_KendallTau_Stats.txt"), sep = "|")
}
```

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.

```
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,
                 color=VQ_Plot)) +
   geom_point(size = 1.5) +
```

```
   labs(subtitle = "Autoscale",
        x = "Year", y = paste0("Values (", unit, ")"),
        color="Value Qualifier") +
   theme_bw() +
   theme(legend.position = "top", legend.box = "horizontal",
         legend.justification = "right",
         axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face="bold")) +
   scale_x_date(labels = date_format("%Y")) +
   {if(inc_H==TRUE){
      scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
                                    "HU" = "#7CAE00"), na.value="black")
   } else {
      scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
   }}


p2 <- ggplot(data = data[data$Include==TRUE,],
             aes(x = SampleDate, y = ResultValue,
                 color=VQ_Plot)) +
   geom_point(size = 1.5) +
   ylim(min_RV, y_scale) +
   labs(subtitle = "Scaled to 4x Standard Deviation",
        x = "Year", y = paste0("Values (", unit, ")")) +
   theme_bw() +
   theme(legend.position = "none",
         axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold")) +
   scale_x_date(labels = date_format("%Y")) +
   {if(inc_H==TRUE){
      scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
                                    "HU" = "#7CAE00"), na.value="black")
   } else {
      scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
   }}

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") +
   theme_bw() + theme(plot.title = element_text(face="bold", hjust = 0.5),
                      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 `Use_In_Analysis` of `TRUE`
2. Set what values are to be used for the x-axis, y-axis, and the variable that should determine groups for the box plots
3. Set the plot type as a box plot with the size of the outlier points
4. Create the title, x-axis, y-axis, and color fill labels
5. Set the y and x limits
6. Make the axis labels bold
7. Plot the arrangement as a set of panels

This set of box plots are grouped by year.

```
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(outlier.size = 0.5) +
   labs(subtitle = "Autoscale", x = "Year",
        y = paste0("Values (", unit, ")")) +
   theme_bw() +
   theme(axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))

p2 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = Year, y = ResultValue, group = Year)) +
   geom_boxplot(outlier.size = 0.5) +
   labs(subtitle = "Scaled to 4x Standard Deviation", x = "Year",
        y = paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   theme_bw() +
   theme(axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))

p3 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = as.integer(Year), y = ResultValue, group = Year)) +
   geom_boxplot(outlier.size = 0.5) +
   labs(subtitle = "Scaled to 4x Standard Deviation, Last 10 Years",
        x = "Year", y = paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits = c(max(data$Year) - 10.5, max(data$Year)+1),
                      breaks = seq(max(data$Year) - 10, max(data$Year), 2)) +
   theme_bw() +
   theme(axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))

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

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

```
                      subtitle = "By Year") + theme_bw() +
   theme(plot.title = element_text(face="bold", hjust = 0.5),
         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(outlier.size = 0.5) +
   labs(subtitle = "Autoscale", x = "Year",
        y = paste0("Values (", unit, ")"), color="Month") +
   theme_bw() +
   theme(legend.position = "top", legend.box = "horizontal",
         axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold")) +
   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(outlier.size = 0.5) +
   labs(subtitle = "Scaled to 4x Standard Deviation",
        x = "Year", y = paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   theme_bw() +
   theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))

p3 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = YearMonthDec, y = ResultValue,
                 group = YearMonth, color = as.factor(Month))) +
   geom_boxplot(outlier.size = 0.5) +
   labs(subtitle = "Scaled to 4x Standard Deviation, Last 10 Years",
        x = "Year", y = paste0("Values (", unit, ")")) +
   ylim(min_RV, y_scale) +
   scale_x_continuous(limits = c(max(data$Year) - 10.5, max(data$Year)+1),
                      breaks = seq(max(data$Year) - 10, max(data$Year), 2)) +
   theme_bw() +
   theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))
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") + theme_bw() +
   theme(plot.title = element_text(face="bold", hjust = 0.5),
         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))
```

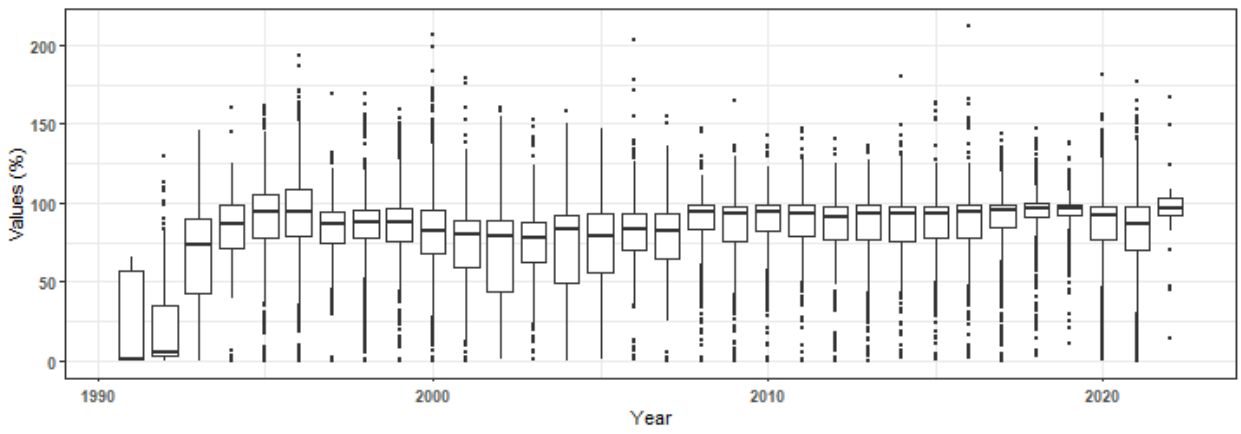
The following box plots are grouped by month with fill color being related to the month. This is designed to view potential seasonal trends.

```
p1 <- ggplot(data = data[data$Include == TRUE, ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
   geom_boxplot(outlier.size = 0.5) +
   labs(subtitle = "Autoscale", x = "Month",
        y = paste0("Values (", unit, ")"), fill="Month") +
   scale_x_continuous(limits = c(0, 13), breaks = seq(3, 12, 3)) +
   theme_bw() +
   theme(legend.position = "top", legend.box = "horizontal",
         axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold")) +
   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(outlier.size = 0.5) +
   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)) +
   theme_bw() +
   theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))

p3 <- ggplot(data = data[data$Include == TRUE &
                           data$Year >= max(data$Year) - 10, ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
   geom_boxplot(outlier.size = 0.5) +
   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)) +
   theme_bw() +
   theme(legend.position = "none", axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))
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") + theme_bw() +
   theme(plot.title = element_text(face="bold", hjust = 0.5),
         panel.border = element_blank(), panel.grid.major = element_blank(),
         panel.grid.minor = element_blank(), axis.line = element_blank())

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

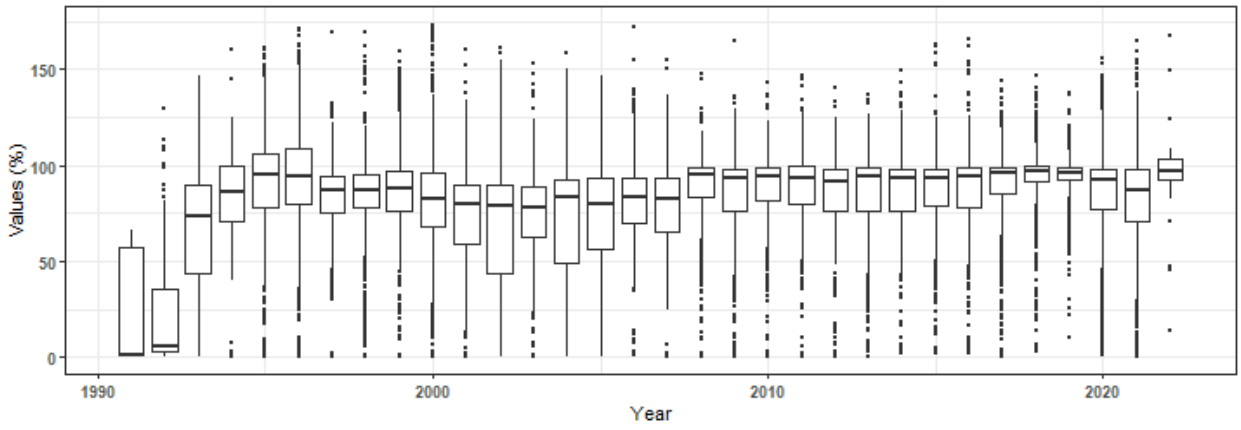
Summary Box Plots for Entire Data

By Year

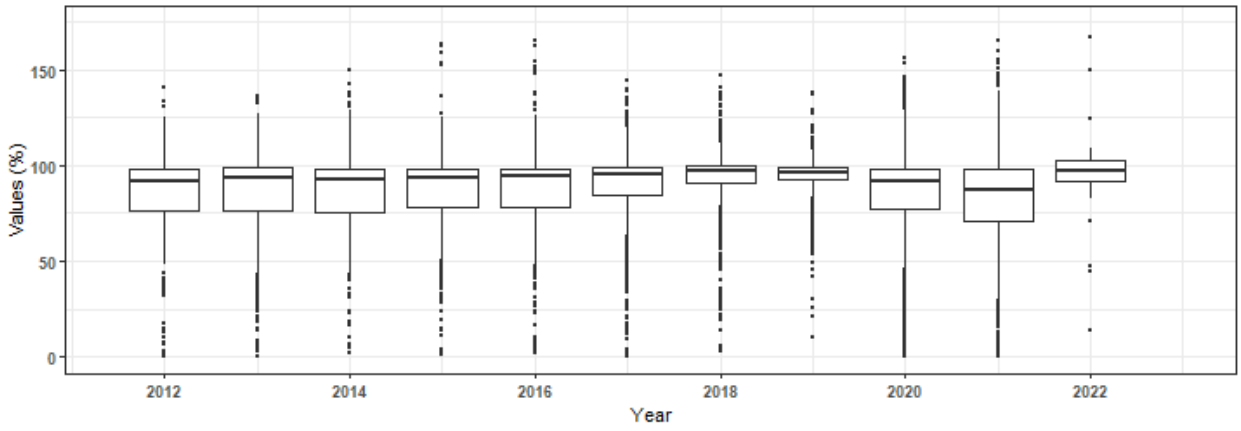
Autoscale



Scaled to 4x Standard Deviation

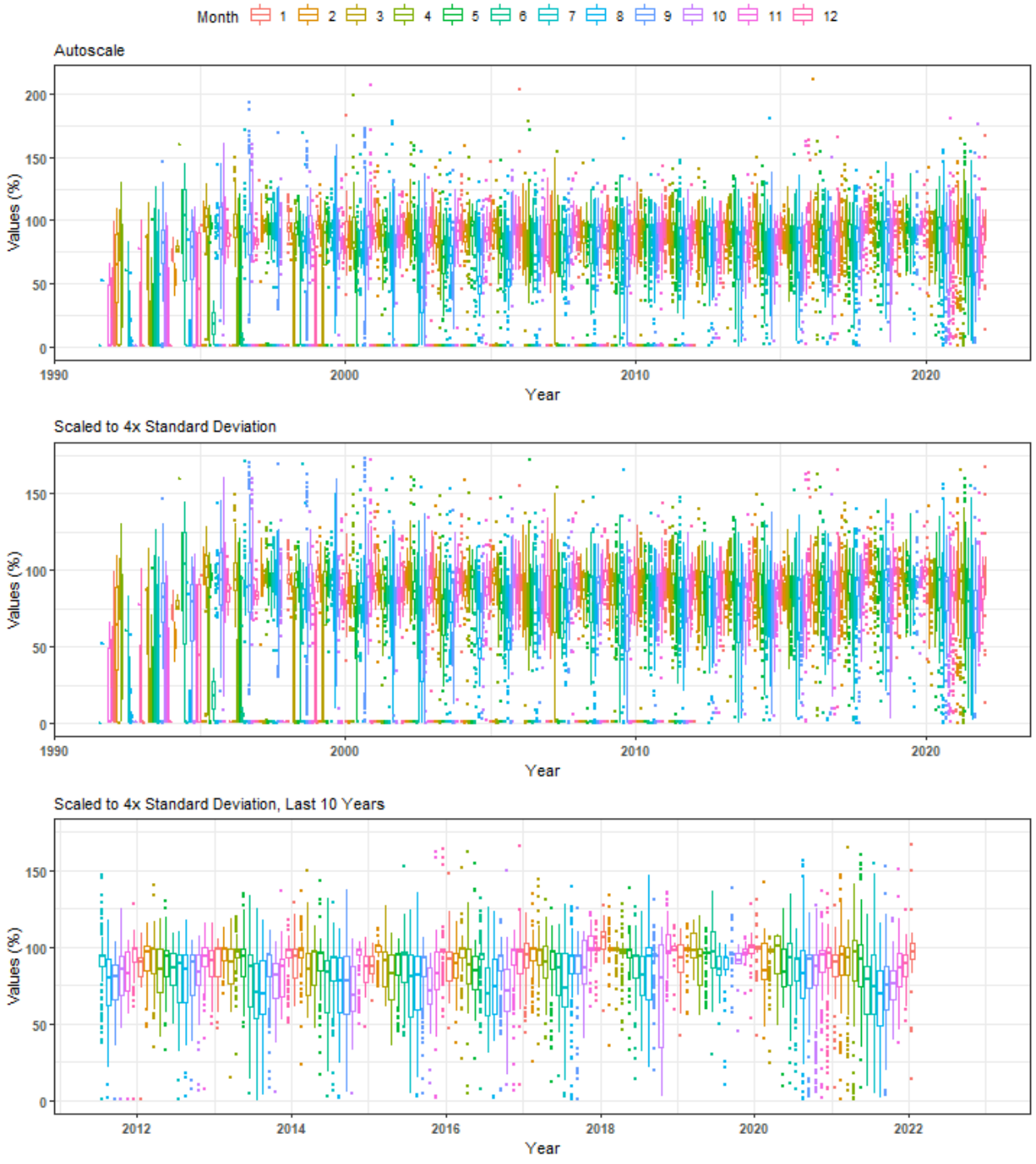


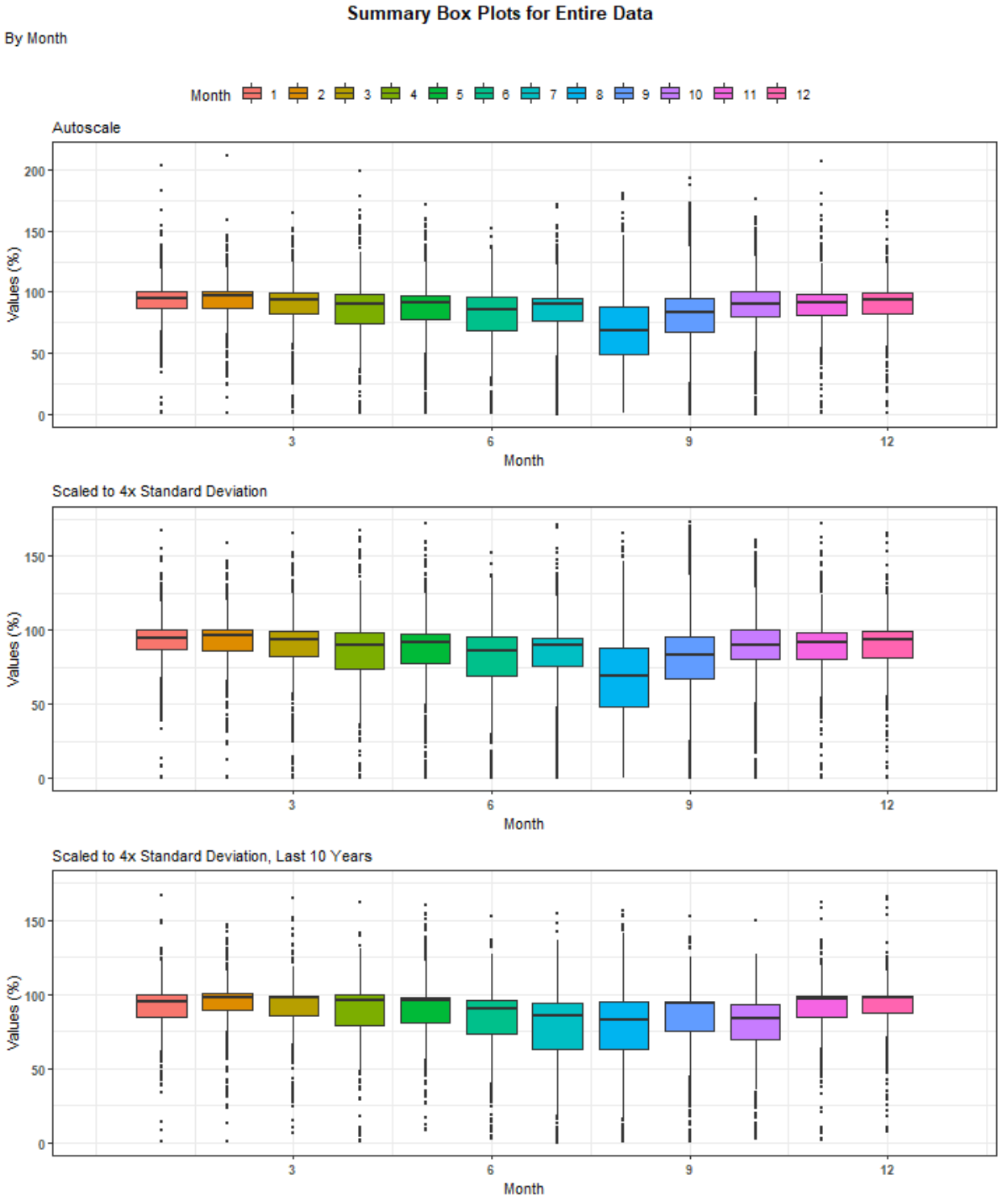
Scaled to 4x Standard Deviation, Last 10 Years



Summary Box Plots for Entire Data

By Year & Month

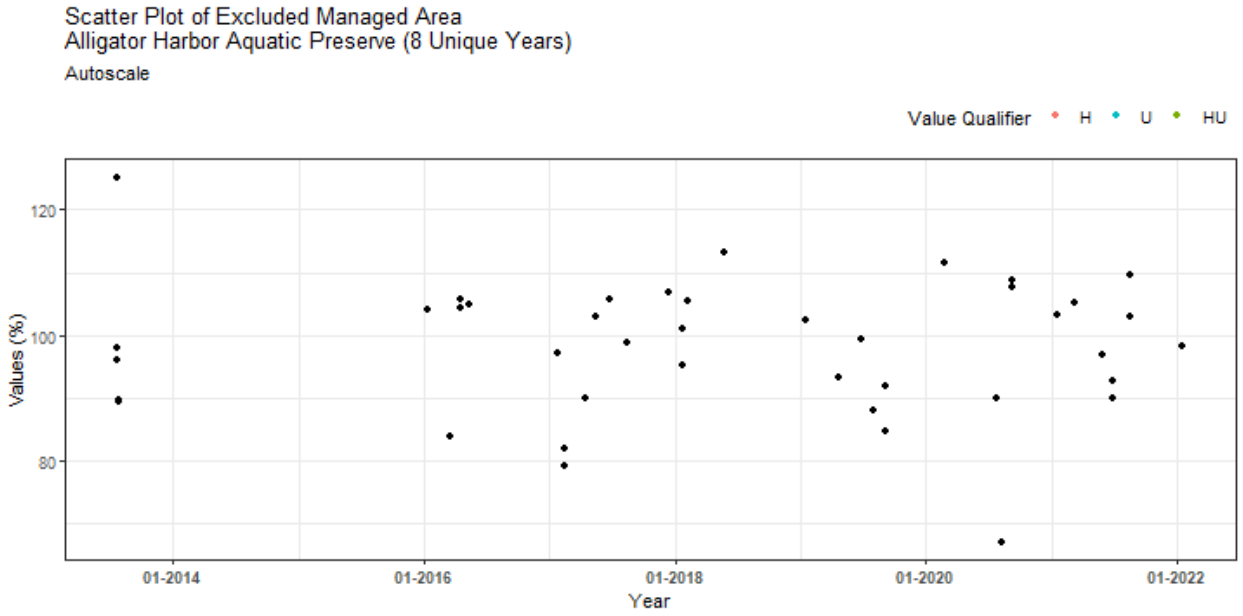




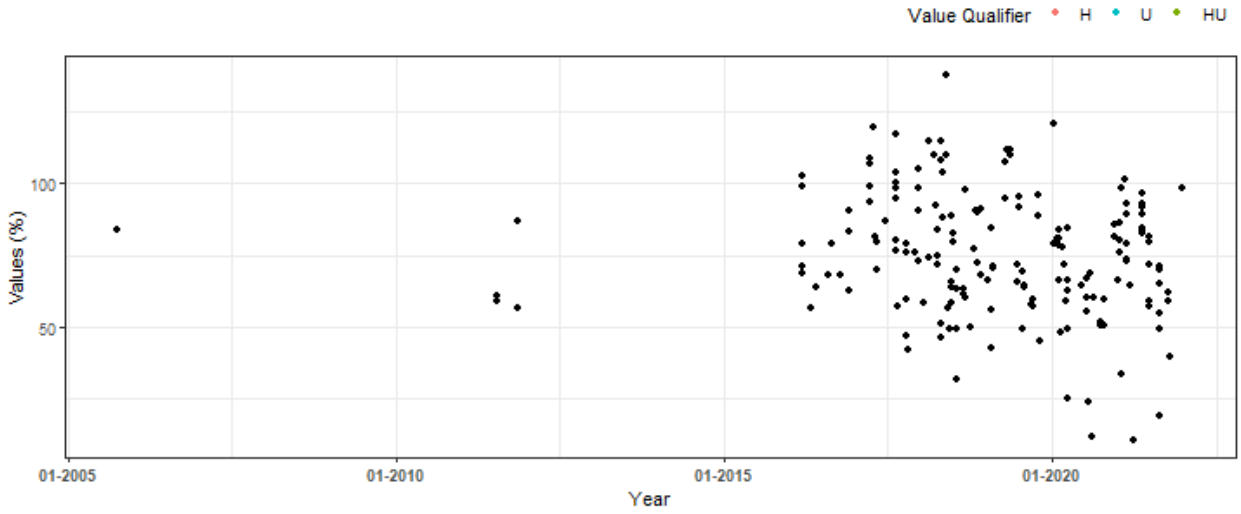
Appendix III: Excluded Managed Areas

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.

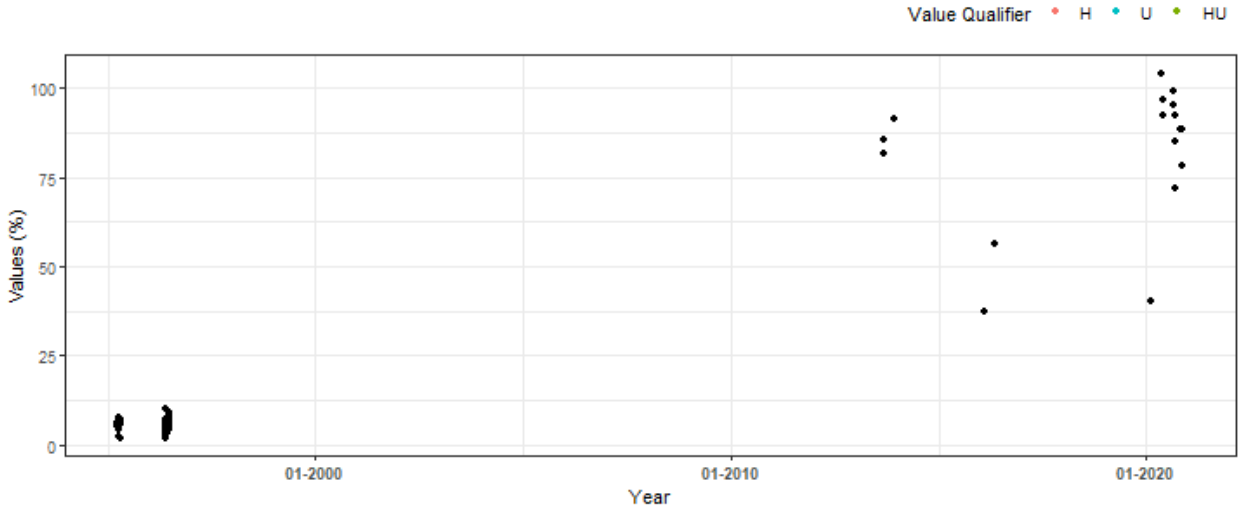
```
MA_Exclude <- MA_Years[MA_Years$Enough_Time==FALSE,]
MA_Exclude <- MA_Exclude[order(MA_Exclude$ManagedAreaName),]
z=length(MA_Exclude$ManagedAreaName)

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$Include == TRUE, ],
                 aes(x = SampleDate, y = ResultValue, color=VQ_Plot)) +
         geom_point() +
         labs(title = paste0("Scatter Plot of Excluded Managed Area\n",
                             MA_Exclude$ManagedAreaName[i], " (",
                             MA_Exclude$N[i], " Unique Years)"),
              subtitle="Autoscale", x = "Year",
              y = paste0("Values (", unit, ")"), color="Value Qualifier") +
         theme_bw() +
         theme(legend.position = "top", legend.box = "horizontal",
               legend.justification = "right",

               axis.text.x = element_text(face = "bold")) +
         scale_x_date(labels = date_format("%m-%Y")) +
         {if(inc_H==TRUE){
            scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
                                          "HU" = "#7CAE00"), na.value="black")
         } else {
            scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
         }}
      print(p1)
   }
}
```



Scatter Plot of Excluded Managed Area
Big Bend Seagrasses Aquatic Preserve (8 Unique Years)
Autoscale

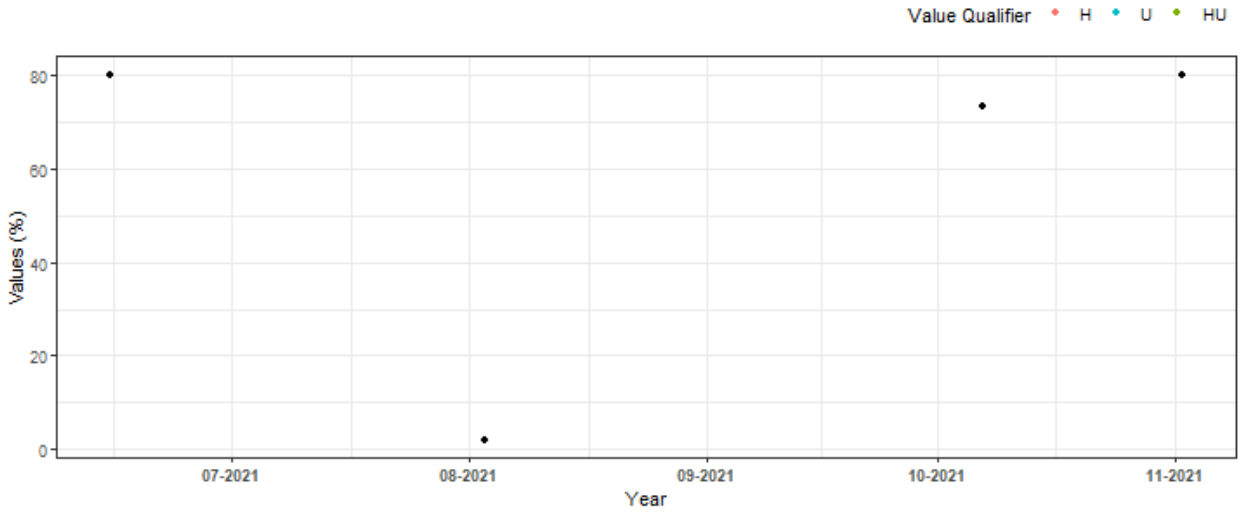


Scatter Plot of Excluded Managed Area
Biscayne Bay Aquatic Preserve (5 Unique Years)
Autoscale



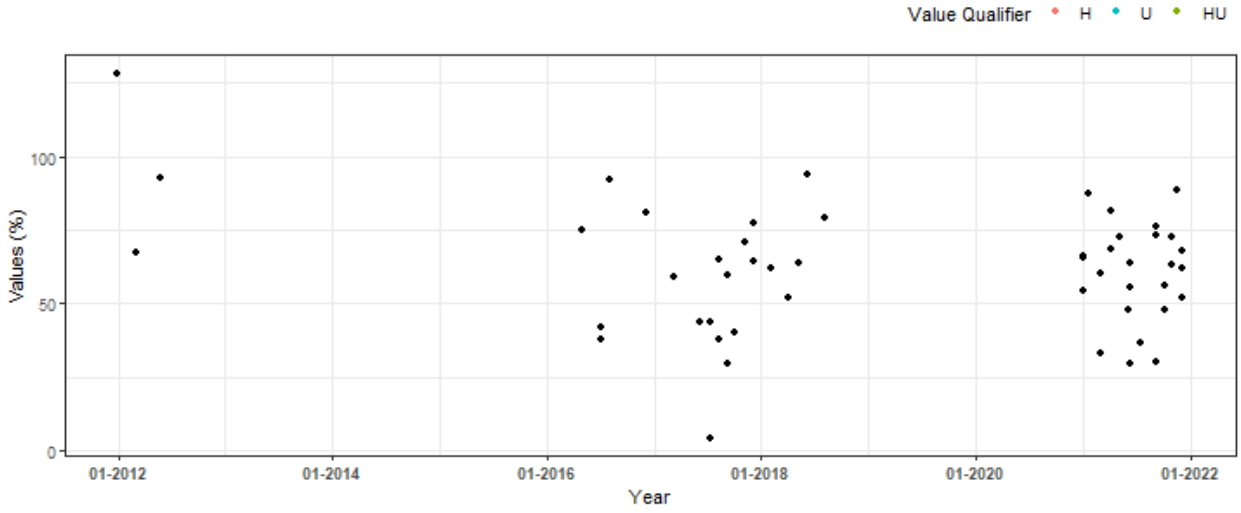
Scatter Plot of Excluded Managed Area
Cape Haze Aquatic Preserve (1 Unique Years)

Autoscale



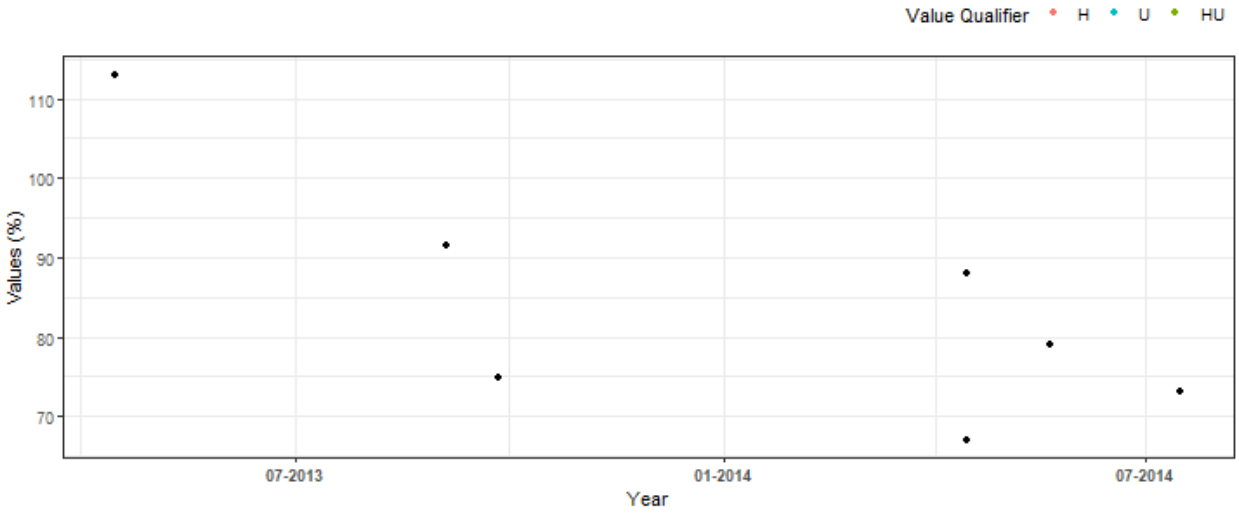
Scatter Plot of Excluded Managed Area
Estero Bay Aquatic Preserve (6 Unique Years)

Autoscale



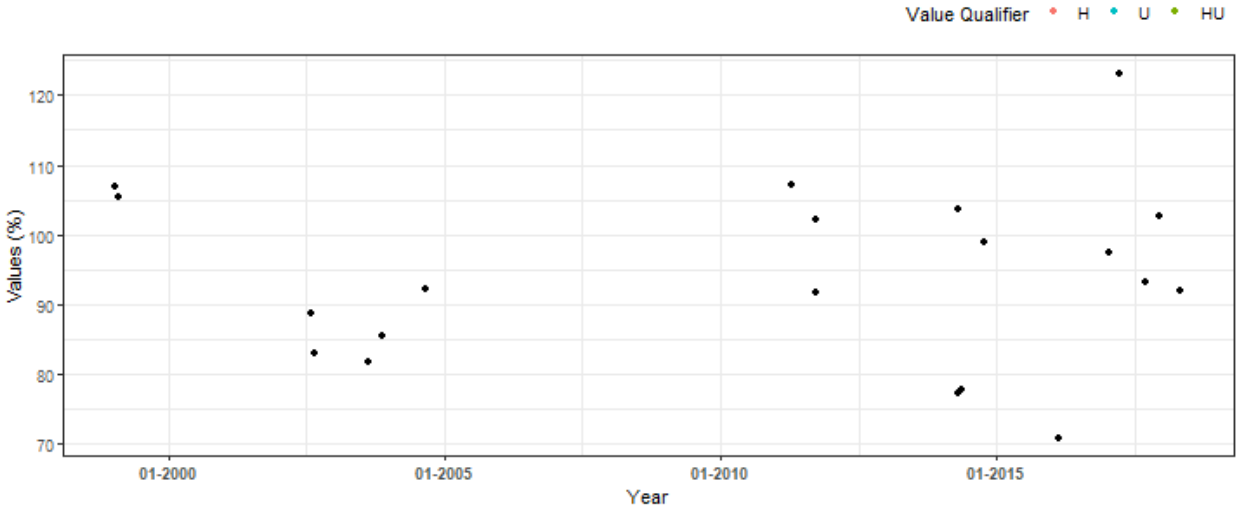
Scatter Plot of Excluded Managed Area
Fort Clinch State Park Aquatic Preserve (2 Unique Years)

Autoscale

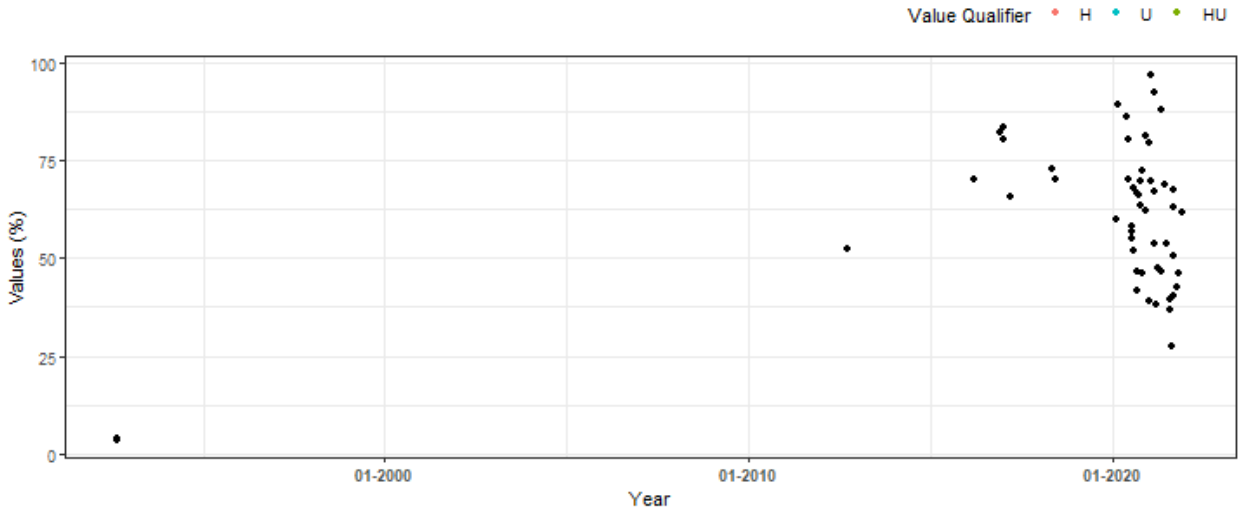


Scatter Plot of Excluded Managed Area
Fort Pickens State Park Aquatic Preserve (9 Unique Years)

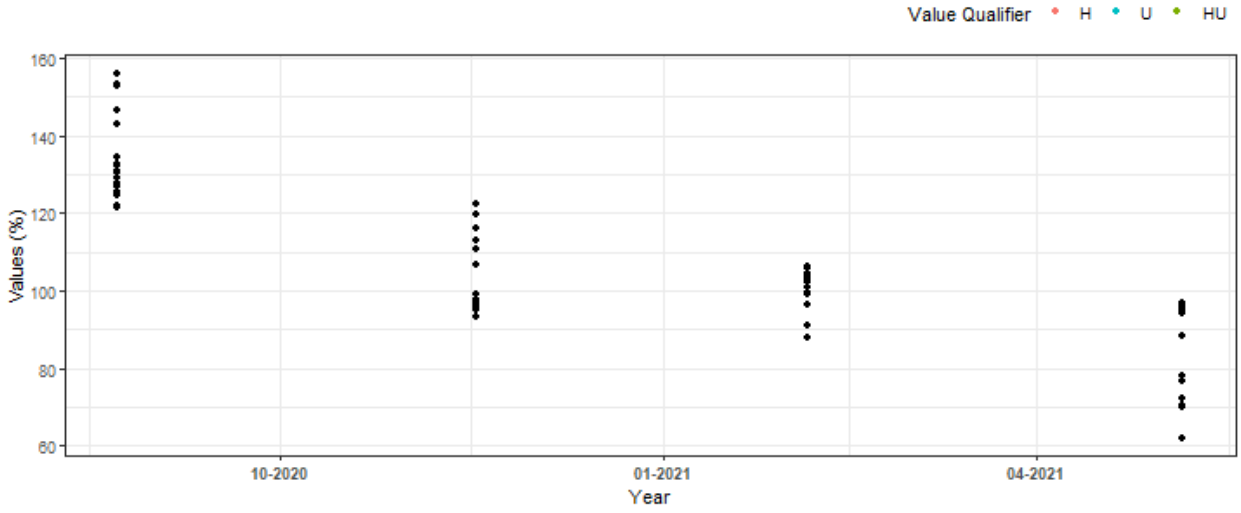
Autoscale

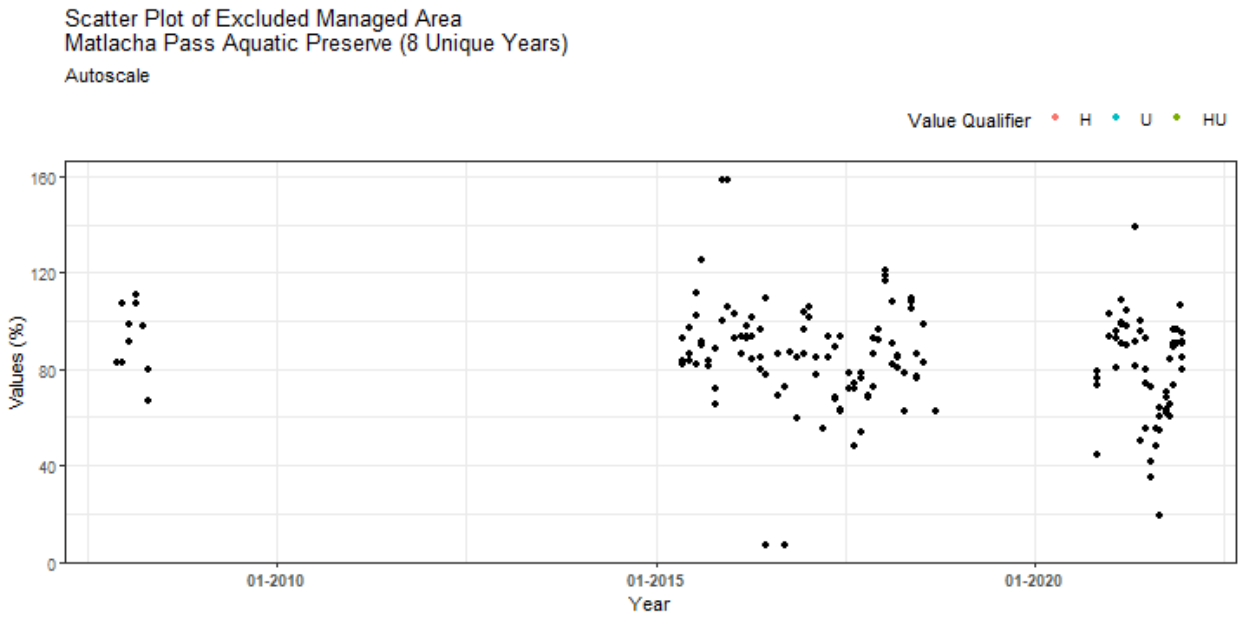


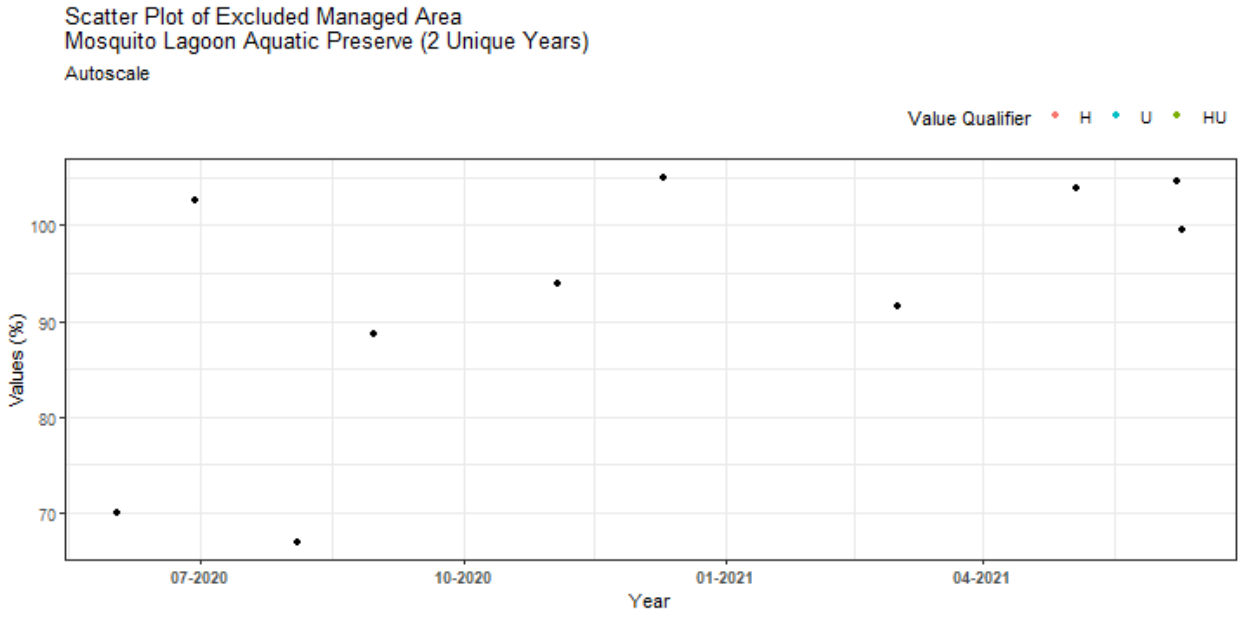
Scatter Plot of Excluded Managed Area
Gasparilla Sound-Charlotte Harbor Aquatic Preserve (7 Unique Years)
Autoscale



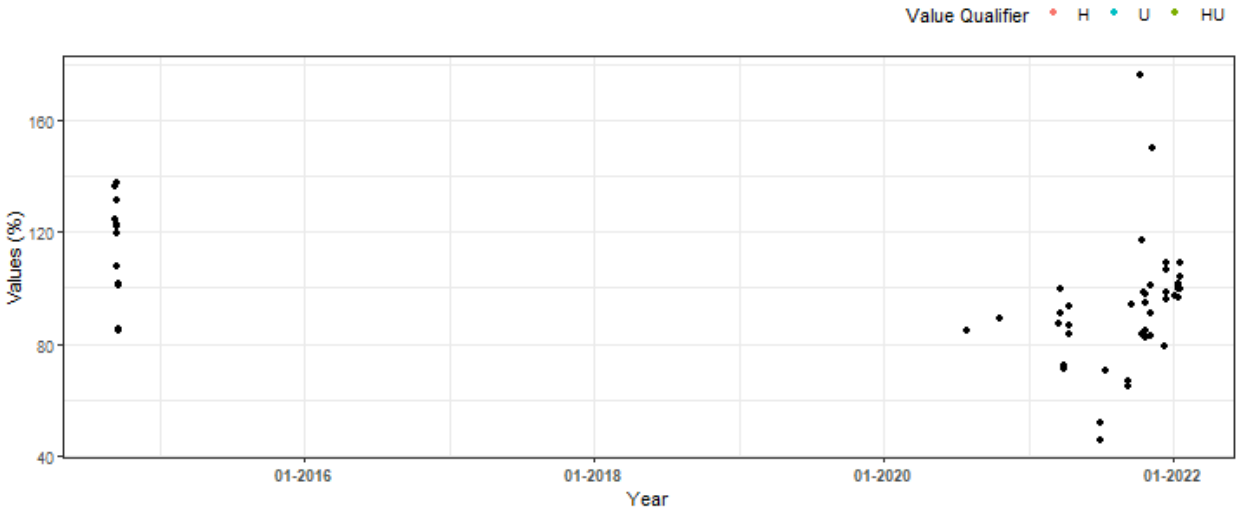
Scatter Plot of Excluded Managed Area
Lignumvitae Key Aquatic Preserve (2 Unique Years)
Autoscale



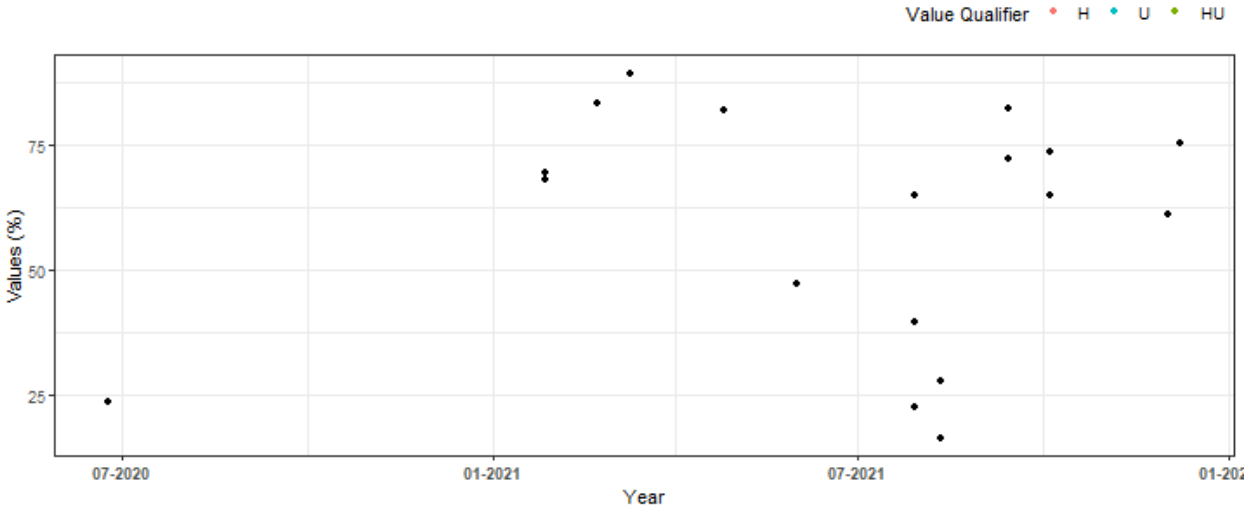

Scatter Plot of Excluded Managed Area
Matlacha Pass Aquatic Preserve (8 Unique Years)
Autoscale
Value Qualifier H U HU
160
120
80
40
0
Values (%)
01-2010
01-2015
01-2020
Year


Scatter Plot of Excluded Managed Area
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

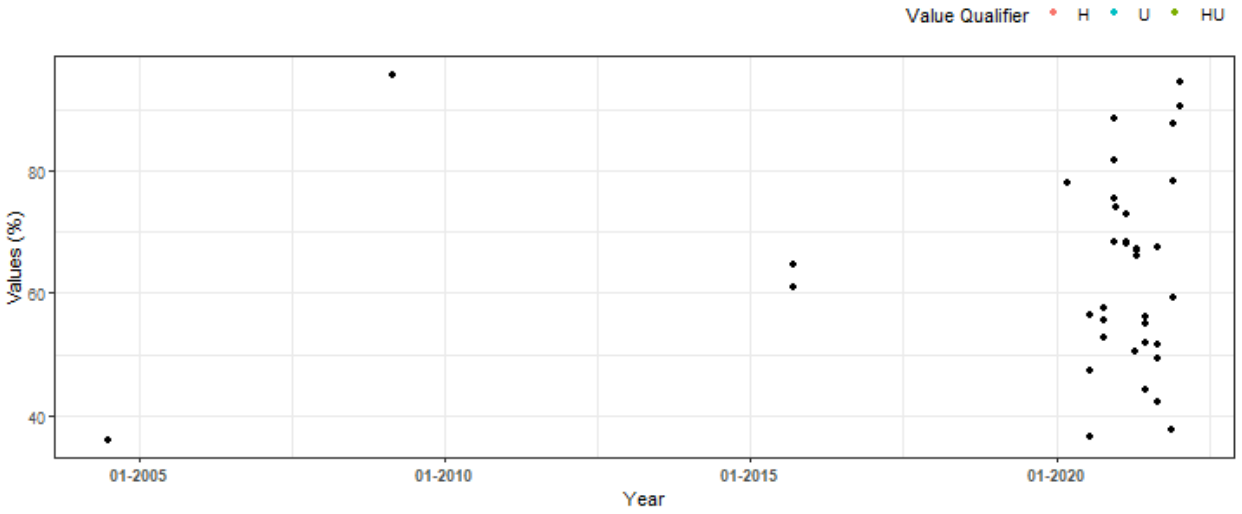
Scatter Plot of Excluded Managed Area
Nature Coast Aquatic Preserve (4 Unique Years)
Autoscale



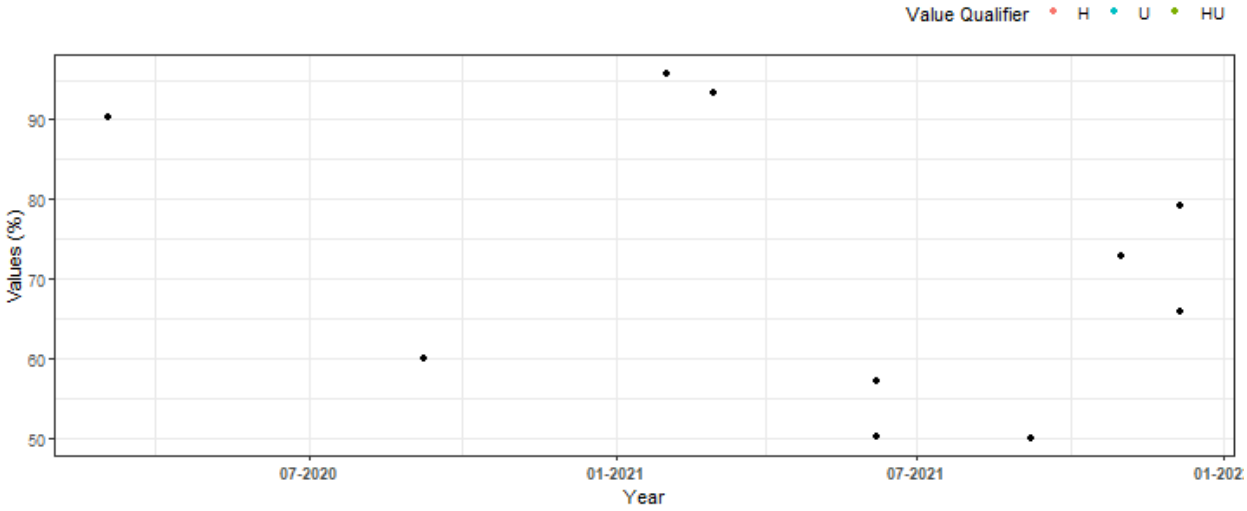
Scatter Plot of Excluded Managed Area
North Fork St. Lucie Aquatic Preserve (2 Unique Years)
Autoscale



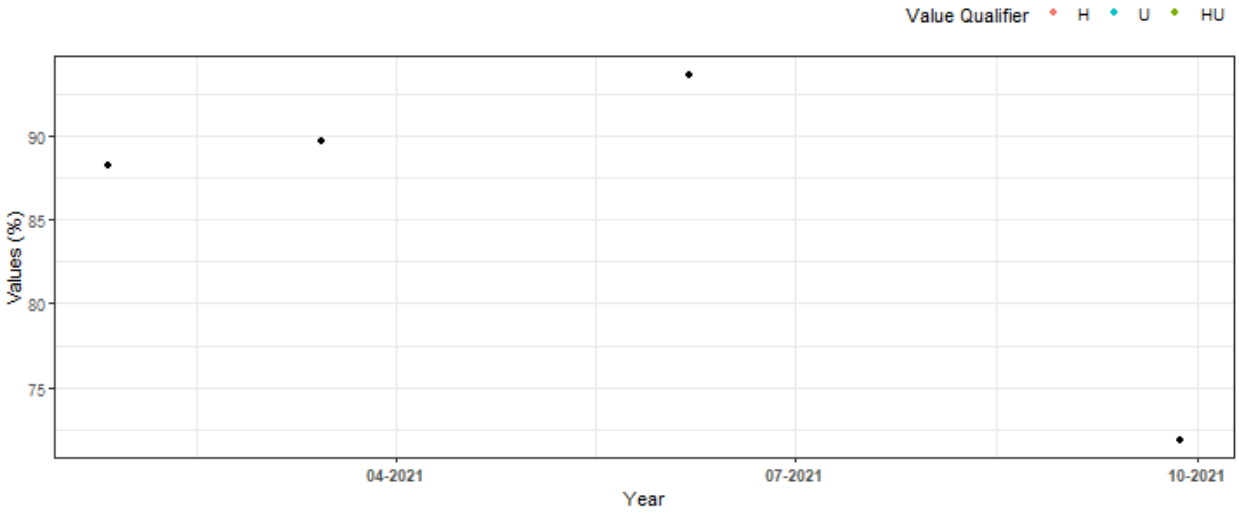
Scatter Plot of Excluded Managed Area
Pellicer Creek Aquatic Preserve (6 Unique Years)
Autoscale



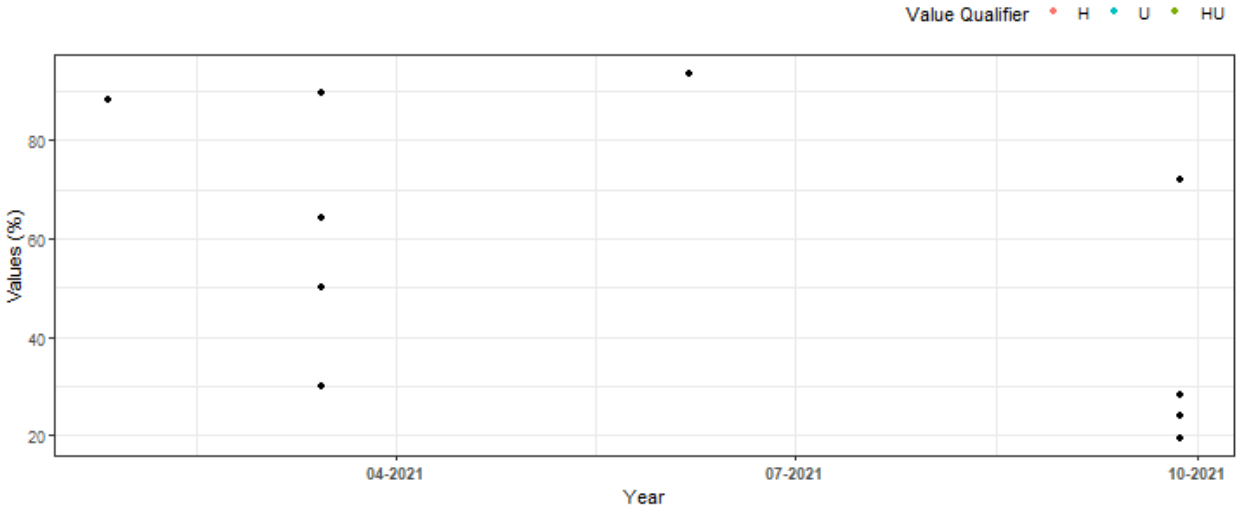
Scatter Plot of Excluded Managed Area
Pine Island Sound Aquatic Preserve (2 Unique Years)
Autoscale



Scatter Plot of Excluded Managed Area
Rookery Bay Aquatic Preserve (1 Unique Years)
Autoscale

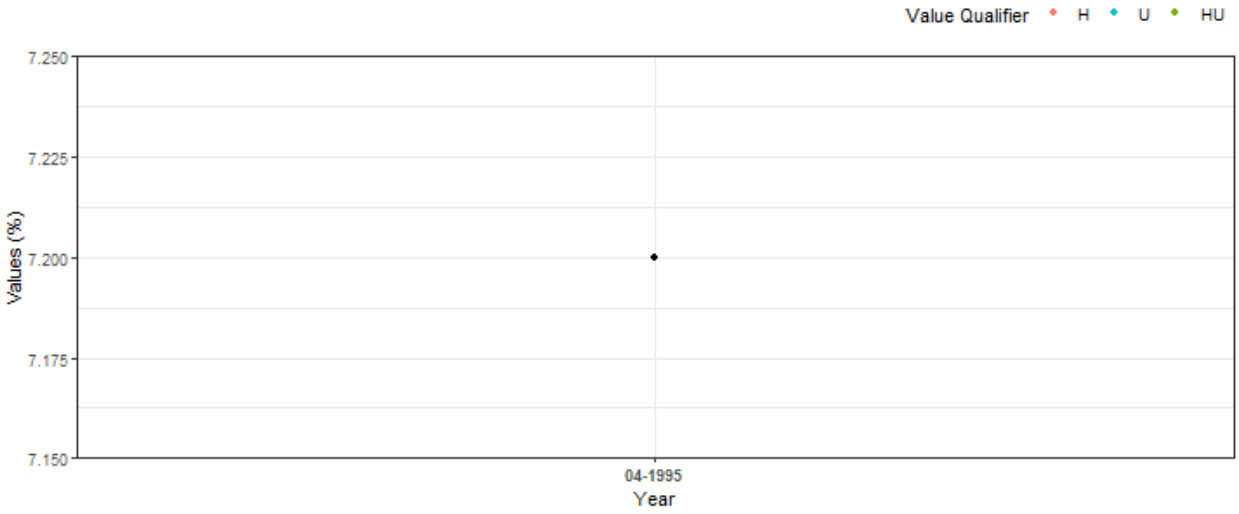


Scatter Plot of Excluded Managed Area
Rookery Bay National Estuarine Research Reserve (1 Unique Years)
Autoscale



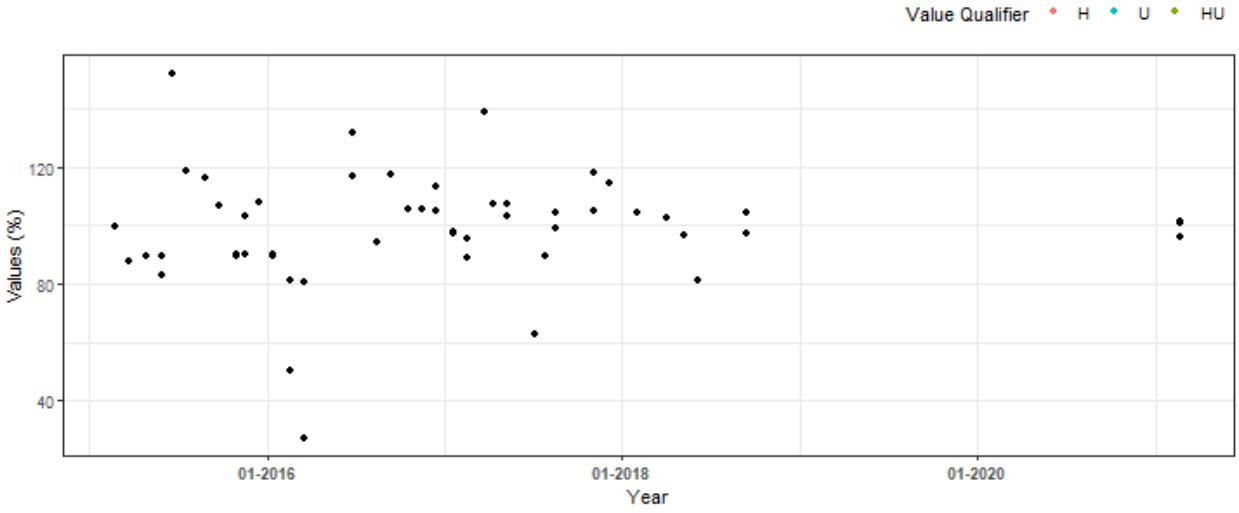
Scatter Plot of Excluded Managed Area
Southeast Florida Coral Reef Ecosystem Conservation Area (1 Unique Years)

Autoscale

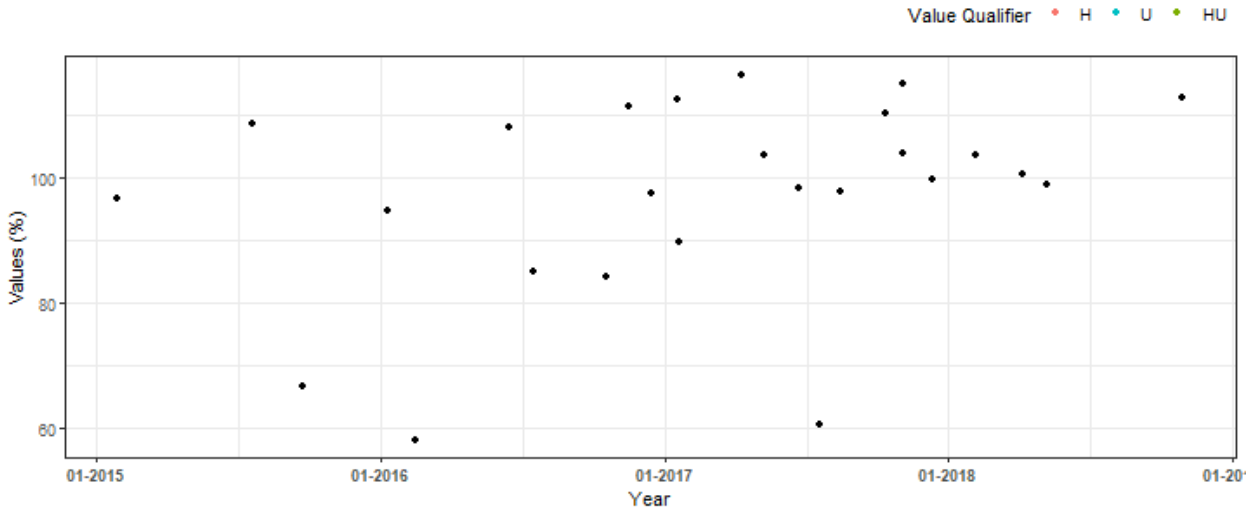


Scatter Plot of Excluded Managed Area
St. Andrews State Park Aquatic Preserve (5 Unique Years)

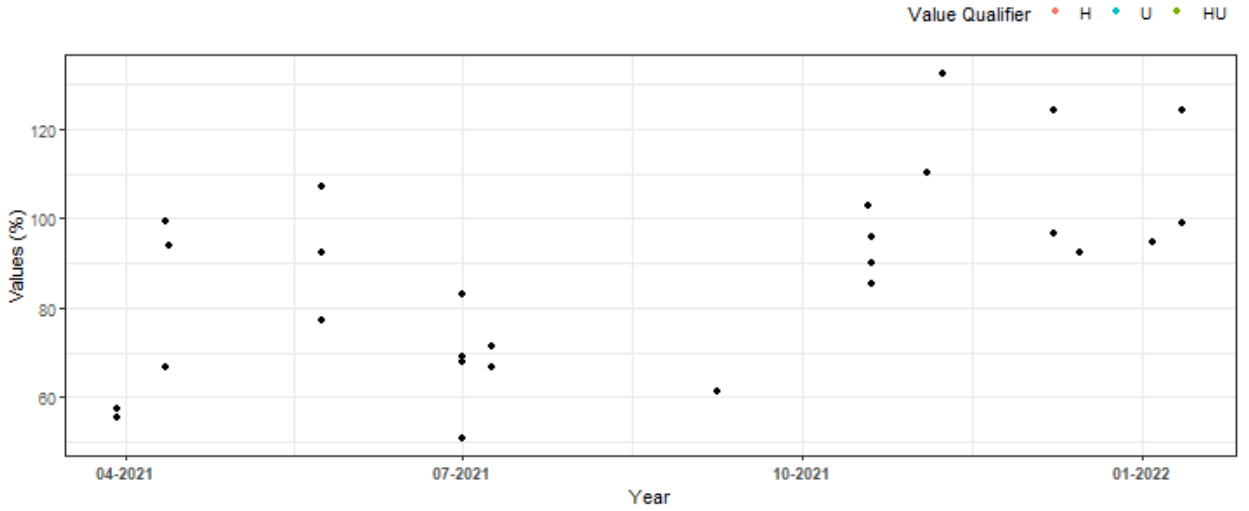
Autoscale

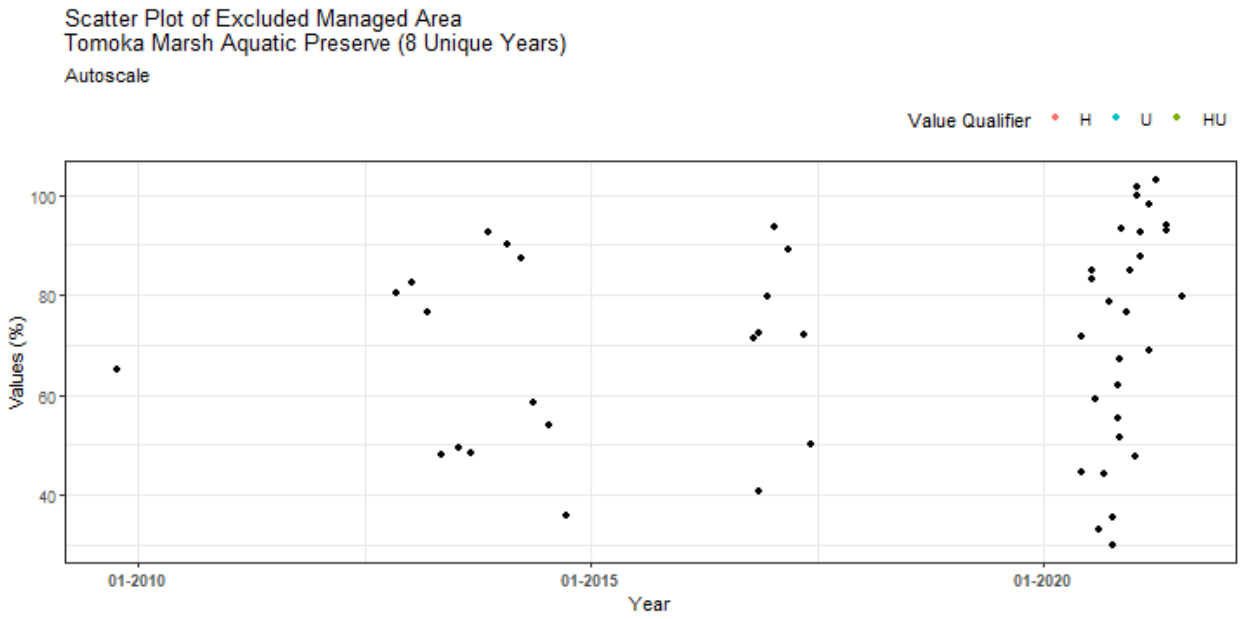


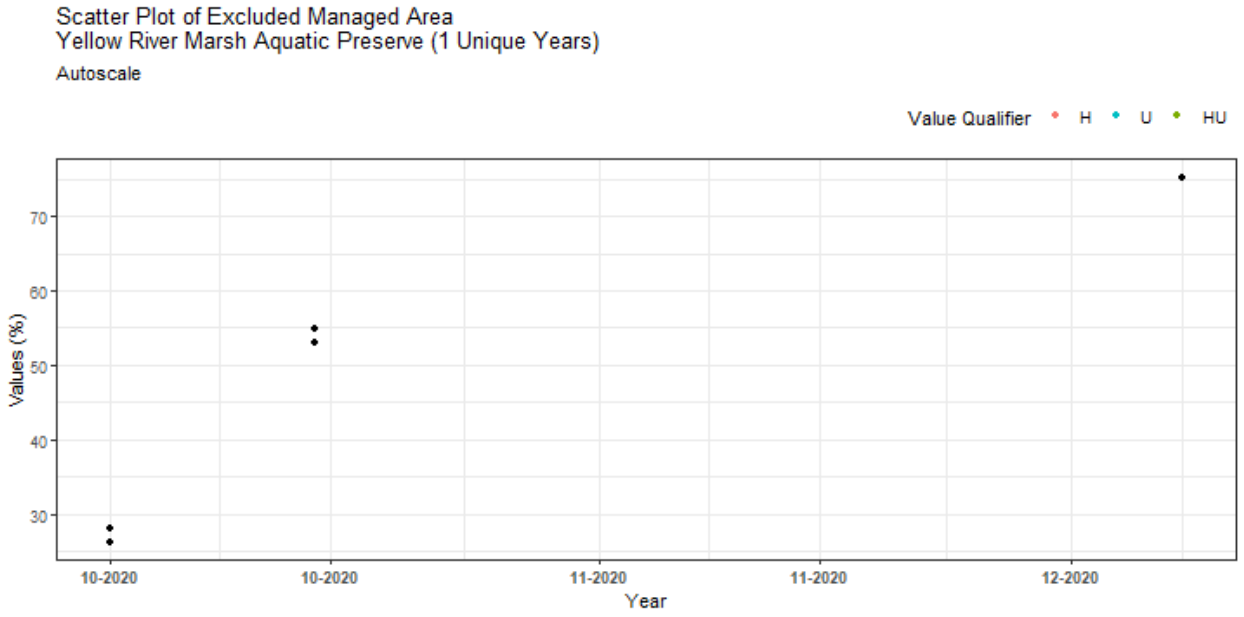
Scatter Plot of Excluded Managed Area
St. Joseph Bay Aquatic Preserve (4 Unique Years)
Autoscale



Scatter Plot of Excluded Managed Area
St. Martins Marsh Aquatic Preserve (2 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 `Use_In_Analysis` 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

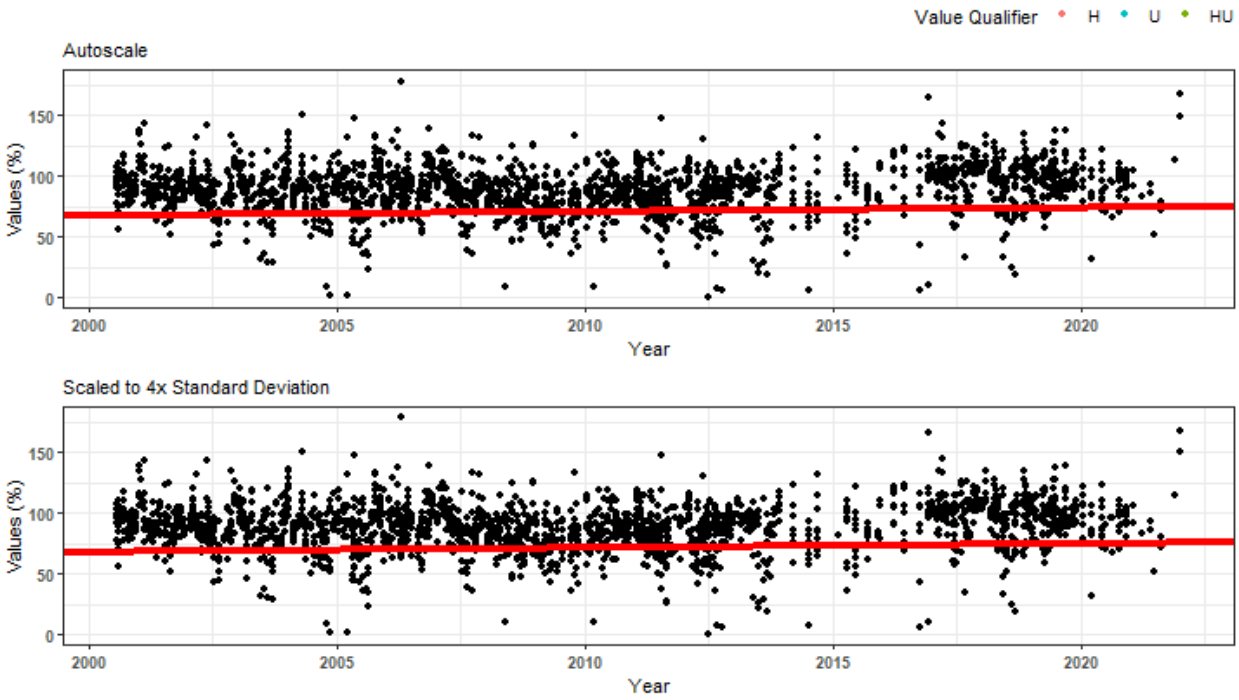
5. Set the plot type as a point plot with the size of the points
6. Add the linear trend
7. Create the title, x-axis, y-axis, and color fill labels
8. Set the y and x limits
9. Make the axis labels bold
10. Plot the arrangement as a set of panels

```
if(n==0){
   print("There are no managed areas that qualify.")
} else {
   for (i in 1:n) {
      plot_data <- data[data$Use_In_Analysis == TRUE &
                                      data$ManagedAreaName == MA_names[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[data$ResultValue <
                                        quantile(data$ResultValue, 0.98)])
      sd_RV <- sd(plot_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

      tau <- KT.Stats$tau[KT.Stats$ManagedAreaName==MA_names[i]]
      s_slope <- KT.Stats$SennSlope[KT.Stats$ManagedAreaName==MA_names[i]]
      s_int <- KT.Stats$SennIntercept[KT.Stats$ManagedAreaName==MA_names[i]]
      trend <- KT.Stats$Trend[KT.Stats$ManagedAreaName==MA_names[i]]
      p <- KT.Stats$p[KT.Stats$ManagedAreaName==MA_names[i]]

      model <- lm(ResultValue ~ DecDate,
                  data = plot_data)
      m_int <- coef(model)[[1]]
      m_slope <- coef(model)[[2]]
      p1 <- ggplot(data = plot_data,
                   aes(x = DecDate, y = ResultValue,
                       color=VQ_Plot)) +
         geom_point(size = 1.5) +
         geom_abline(aes(slope=s_slope, intercept=s_int),
                     color="red", size=1.5) +
         labs(subtitle = "Autoscale",
              x = "Year", y = paste0("Values (", unit, ")"),
              color="Value Qualifier") +
         theme_bw() +
         theme(legend.position = "top", legend.box = "horizontal",
               legend.justification = "right",
               axis.text.x = element_text(face = "bold"),
               axis.text.y = element_text(face="bold")) +
         {if(inc_H==TRUE){
            scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
                                          "HU" = "#7CAE00"), na.value="black")
         } else {
            scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
         }}
```

```
      p2 <- ggplot(data = plot_data,
                   aes(x = DecDate, y = ResultValue,
                       color=VQ_Plot)) +
         geom_point(size = 1.5) +
         geom_abline(aes(slope=s_slope, intercept=s_int),
                     color="red", size=1.5) +
         ylim(min_RV, y_scale) +
         labs(subtitle = "Scaled to 4x Standard Deviation",
              x = "Year", y = paste0("Values (", unit, ")")) +
         theme_bw() +
         theme(legend.position = "none",
               axis.text.x = element_text(face = "bold"),
               axis.text.y = element_text(face="bold")) +
         {if(inc_H==TRUE){
            scale_color_manual(values = c("H"= "#F8766D", "U"= "#00BFC4",
                                          "HU" = "#7CAE00"), na.value="black")
         } else {
            scale_color_manual(values = c("U"= "#00BFC4"), na.value="black")
         }}
      leg <- get_legend(p1)
      KTset <- ggarrange(leg, p1 + theme(legend.position = "none"), p2,
                         ncol = 1, heights = c(0.1, 1, 1))

      p0 <- ggplot() + labs(title = paste0("Data Points with Trendlines for ",
                                           MA_names[i]),
                            subtitle =paste0("Senn Slope = ", s_slope,
                                             ",     Senn Intercept = ", s_int,
                                             "\nTrend = ", trend,
                                             ",     tau = ", tau,
                                             ",     p = ", p,
                                             "\nLinear Trendline: ",
                                             "y = ", m_slope,"x + ",m_int)) +
         theme_bw() + theme(plot.title = element_text(face="bold", hjust = 0.5),
                            panel.border = element_blank(),
                            panel.grid.major = element_blank(),
                            panel.grid.minor = element_blank(),
                            axis.line = element_blank())

      print(ggarrange(p0, KTset, ncol = 1, heights = c(0.15, 1)))
      rm(plot_data)
   }
}
```

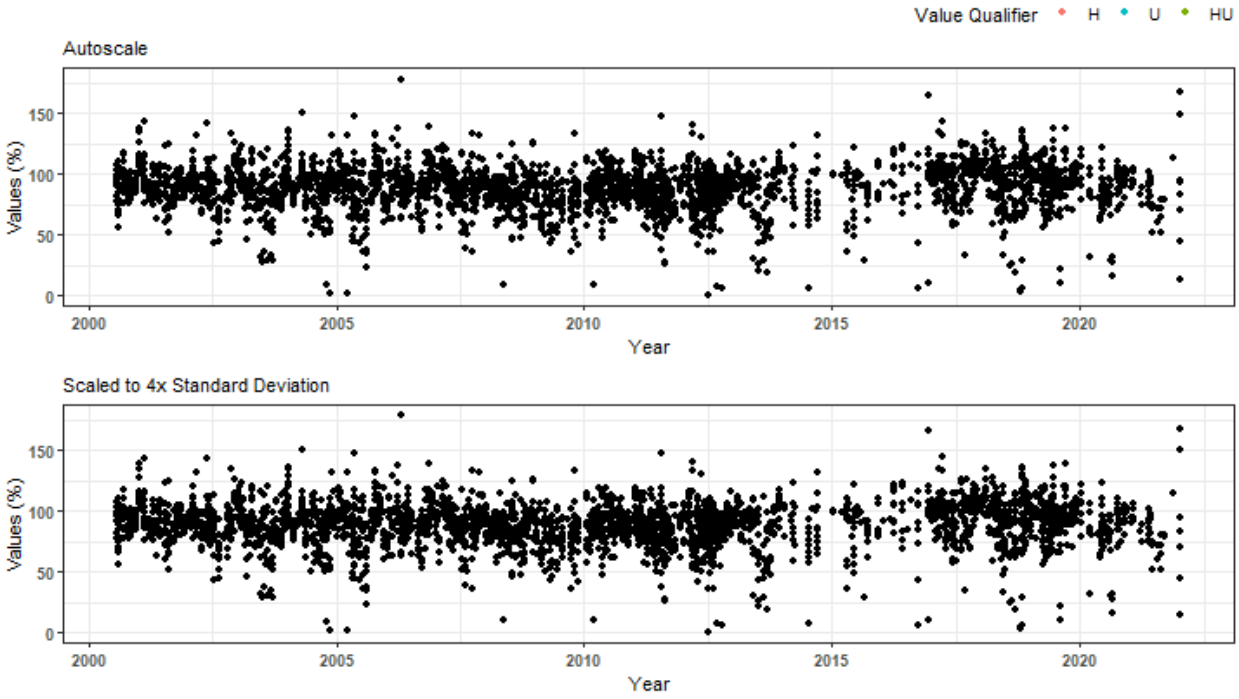
Data Points with Trendlines for Apalachicola Bay Aquatic Preserve

Senn Slope = 0.338461538461538, Senn Intercept = -608.827417582417
Trend = 1, tau = 0.0753, p = 0
Linear Trendline: y = 0.334942290296017x + -584.491662063944



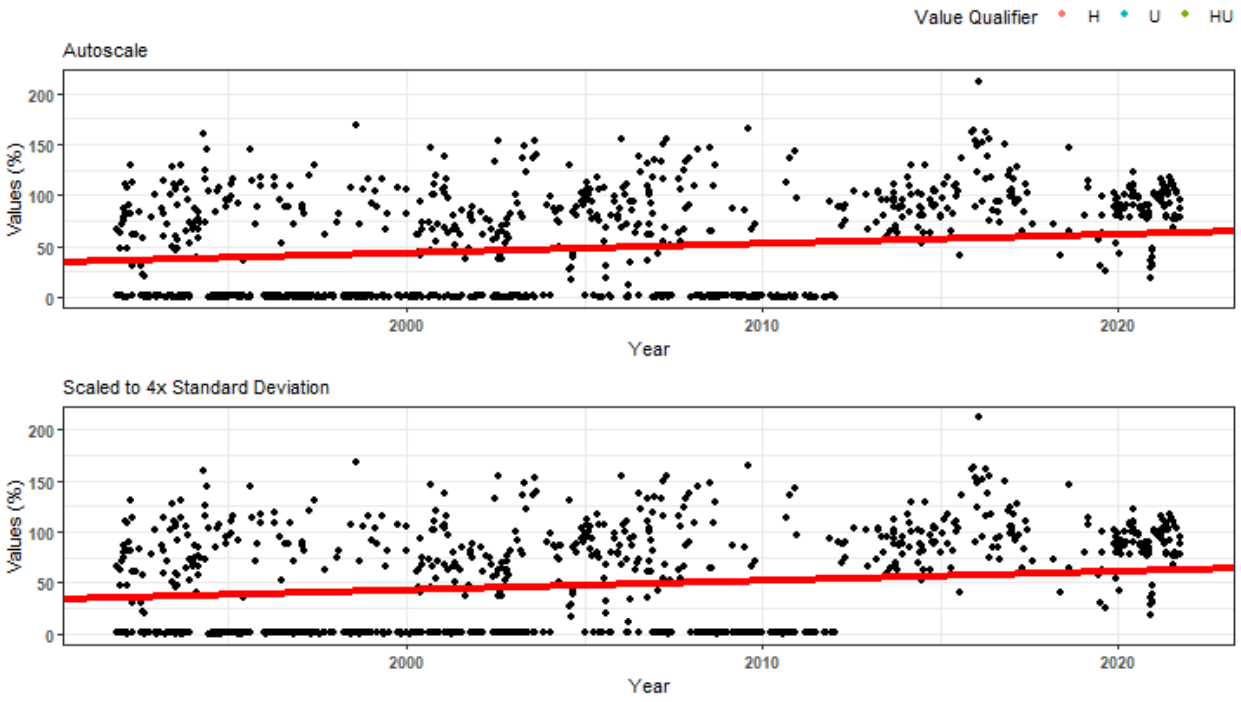
Data Points with Trendlines for Apalachicola National Estuarine Research Reserve

Senn Slope = 0.169230769230769, Senn Intercept = -379.530909090911
Trend = 1, tau = 0.0423, p = 0.0004
Linear Trendline: y = 0.113597282574006x + -139.829923817056



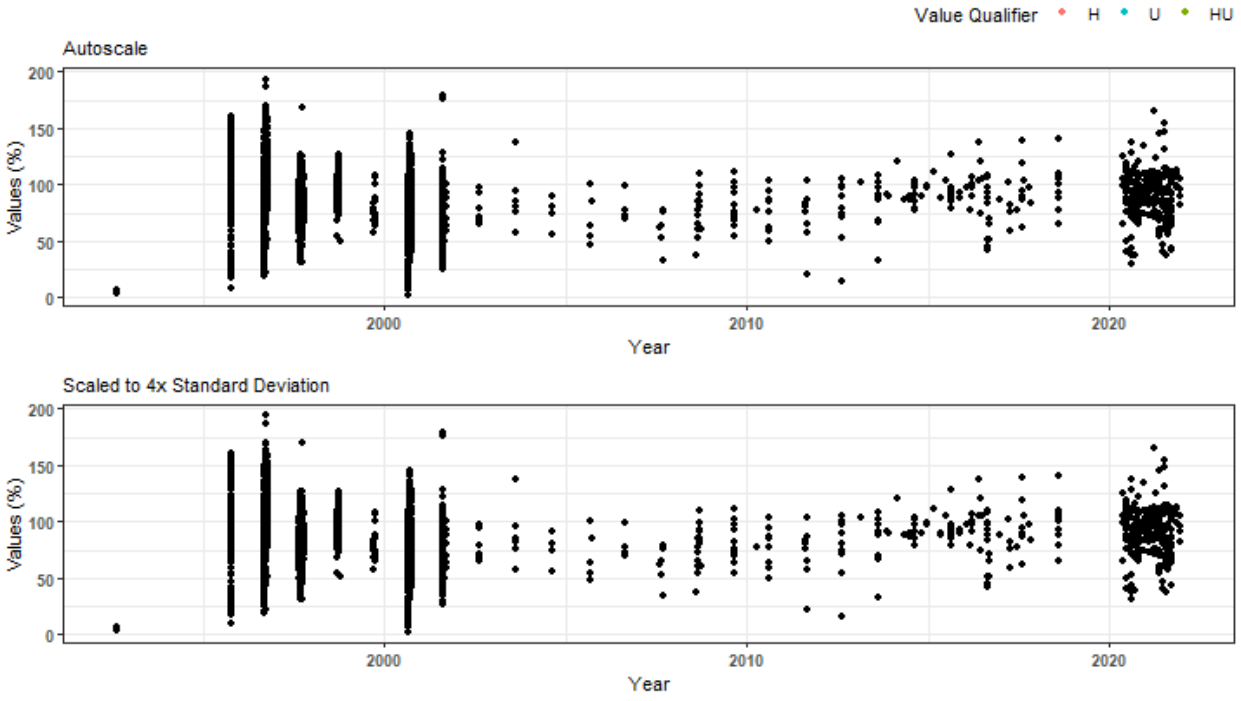
Data Points with Trendlines for Banana River Aquatic Preserve

Senn Slope = 0.912952631578947, Senn Intercept = -1782.36150747863
Trend = 1, tau = 0.2609, p = 0
Linear Trendline: y = 2.27543722657481x + -4512.62692902252



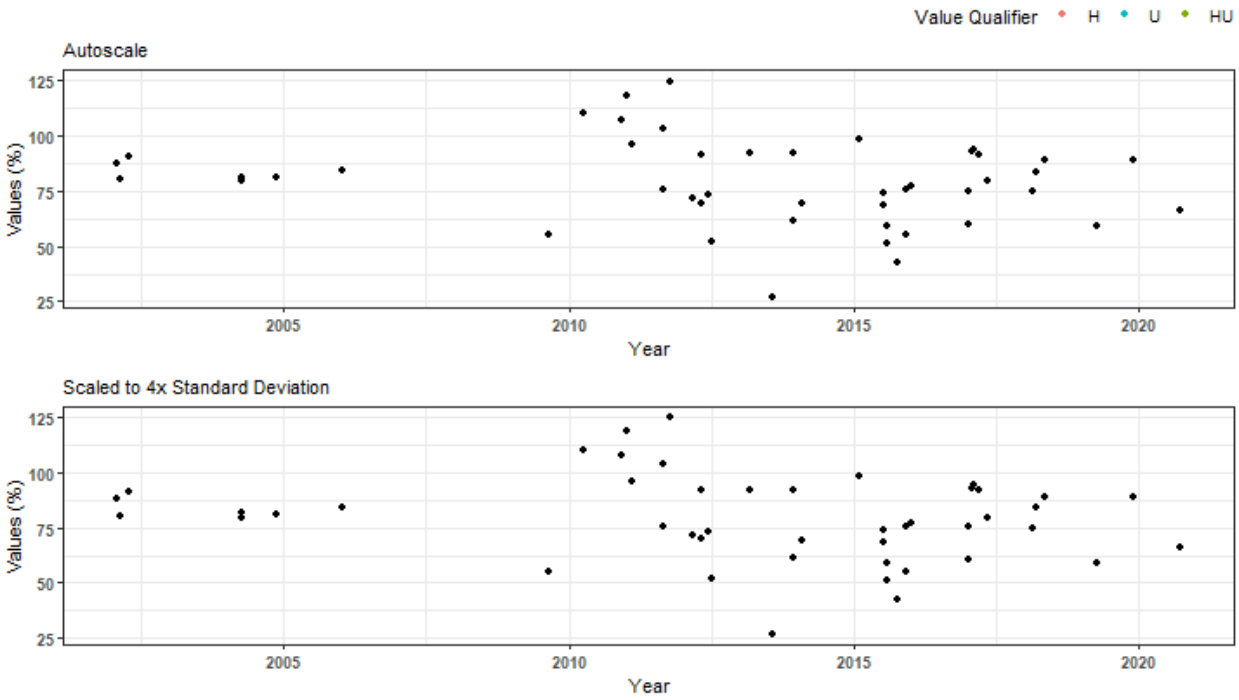
Data Points with Trendlines for Boca Ciega Bay Aquatic Preserve

Senn Slope = -1.26363636363636, Senn Intercept = 1353.60416666667
Trend = -1, tau = -0.0625, p = 0
Linear Trendline: y = -0.0876395628440521x + 261.338417993915



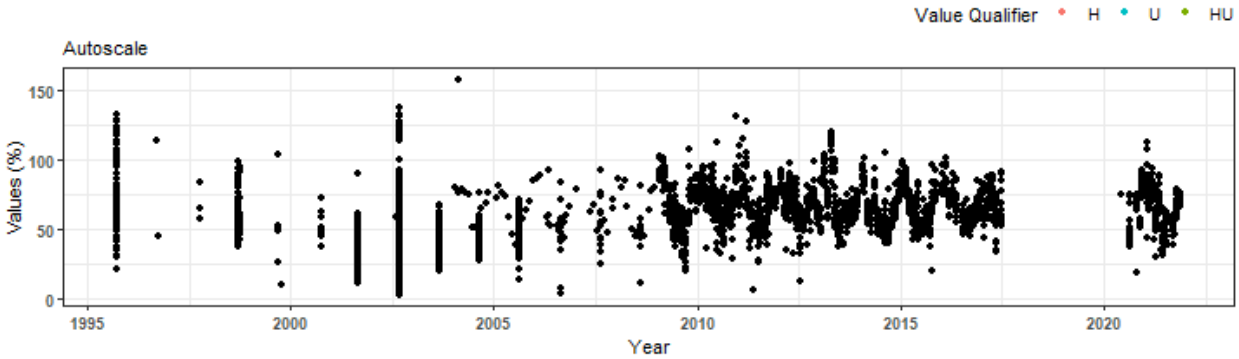
Data Points with Trendlines for Cape Romano-Ten Thousand Islands Aquatic Preserve

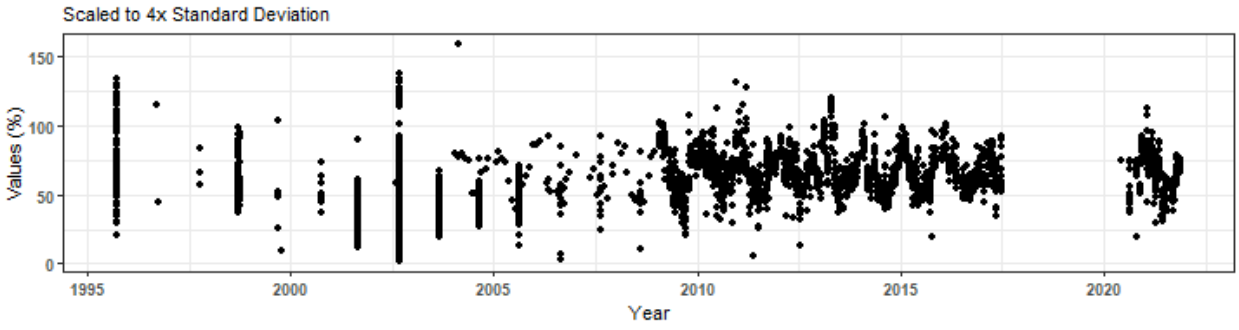
Senn Slope = -0.787597402597403, Senn Intercept = -213.248333333333
Trend = 0, tau = -0.1241, p = 0.2829
Linear Trendline: y = -0.890180957503695x + 1870.82291902259



Data Points with Trendlines for Cockroach Bay Aquatic Preserve

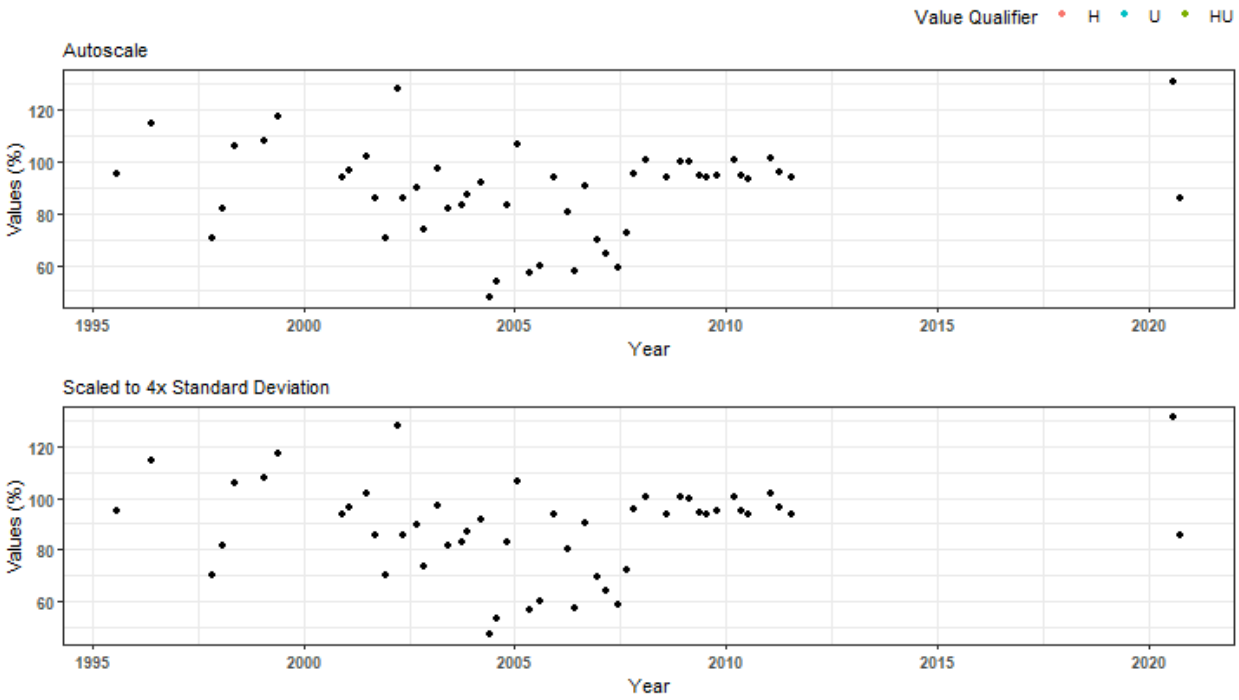
Senn Slope = 0.925, Senn Intercept = 398.385714285717
Trend = 1, tau = 0.0673, p = 0
Linear Trendline: y = 1.08895749472812x + -2130.71958709271





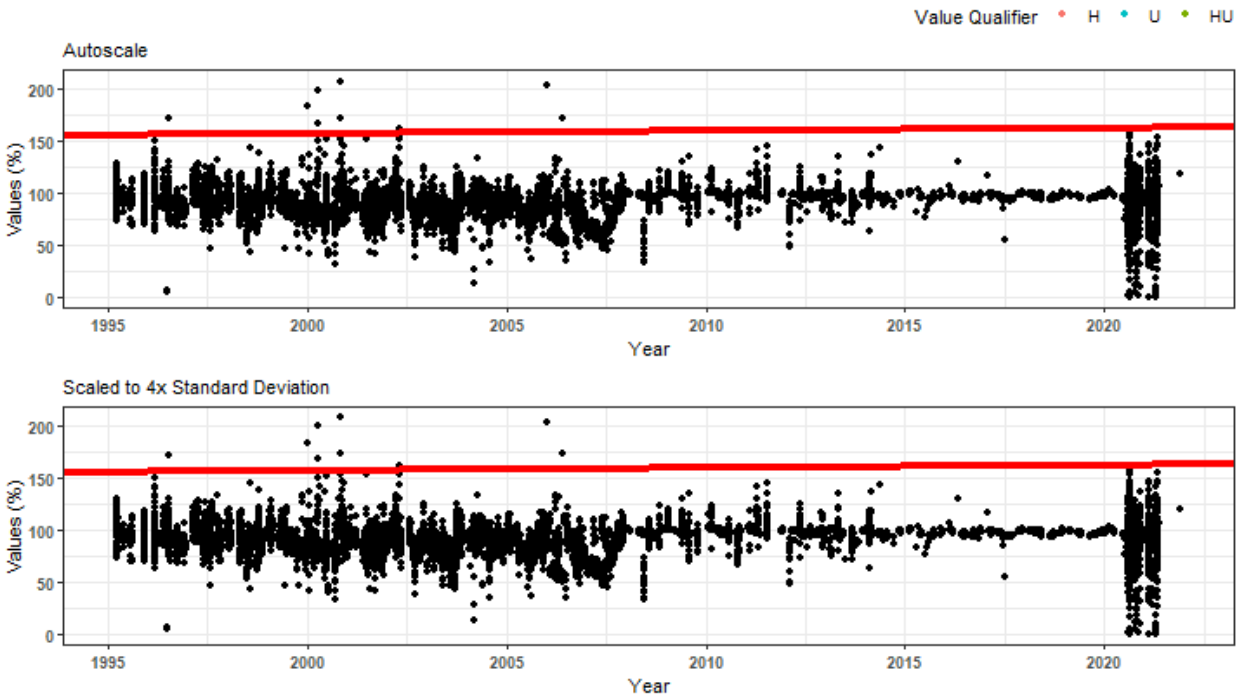
Data Points with Trendlines for Coupon Bight Aquatic Preserve

Senn Slope = 0.313883435416669, Senn Intercept = -865.859238545
Trend = 0, tau = 0.1093, p = 0.6301
Linear Trendline: y = 0.227090061010621x + -366.787359045608



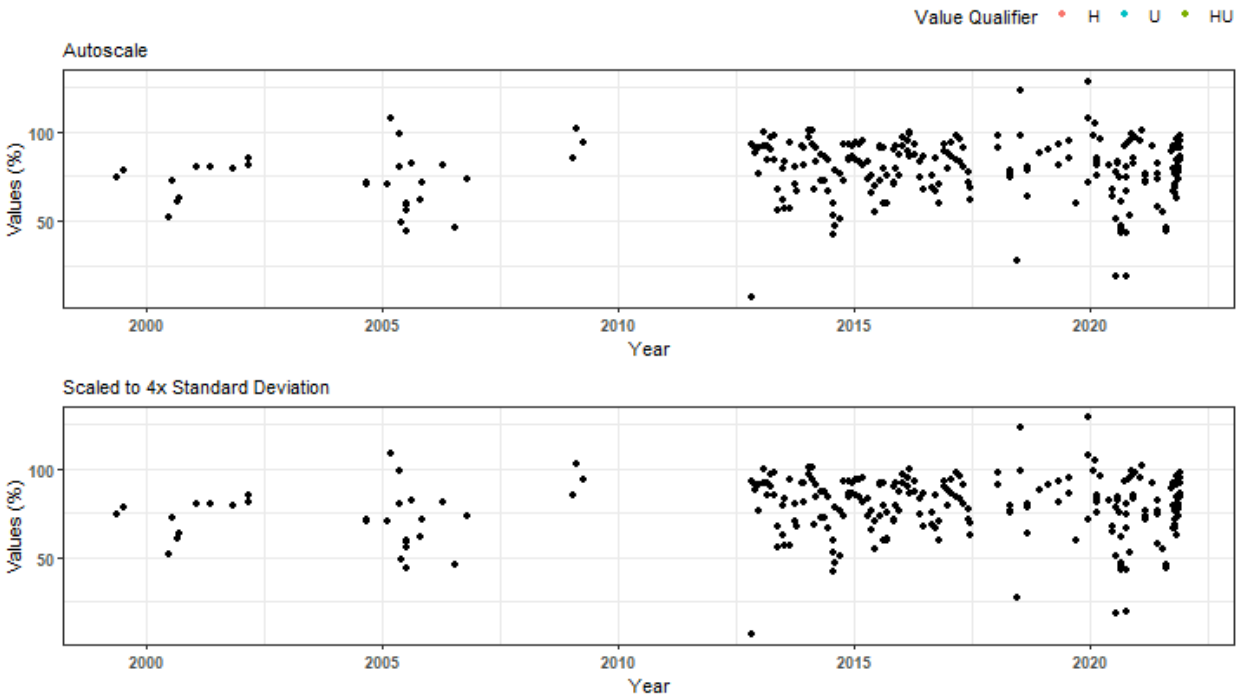
Data Points with Trendlines for Florida Keys National Marine Sanctuary

Senn Slope = 0.26343980375, Senn Intercept = -369.166276817208
Trend = 1, tau = 0.1734, p = 0
Linear Trendline: y = 0.297871364852394x + -506.856590016665



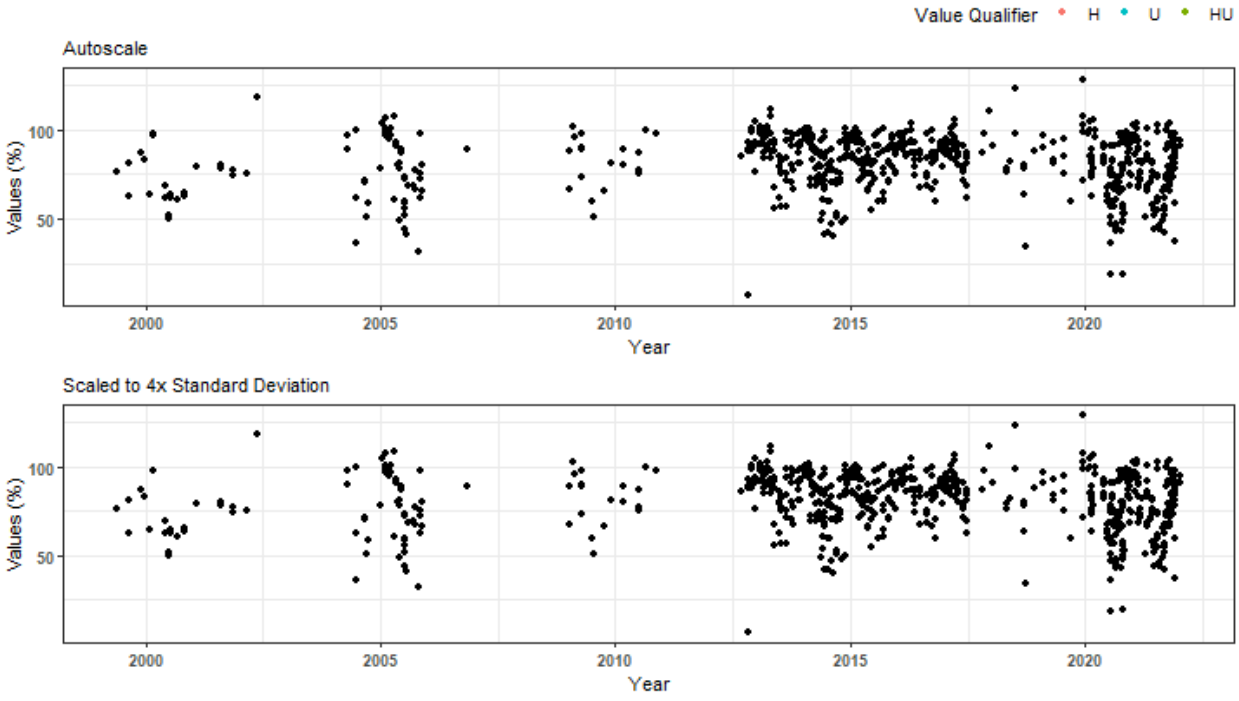
Data Points with Trendlines for Guana River Marsh Aquatic Preserve

Senn Slope = 0.0142857142857135, Senn Intercept = -693.342522321429
Trend = 0, tau = 0.0222, p = 0.9542
Linear Trendline: y = 0.217730122075694x + -360.871735808857



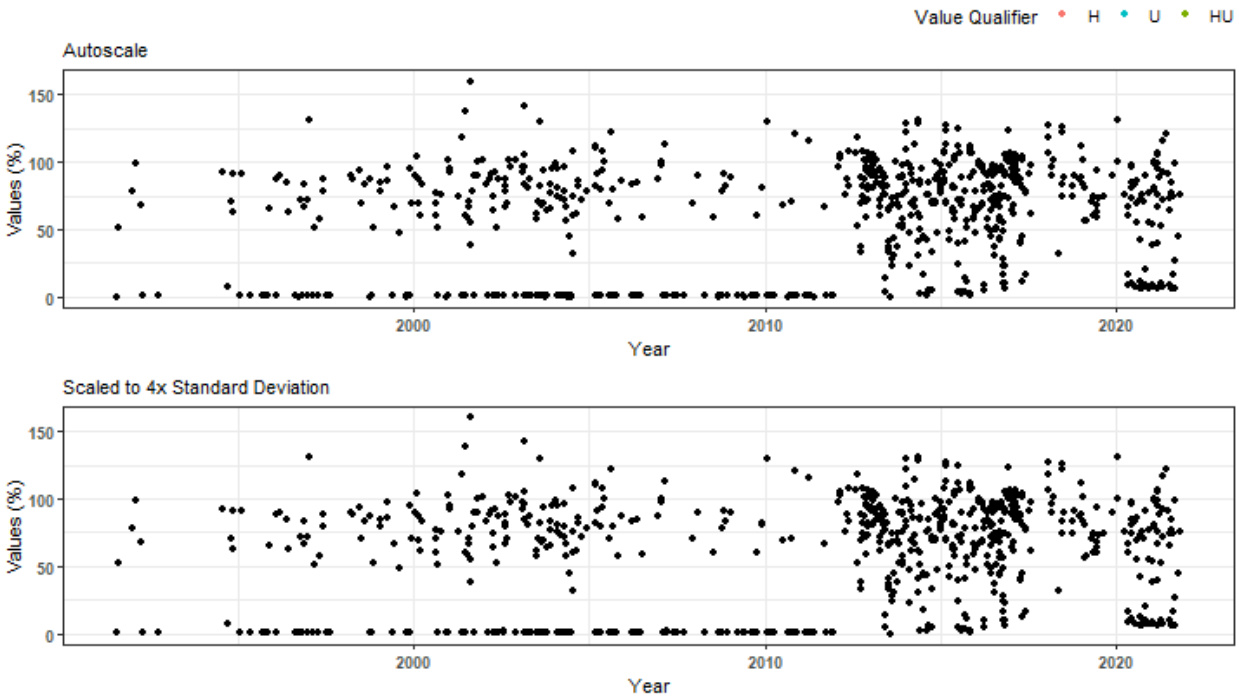
Data Points with Trendlines for Guana Tolomato Matanzas National Estuarine Research Reserve

Senn Slope = -0.327922077922078, Senn Intercept = 559.150000000003
Trend = -1, tau = -0.0837, p = 0.0007
Linear Trendline: y = -0.0201673275608386x + 121.993889200401



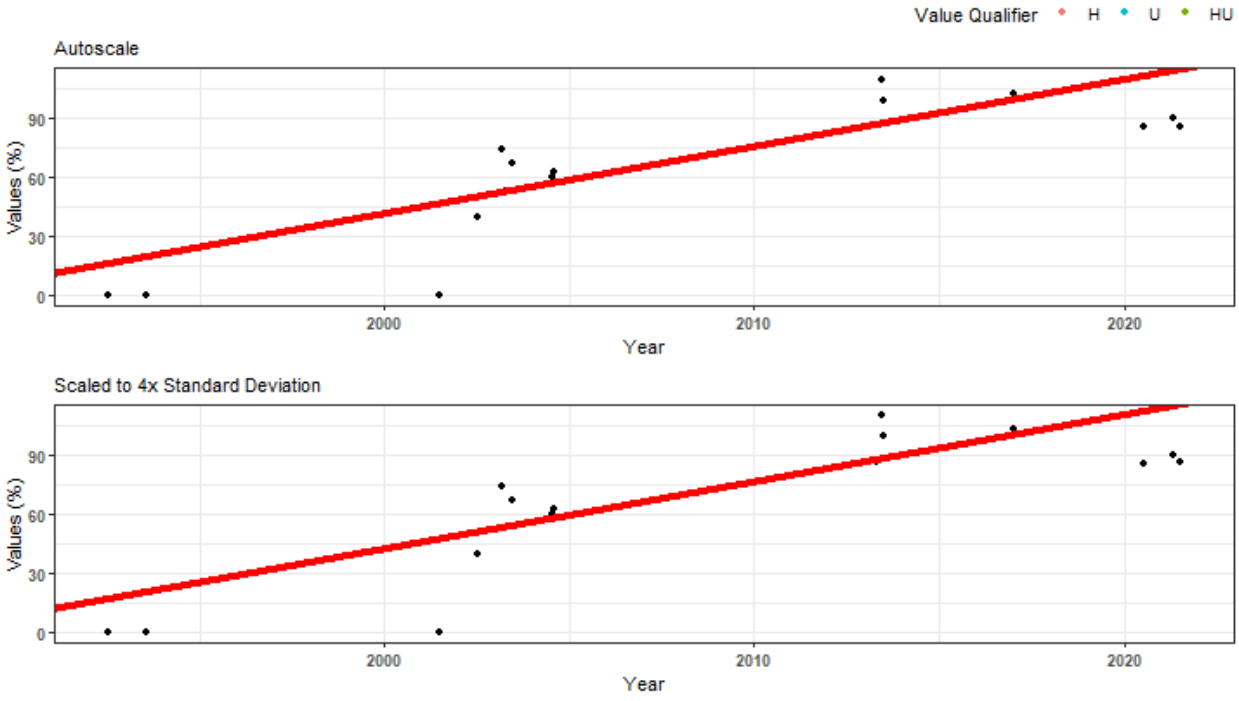
Data Points with Trendlines for Indian River-Malabar to Vero Beach Aquatic Preserve

Senn Slope = 0.457142857142857, Senn Intercept = -676.215389409722
Trend = 1, tau = 0.1205, p = 0
Linear Trendline: y = 0.971576206721849x + -1893.2720052595



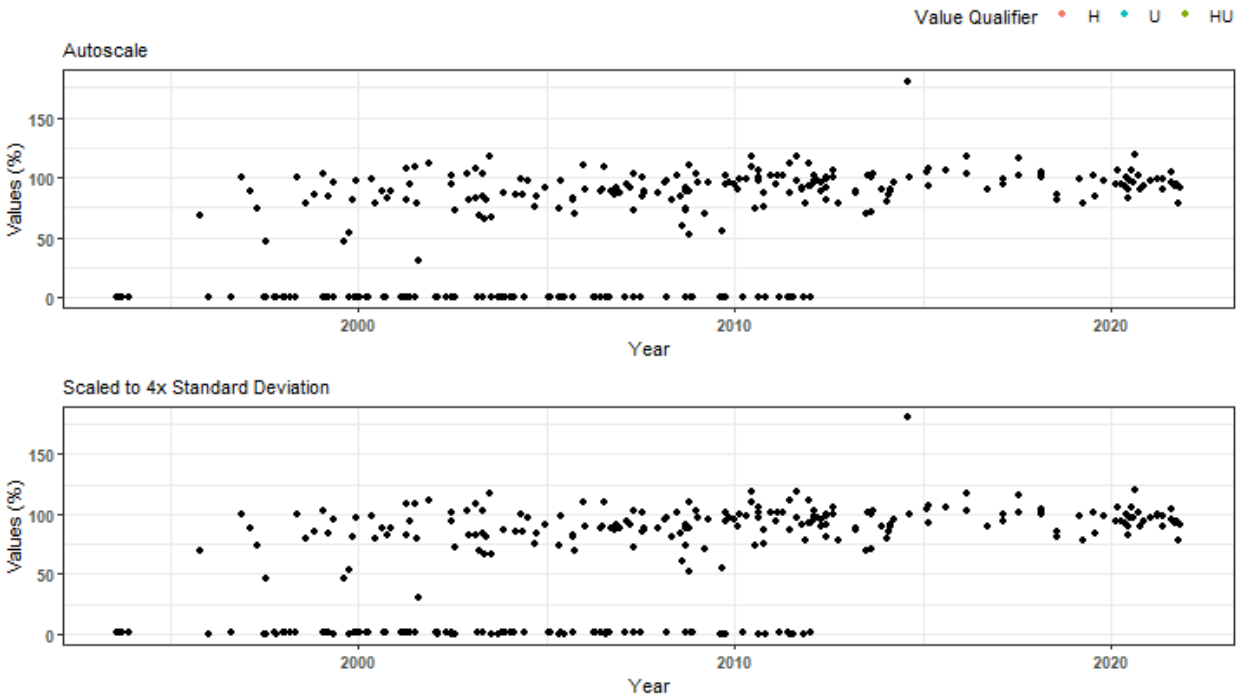
Data Points with Trendlines for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Senn Slope = 3.4, Senn Intercept = -6757.2
Trend = 1, tau = 0.8889, p = 0.0043
Linear Trendline: y = 3.31152395512285x + -6586.13194858155



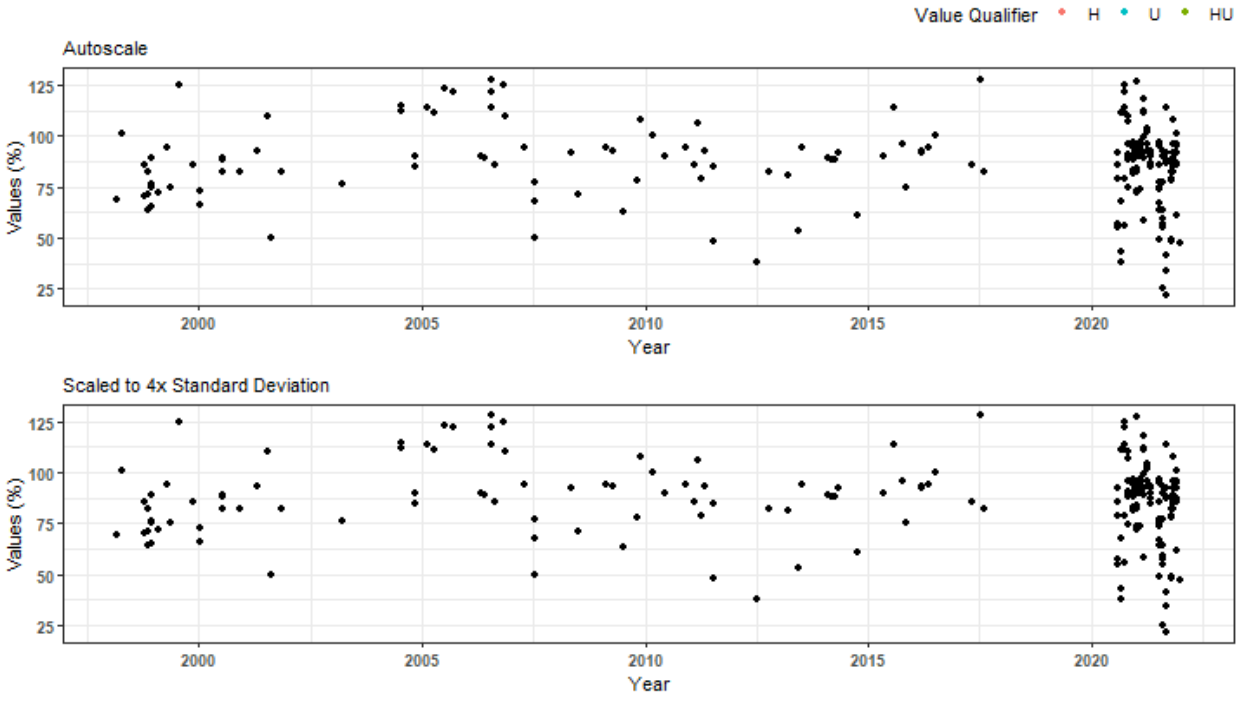
Data Points with Trendlines for Jensen Beach to Jupiter Inlet Aquatic Preserve

Senn Slope = 1.7, Senn Intercept = -2660.90083333333
Trend = 1, tau = 0.3232, p = 0
Linear Trendline: y = 3.20295396184345x + -6368.35047542717



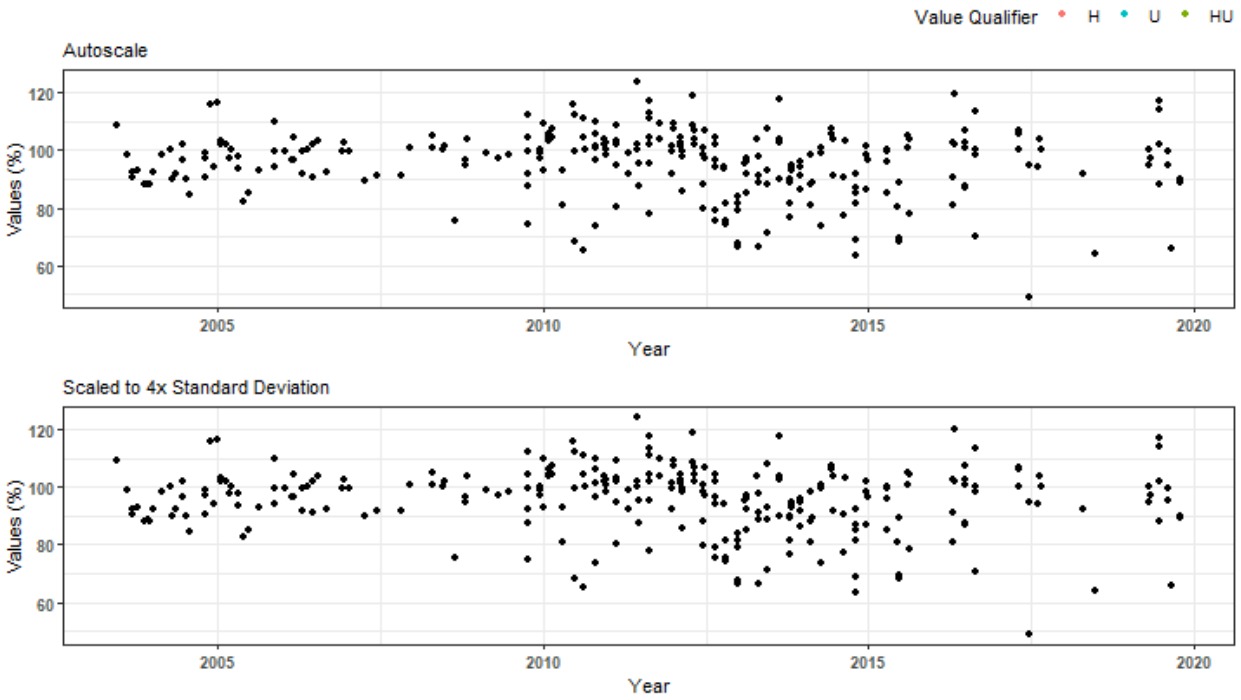
Data Points with Trendlines for Lemon Bay Aquatic Preserve

Senn Slope = 0, Senn Intercept = 0.590909090909093
Trend = 0, tau = 0.0188, p = 0.7961
Linear Trendline: y = -0.110823237233144x + 309.966444539463



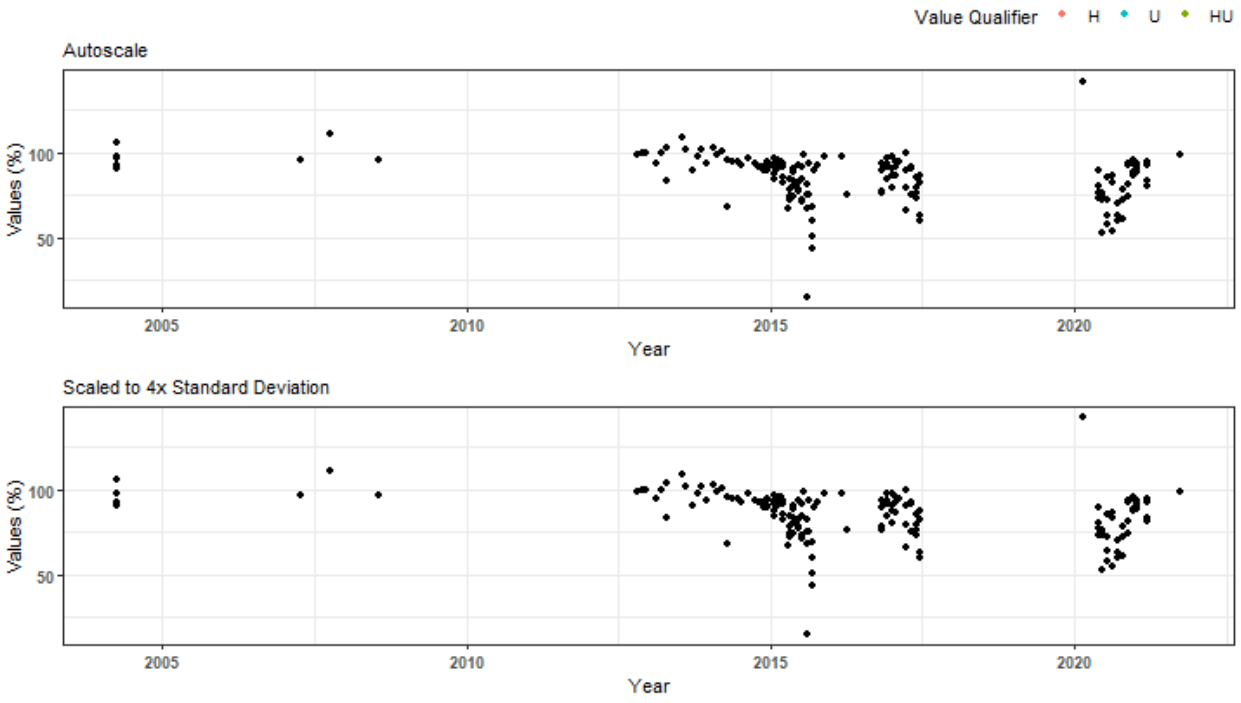
Data Points with Trendlines for Loxahatchee River-Lake Worth Creek Aquatic Preserve

Senn Slope = -0.555555555555556, Senn Intercept = -259.014583333335
Trend = -1, tau = -0.1176, p = 0.0027
Linear Trendline: y = -0.370170675154484x + 840.179344057597



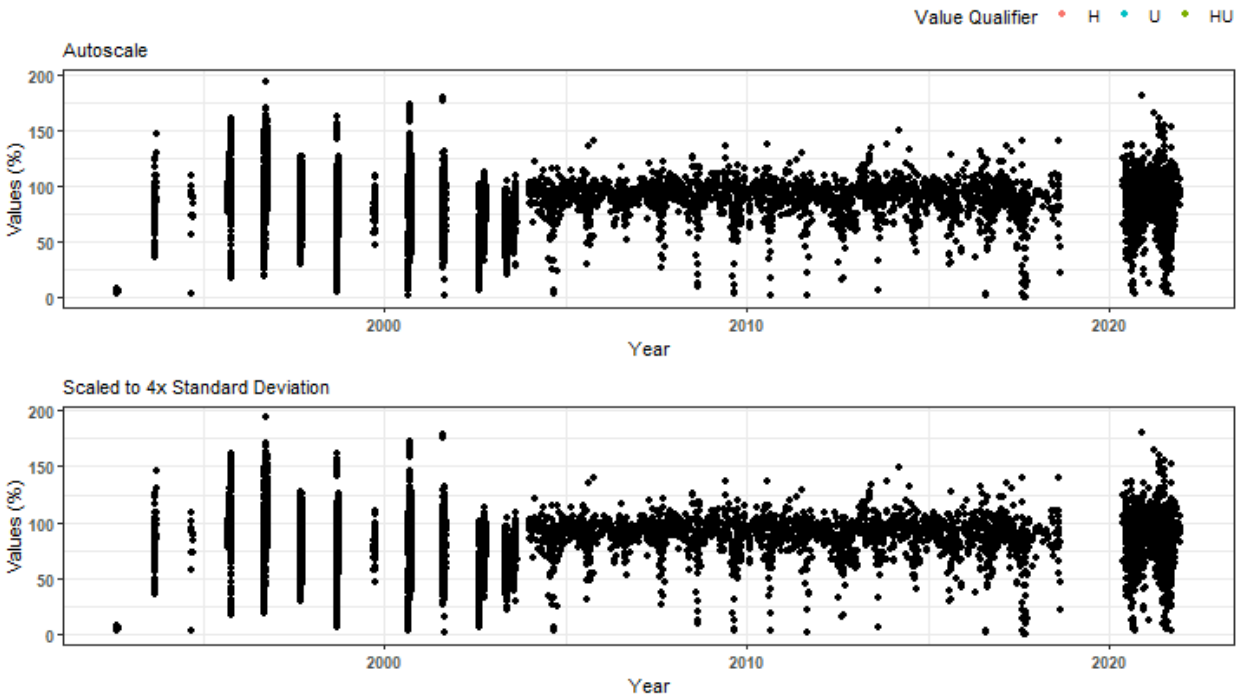
Data Points with Trendlines for Nassau River-St. Johns River Marshes Aquatic Preserve

Senn Slope = -1.24, Senn Intercept = 3001.8625
Trend = -1, tau = -0.2928, p = 0
Linear Trendline: y = -1.25681274345237x + 2619.39222599161



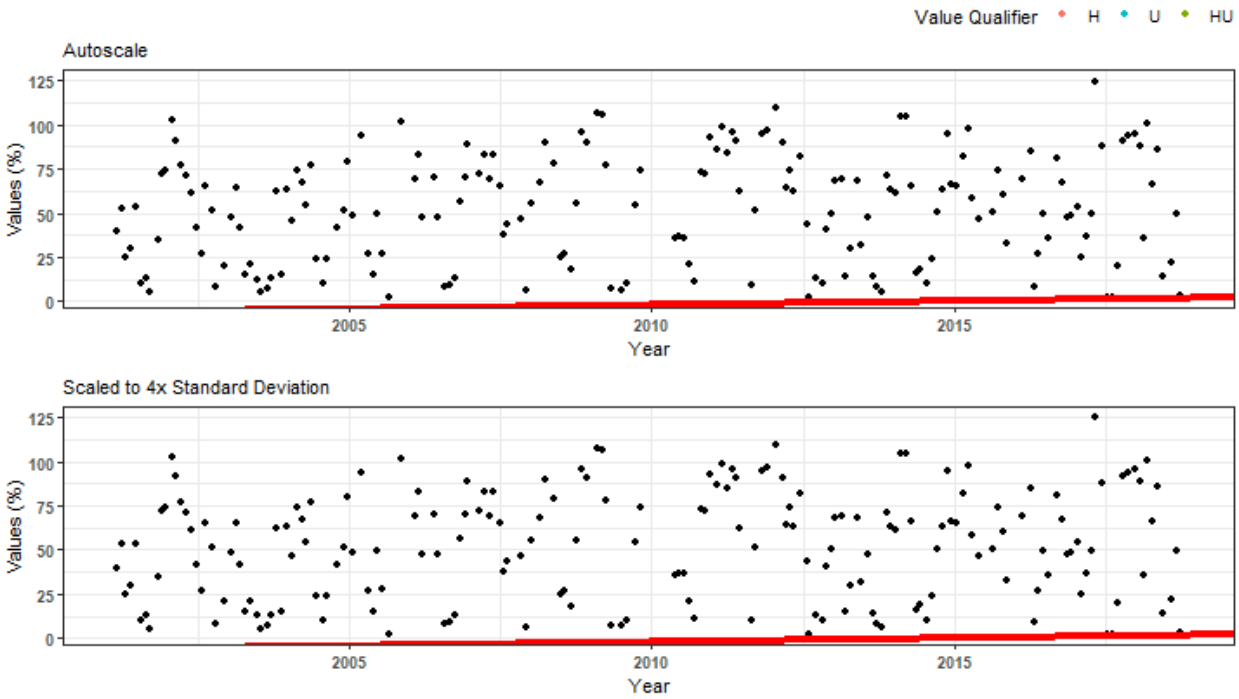
Data Points with Trendlines for Pinellas County Aquatic Preserve

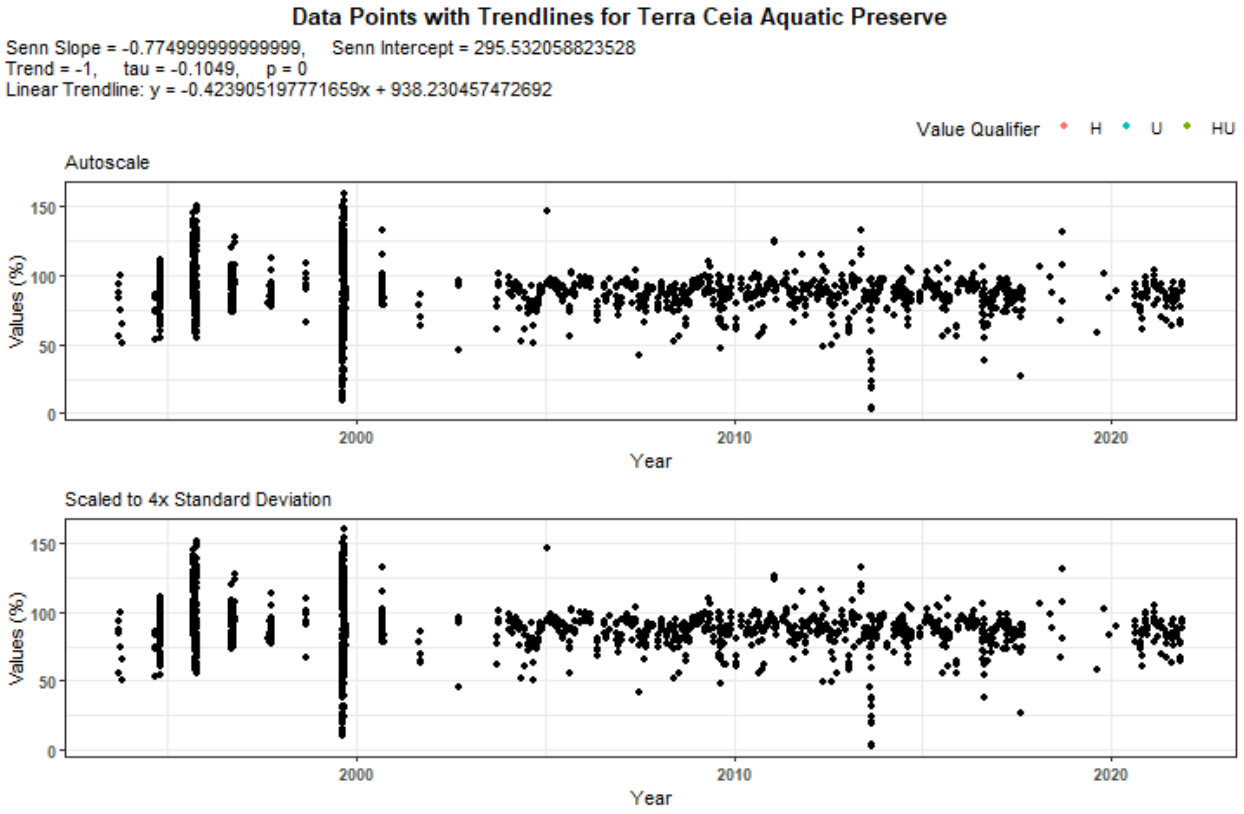
Senn Slope = -0.554166666666667, Senn Intercept = 273.1945
Trend = -1, tau = -0.0166, p = 0
Linear Trendline: y = 0.0469446003690803x + -11.6059162450431



Data Points with Trendlines for Rocky Bayou State Park Aquatic Preserve

Senn Slope = 0.45, Senn Intercept = -906
Trend = 0, tau = 0.0633, p = 0.1722
Linear Trendline: y = 0.6979011656289x + -1350.04330842177





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 `Use_In_Analysis` 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) {
      year_lower <- min(data$Year[data$Use_In_Analysis == TRUE &
                                  data$ManagedAreaName == MA_names[i]])
      year_upper <- max(data$Year[data$Use_In_Analysis == TRUE &
                                  data$ManagedAreaName == MA_names[i]])
      min_RV <- min(data$ResultValue[data$Use_In_Analysis == TRUE &
                                     data$ManagedAreaName == MA_names[i]])
      mn_RV <- mean(data$ResultValue[data$Use_In_Analysis == TRUE &
                                     data$ManagedAreaName == MA_names[i] &
                                     data$ResultValue <
                                     quantile(data$ResultValue, 0.98)])
      sd_RV <- sd(data$ResultValue[data$Use_In_Analysis == TRUE &
                                   data$ManagedAreaName == MA_names[i] &
                                   data$ResultValue <
                                   quantile(data$ResultValue, 0.98)])
      x_scale <- ifelse(year_upper - year_lower > 30, 10, 5)
      y_scale <- mn_RV + 4 * sd_RV

      ##Year plots
      p1 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                               data$ManagedAreaName == MA_names[i], ],
                   aes(x = Year, y = ResultValue, group = Year)) +
         geom_boxplot(outlier.size = 0.5) +
         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))) +
         theme_bw() +
         theme(axis.text.x = element_text(face = "bold"),
               axis.text.y = element_text(face = "bold"))

      p2 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                               data$ManagedAreaName == MA_names[i], ],
                   aes(x = Year, y = ResultValue, group = Year)) +
         geom_boxplot(outlier.size = 0.5) +
         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))) +
         theme_bw() +
         theme(axis.text.x = element_text(face = "bold"),
               axis.text.y = element_text(face = "bold"))

      p3 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                               data$ManagedAreaName == MA_names[i] &
                               data$Year>=year_upper-10, ],
                   aes(x = Year, y = ResultValue, group = Year)) +
```

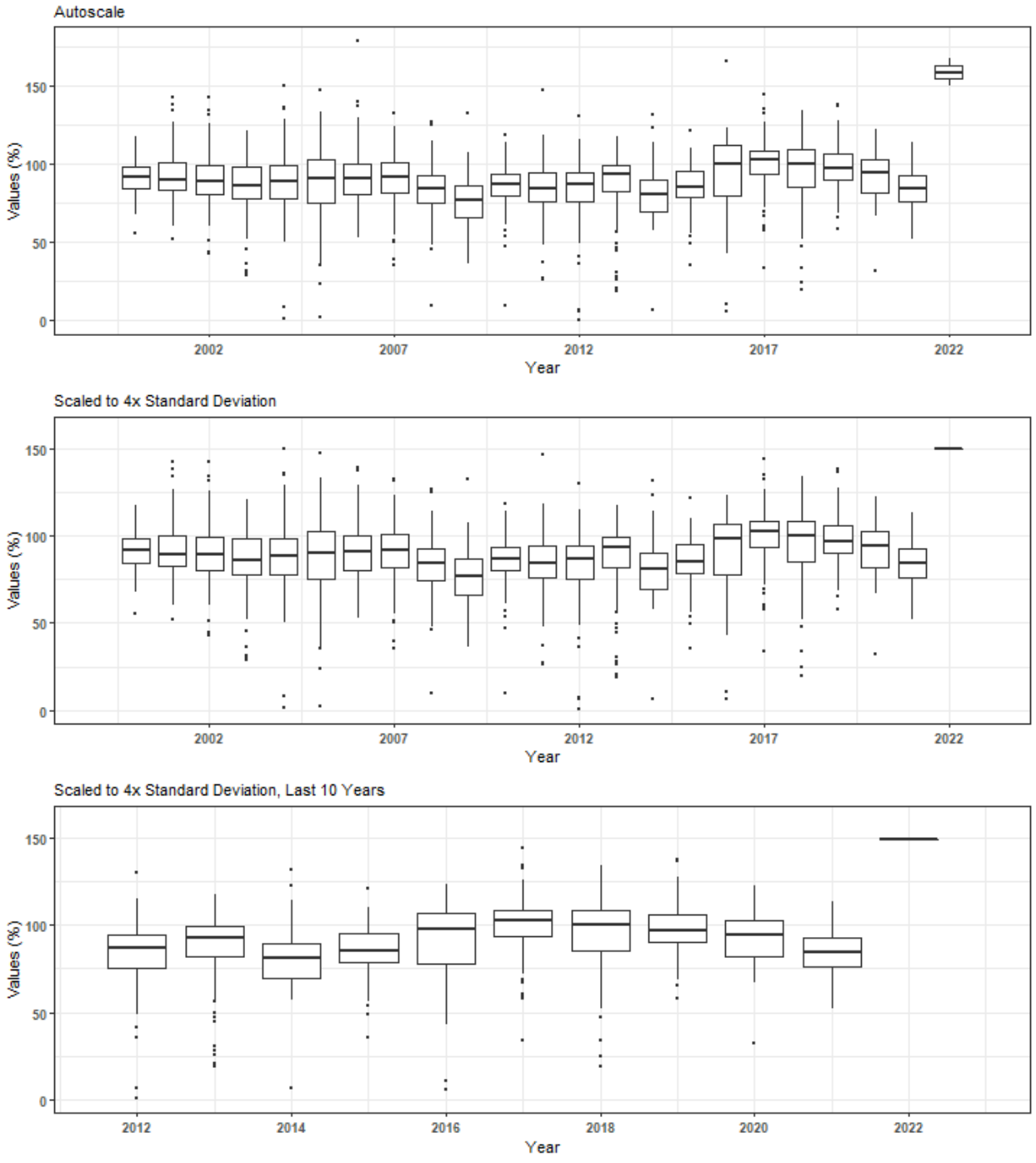
```
    geom_boxplot(outlier.size = 0.5) +
    labs(subtitle = "Scaled to 4x Standard Deviation, Last 10 Years",
         x = "Year", y = paste0("Values (", unit, ")")) +
    ylim(min_RV, y_scale) +
    scale_x_continuous(limits = c(year_upper - 10.5, year_upper + 1),
                       breaks = rev(seq(year_upper, year_upper - 10,-2))) +
    theme_bw() +
    theme(axis.text.x = element_text(face = "bold"),
          axis.text.y = element_text(face = "bold"))

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

p0 <- ggplot() + labs(title = paste0("Summary Box Plots for ",
                                     MA_names[i]), subtitle = "By Year") +
    theme_bw() + theme(plot.title = element_text(face="bold", hjust = 0.5),
                       panel.border = element_blank(),
                       panel.grid.major = element_blank(),
                       panel.grid.minor = element_blank(), axis.line = element_blank())


## Year & Month Plots
p4 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                           data$ManagedAreaName == MA_names[i], ],
             aes(x = YearMonthDec, y = ResultValue,
                 group = YearMonth, color = as.factor(Month))) +
    geom_boxplot(outlier.size = 0.5) +
    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))) +
    theme_bw() +
    theme(legend.position = "none",
          axis.text.x = element_text(face = "bold"),
          axis.text.y = element_text(face = "bold"))

p5 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                           data$ManagedAreaName == MA_names[i], ],
             aes(x = YearMonthDec, y = ResultValue,
                 group = YearMonth, color = as.factor(Month))) +
    geom_boxplot(outlier.size = 0.5) +
    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))) +
    theme_bw() +
    theme(legend.position = "top", legend.box = "horizontal",
          axis.text.x = element_text(face = "bold"),
          axis.text.y = element_text(face = "bold")) +
    guides(color = guide_legend(nrow = 1))
```

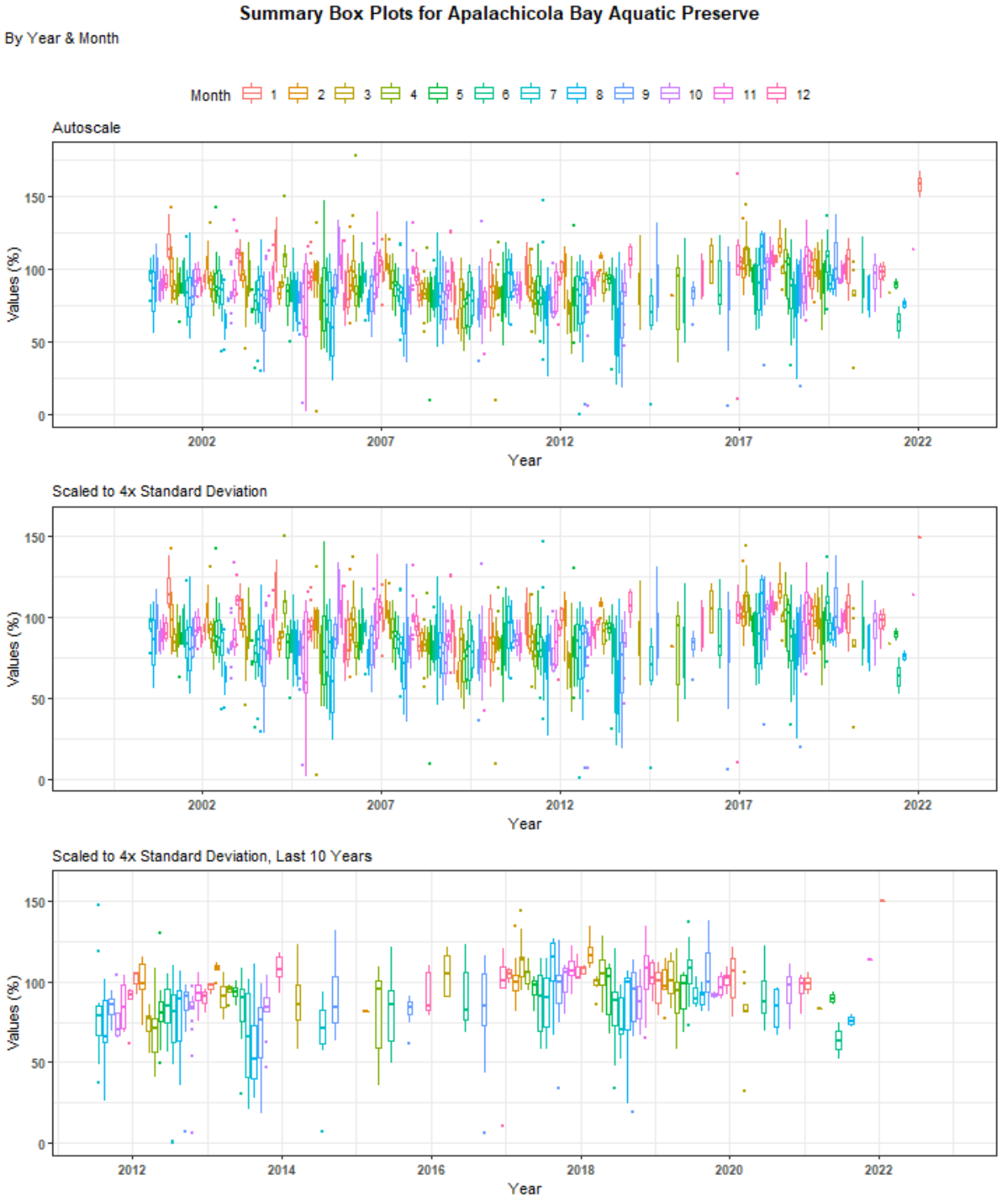
```
p6 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                           data$ManagedAreaName == MA_names[i], ],
             aes(x = YearMonthDec, y = ResultValue,
                 group = YearMonth, color = as.factor(Month)
             )) +
   geom_boxplot(outlier.size = 0.5) +
   labs(subtitle = "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))) +
   theme_bw() +
   theme(legend.position = "none",
         axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))

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("Summary Box Plots for ",
                                      MA_names[i]),
                       subtitle = "By Year & Month") + theme_bw() +
   theme(plot.title = element_text(face="bold", hjust = 0.5),
         panel.border = element_blank(),
         panel.grid.major = element_blank(),
         panel.grid.minor = element_blank(), axis.line = element_blank())

## Month Plots
p7 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                           data$ManagedAreaName == MA_names[i], ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
   geom_boxplot(outlier.size = 0.5) +
   labs(subtitle = "Autoscale",
        x = "Month", y = paste0("Values (", unit, ")"), fill = "Month") +
   scale_x_continuous(limits = c(0, 13), breaks = seq(3, 12, 3)) +
   theme_bw() +
   theme(legend.position = "none",
         axis.text.x = element_text(face = "bold"),
         axis.text.y = element_text(face = "bold"))

p8 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                           data$ManagedAreaName == MA_names[i], ],
             aes(x = Month, y = ResultValue,
                 group = Month, fill = as.factor(Month))) +
   geom_boxplot(outlier.size = 0.5) +
   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)) +
   theme_bw() +
   theme(legend.position = "top", legend.box = "horizontal",
```

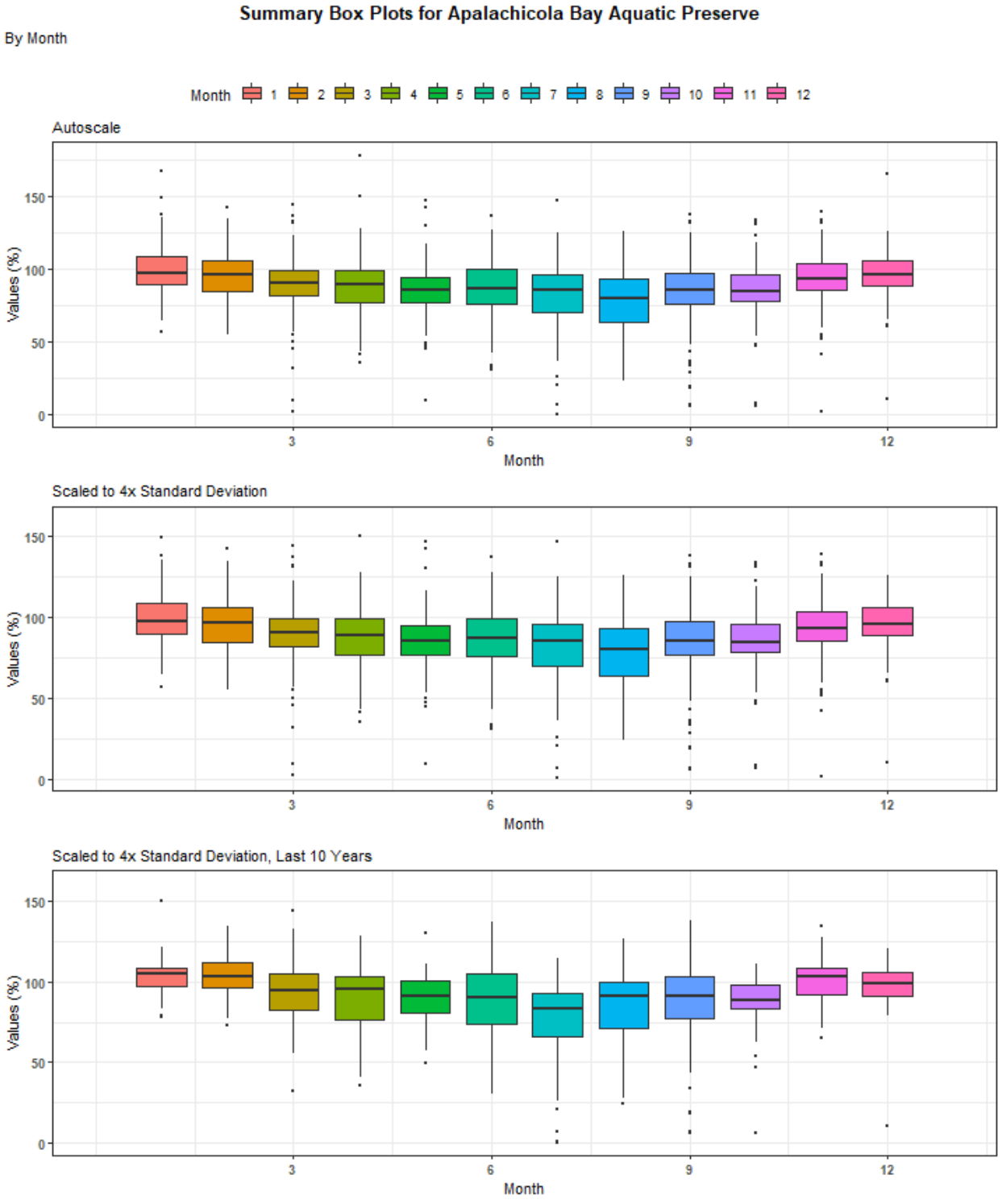
```
            axis.text.x = element_text(face = "bold"),
            axis.text.y = element_text(face = "bold")) +
      guides(fill = guide_legend(nrow = 1))

    p9 <- ggplot(data = data[data$Use_In_Analysis == TRUE &
                               data$ManagedAreaName == MA_names[i] &
                               data$Year >= year_upper - 10, ],
                 aes(x = Month, y = ResultValue,
                     group = Month, fill = as.factor(Month))) +
      geom_boxplot(outlier.size = 0.5) +
      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)) +
      theme_bw() +
      theme(legend.position = "none",
            axis.text.x = element_text(face = "bold"),
            axis.text.y = element_text(face = "bold"))

    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("Summary Box Plots for ",
                                           MA_names[i]),
                            subtitle = "By Month") + theme_bw() +
      theme(plot.title = element_text(face="bold", hjust = 0.5),
            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)))
  }
}
```

Summary Box Plots for Apalachicola Bay Aquatic Preserve

By Year



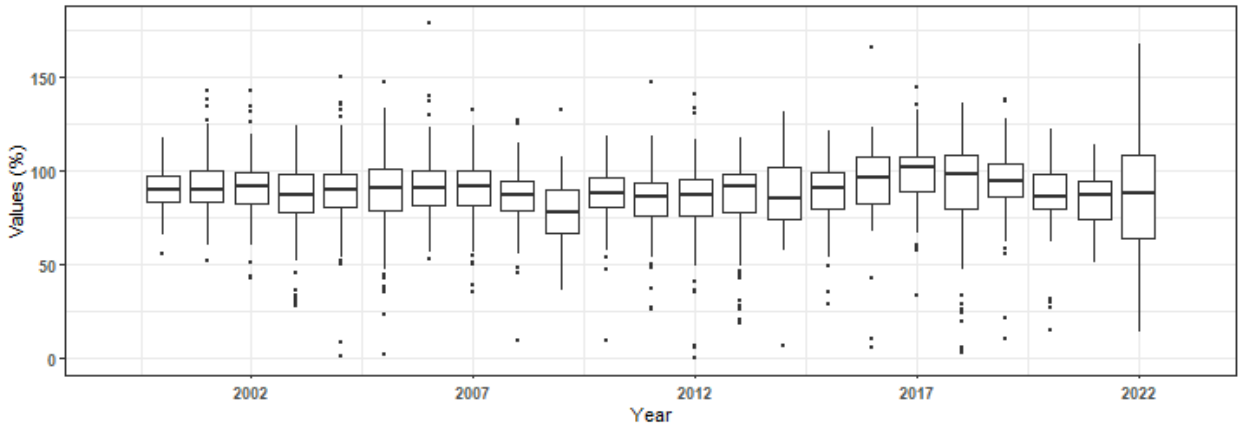

Summary Box Plots for Apalachicola Bay Aquatic Preserve
By Year & Month
Month 1 2 3 4 5 6 7 8 9 10 11 12
Autoscale
Values (%)
Year
Scaled to 4x Standard Deviation
Values (%)
Year
Scaled to 4x Standard Deviation, Last 10 Years
Values (%)
Year


Summary Box Plots for 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

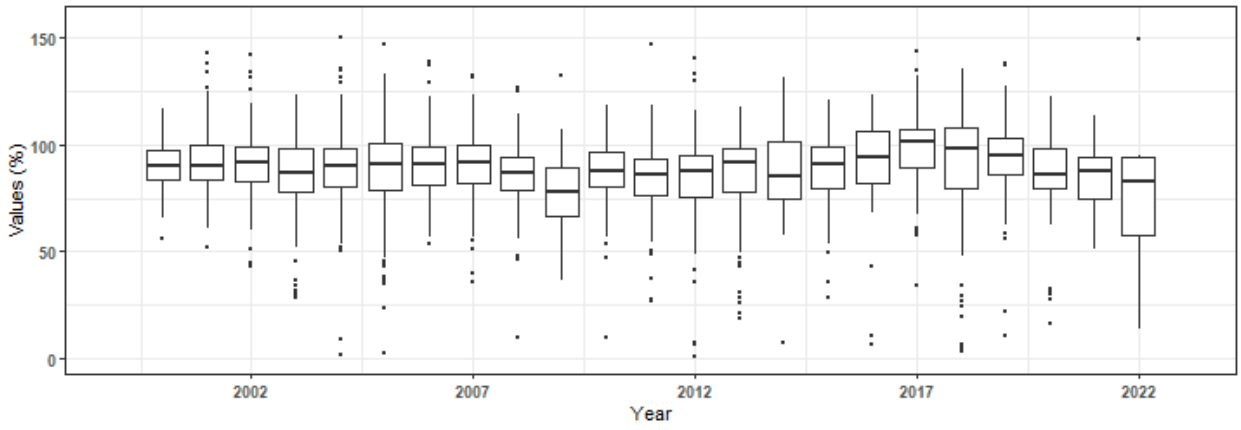
Summary Box Plots for Apalachicola National Estuarine Research Reserve

By Year

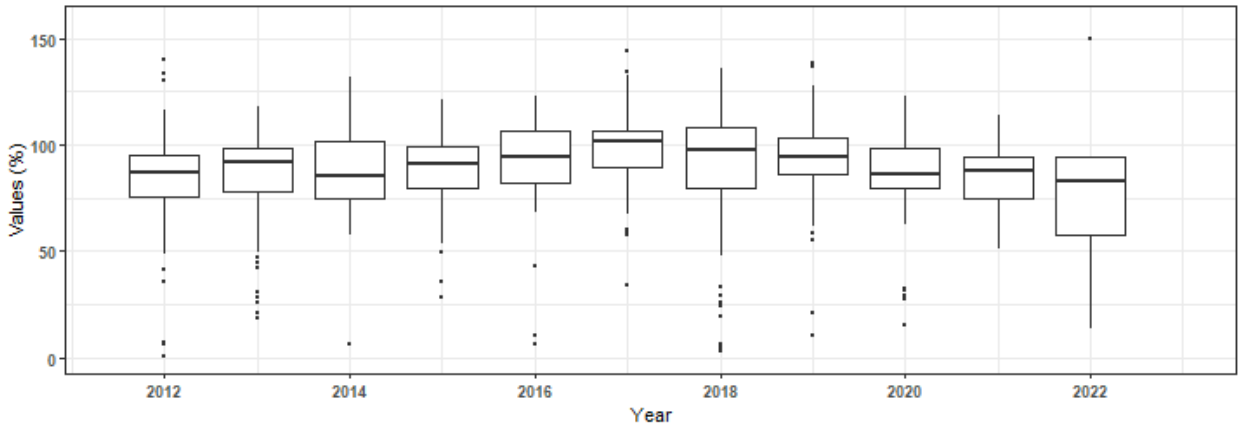
Autoscale



Scaled to 4x Standard Deviation

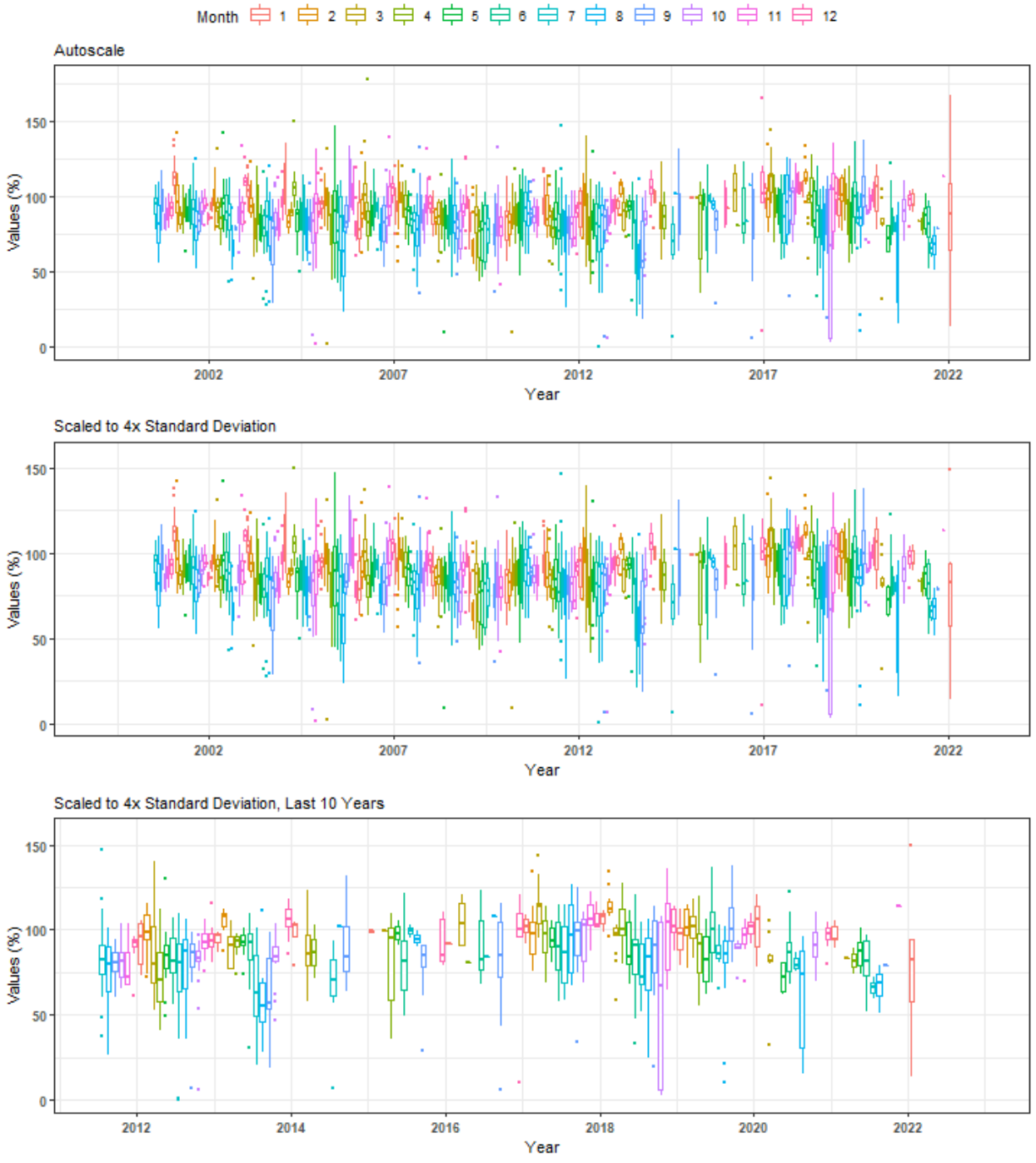


Scaled to 4x Standard Deviation, Last 10 Years



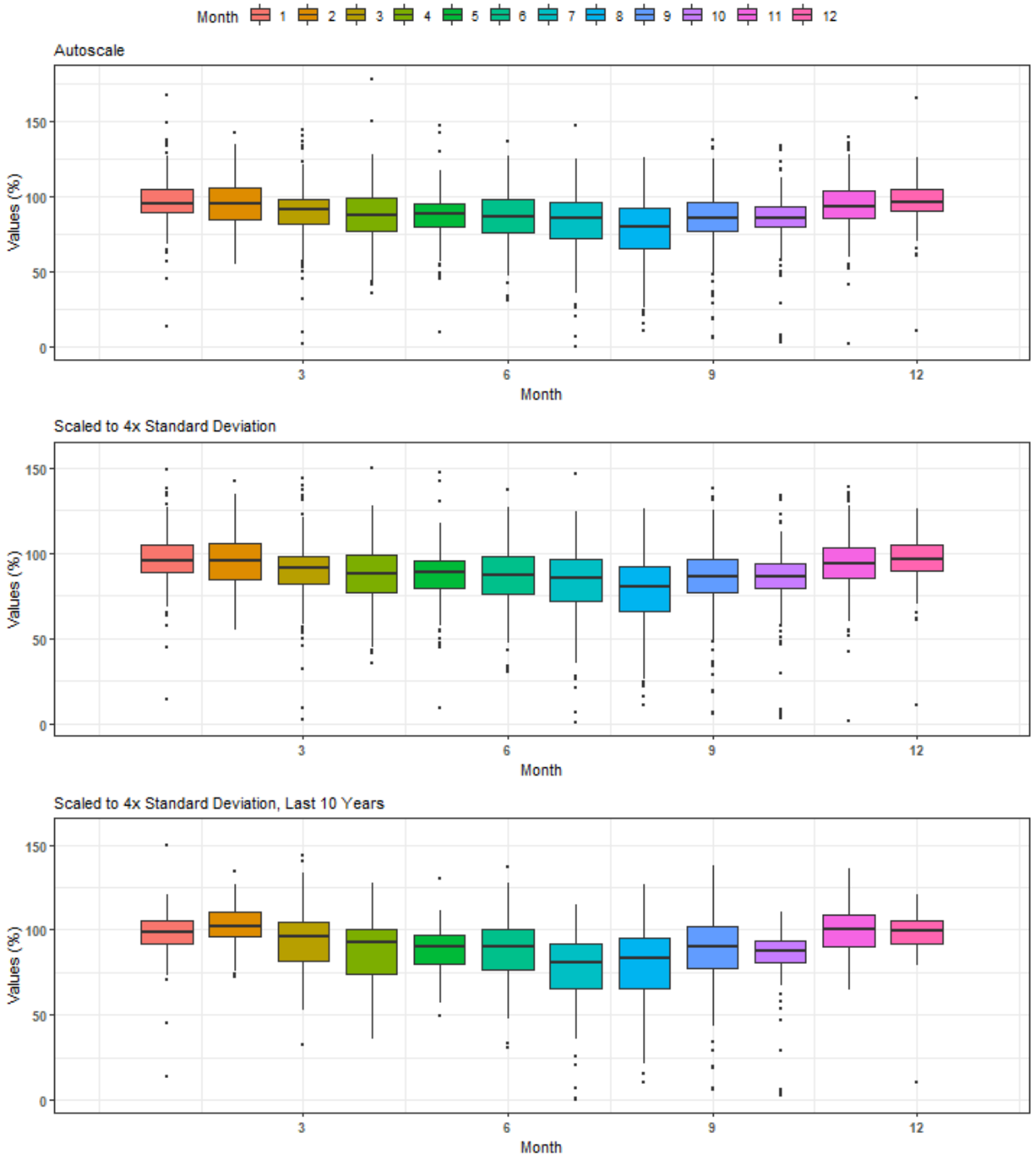
Summary Box Plots for Apalachicola National Estuarine Research Reserve

By Year & Month



Summary Box Plots for Apalachicola National Estuarine Research Reserve

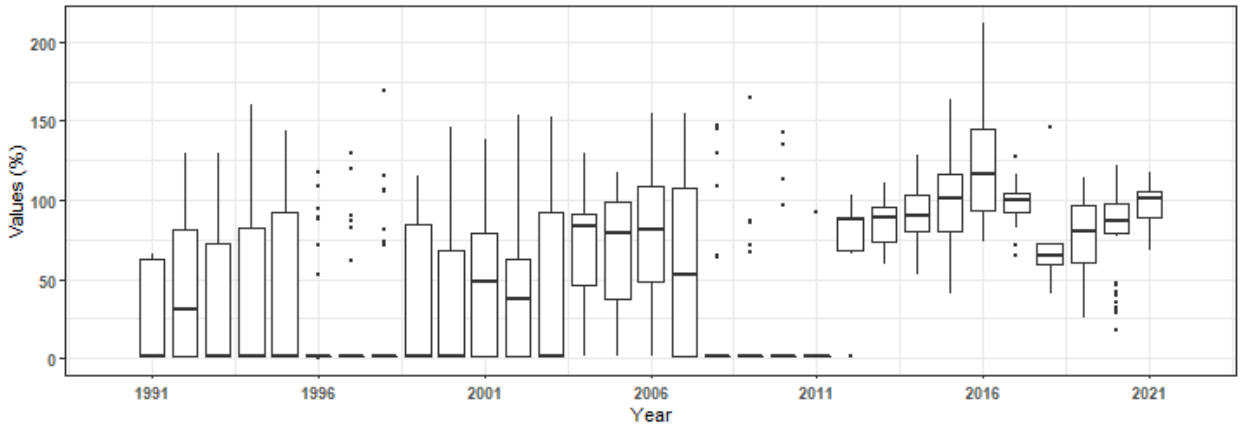
By Month



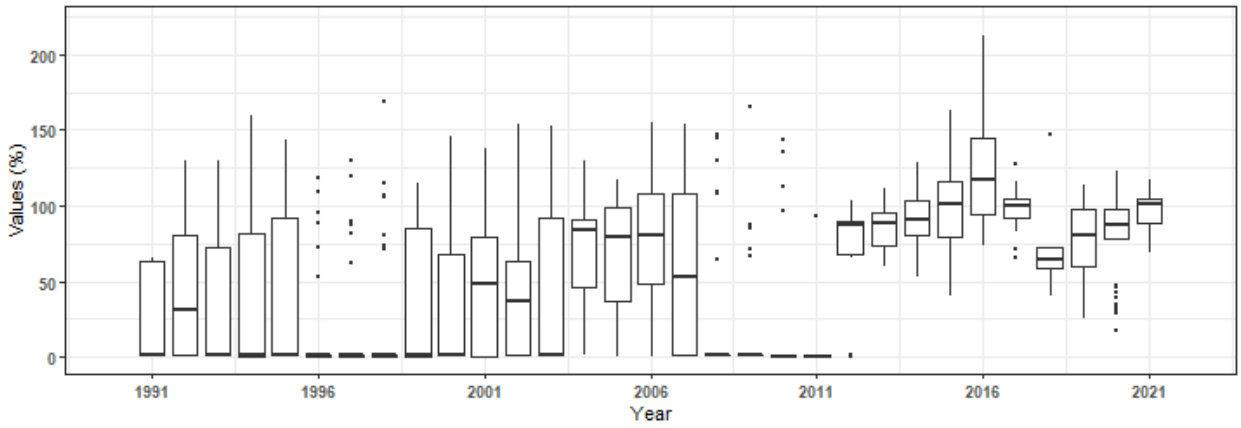
Summary Box Plots for Banana River Aquatic Preserve

By Year

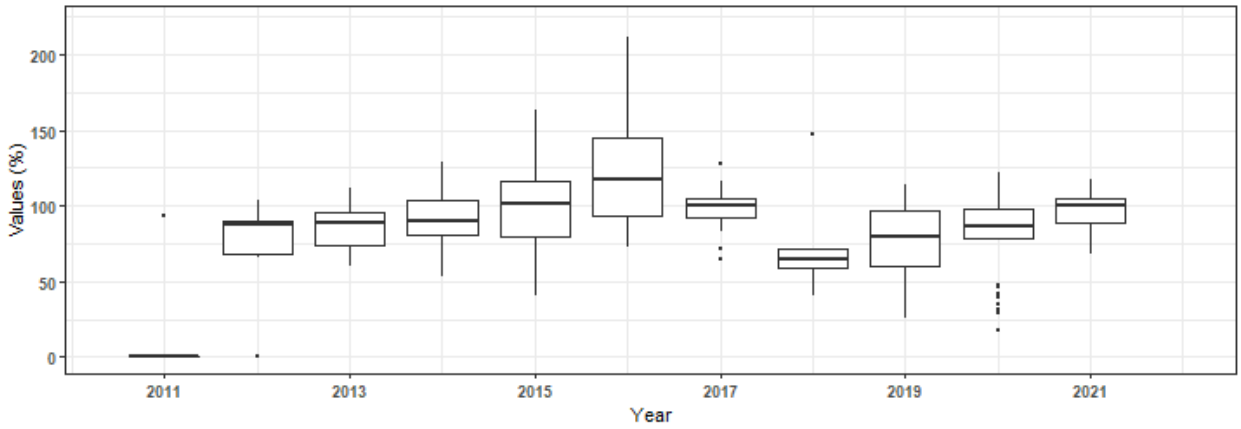
Autoscale



Scaled to 4x Standard Deviation

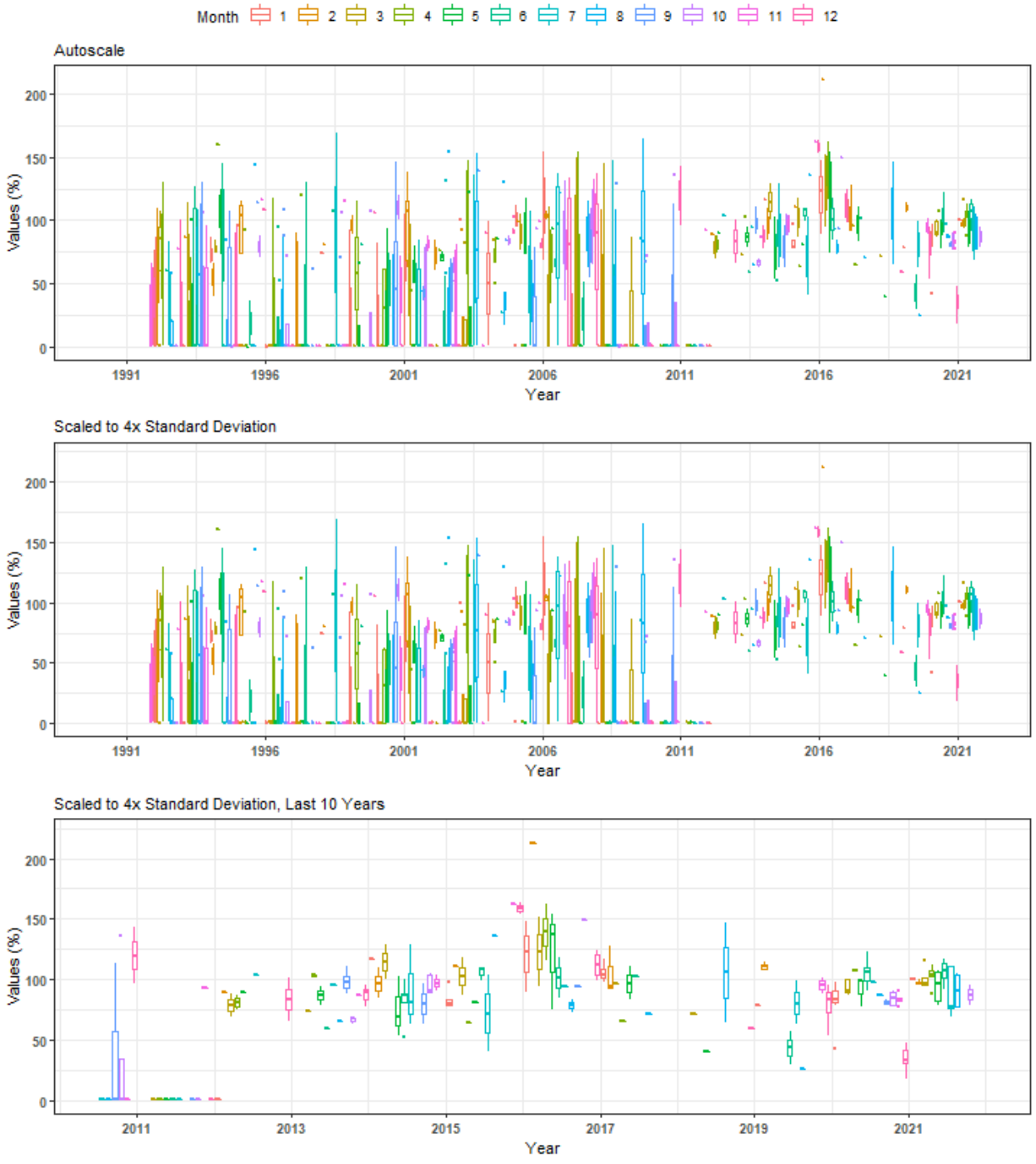


Scaled to 4x Standard Deviation, Last 10 Years



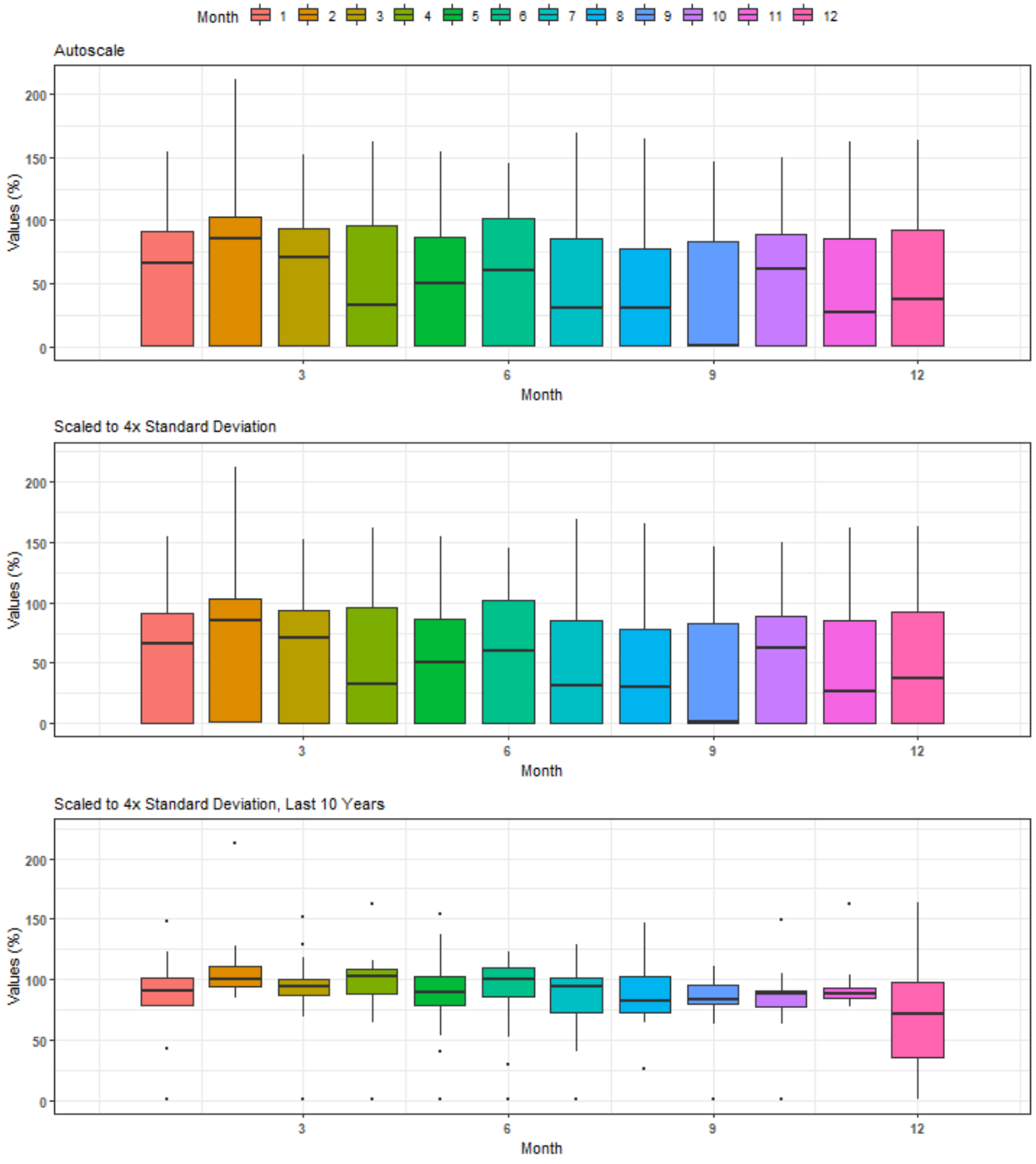
Summary Box Plots for Banana River Aquatic Preserve

By Year & Month



Summary Box Plots for Banana River Aquatic Preserve

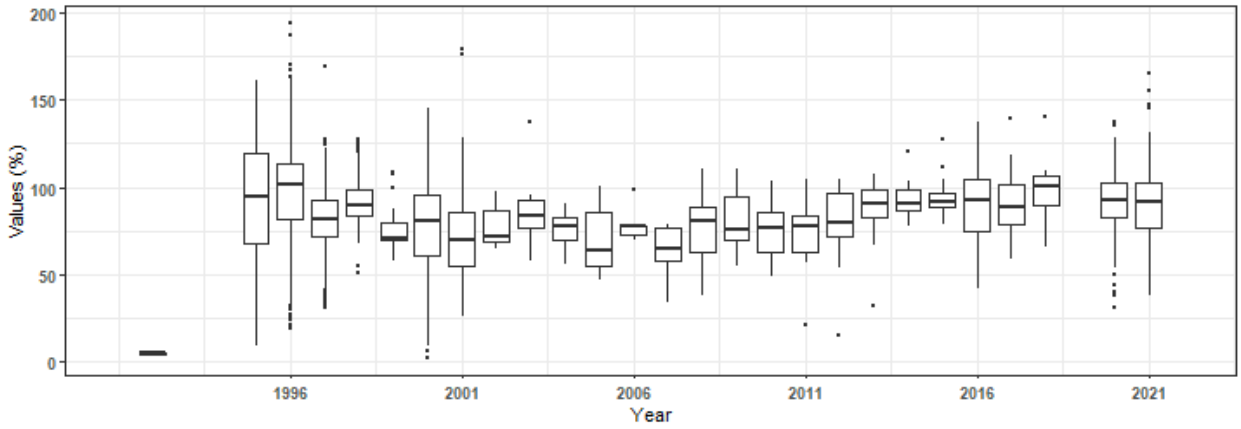
By Month



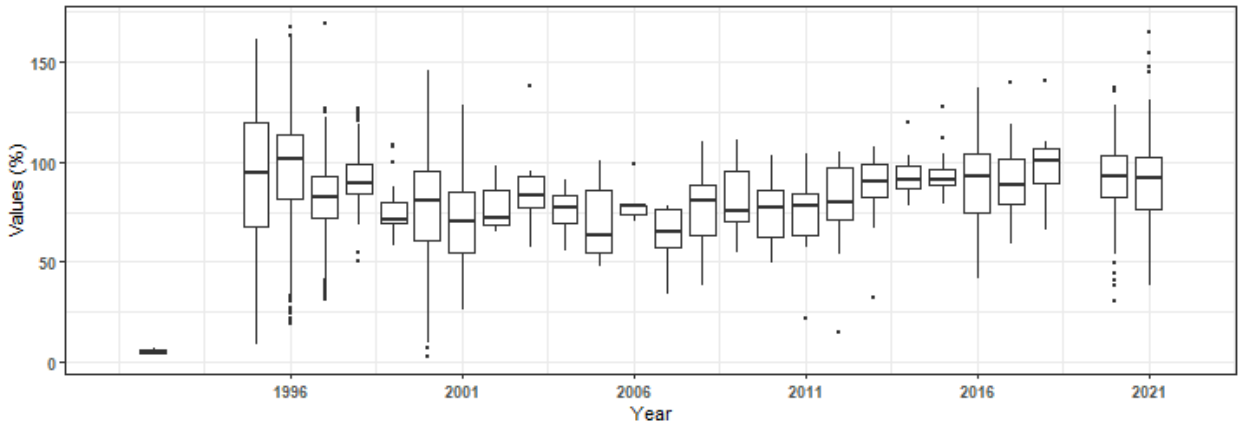
Summary Box Plots for Boca Ciega Bay Aquatic Preserve

By Year

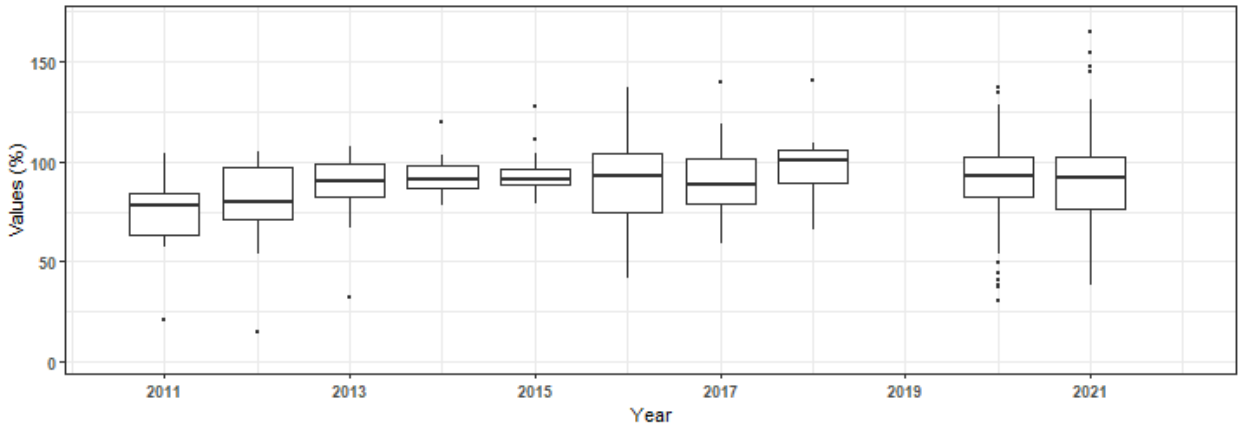
Autoscale



Scaled to 4x Standard Deviation

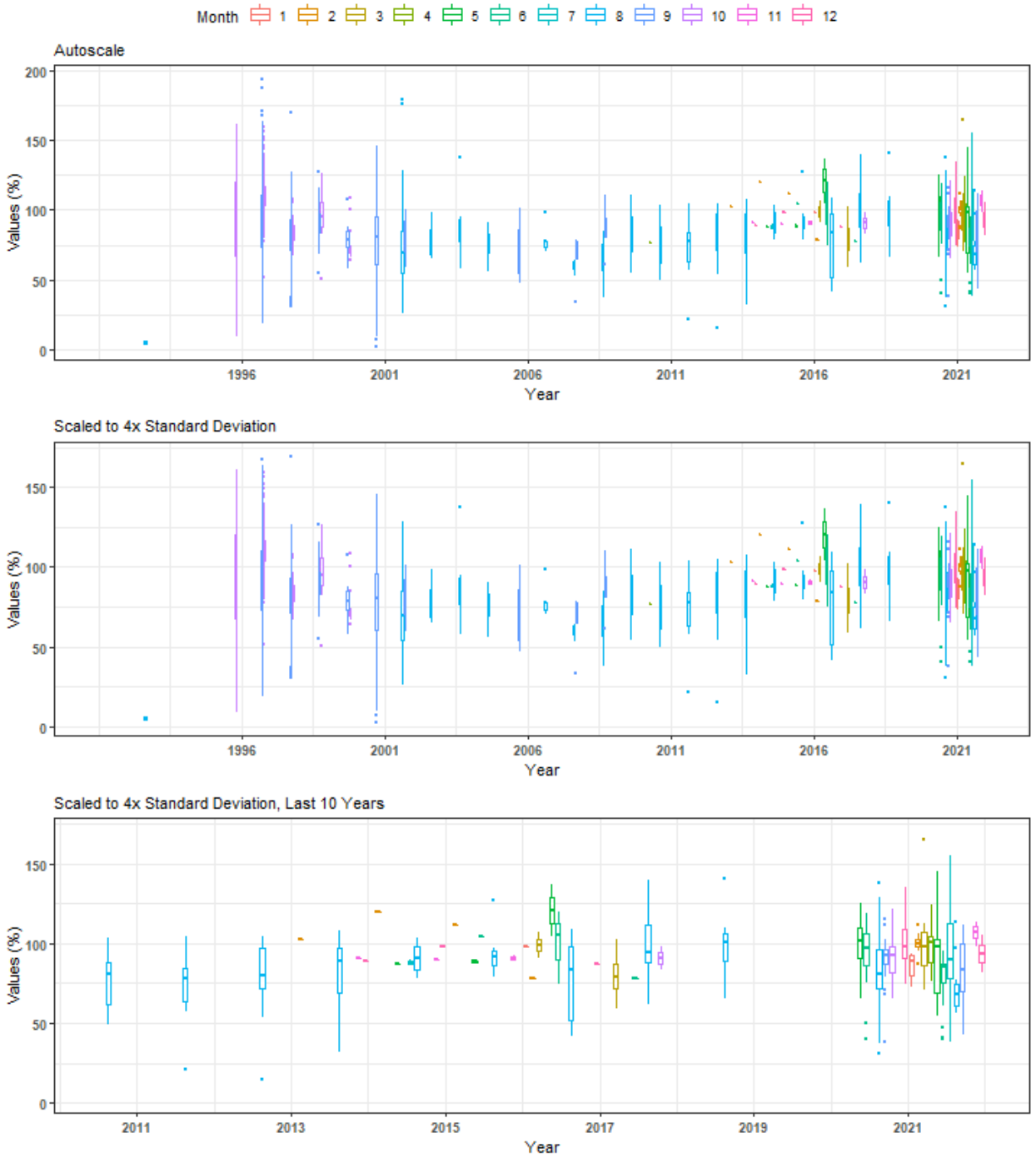


Scaled to 4x Standard Deviation, Last 10 Years



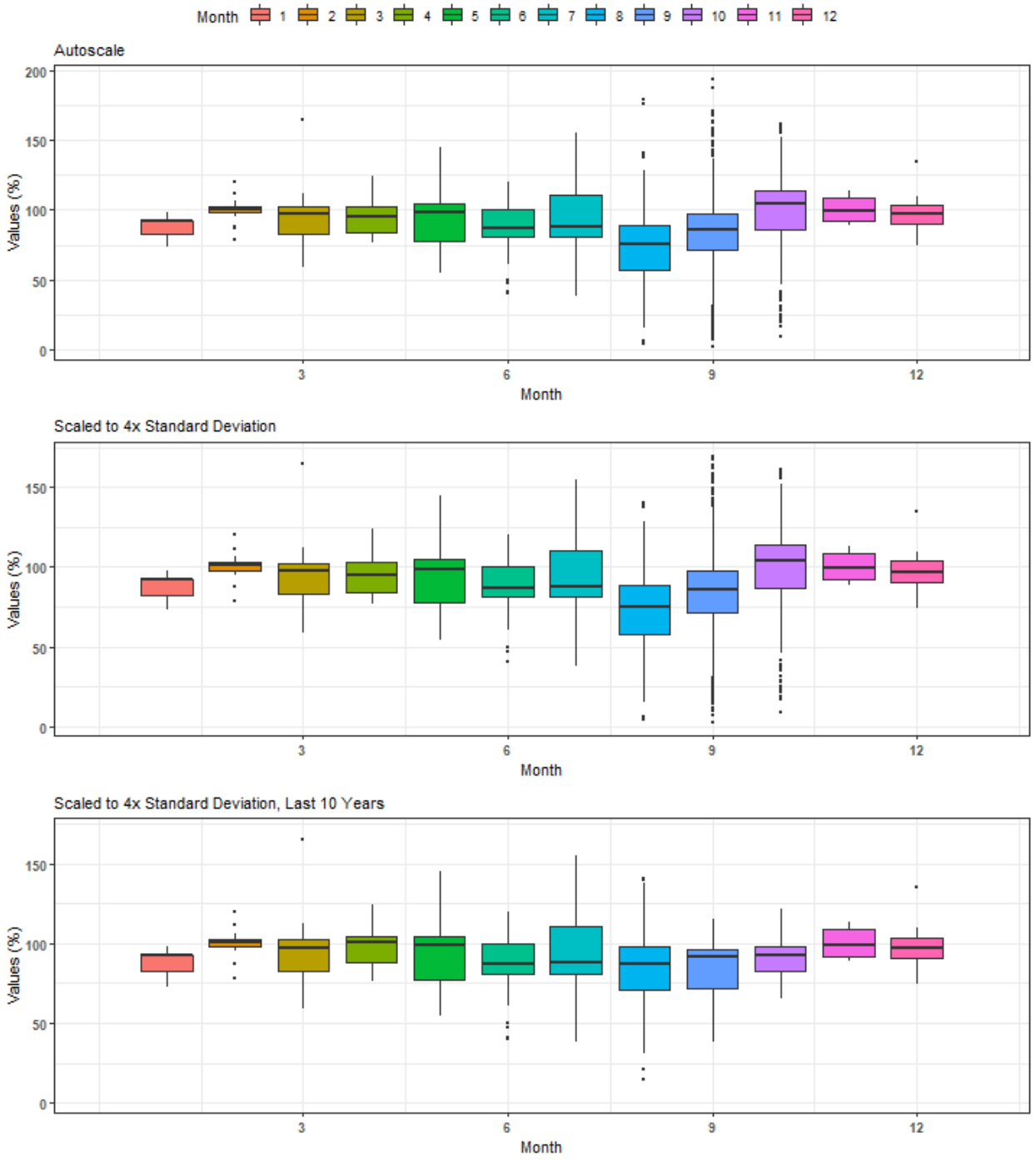
Summary Box Plots for Boca Ciega Bay Aquatic Preserve

By Year & Month



Summary Box Plots for Boca Ciega Bay Aquatic Preserve

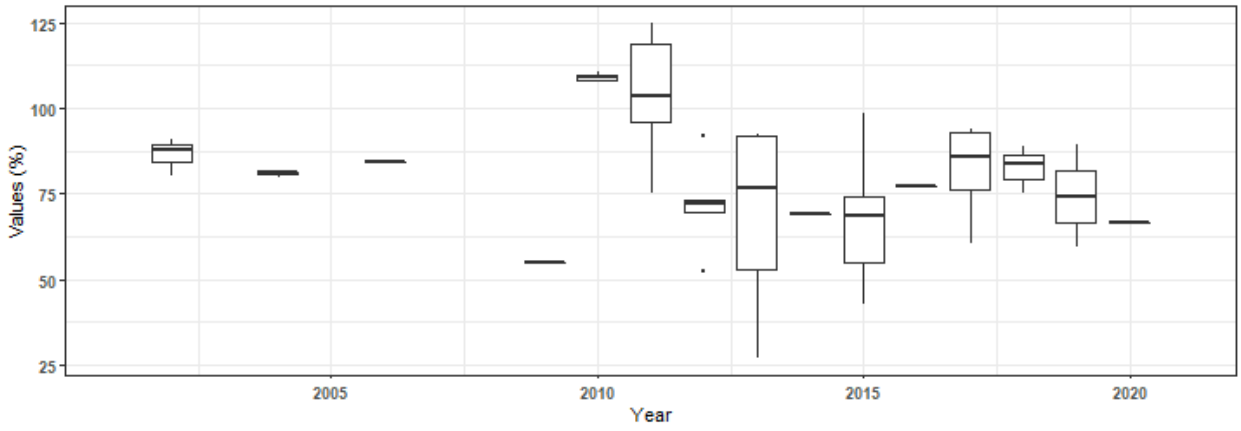
By Month



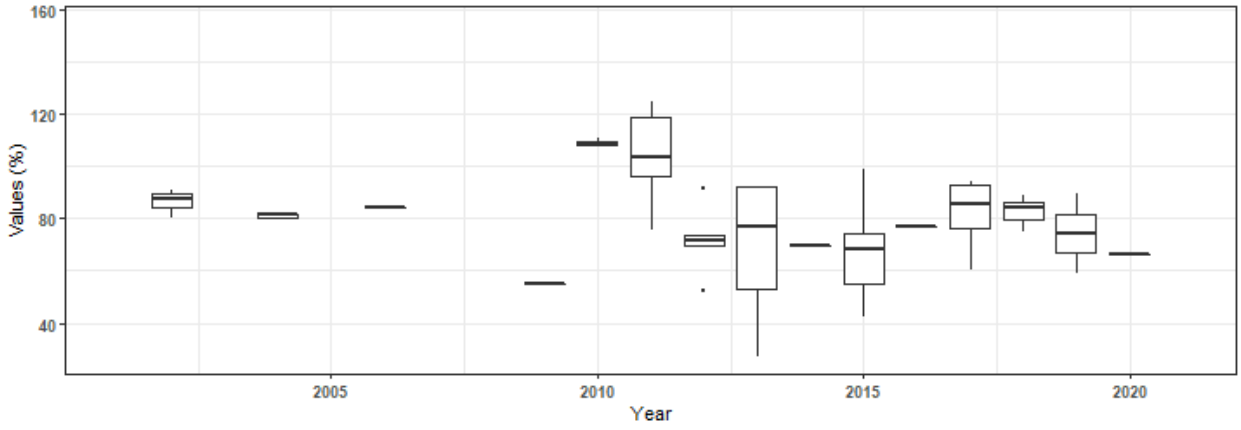
Summary Box Plots for Cape Romano-Ten Thousand Islands Aquatic Preserve

By Year

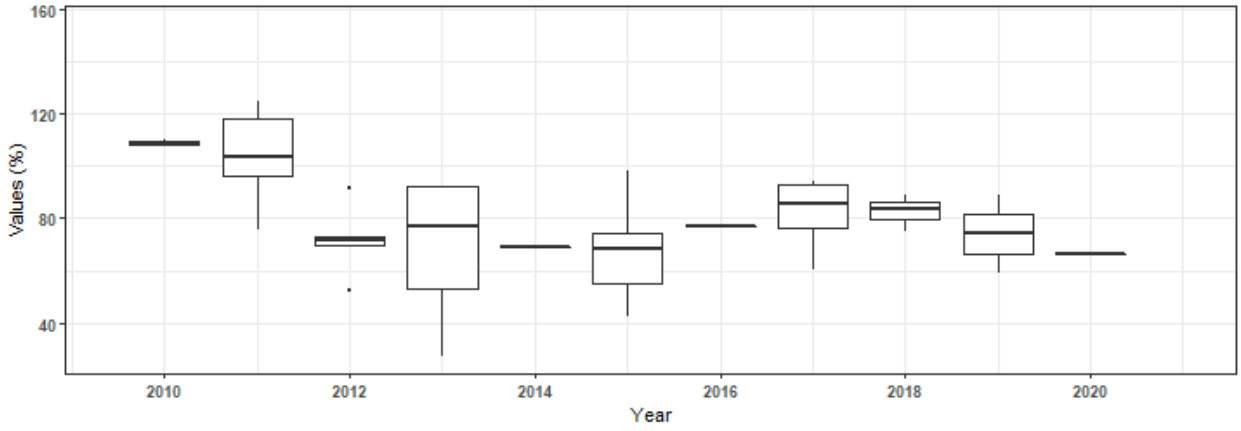
Autoscale



Scaled to 4x Standard Deviation

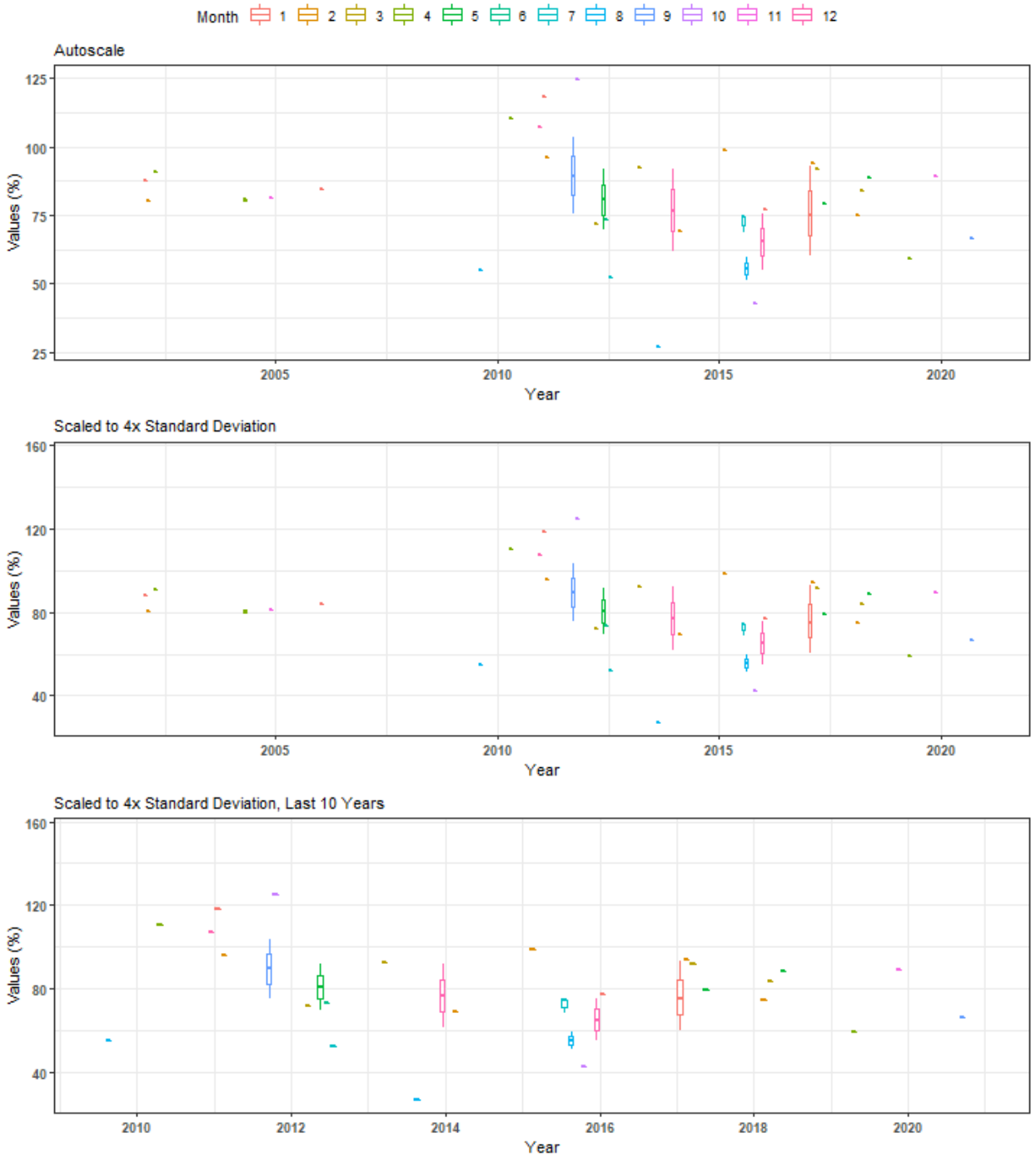


Scaled to 4x Standard Deviation, Last 10 Years



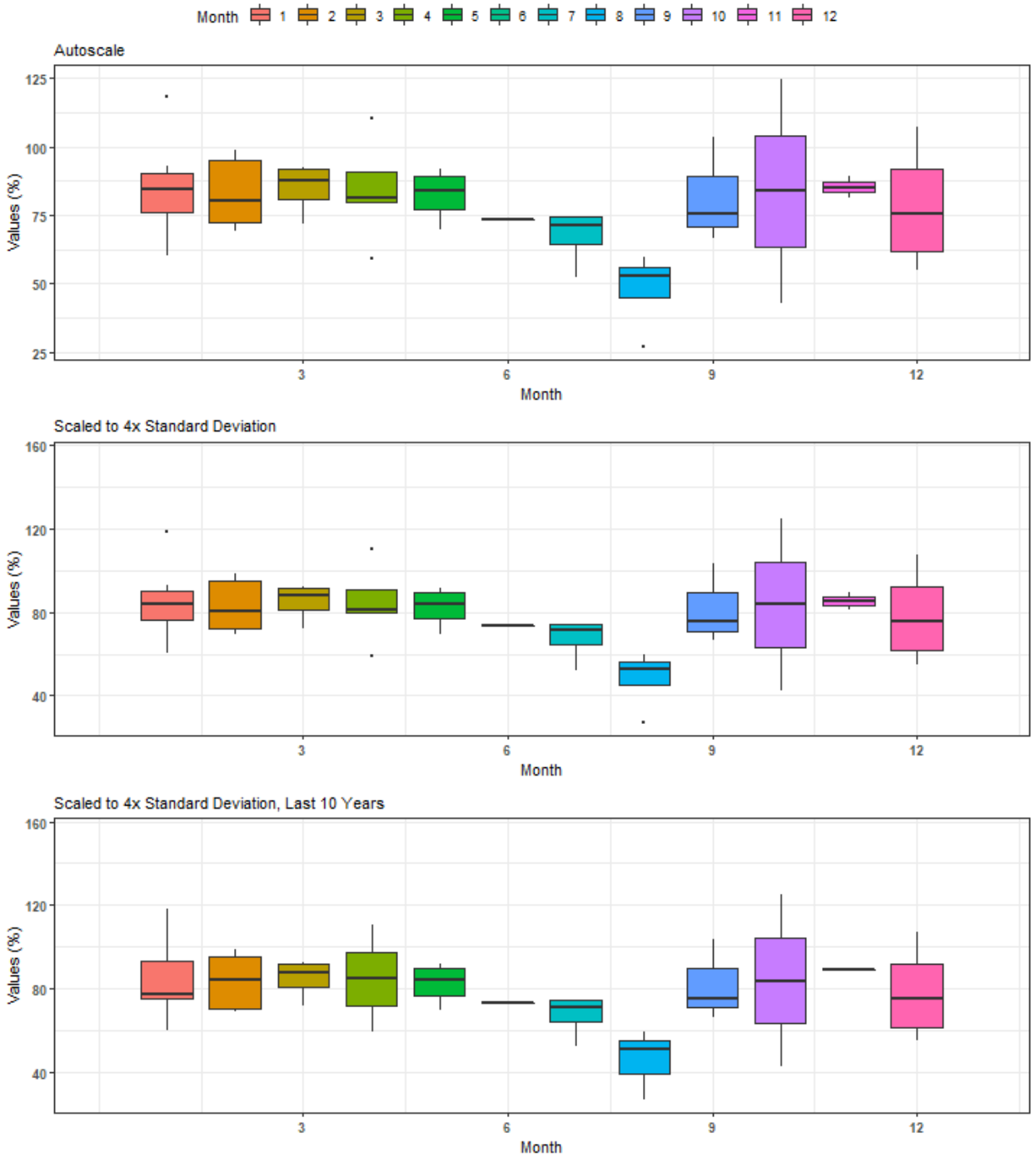
Summary Box Plots for Cape Romano-Ten Thousand Islands Aquatic Preserve

By Year & Month



Summary Box Plots for Cape Romano-Ten Thousand Islands Aquatic Preserve

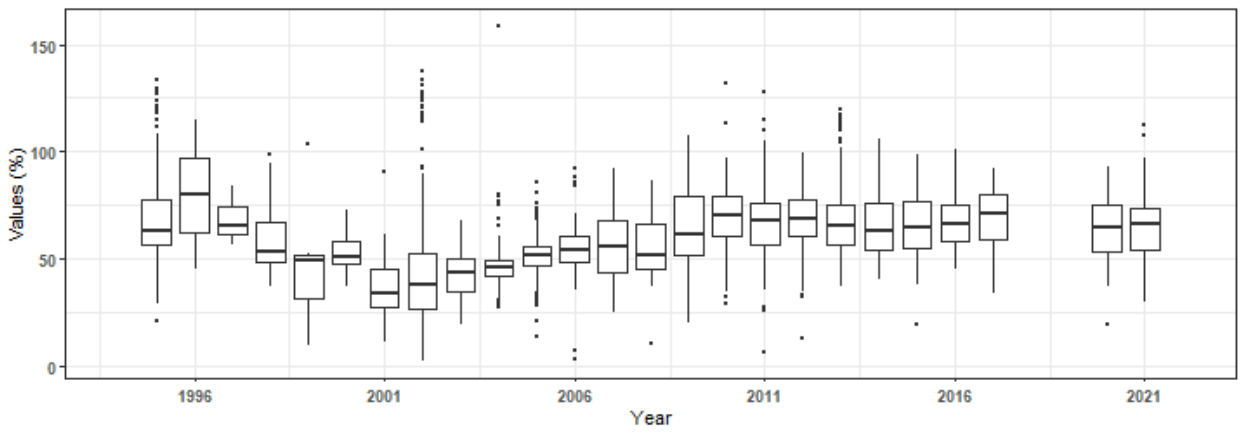
By Month



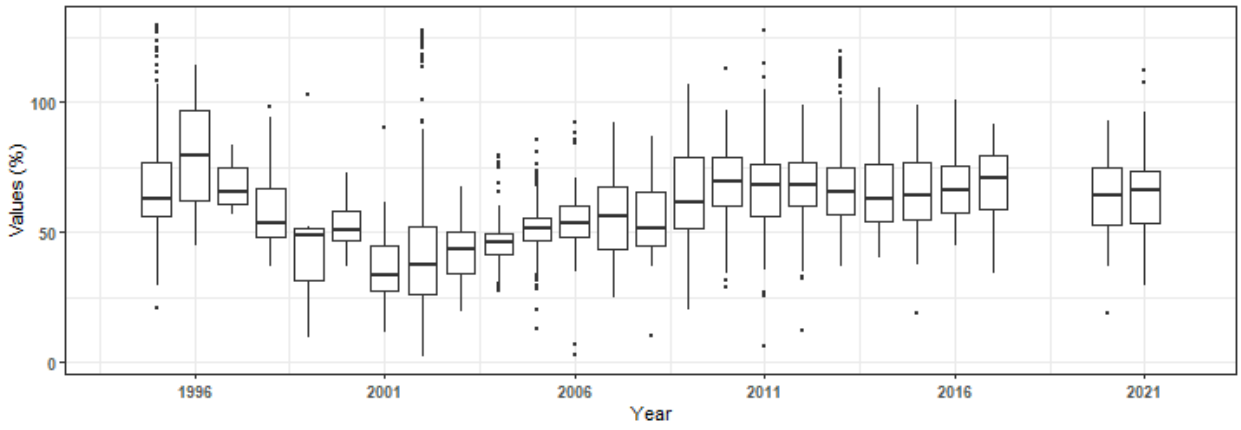
Summary Box Plots for Cockroach Bay Aquatic Preserve

By Year

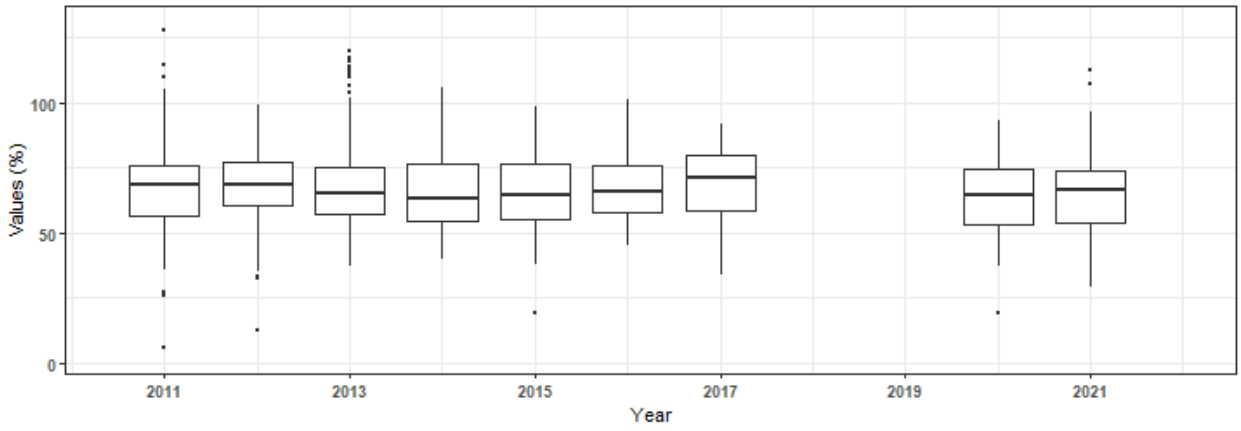
Autoscale

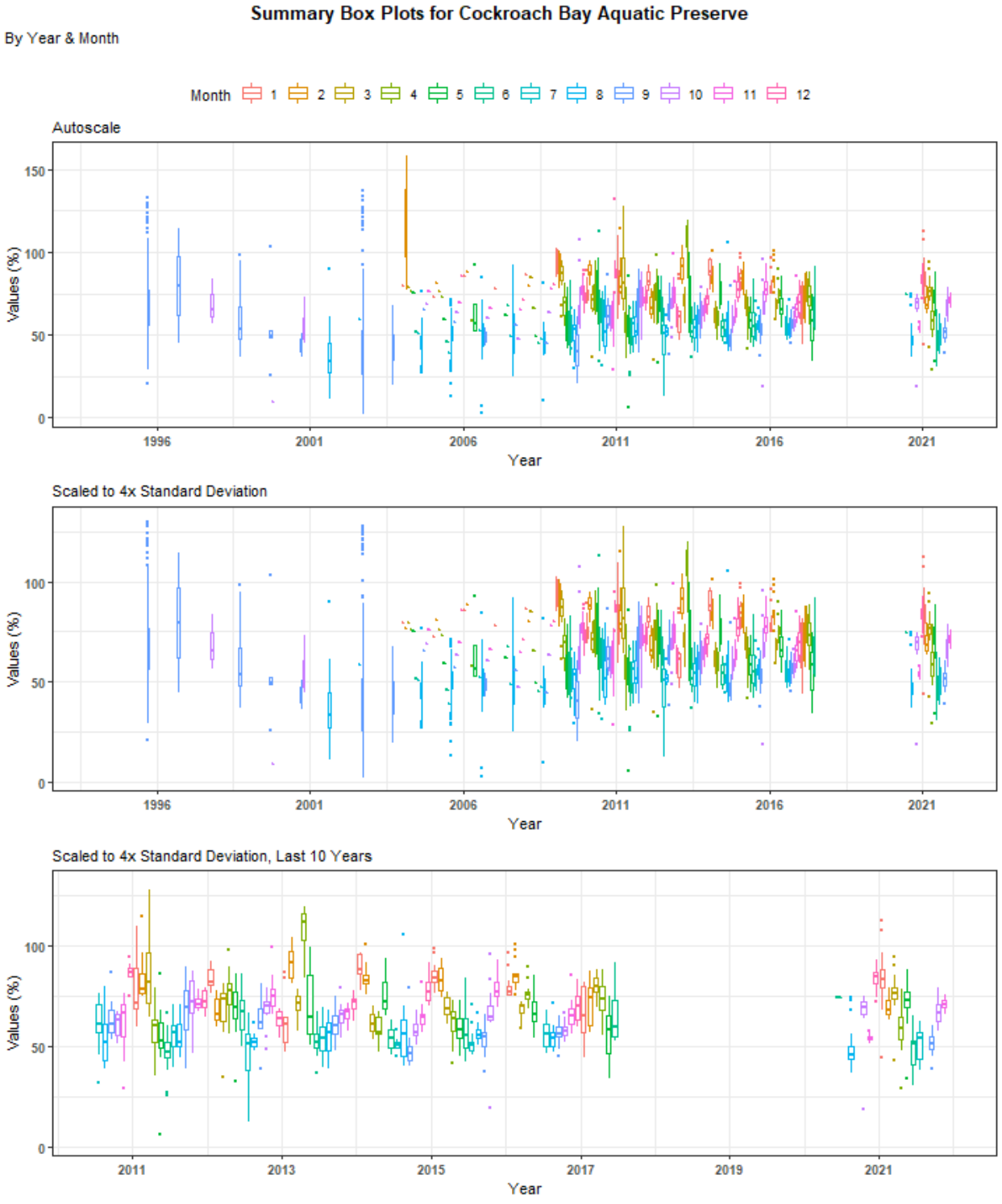


Scaled to 4x Standard Deviation



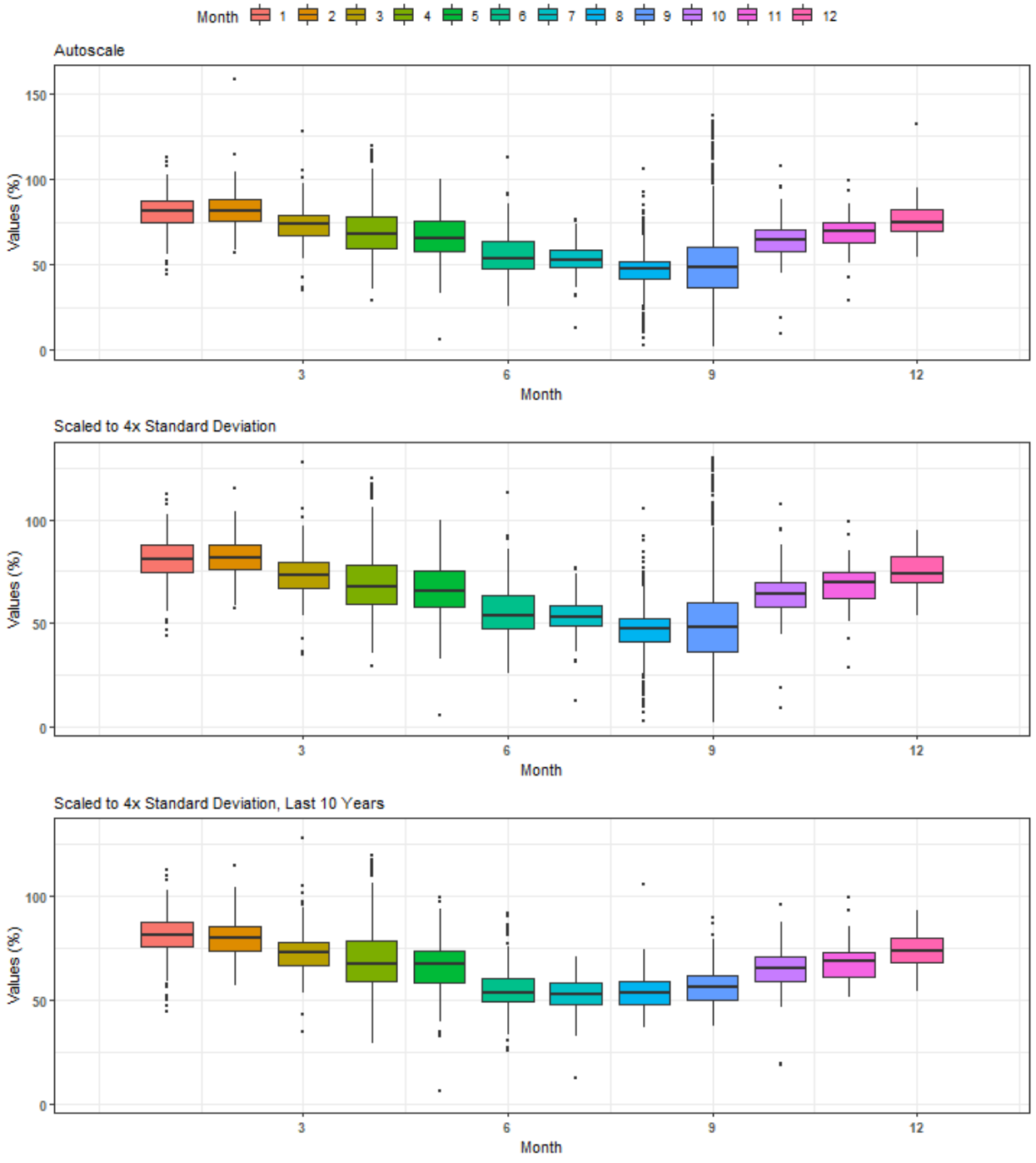
Scaled to 4x Standard Deviation, Last 10 Years




Summary Box Plots for Cockroach 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

Summary Box Plots for Cockroach Bay Aquatic Preserve

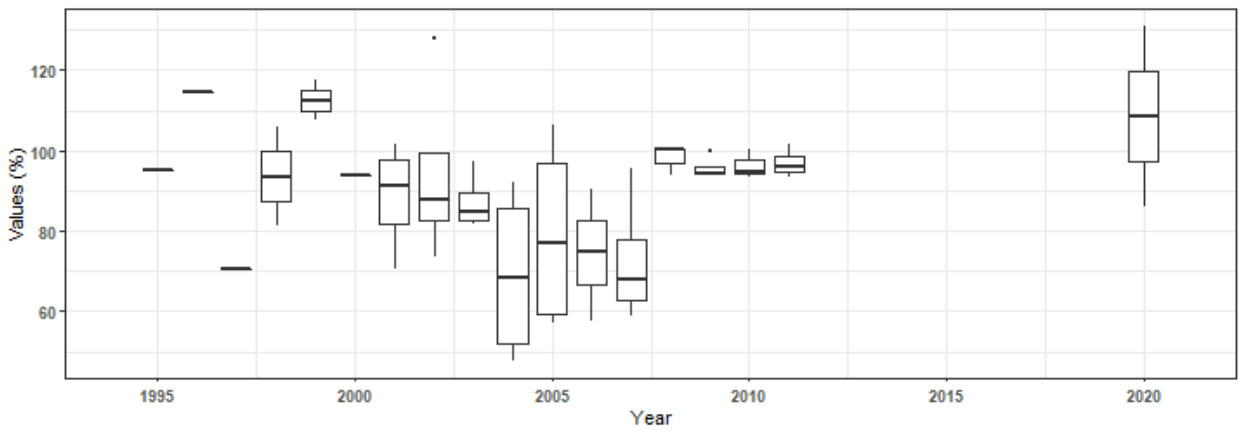
By Month



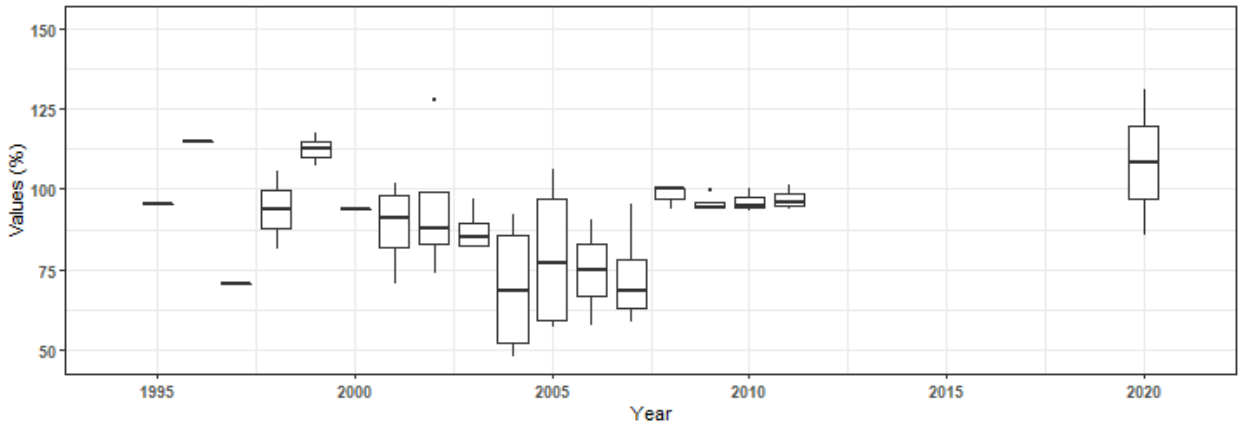
Summary Box Plots for Coupon Bight Aquatic Preserve

By Year

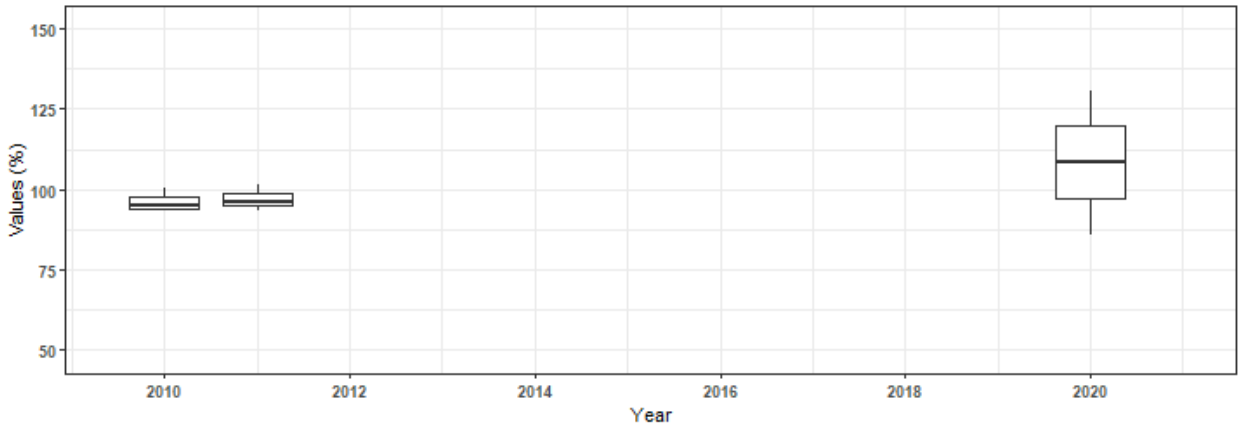
Autoscale



Scaled to 4x Standard Deviation

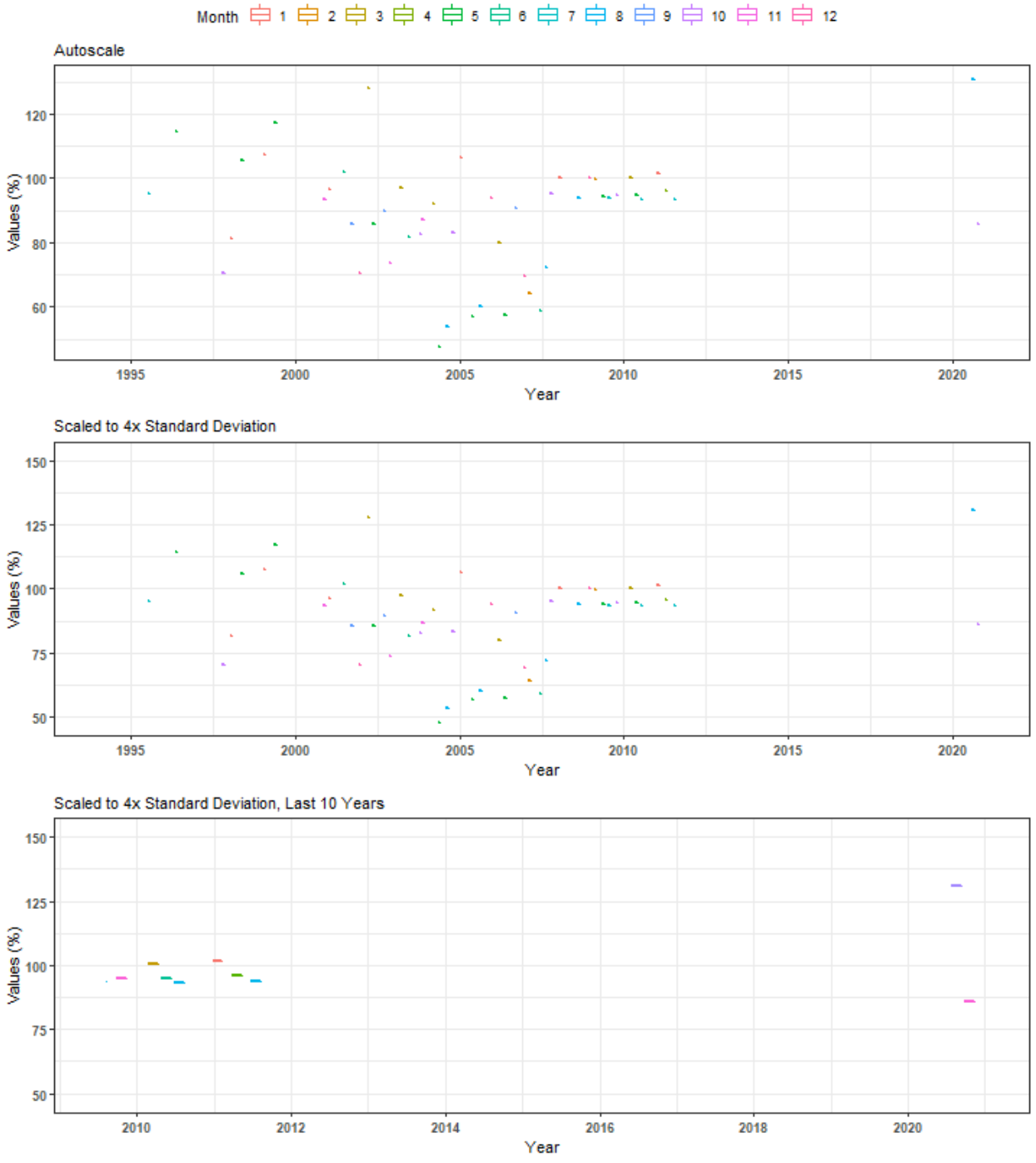


Scaled to 4x Standard Deviation, Last 10 Years



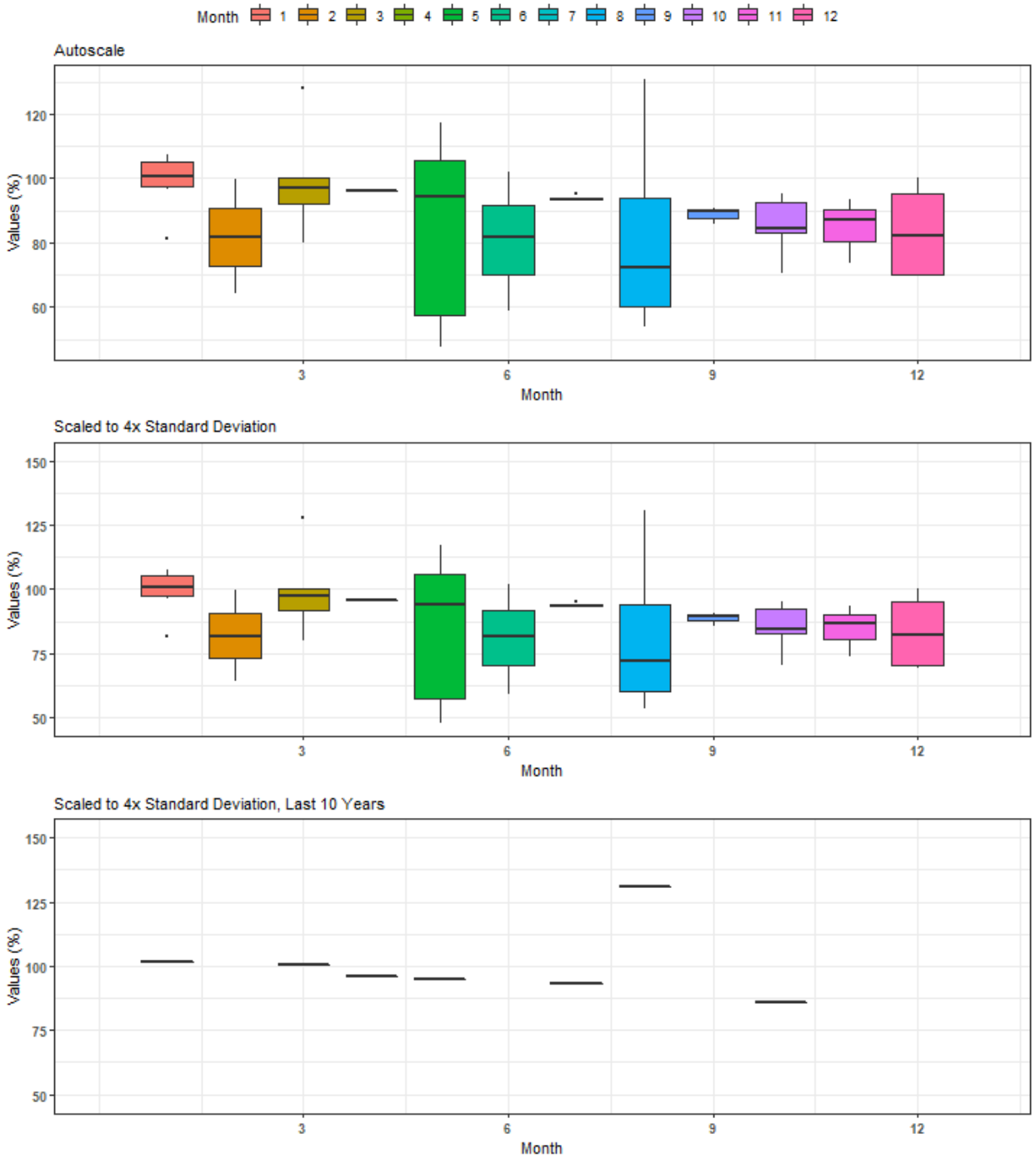
Summary Box Plots for Coupon Bight Aquatic Preserve

By Year & Month



Summary Box Plots for Coupon Bight Aquatic Preserve

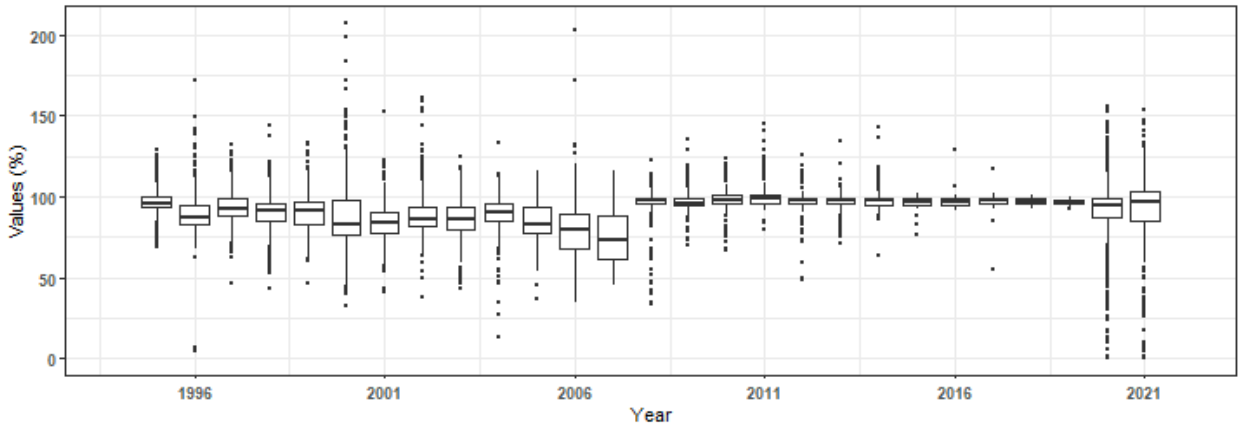
By Month



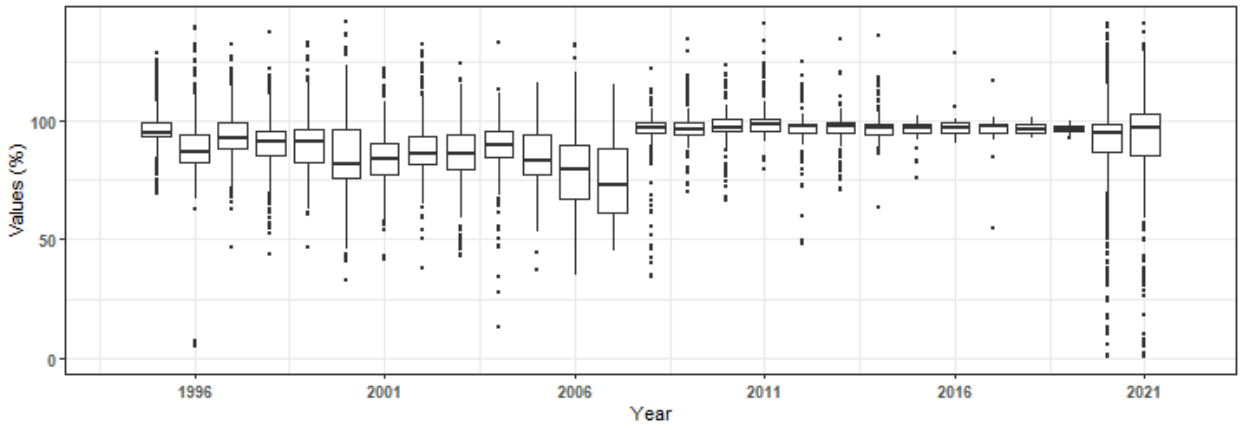
Summary Box Plots for Florida Keys National Marine Sanctuary

By Year

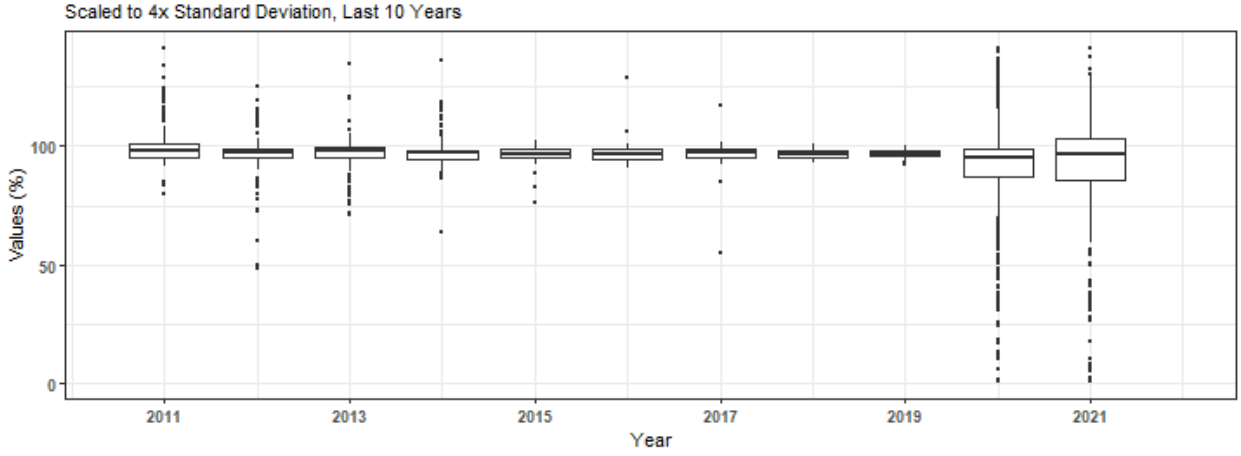
Autoscale



Scaled to 4x Standard Deviation

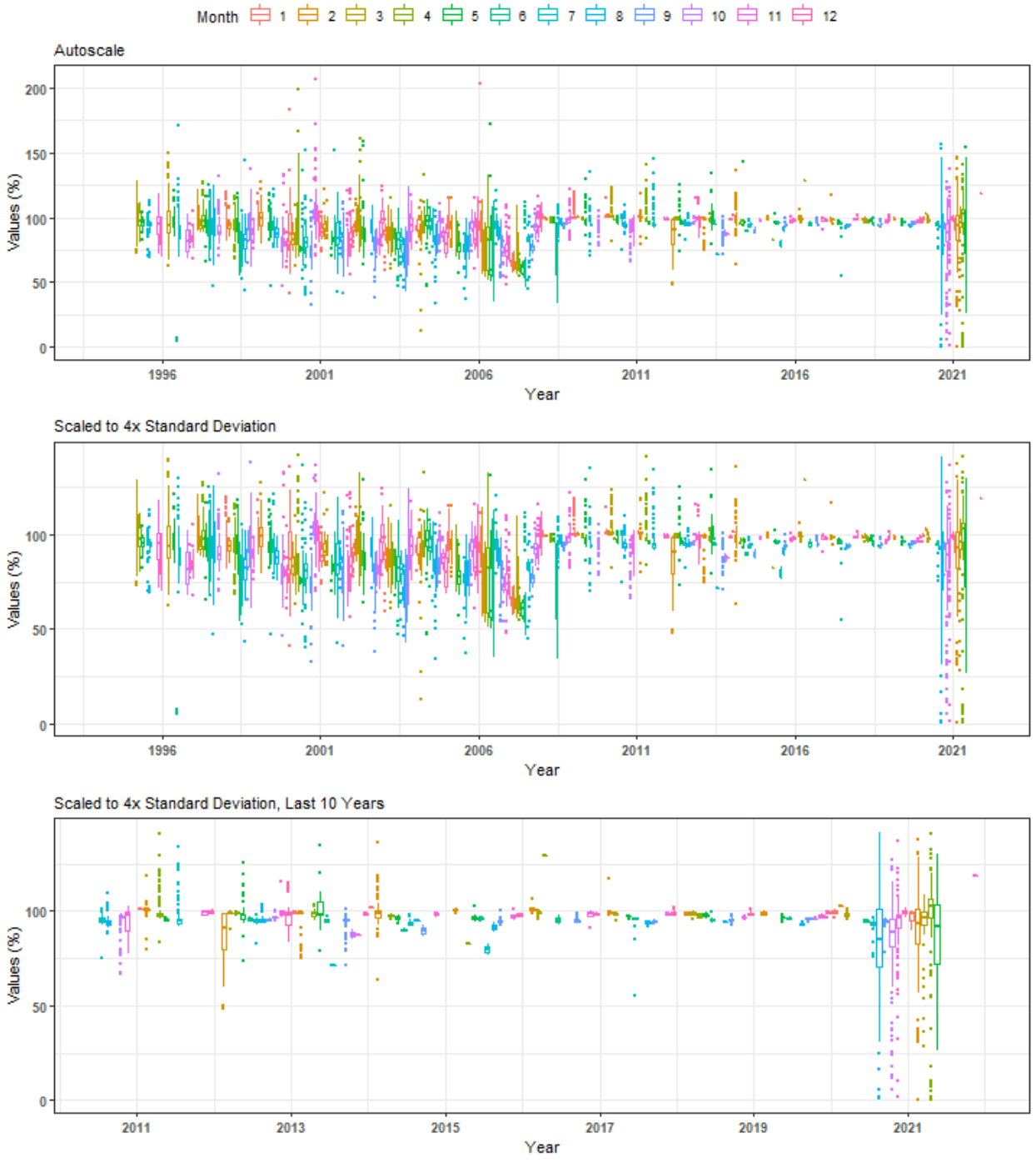


Scaled to 4x Standard Deviation, Last 10 Years



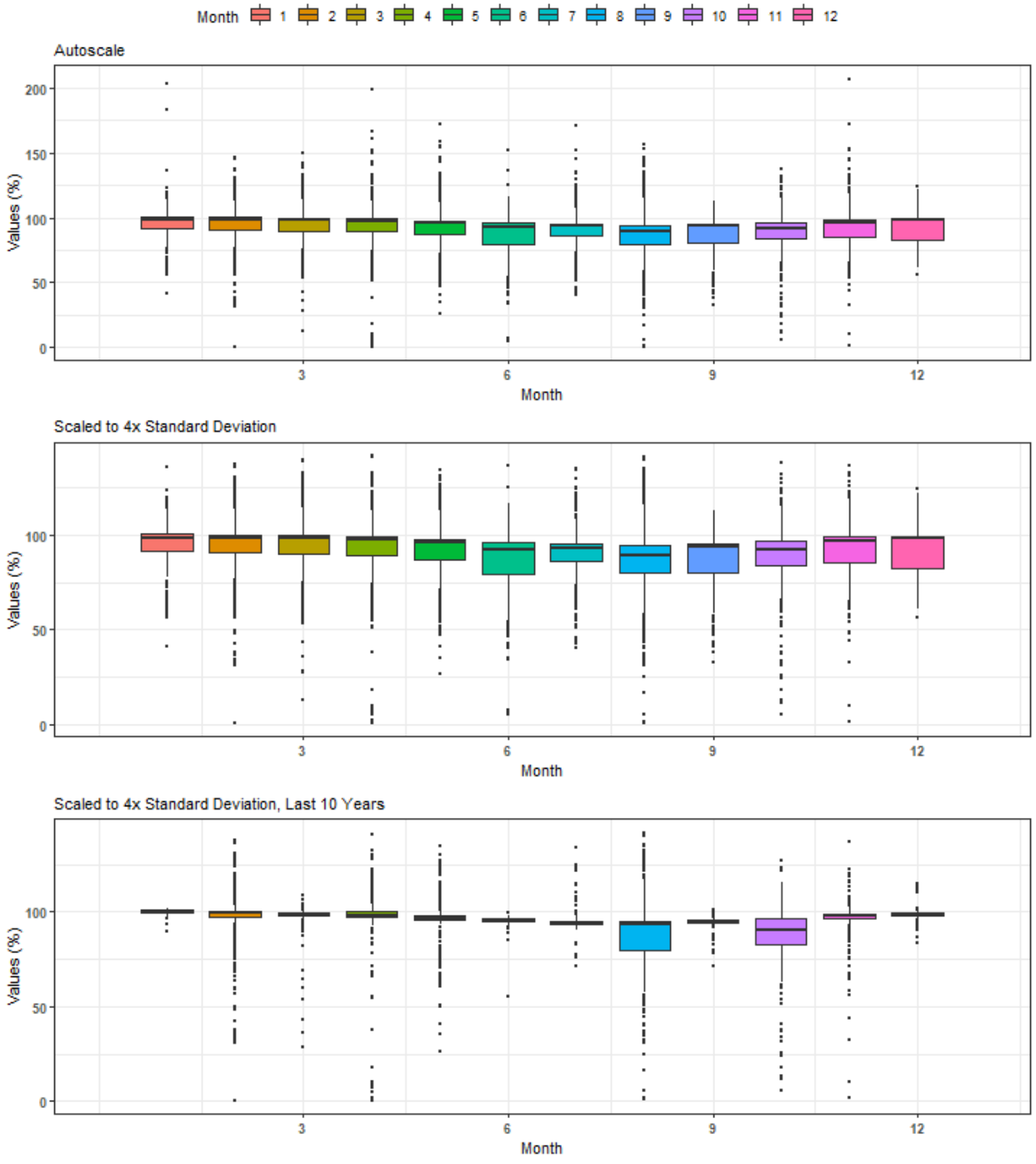
Summary Box Plots for Florida Keys National Marine Sanctuary

By Year & Month



Summary Box Plots for Florida Keys National Marine Sanctuary

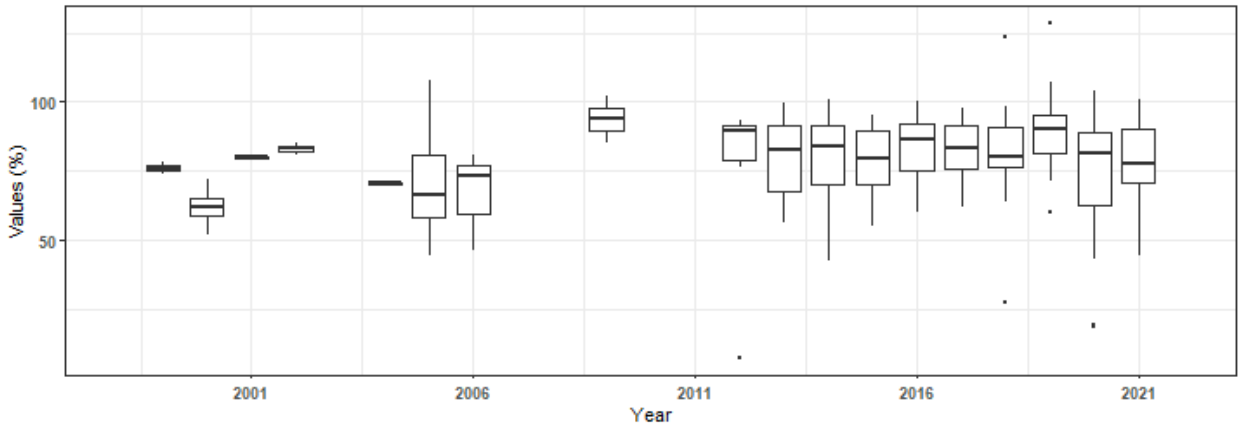
By Month



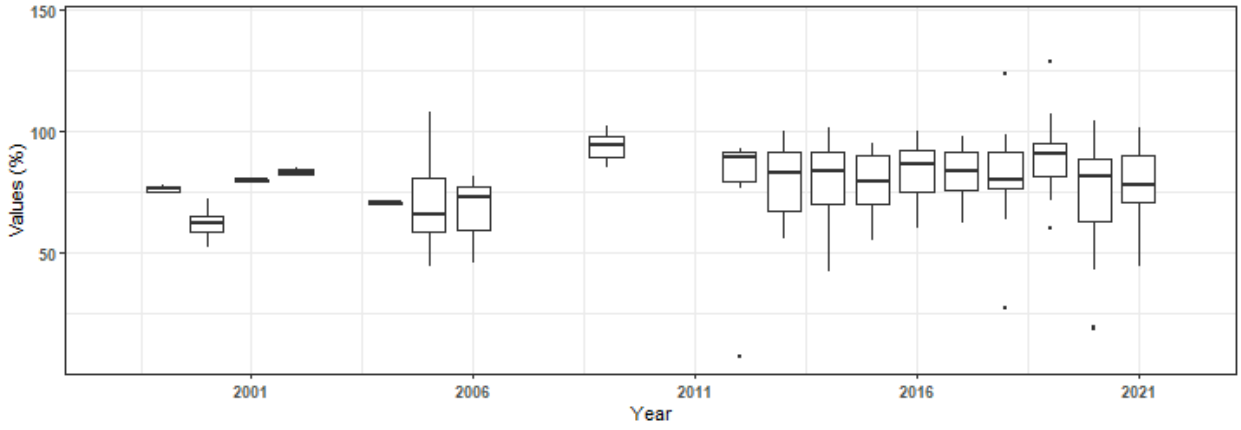
Summary Box Plots for Guana River Marsh Aquatic Preserve

By Year

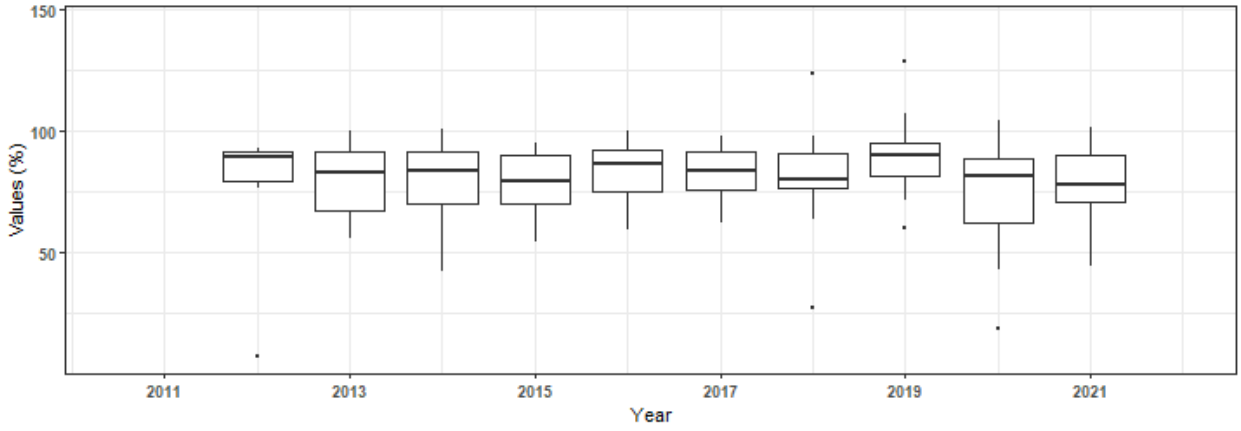
Autoscale



Scaled to 4x Standard Deviation

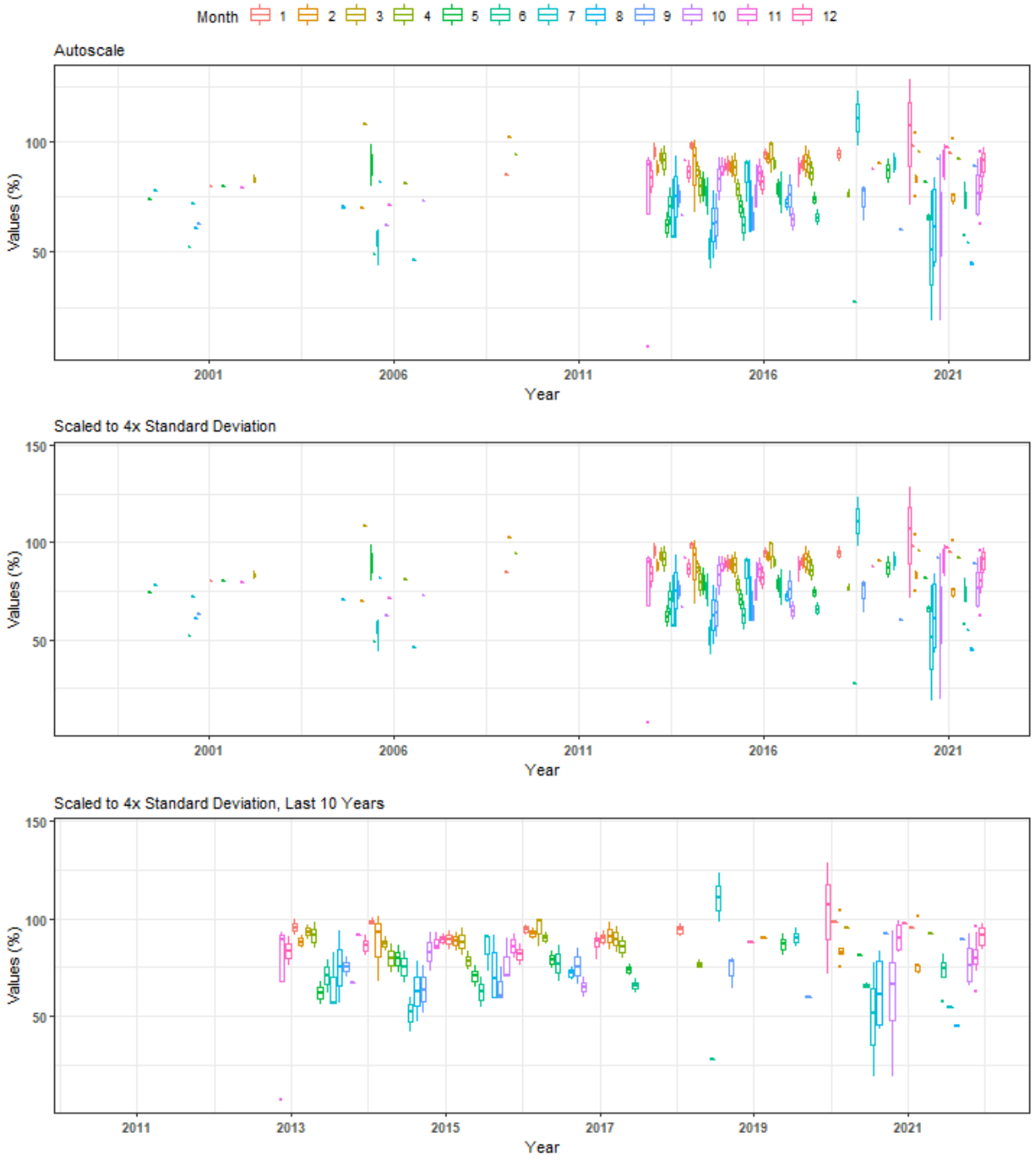


Scaled to 4x Standard Deviation, Last 10 Years



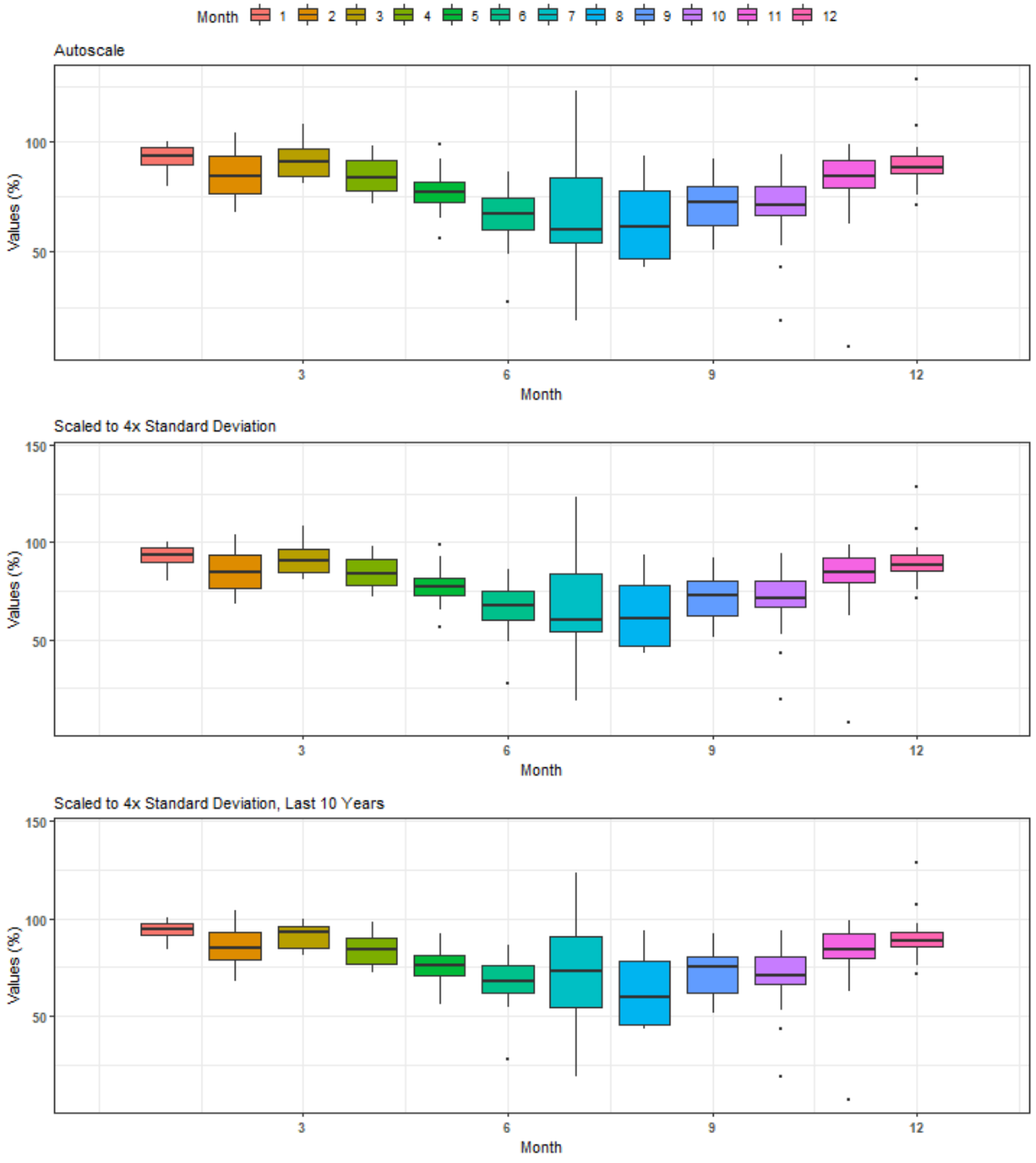
Summary Box Plots for Guana River Marsh Aquatic Preserve

By Year & Month



Summary Box Plots for Guana River Marsh Aquatic Preserve

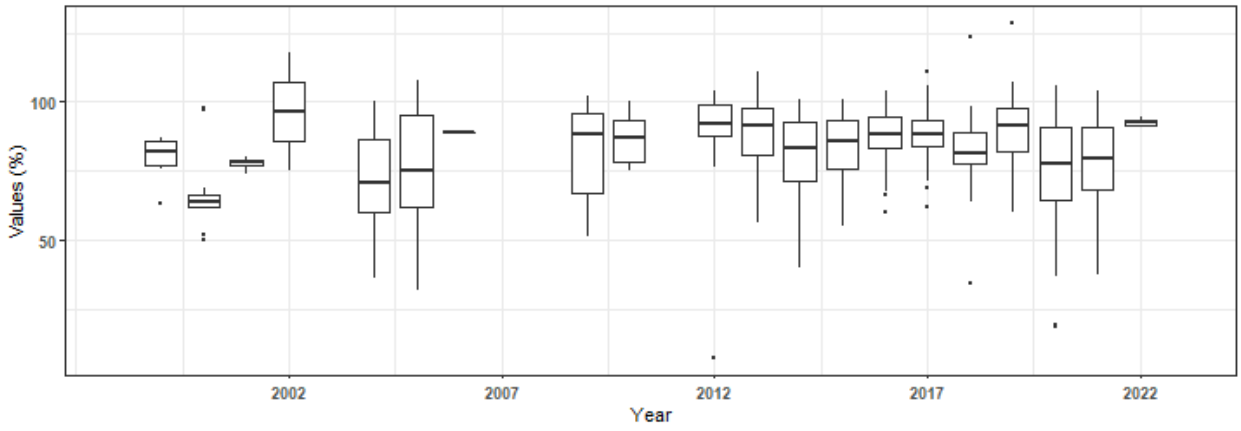
By Month



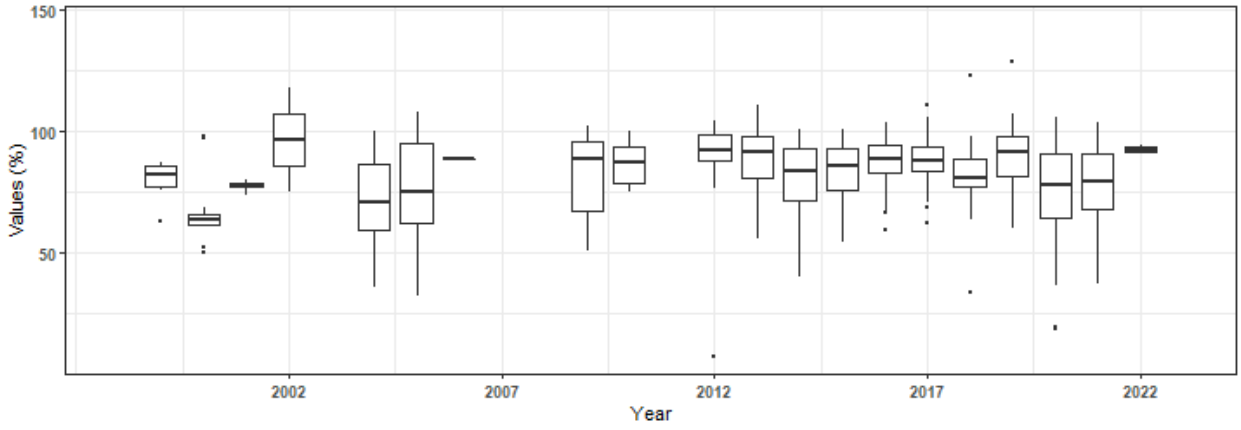
Summary Box Plots for Guana Tolomato Matanzas National Estuarine Research Reserve

By Year

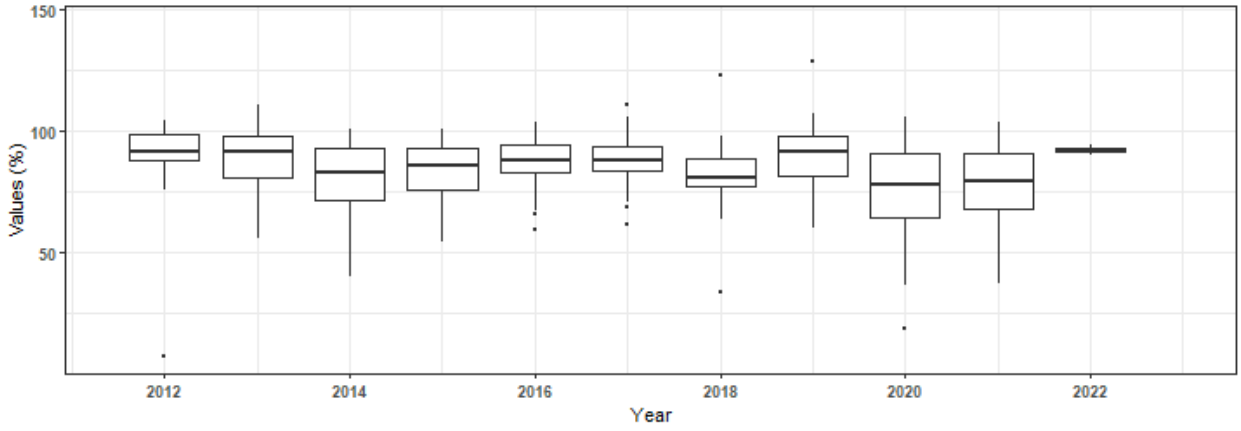
Autoscale



Scaled to 4x Standard Deviation

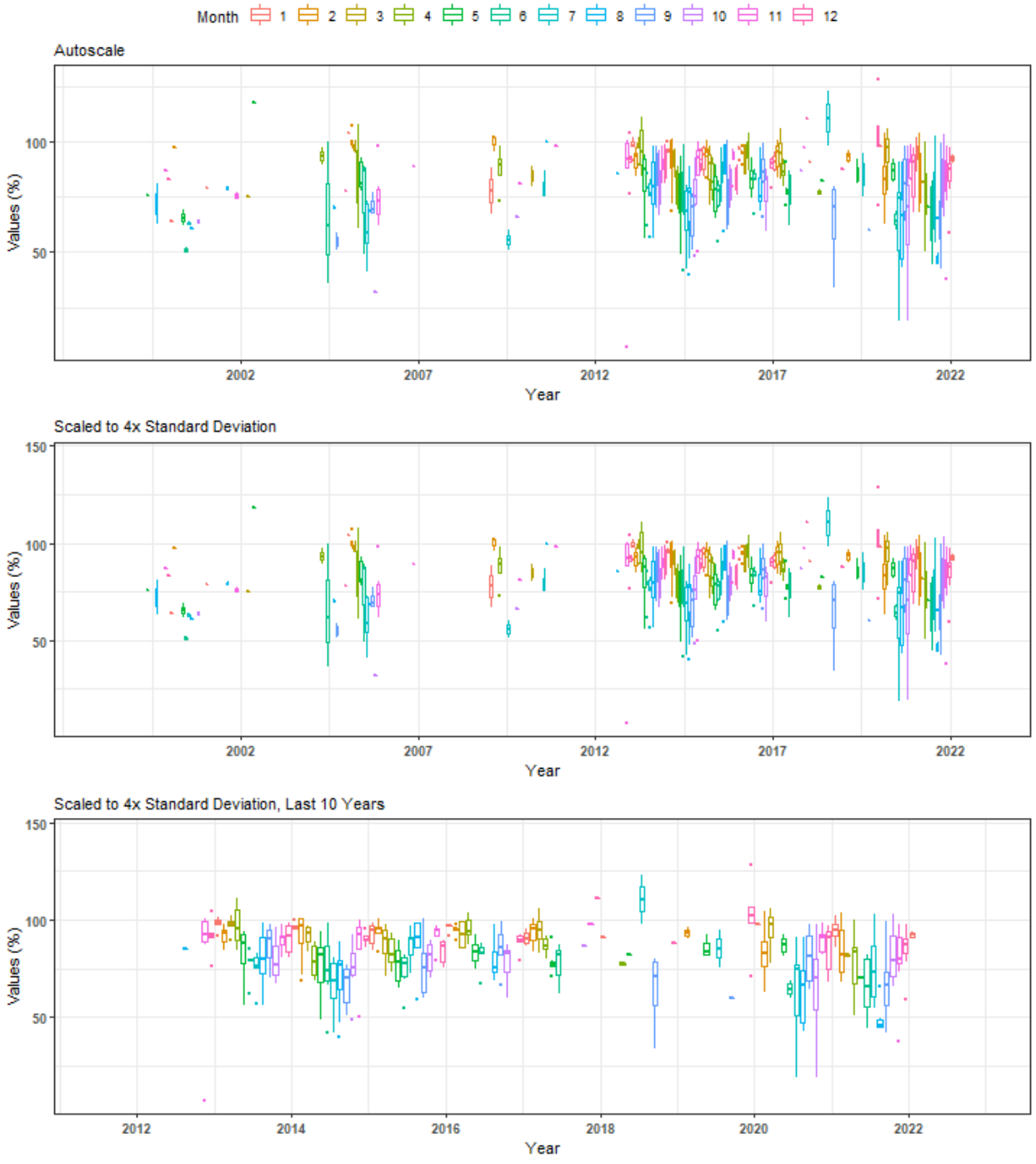


Scaled to 4x Standard Deviation, Last 10 Years



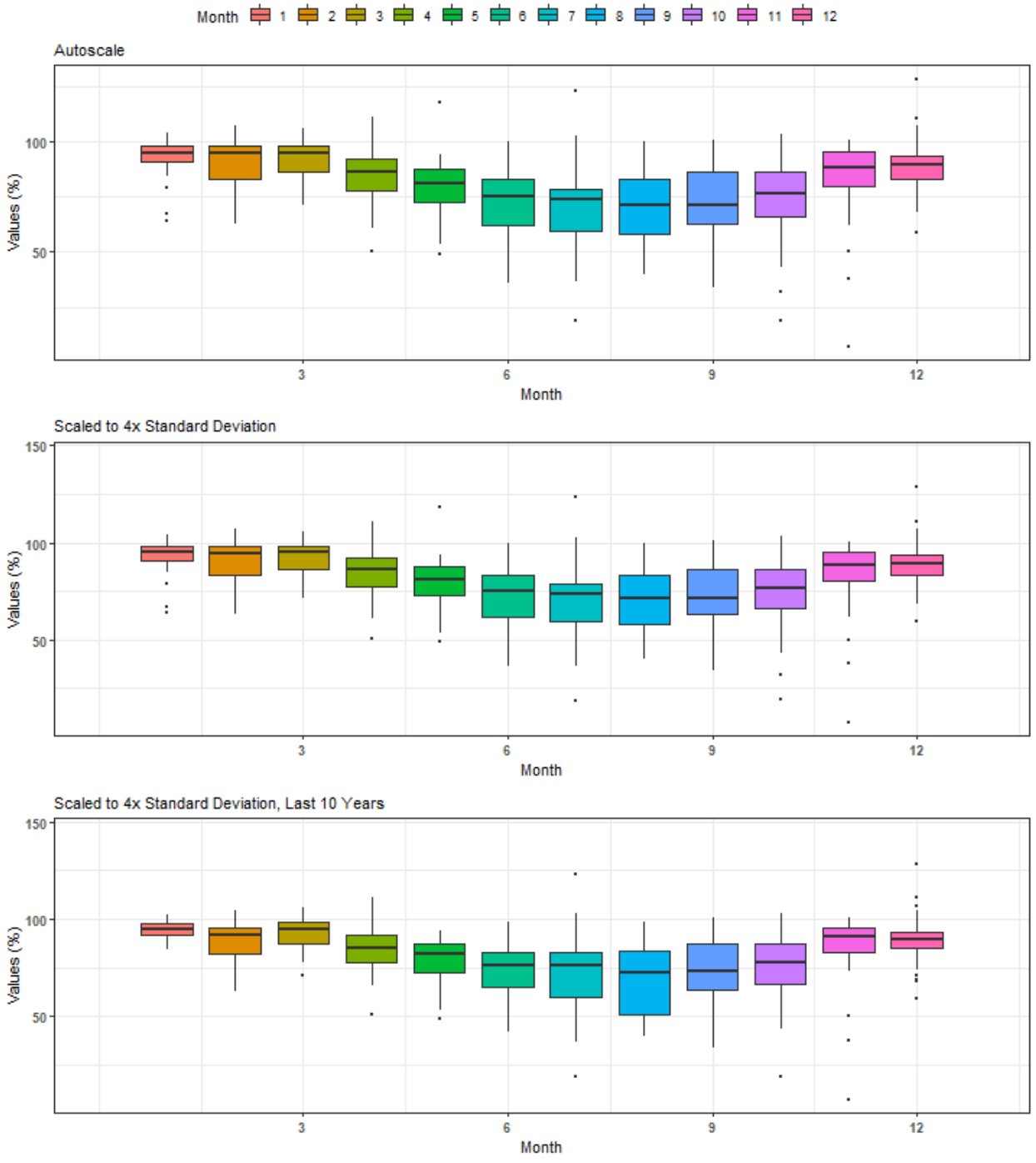
Summary Box Plots for Guana Tolomato Matanzas National Estuarine Research Reserve

By Year & Month



Summary Box Plots for Guana Tolomato Matanzas National Estuarine Research Reserve

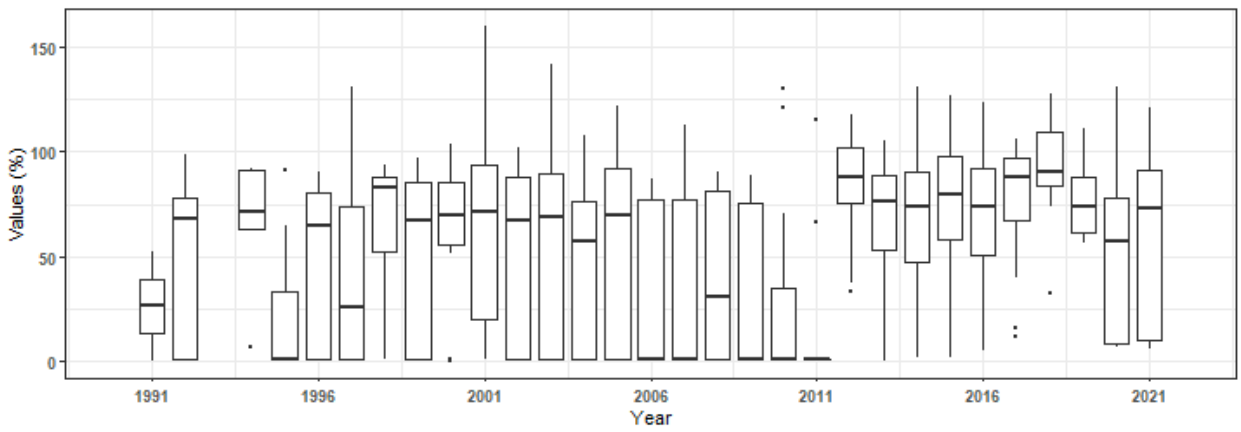
By Month



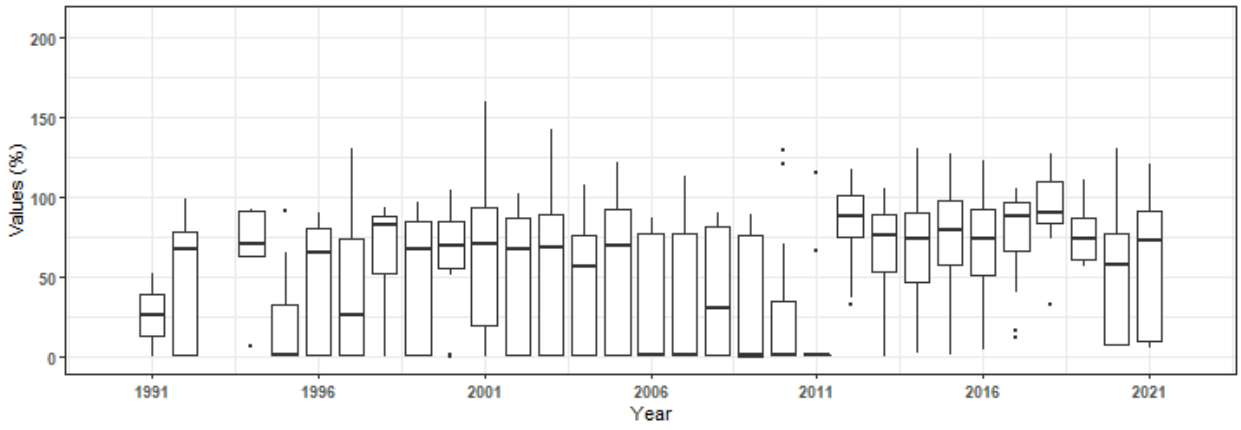
Summary Box Plots for Indian River-Malabar to Vero Beach Aquatic Preserve

By Year

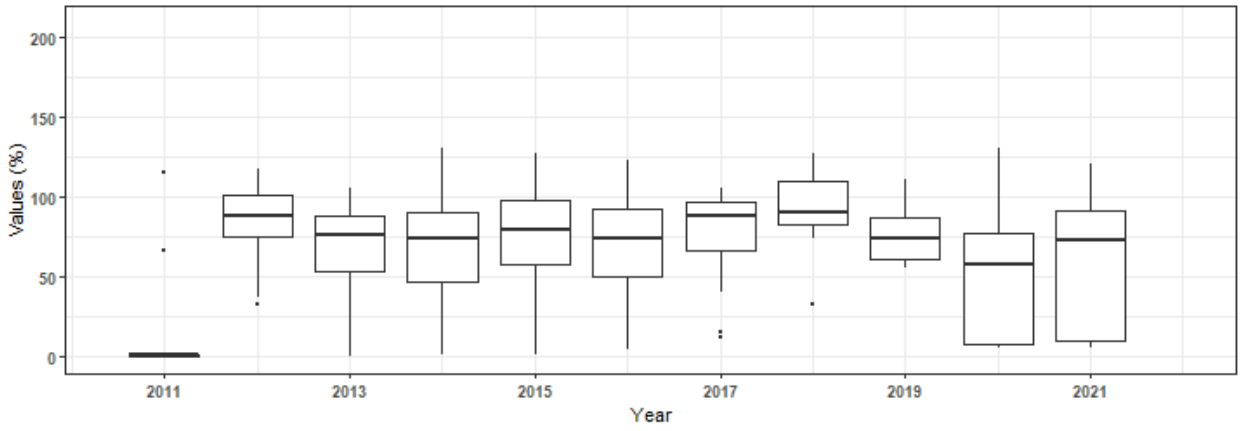
Autoscale



Scaled to 4x Standard Deviation

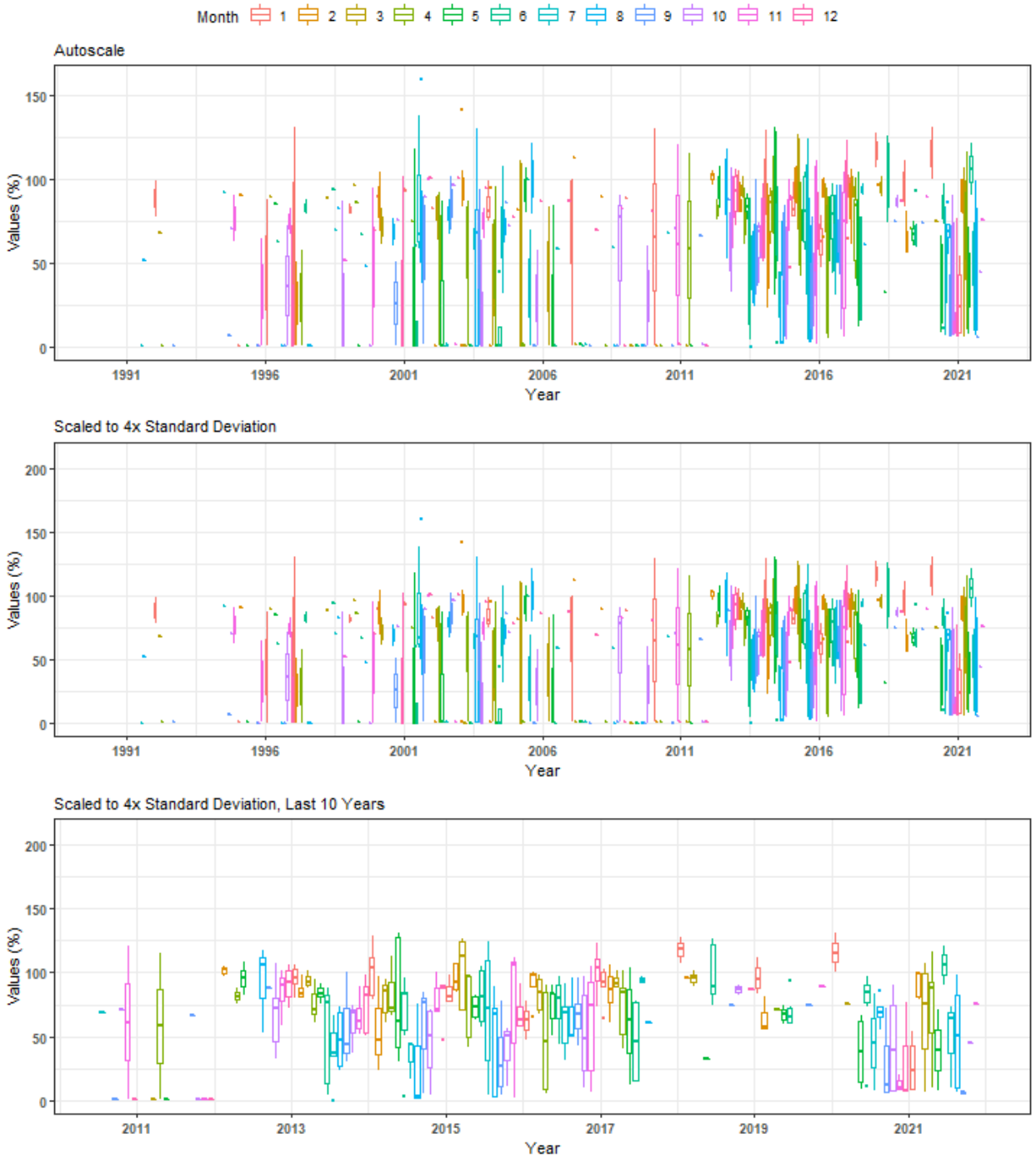


Scaled to 4x Standard Deviation, Last 10 Years



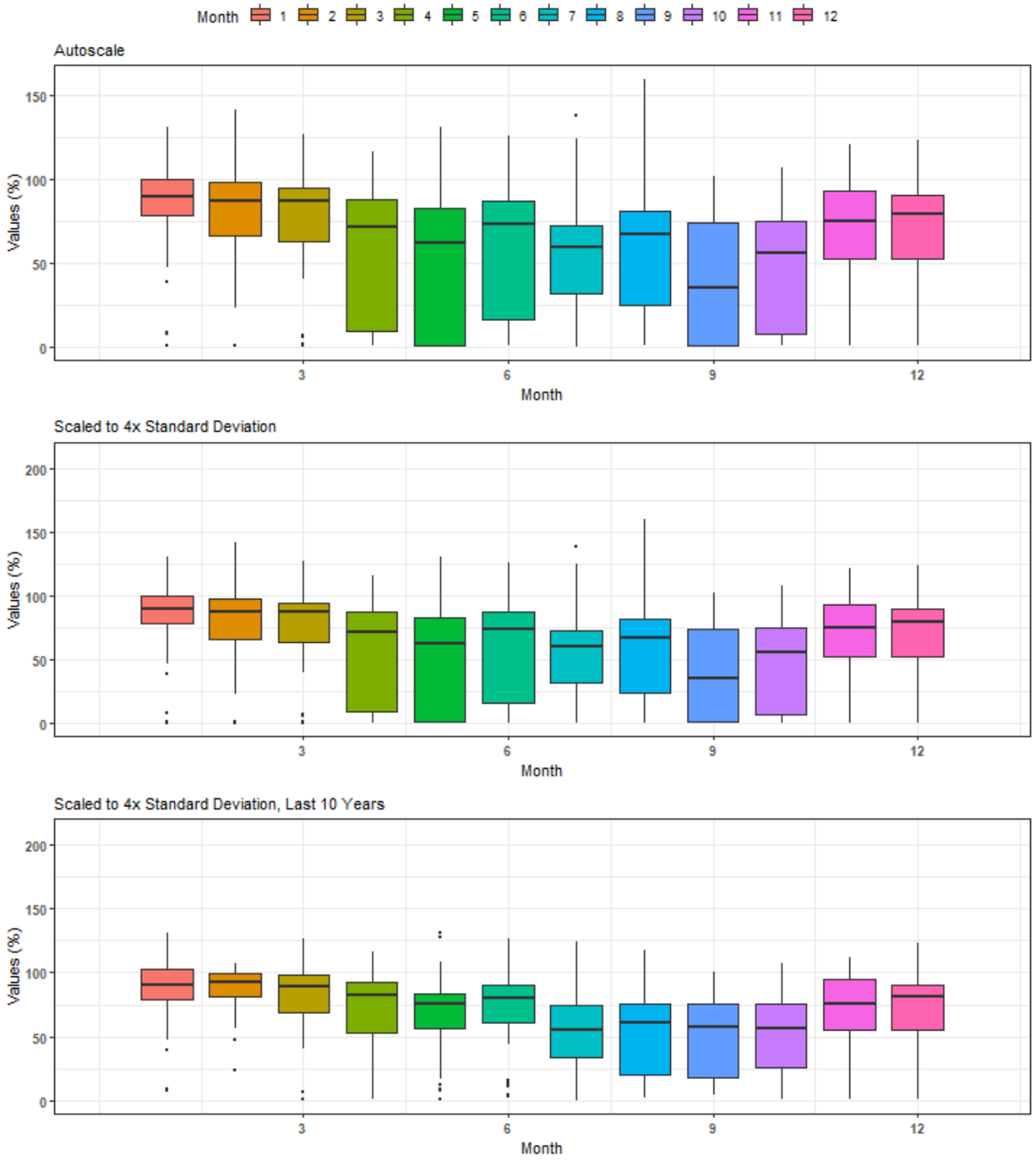
Summary Box Plots for Indian River-Malabar to Vero Beach Aquatic Preserve

By Year & Month



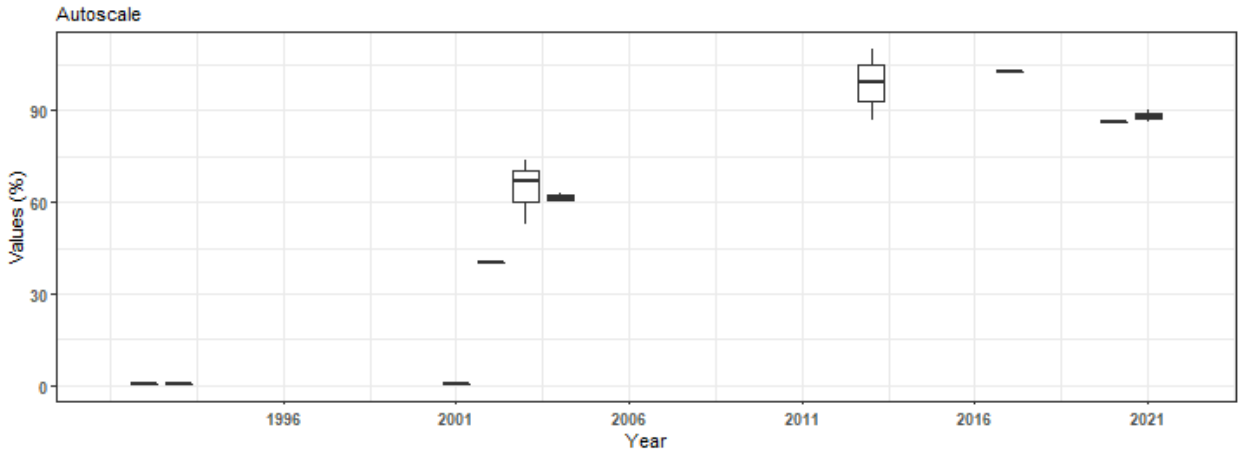
Summary Box Plots for Indian River-Malabar to Vero Beach Aquatic Preserve

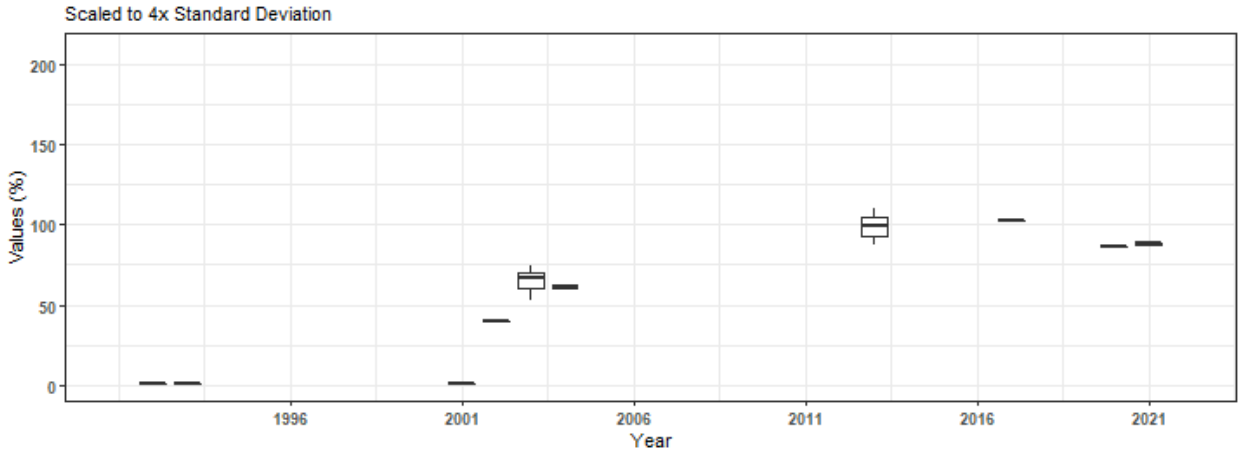
By Month

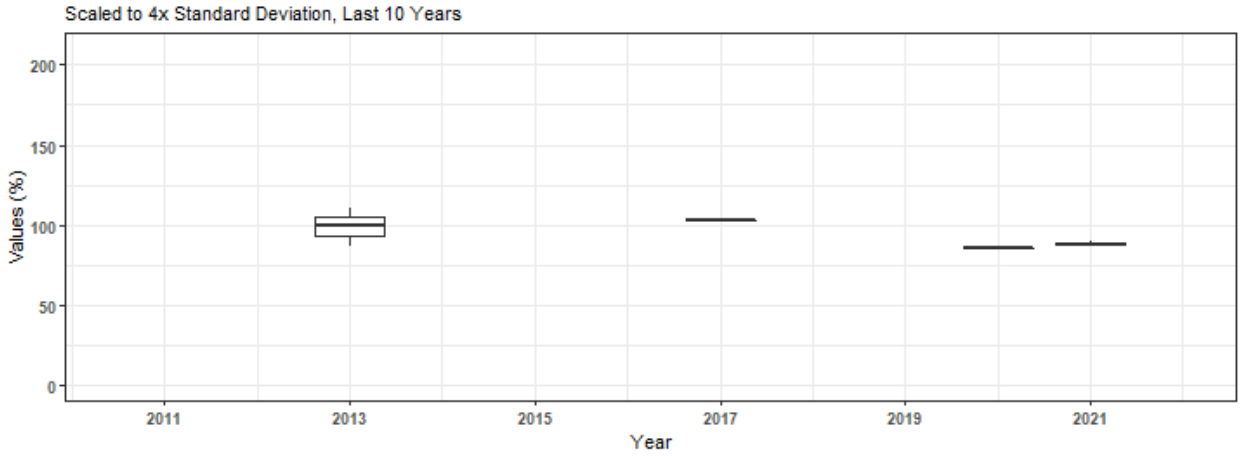


Summary Box Plots for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

By Year

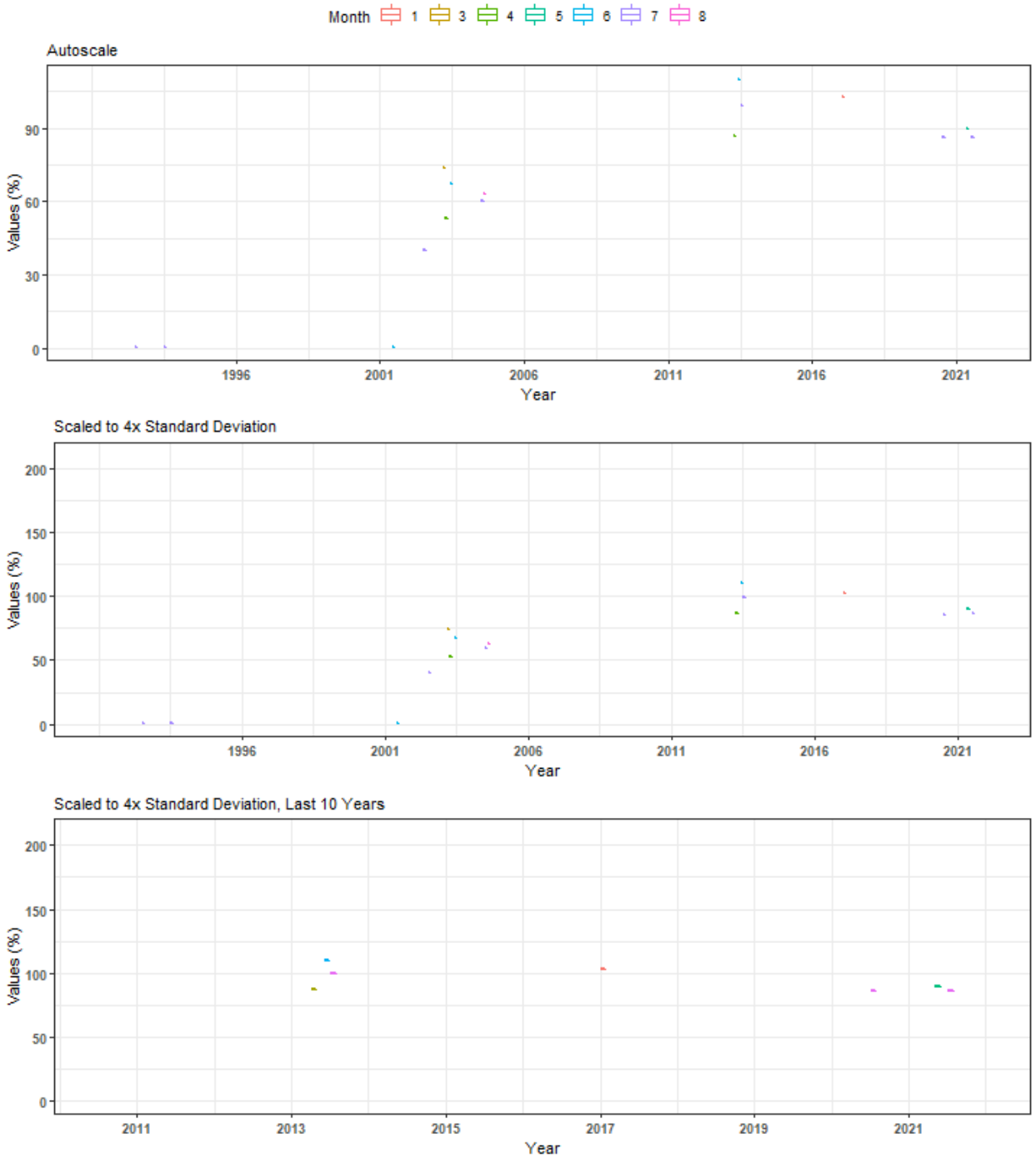






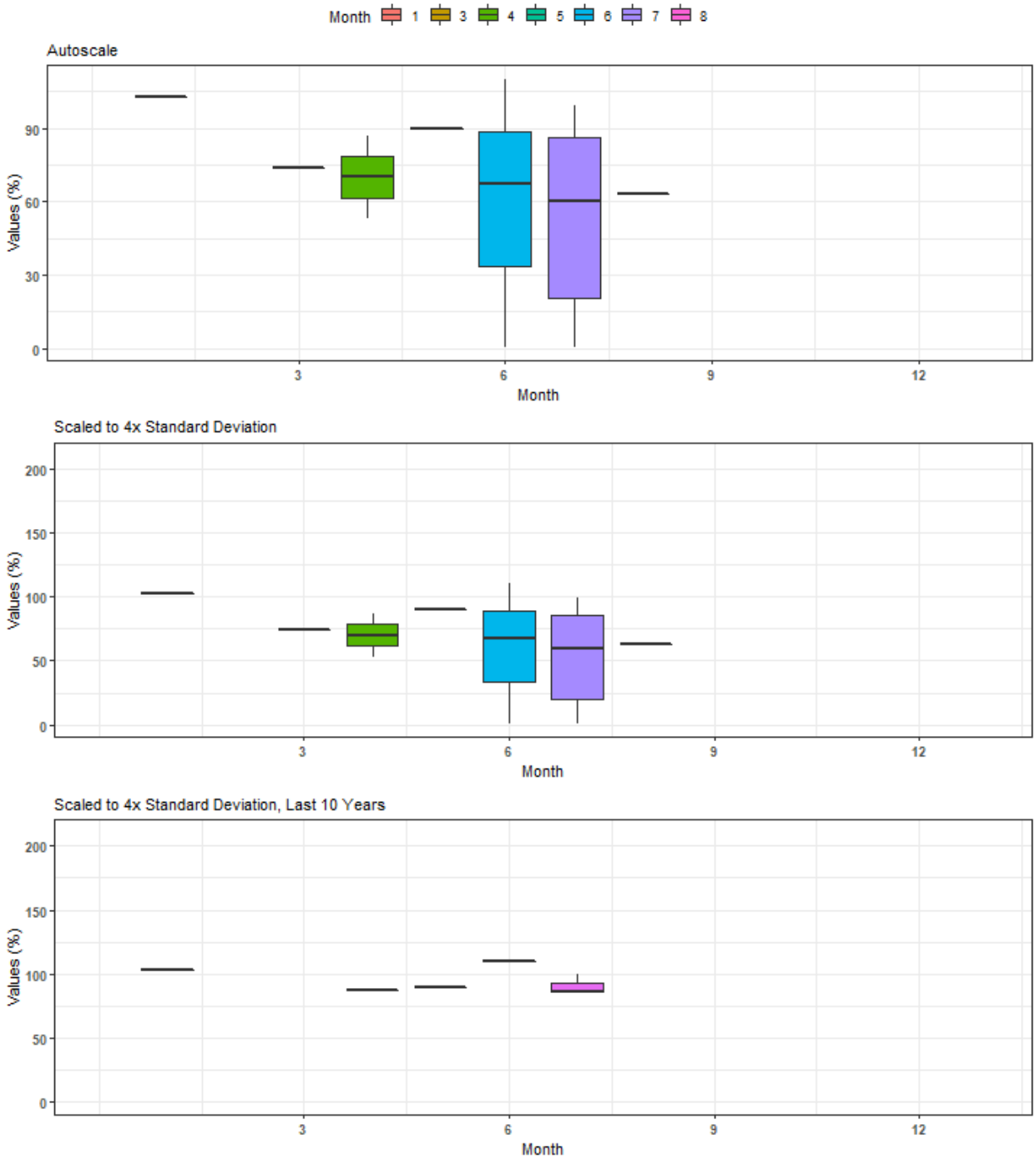
Summary Box Plots for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

By Year & Month



Summary Box Plots for Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

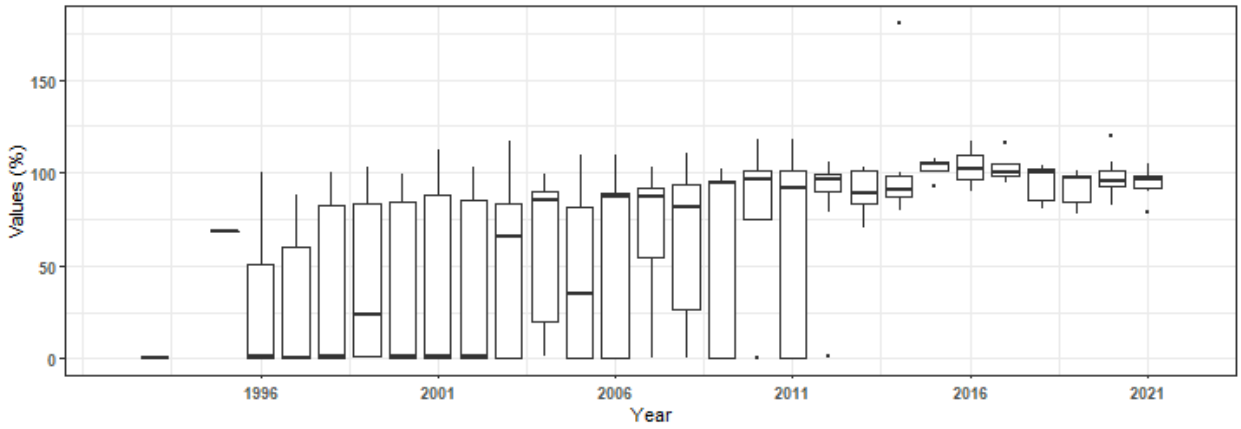
By Month



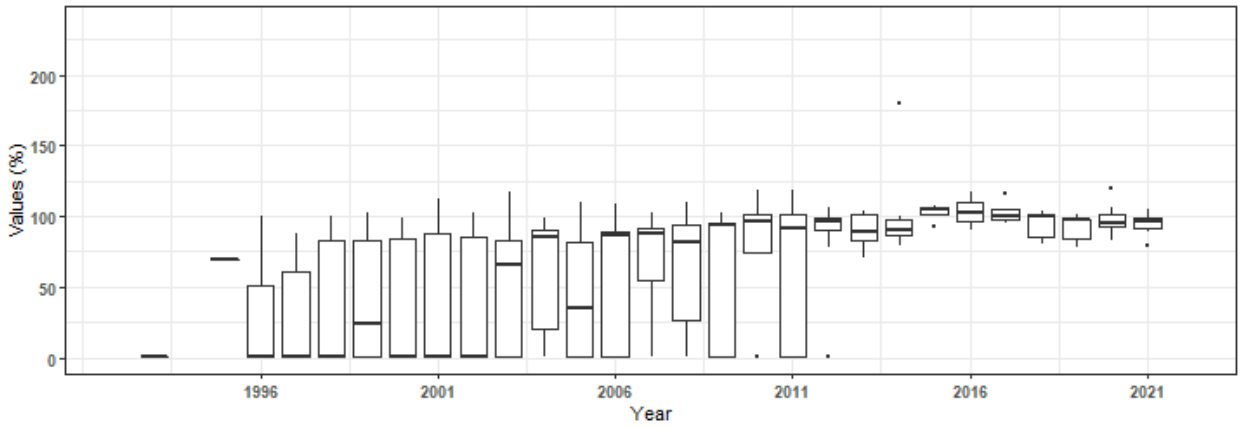
Summary Box Plots for Jensen Beach to Jupiter Inlet Aquatic Preserve

By Year

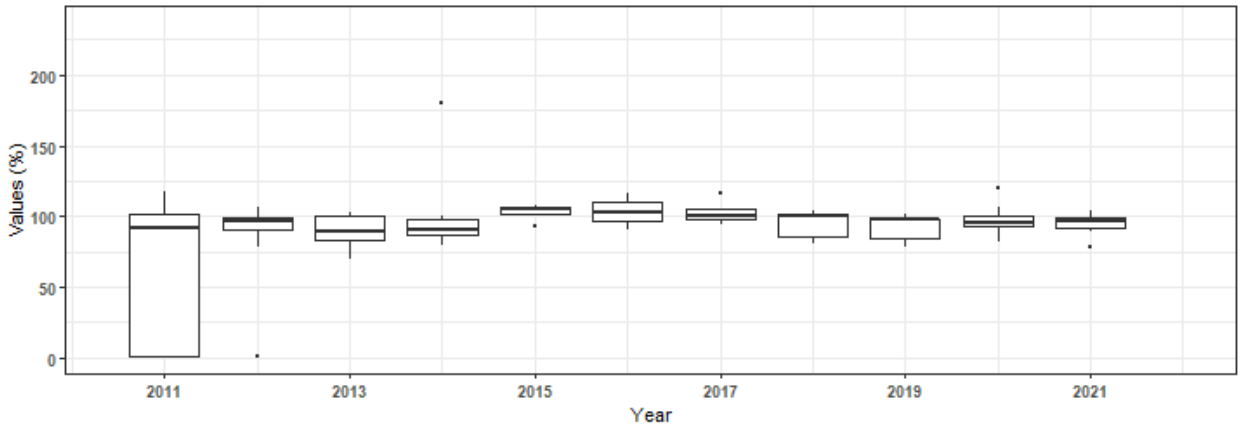
Autoscale



Scaled to 4x Standard Deviation

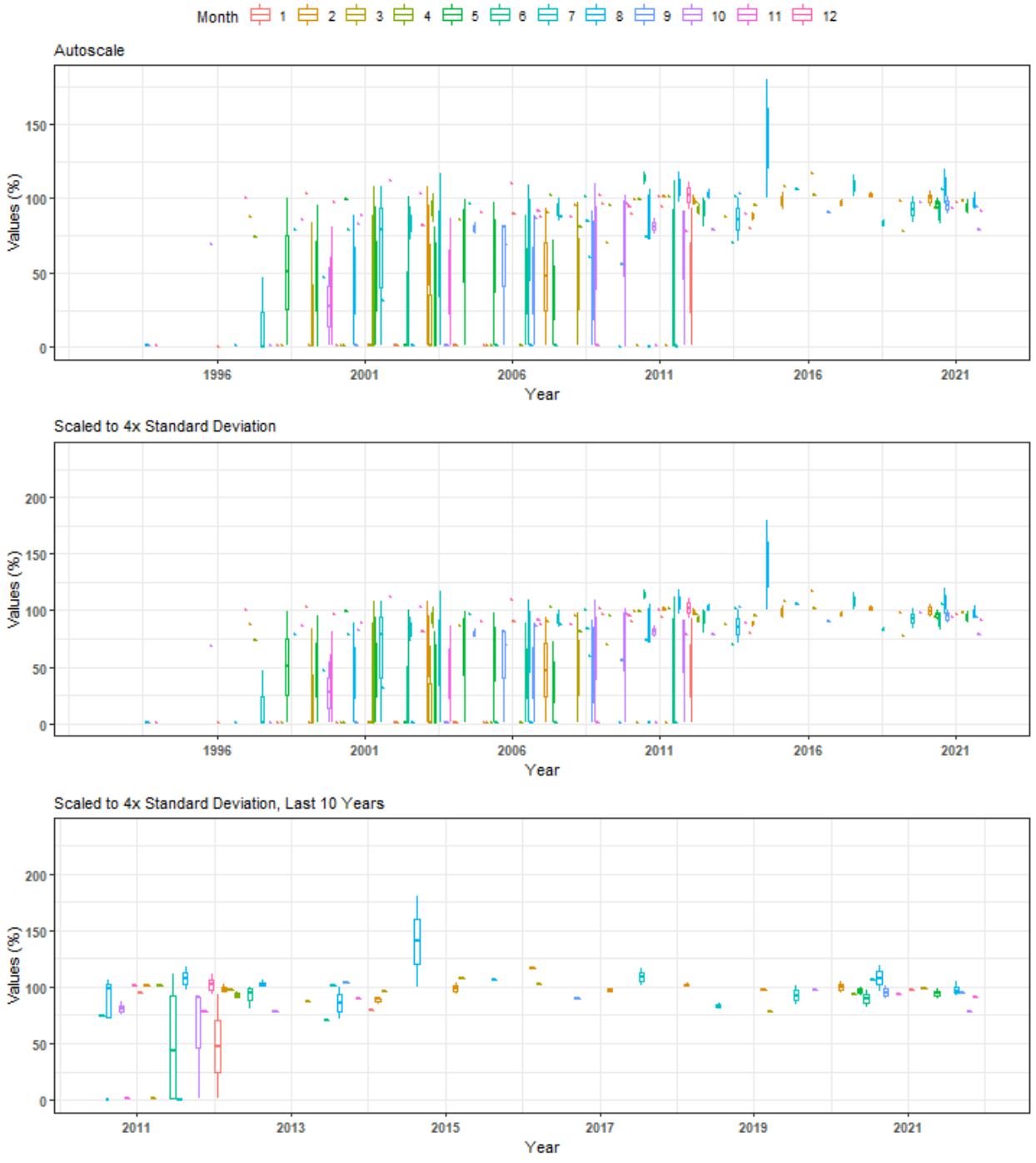


Scaled to 4x Standard Deviation, Last 10 Years



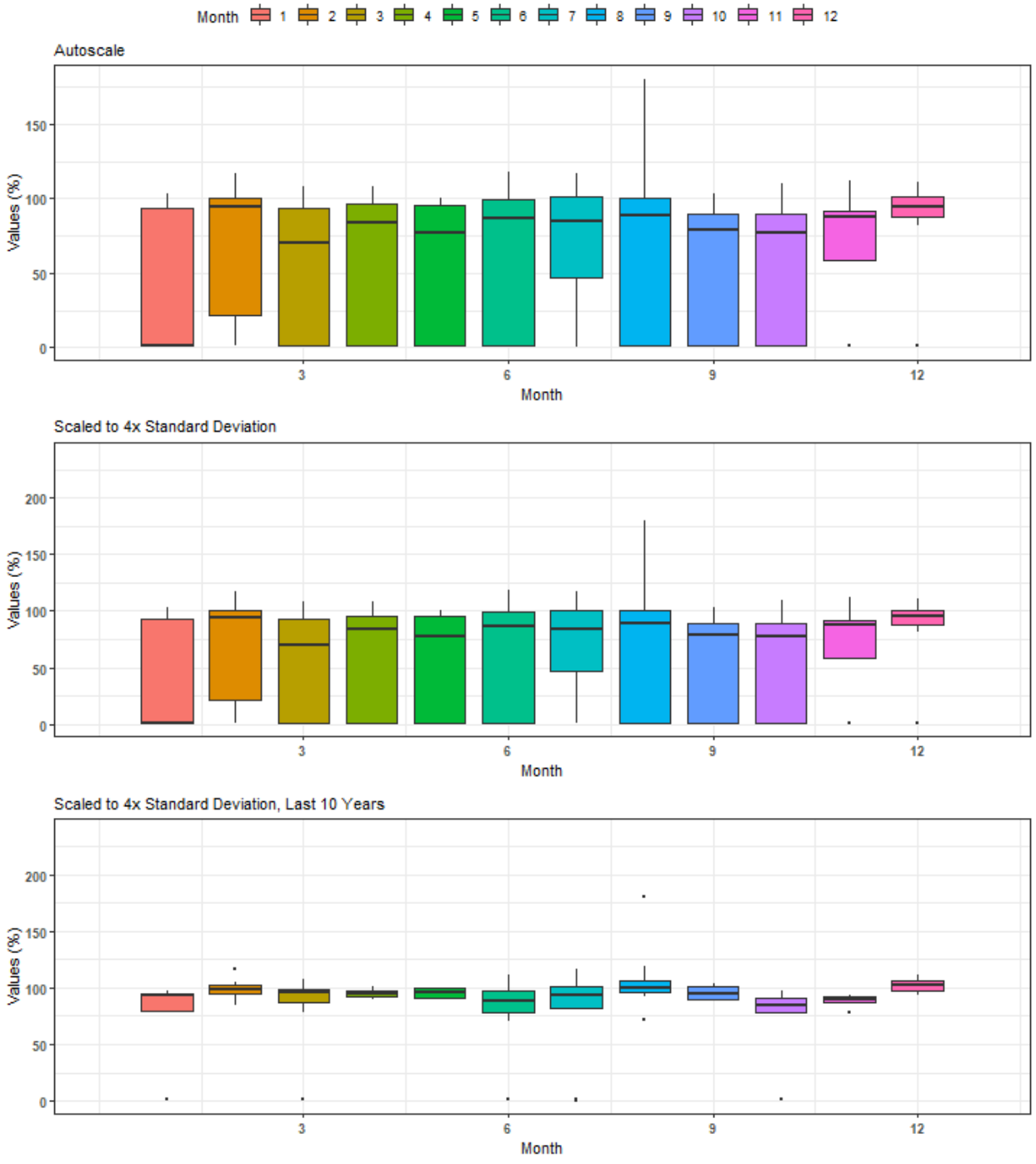
Summary Box Plots for Jensen Beach to Jupiter Inlet Aquatic Preserve

By Year & Month



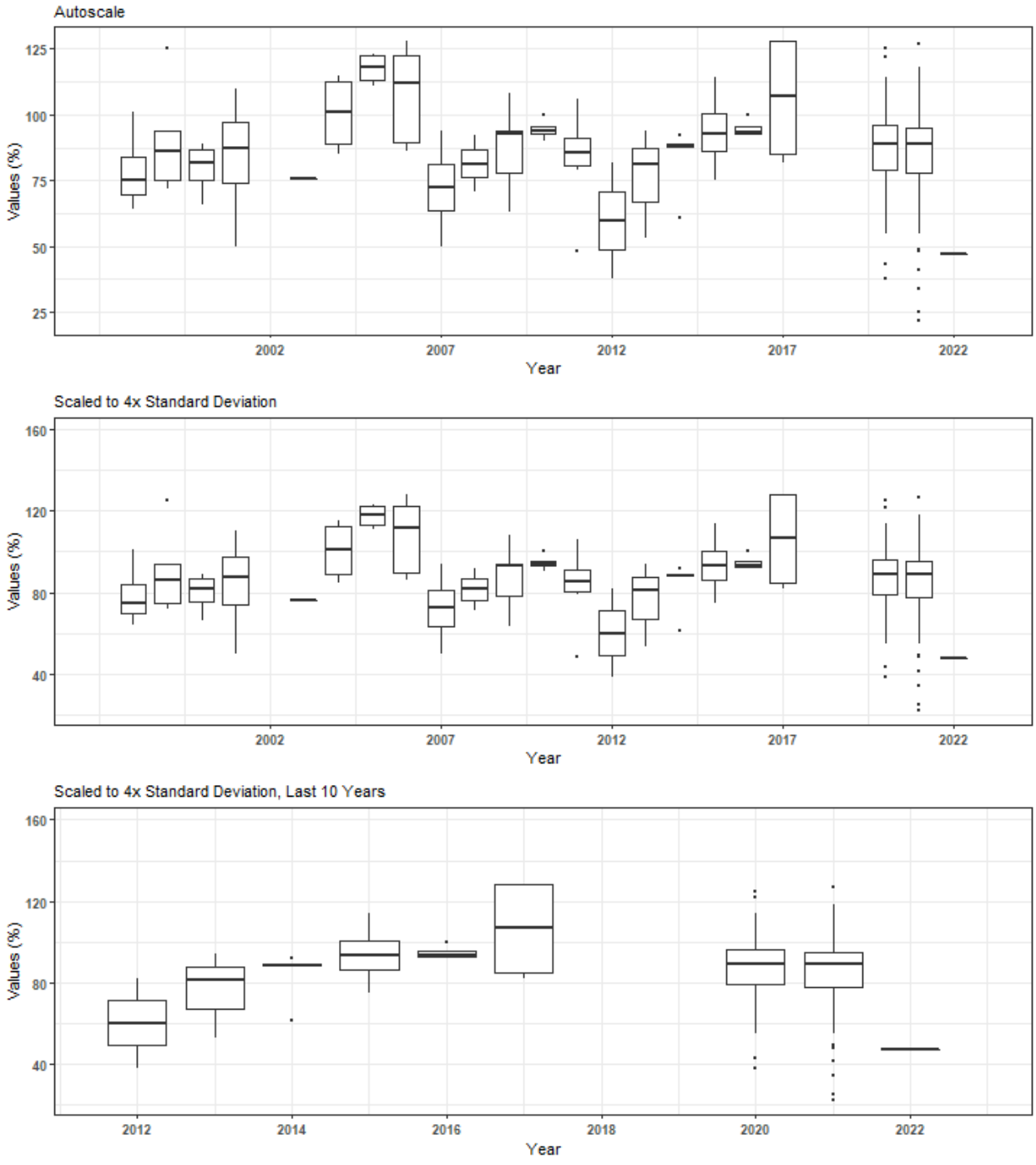
Summary Box Plots for Jensen Beach to Jupiter Inlet Aquatic Preserve

By Month



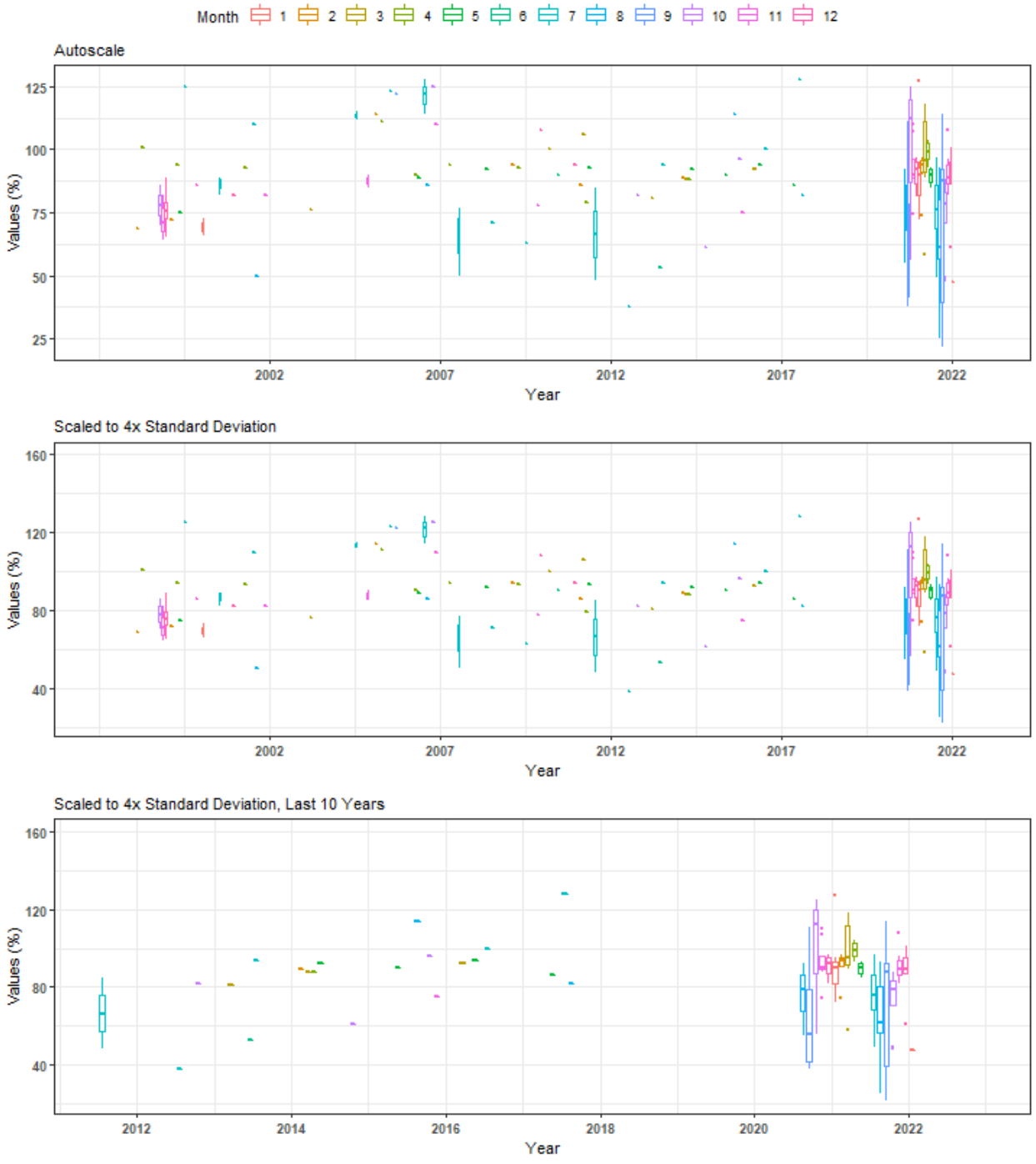
Summary Box Plots for Lemon Bay Aquatic Preserve

By Year



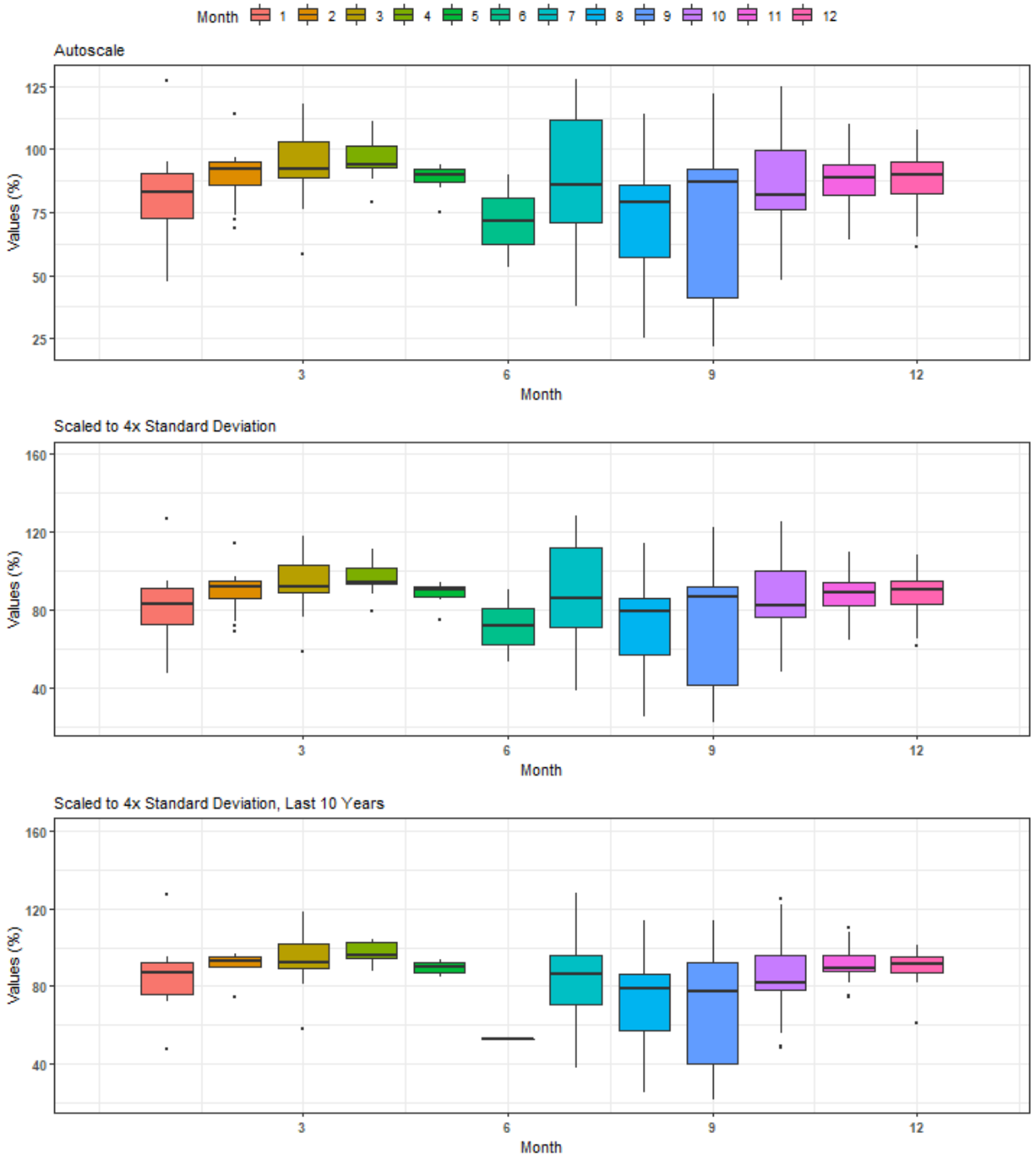
Summary Box Plots for Lemon Bay Aquatic Preserve

By Year & Month



Summary Box Plots for Lemon Bay Aquatic Preserve

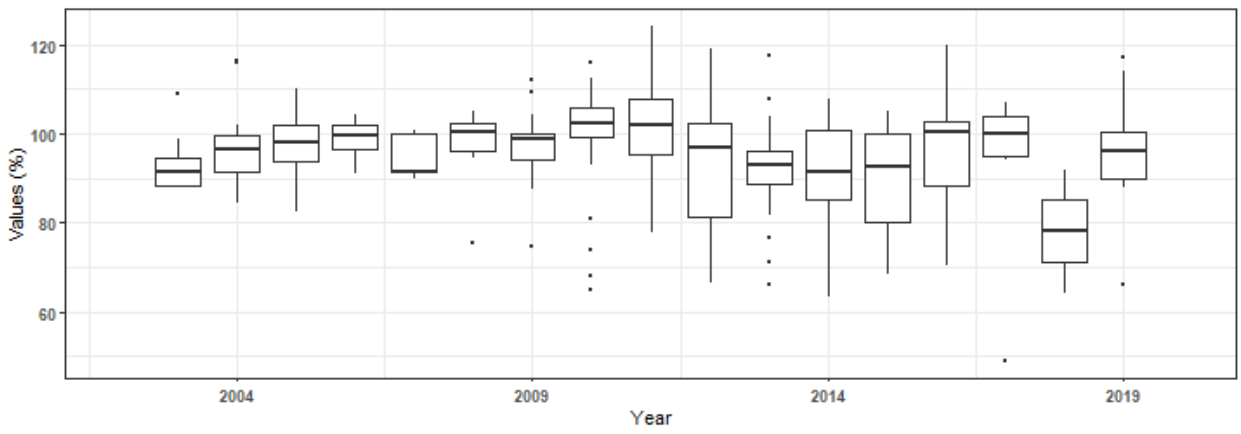
By Month



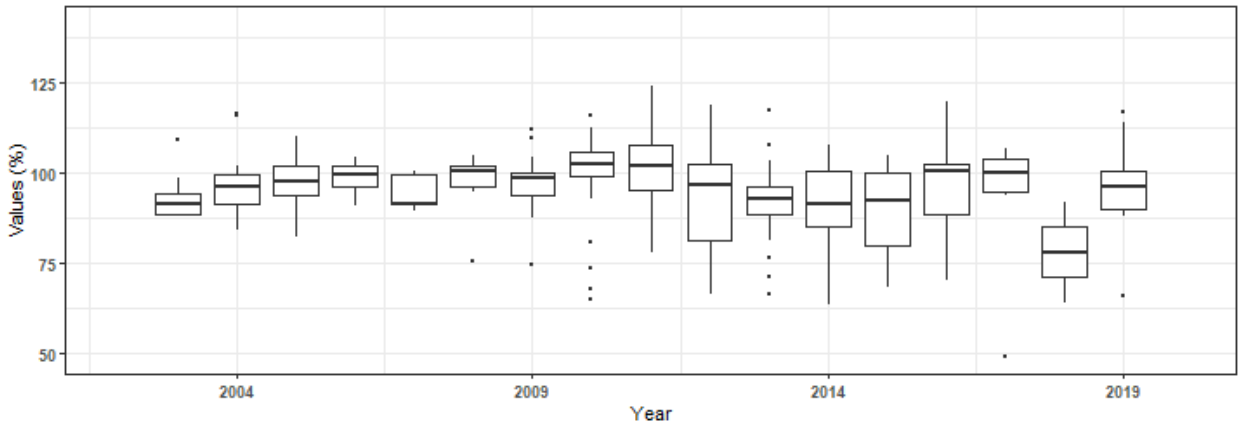
Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve

By Year

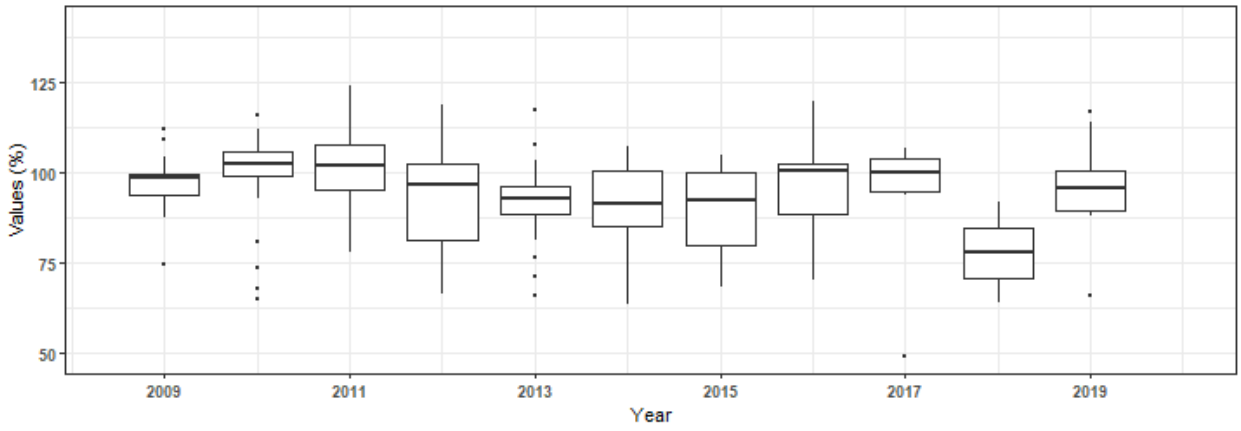
Autoscale



Scaled to 4x Standard Deviation

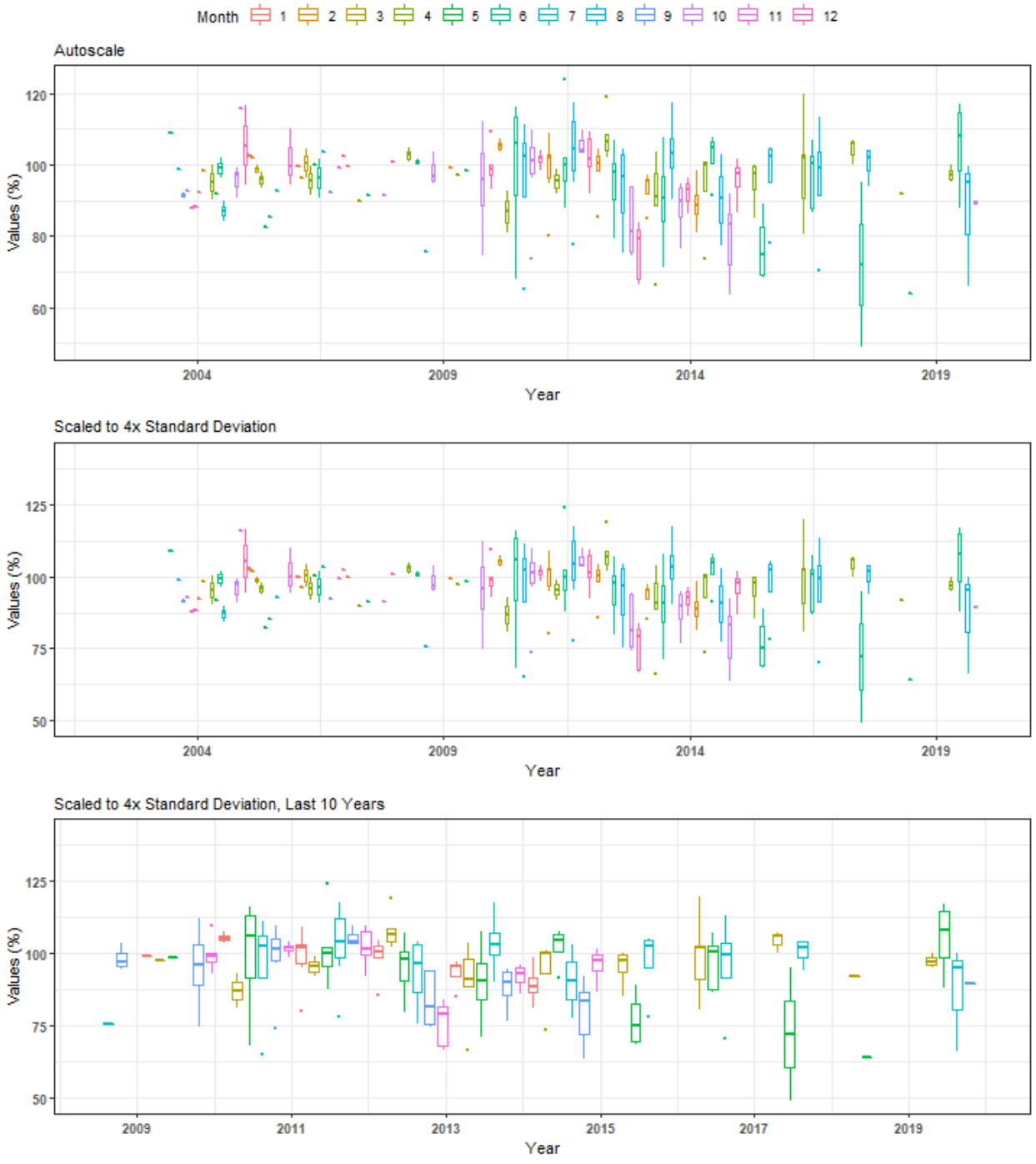


Scaled to 4x Standard Deviation, Last 10 Years



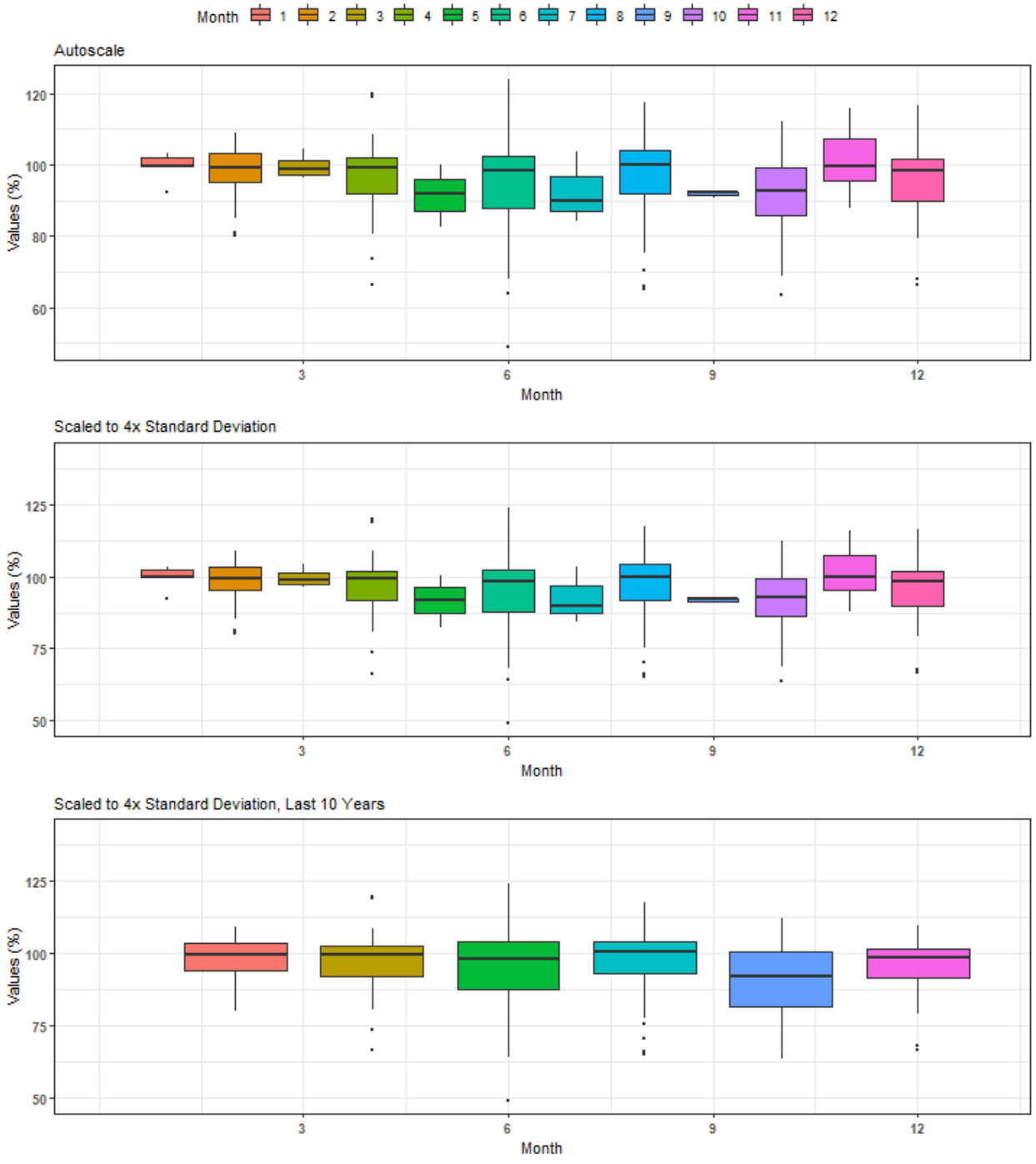
Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve

By Year & Month



Summary Box Plots for Loxahatchee River-Lake Worth Creek Aquatic Preserve

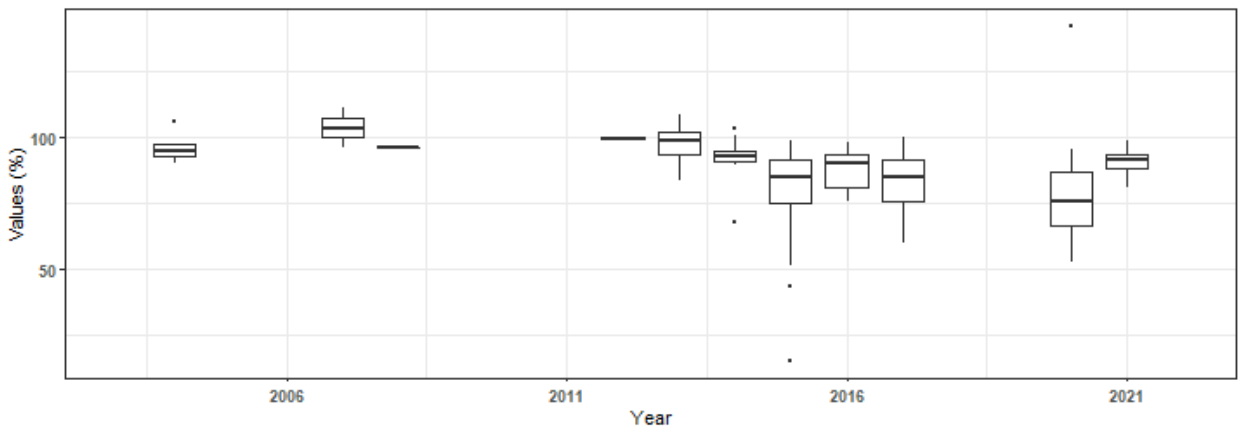
By Month



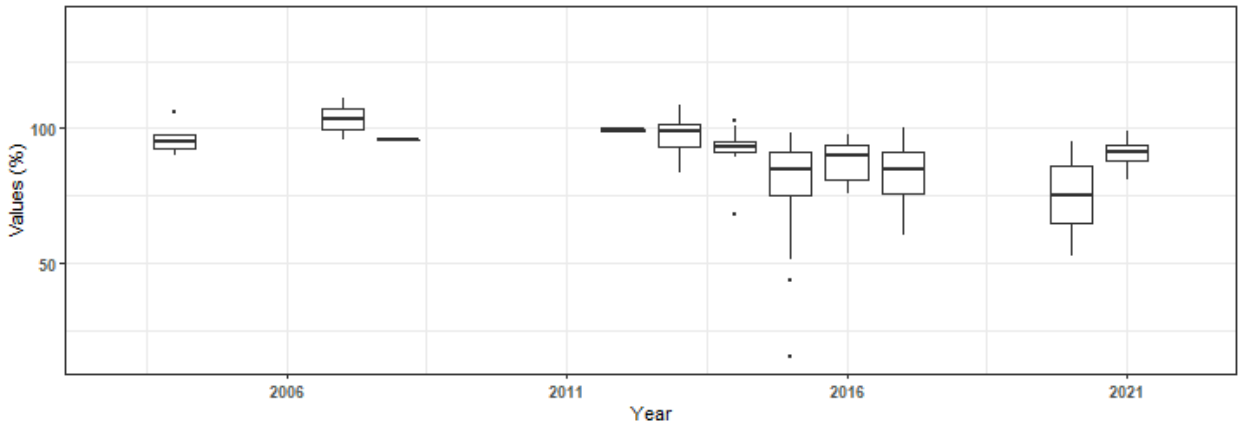
Summary Box Plots for Nassau River-St. Johns River Marshes Aquatic Preserve

By Year

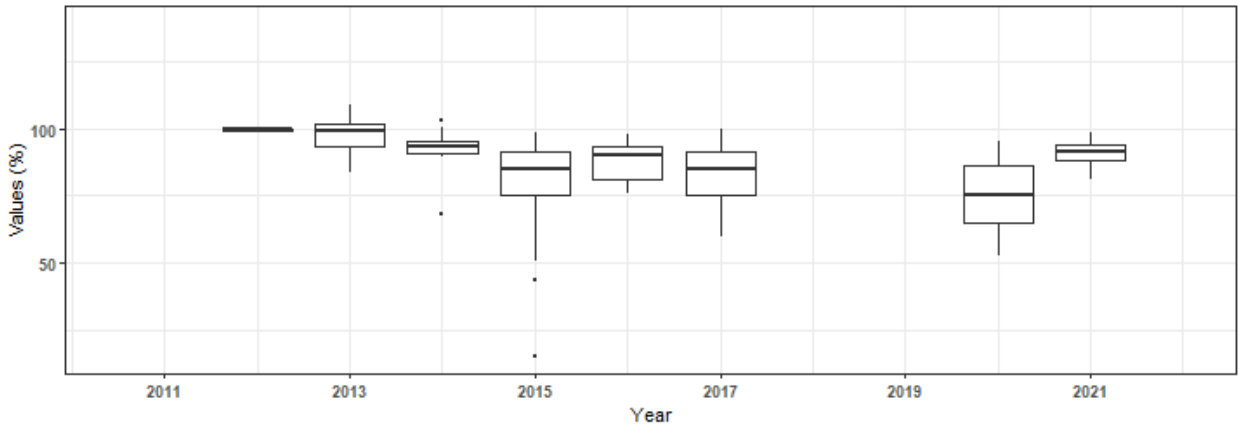
Autoscale

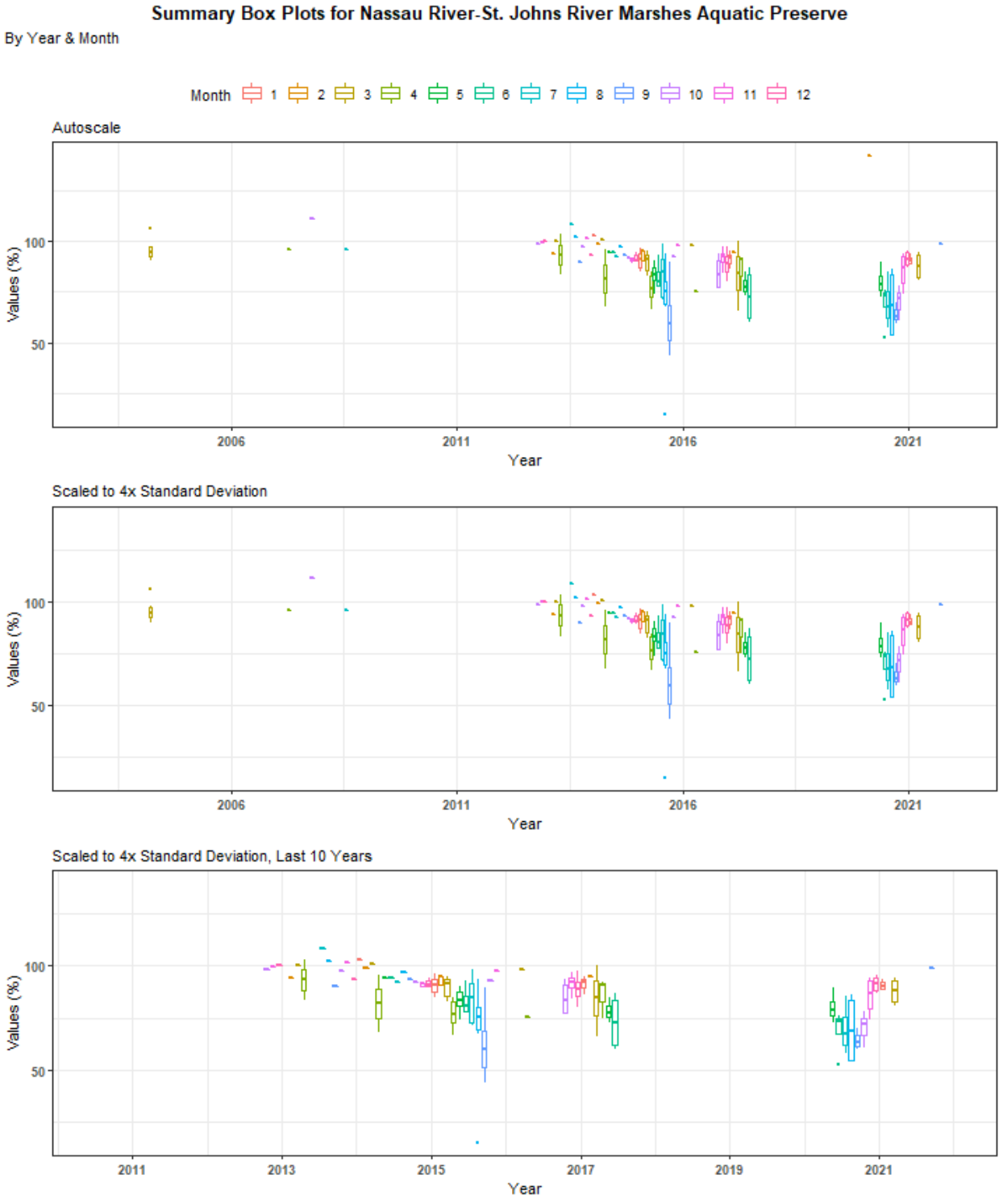


Scaled to 4x Standard Deviation



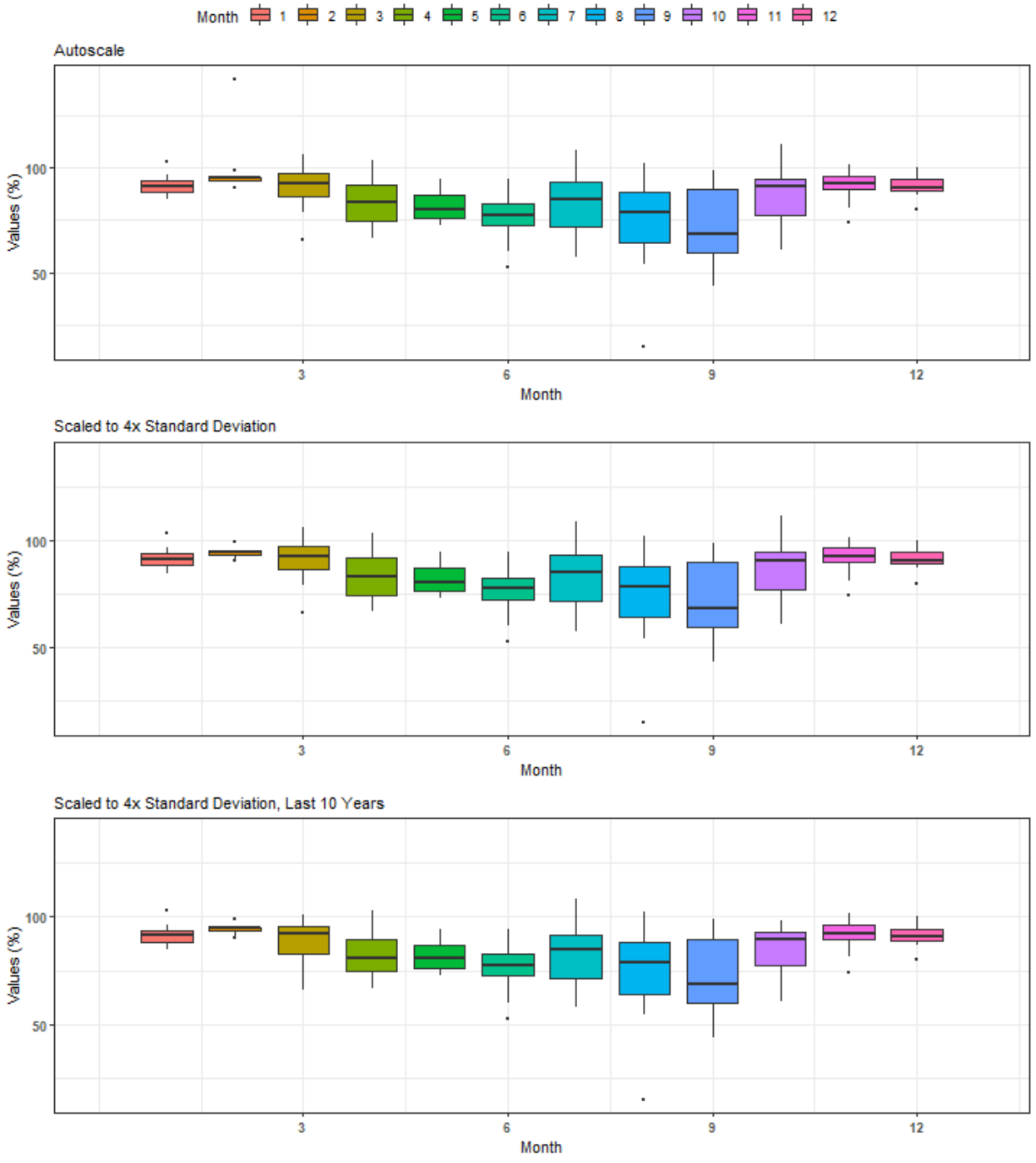
Scaled to 4x Standard Deviation, Last 10 Years




Summary Box Plots for 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

Summary Box Plots for Nassau River-St. Johns River Marshes Aquatic Preserve

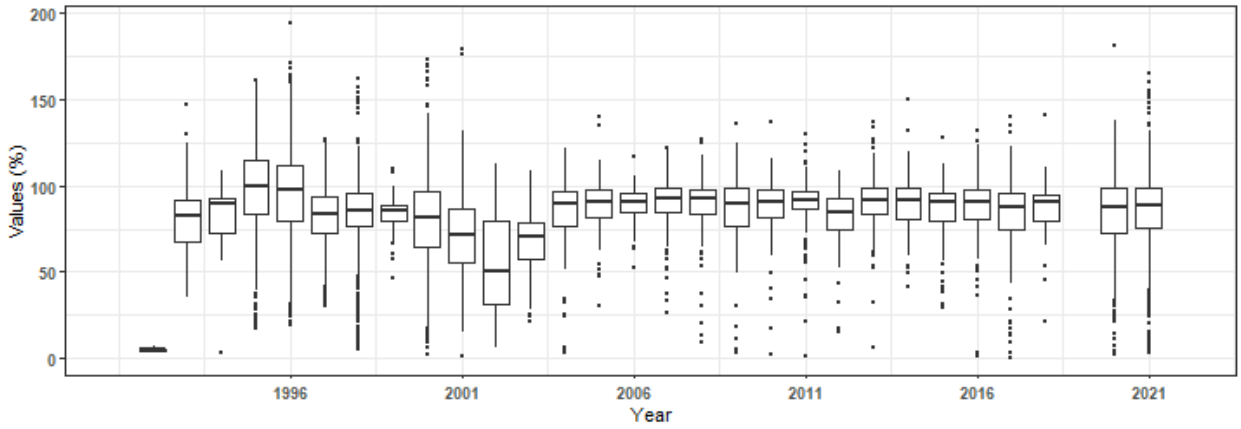
By Month



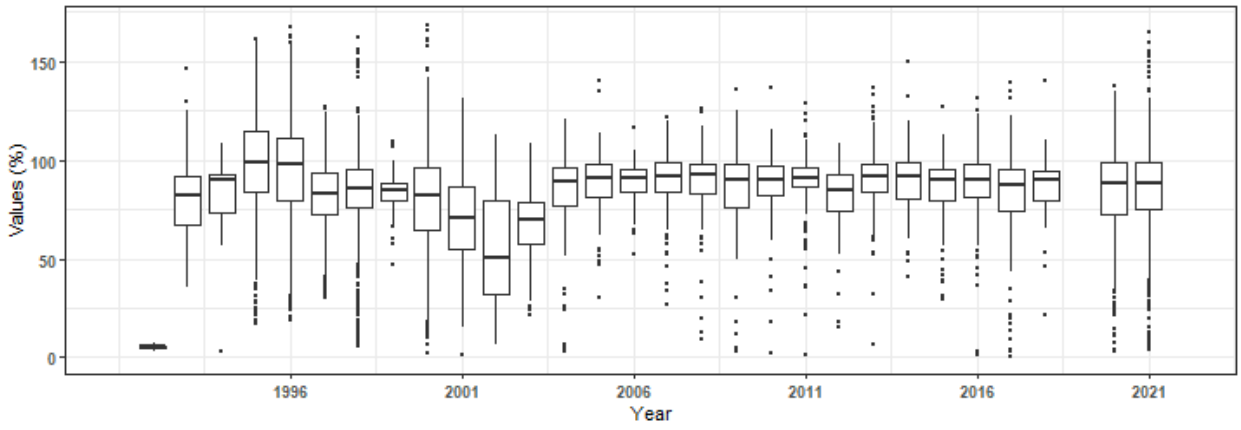
Summary Box Plots for Pinellas County Aquatic Preserve

By Year

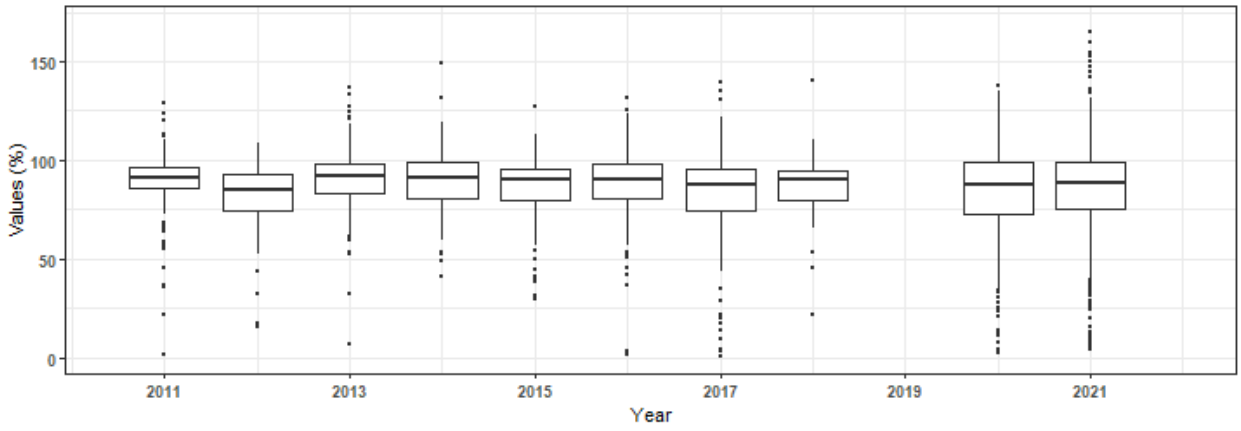
Autoscale



Scaled to 4x Standard Deviation

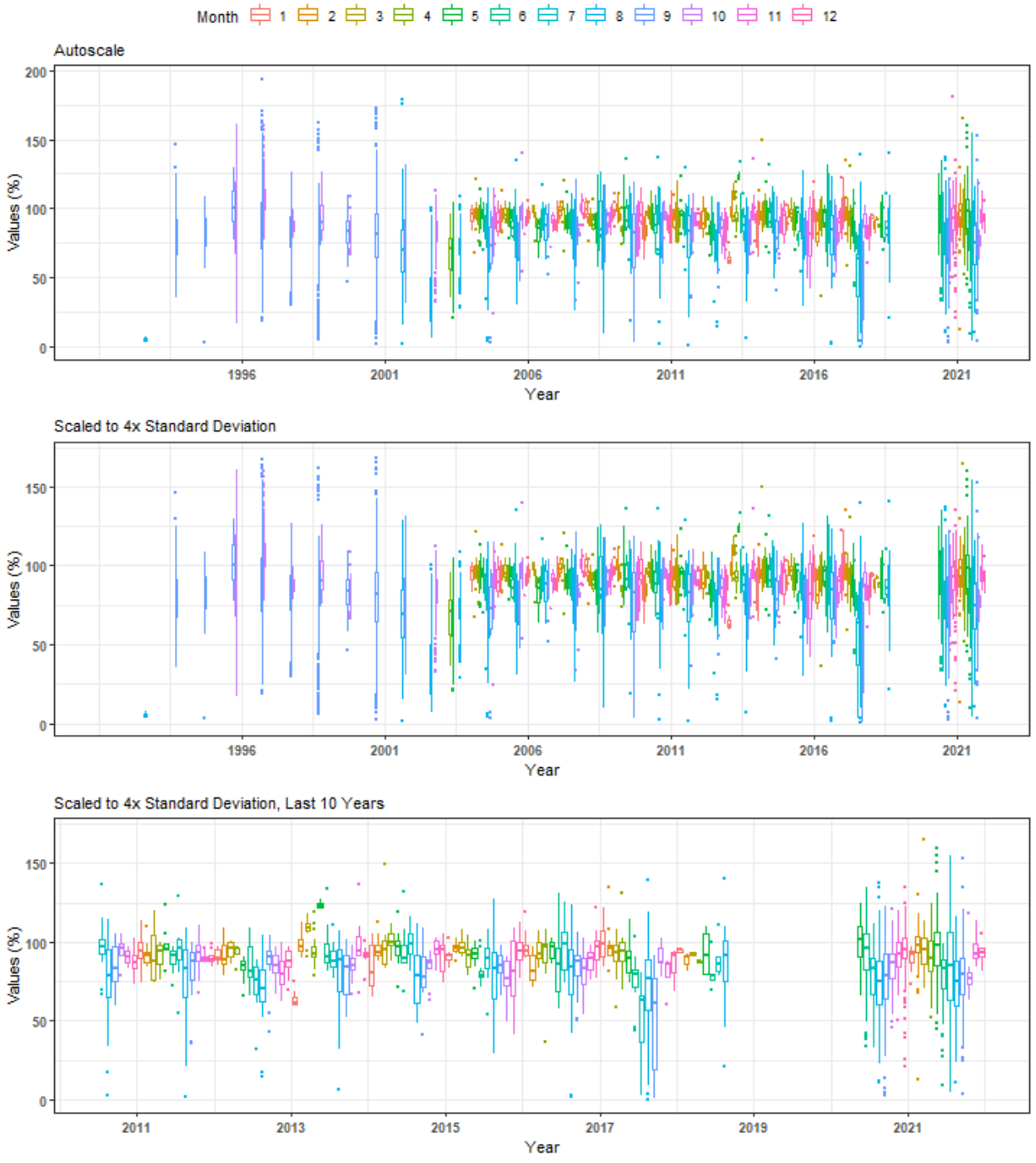


Scaled to 4x Standard Deviation, Last 10 Years



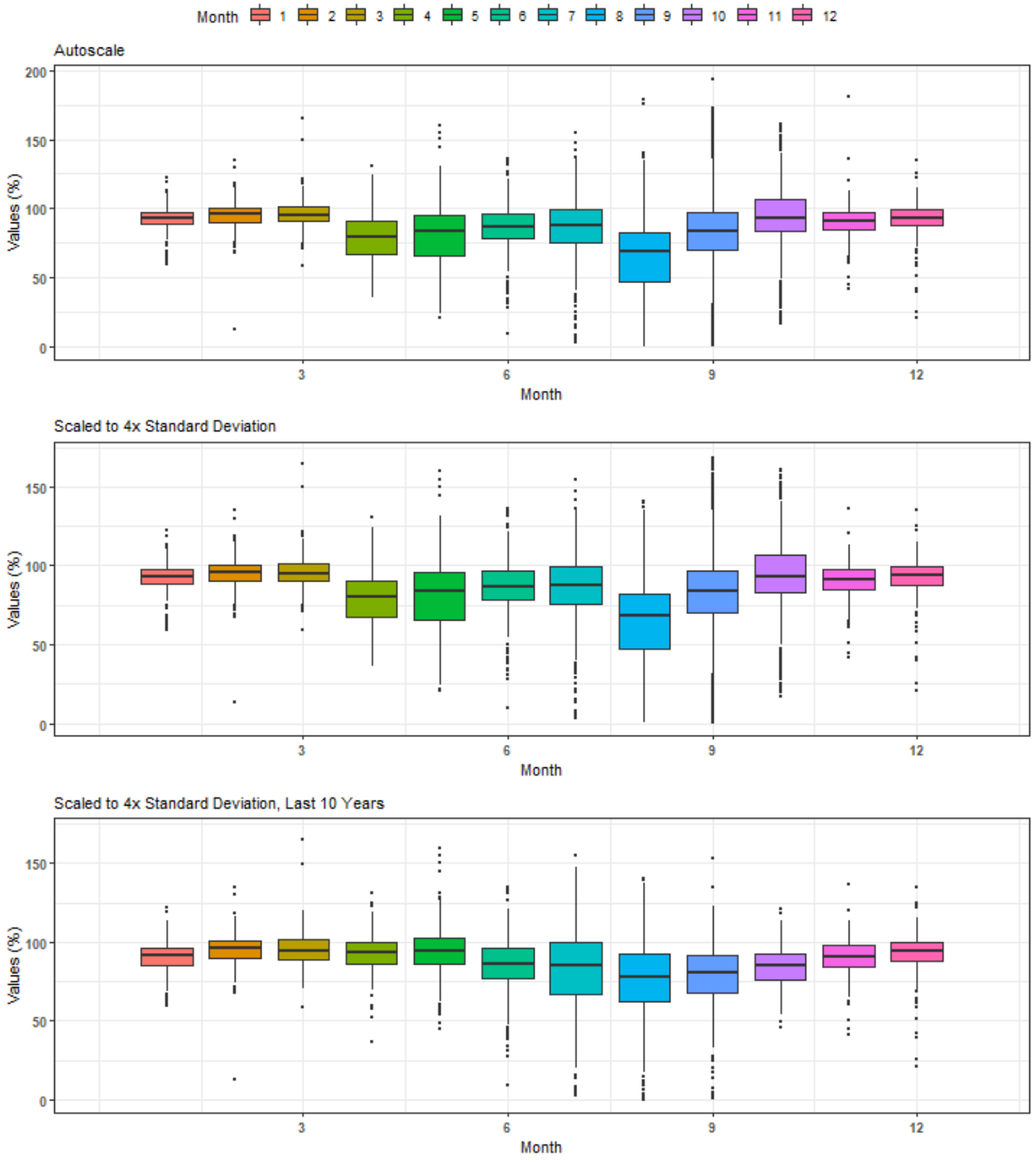
Summary Box Plots for Pinellas County Aquatic Preserve

By Year & Month



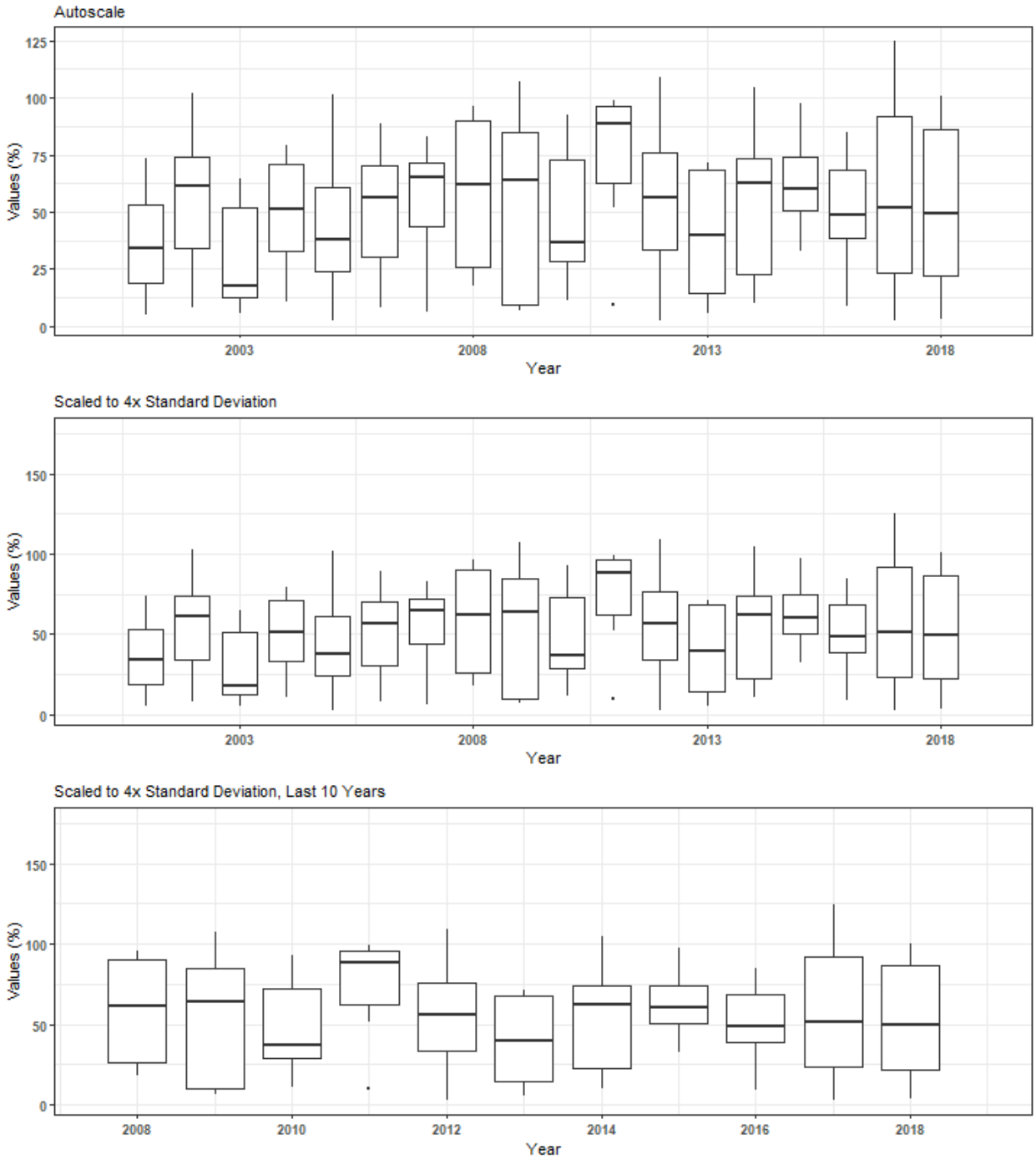
Summary Box Plots for Pinellas County Aquatic Preserve

By Month



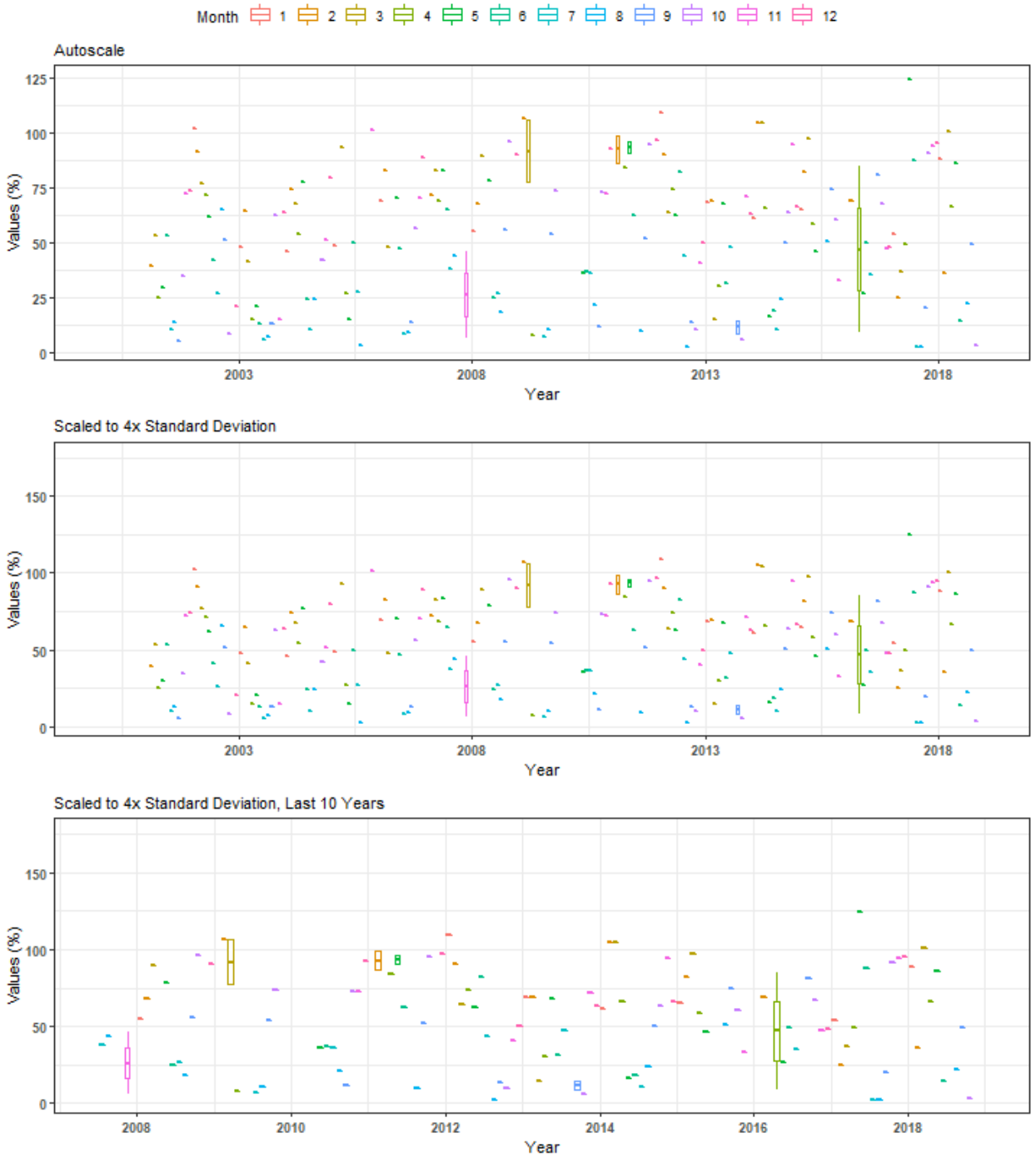
Summary Box Plots for Rocky Bayou State Park Aquatic Preserve

By Year



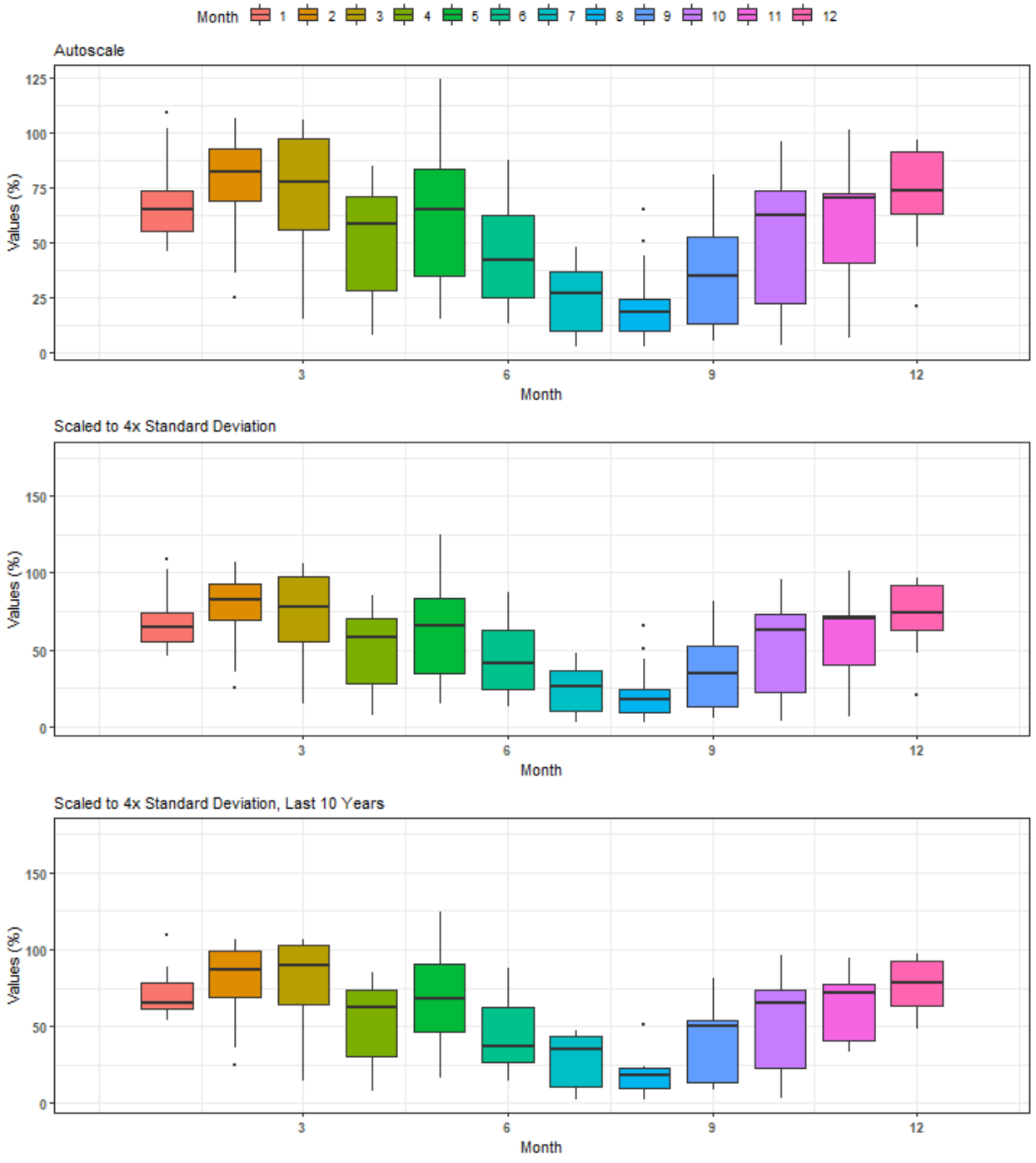
Summary Box Plots for Rocky Bayou State Park Aquatic Preserve

By Year & Month



Summary Box Plots for Rocky Bayou State Park Aquatic Preserve

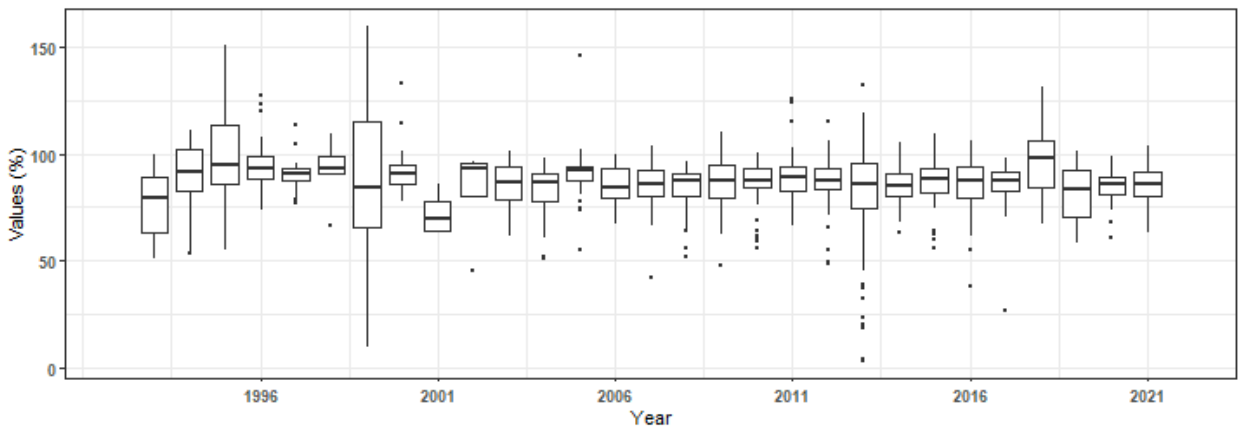
By Month



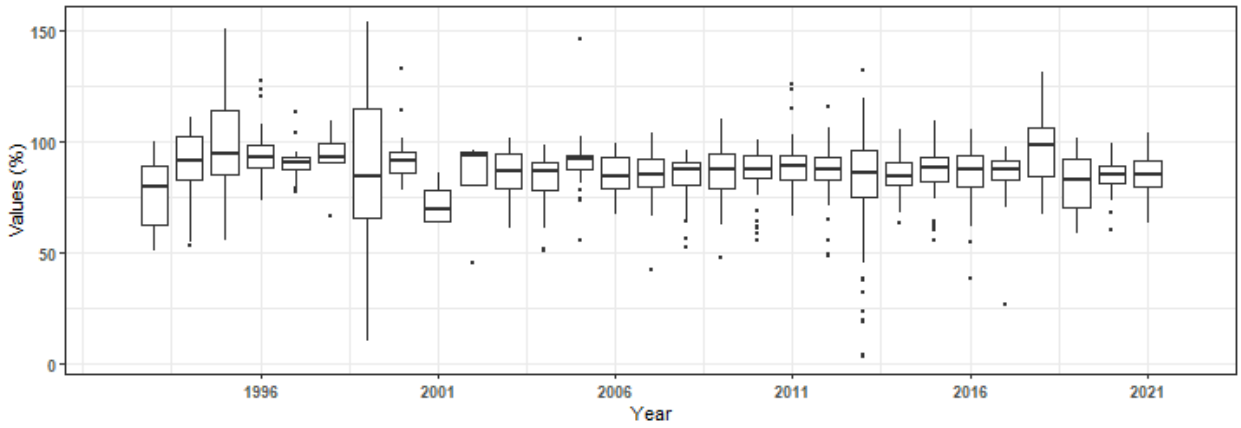
Summary Box Plots for Terra Ceia Aquatic Preserve

By Year

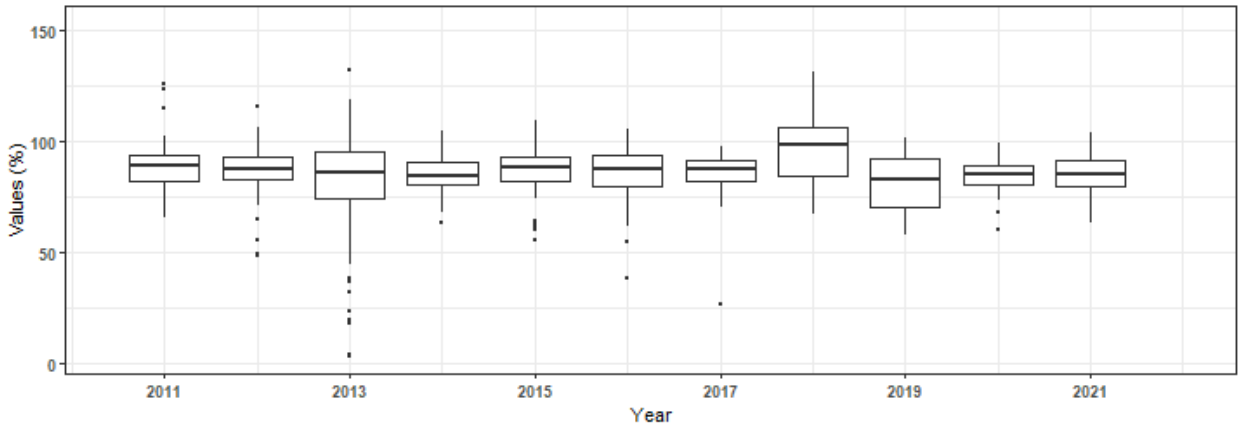
Autoscale

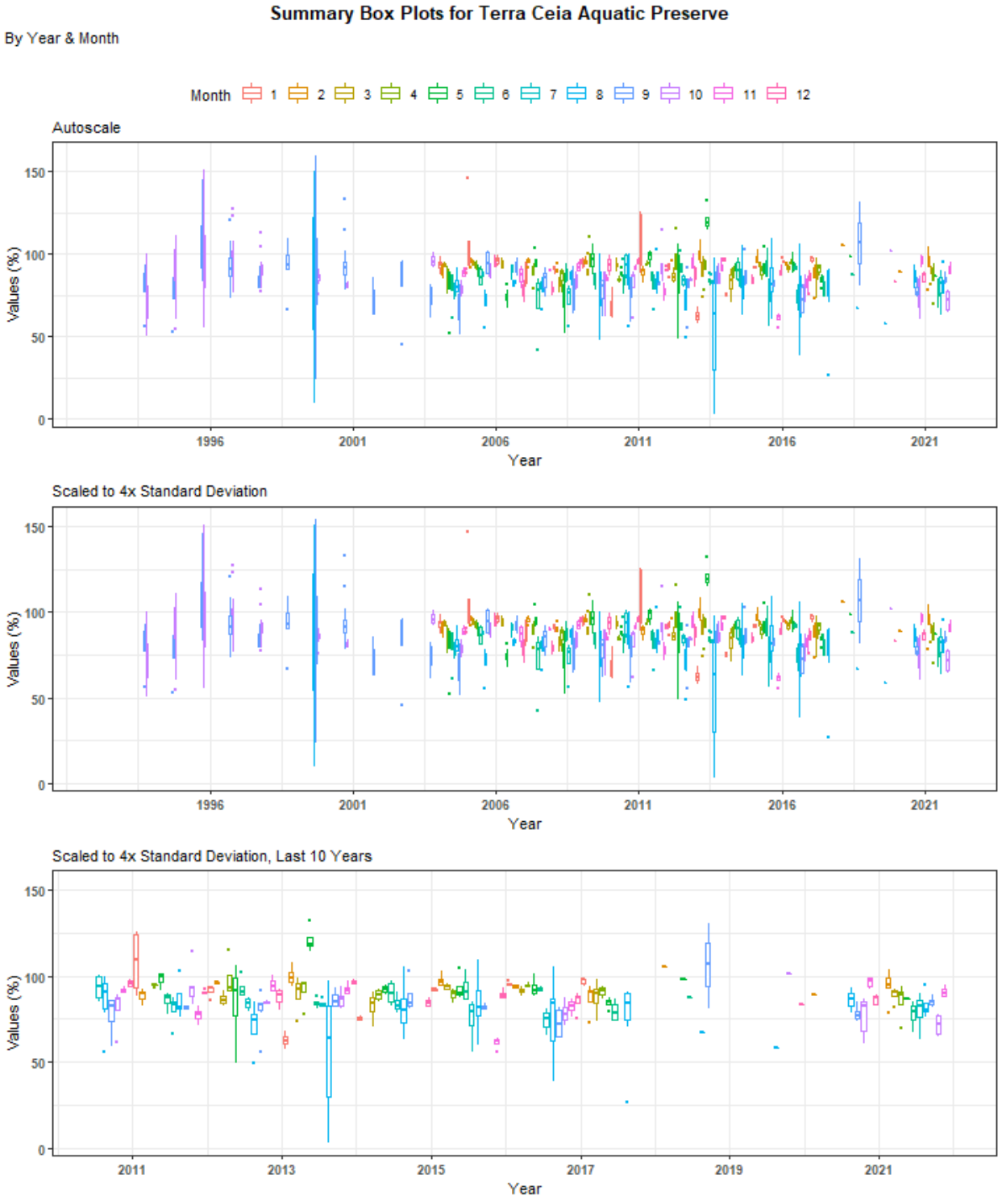


Scaled to 4x Standard Deviation



Scaled to 4x Standard Deviation, Last 10 Years




Summary Box Plots for Terra Ceia 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

Summary Box Plots for Terra Ceia Aquatic Preserve

By Month