SEACAR Continuous Water Quality Analysis: SE Region for Salinity

Last compiled on 30 June, 2022

# Important Notes

All scripts and outputs can be found on the SEACAR GitHub repository:

<https://github.com/FloridaSEACAR/SEACAR_Panzik>

Note: The top 2% of data is excluded when computing mean and standard deviations in plotting sections solely for the purpose of getting y-axis scales. The exclusion of the top 2% is not used in any statistics that are exported.

# Libraries and Settings

Loads libraries used in the script. The inclusion of scipen option limits how frequently R defaults to scientific notation.

library(knitr)  
library(data.table)  
library(dplyr)  
library(lubridate)  
library(ggplot2)  
library(ggpubr)  
library(scales)  
library(EnvStats)  
library(tidyr)  
library(kableExtra)  
windowsFonts(`Segoe UI` = windowsFont('Segoe UI'))  
options(scipen=999)  
opts\_chunk$set(warning=FALSE, message=FALSE, dpi=200)

# File Import

Imports file that is determined in the WC\_Continuous\_parameter\_ReportCompile.R script.

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

The script then gets the name of the parameter as it appears in the data file and units of the parameter.

data <- fread(file\_in, sep="|", header=TRUE, stringsAsFactors=FALSE,  
 select=c("ManagedAreaName", "ProgramID", "ProgramName",  
 "ProgramLocationID", "SampleDate", "Year", "Month",  
 "RelativeDepth", "ActivityType", "ParameterName",  
 "ResultValue", "ParameterUnits", "ValueQualifier",  
 "SEACAR\_QAQCFlagCode", "Include"),  
 na.strings="")  
parameter <- unique(data$ParameterName)  
unit <- unique(data$ParameterUnits)

# Data Filtering

Most data filtering is performed on export from the database, and is indicated by the Include variable. Include values of 1 indicate the data should be used for analysis, values of 0 indicate the data should not be used for analysis. Documentation on the database filtering is provided here: [SEACAR Documentation- Analysis Filters and Calculations.docx](https://github.com/FloridaSEACAR/SEACAR_Panzik/blob/main/SEACAR%20Documentation%20-%20Analysis%20Filters%20and%20Calculations.docx)

The filtering that is performed by the script at this point removes rows that are missing values for ResultValue and RelativeDepth, and removes any activity type that has “Blank” in the description. Data passes the filtering the process if it is has an Include value of 1.

The script then gets the units of the parameter, sets the SampleDate as a date object, and creates various scales of the date to be used by plotting functions.

Because the continuous data is extensive and most measurements are taken every 15 minutes, a daily average is determined and used based on grouping ManagedAreaName, ProgramID, ProgramName, ProgramLocationID, and SampleDate. The new ResultValue is the mean of all values on that date from that specific monitoring location. Sets the SampleDate as a date object, and creates various scales of the date to be used by plotting functions.

Creates a variable for each MonitoringID which is defined as a unique combination of ManagedAreaName, ProgramID, ProgramAreaName, and ProgramLocationID.

After the initial filtering, a second filter variable is created to determine whether enough time is represented in the managed area, which is that each managed area has 5 year or more of unique year entries for observation that pass the initial filter. If data passes the first set of filtering criteria and the time criteria, they are used in the analysis.

data$Include <- as.logical(data$Include)  
data <- data[data$Include==TRUE,]  
data <- data[!is.na(data$ResultValue),]  
data <- data[!is.na(data$RelativeDepth),]  
data <- data[!grep("Blank", data$ActivityType),]  
  
if(param\_name=="Water\_Temperature"){  
 data <- data[data$ResultValue>=-5,]  
} else{  
 data <- data[data$ResultValue>=0,]  
}  
  
  
  
data <- data %>%  
 group\_by(ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,  
 SampleDate) %>%  
 summarise(Year=unique(Year), Month=unique(Month),  
 RelativeDepth=unique(RelativeDepth),   
 ResultValue=mean(ResultValue), Include=unique(Include))  
  
data <- merge.data.frame(MA\_All[,c("AreaID", "ManagedAreaName")],  
 data, by="ManagedAreaName", all=TRUE)  
  
data$SampleDate <- as.Date(data$SampleDate)  
data$YearMonth <- format(data$SampleDate, format = "%m-%Y")  
data$YearMonthDec <- data$Year + ((data$Month-0.5) / 12)  
data$DecDate <- decimal\_date(data$SampleDate)  
  
data <- data %>%  
 group\_by(ManagedAreaName, ProgramID, ProgramName, ProgramLocationID) %>%  
 mutate(MonitoringID=cur\_group\_id())  
  
Mon\_Summ <- data %>%  
 group\_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,  
 ProgramLocationID) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=unique(RelativeDepth),  
 N\_Data=length(ResultValue[Include==TRUE & !is.na(ResultValue)]),  
 N\_Years=length(unique(Year[Include==TRUE & !is.na(Year)])),  
 EarliestYear=min(Year[Include==TRUE]),  
 LatestYear=max(Year[Include==TRUE]),  
 LastSampleDate=max(SampleDate[Include==TRUE]),  
 SufficientData=ifelse(N\_Data>0 & N\_Years>=10, TRUE, FALSE))  
  
Mon\_Summ <- as.data.table(Mon\_Summ[order(Mon\_Summ$MonitoringID), ])  
  
data <- data %>%  
 group\_by(MonitoringID) %>%  
 mutate(YearFromStart=Year-min(Year))  
  
data <- merge.data.frame(data, Mon\_Summ[,c("MonitoringID", "SufficientData")],  
 by="MonitoringID")  
  
data$Use\_In\_Analysis <- ifelse(data$Include==TRUE &  
 data$SufficientData==TRUE, TRUE, FALSE)  
  
Mon\_IDs <- unique(data$MonitoringID[data$Use\_In\_Analysis==TRUE])  
Mon\_IDs <- Mon\_IDs[order(Mon\_IDs)]  
n <- length(Mon\_IDs)

# Monitoring Location Statistics

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

1. Take the data variable and only include rows that have a Use\_In\_Analysis value of TRUE
2. Group data that have the same ManagedAreaName, ProgramID, ProgramName, ProgramLocationID, Year, and Month.
   * Second summary statistics consider the monitoring location grouping and Year.
   * Third summary statistics consider the monitoring location grouping and Month.
3. For each group, provide the following information: Earliest Sample Date (EarliestSampleDate), Latest Sample Date (LastSampleDate), Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, and a list of all Program IDs included in these measurements.
4. Sort the data in ascending (A to Z and 0 to 9) order based on ManagedAreaName, ProgramID, ProgramName, ProgramLocationID, Year, and Month in that order.
5. Write summary stats to a pipe-delimited .txt file in the output directory

Mon\_YM\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,  
 Year, Month) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=unique(RelativeDepth),  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate), N=length(ResultValue),  
 Min=min(ResultValue), Max=max(ResultValue),  
 Median=median(ResultValue), Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
Mon\_YM\_Stats <- as.data.table(Mon\_YM\_Stats[order(Mon\_YM\_Stats$ManagedAreaName,  
 Mon\_YM\_Stats$ProgramID,  
 Mon\_YM\_Stats$ProgramName,  
 Mon\_YM\_Stats$ProgramLocationID,  
 Mon\_YM\_Stats$Year,  
 Mon\_YM\_Stats$Month), ])  
fwrite(Mon\_YM\_Stats, paste0(out\_dir,"/", param\_name, "\_", region,  
 "\_MonitoringLoc\_YearMonth\_Stats.txt"), sep="|")  
  
Mon\_Y\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,  
 Year) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=unique(RelativeDepth),  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate), N=length(ResultValue),  
 Min=min(ResultValue), Max=max(ResultValue),  
 Median=median(ResultValue), Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
Mon\_Y\_Stats <- as.data.table(Mon\_Y\_Stats[order(Mon\_Y\_Stats$ManagedAreaName,  
 Mon\_Y\_Stats$ProgramID,  
 Mon\_Y\_Stats$ProgramName,  
 Mon\_Y\_Stats$ProgramLocationID,  
 Mon\_Y\_Stats$Year), ])  
fwrite(Mon\_Y\_Stats, paste0(out\_dir,"/", param\_name, "\_", region,  
 "\_MonitoringLoc\_Year\_Stats.txt"), sep="|")  
  
Mon\_M\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,  
 Month) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=unique(RelativeDepth),  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate), N=length(ResultValue),  
 Min=min(ResultValue), Max=max(ResultValue),  
 Median=median(ResultValue), Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
Mon\_M\_Stats <- as.data.table(Mon\_M\_Stats[order(Mon\_M\_Stats$ManagedAreaName,  
 Mon\_M\_Stats$ProgramID,  
 Mon\_M\_Stats$ProgramName,  
 Mon\_M\_Stats$ProgramLocationID,  
 Mon\_M\_Stats$Month), ])  
  
fwrite(Mon\_M\_Stats, paste0(out\_dir,"/", param\_name, "\_", region,  
 "\_MonitoringLoc\_Month\_Stats.txt"), sep="|")

# Seasonal Kendall Tau Analysis

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

The following steps are performed:

1. Define the trend function.
2. Take the data variable and only include rows that have a Use\_In\_Analysis value of TRUE
3. Group data that have the same ManagedAreaName, ProgramID, ProgramName, and ProgramLocationID.
4. For each group, provides the following information: Earliest Sample Date (EarliestSampleDate), Latest Sample Date (LastSampleDate), Number of Entries (N), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation,
5. For each group, a temporary variable is created to run the kendallSeasonalTrendTest function using the Year values for year, and Month as the seasonal qualifier, and Trend.
   * An independent.obs value of TRUE indicates that the data should be treated as not being serially auto-correlated. An independent.obs value of FALSE indicates that it is treated as being serially auto-correlated, but also requires one observation per season per year for the full time of observation.
   * tau, Senn Slope (SennSlope), Senn Intercept (SennIntercept), and p are extracted from the model results.
6. The two stats tables are merged based on similar groups, and then Trend is determined from the user-defined function.
7. Write summary stats to a pipe-delimited .txt file in the output directory
   * [Click this text to open Git directory with output files](https://github.com/FloridaSEACAR/SEACAR_Panzik/tree/main/output/WQ)
8. Add the Monitoring IDS to KT.Stats for easier use while plotting.

tauSeasonal <- function(data, independent, stats.median, stats.minYear,  
 stats.maxYear) {  
 tau <- NULL  
 tryCatch({ken <- kendallSeasonalTrendTest(  
 y=data$ResultValue,  
 season=data$Month,  
 year=data$YearFromStart,  
 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)  
 rm(ken)  
 }, warning=function(w) {  
 print(w)  
 }, error=function(e) {  
 print(e)  
 }, finally={  
 if (!exists("tau")) {  
 tau <- NA  
 }  
 if (!exists("p")) {  
 p <- NA  
 }  
 if (!exists("slope")) {  
 slope <- NA  
 }  
 if (!exists("intercept")) {  
 intercept <- NA  
 }  
 if (!exists("trend")) {  
 trend <- NA  
 }  
 })  
 KT <-c(unique(data$MonitoringID),  
 stats.median,  
 independent,  
 tau,  
 p,  
 slope,  
 intercept,  
 trend)  
 return(KT)  
}  
runStats <- function(data) {  
 data$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[8])) {  
 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$ManagedAreaName  
c\_names <- c("MonitoringID", "Median", "Independent", "tau", "p",  
 "SennSlope", "SennIntercept", "Trend")  
if(n==0){  
 KT.Stats <- data.frame(matrix(ncol=length(c\_names),  
 nrow=nrow(Mon\_Summ)))  
 colnames(KT.Stats) <- c\_names  
 KT.Stats[, c("MonitoringID")] <- Mon\_Summ[, c("MonitoringID")]  
} else{  
 for (i in 1:n) {  
 x <- nrow(data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ])  
 if (x>0) {  
 KT.Stats <- runStats(data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ])  
 }  
 }  
 KT.Stats <- as.data.frame(KT.Stats)  
   
 if(dim(KT.Stats)[2]==1){  
 KT.Stats <- as.data.frame(t(KT.Stats))  
 }  
 colnames(KT.Stats) <- c\_names  
 rownames(KT.Stats) <- seq(1:nrow(KT.Stats))  
 KT.Stats$tau <- round(as.numeric(KT.Stats$tau), digits=4)  
 KT.Stats$p <- round(as.numeric(KT.Stats$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)  
}  
  
KT.Stats <- merge.data.frame(Mon\_Summ, KT.Stats,  
 by=c("MonitoringID"), all=TRUE)  
  
KT.Stats <- as.data.table(KT.Stats[order(KT.Stats$MonitoringID), ])  
  
KT.Stats$MonitoringID <- NULL  
fwrite(KT.Stats, paste0(out\_dir,"/", param\_name, "\_", region,  
 "\_KendallTau\_Stats.txt"), sep="|")  
  
KT.Stats$MonitoringID <- Mon\_Summ$MonitoringID  
data <- data[!is.na(data$ResultValue),]  
  
KT.Plot <- KT.Stats %>%  
 group\_by(MonitoringID) %>%  
 summarize(x=EarliestYear, y=SennIntercept)  
KT.Plot2 <- KT.Stats %>%  
 group\_by(MonitoringID) %>%  
 summarize(x=decimal\_date(LastSampleDate),  
 y=(x-EarliestYear)\*SennSlope+SennIntercept)  
KT.Plot <- bind\_rows(KT.Plot, KT.Plot2)  
rm(KT.Plot2)  
KT.Plot <- as.data.table(KT.Plot[order(KT.Plot$MonitoringID), ])  
KT.Plot <- KT.Plot[!is.na(KT.Plot$y),]

# Appendix I: Dataset Summary Box Plots

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

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

This set of box plots are grouped by year.

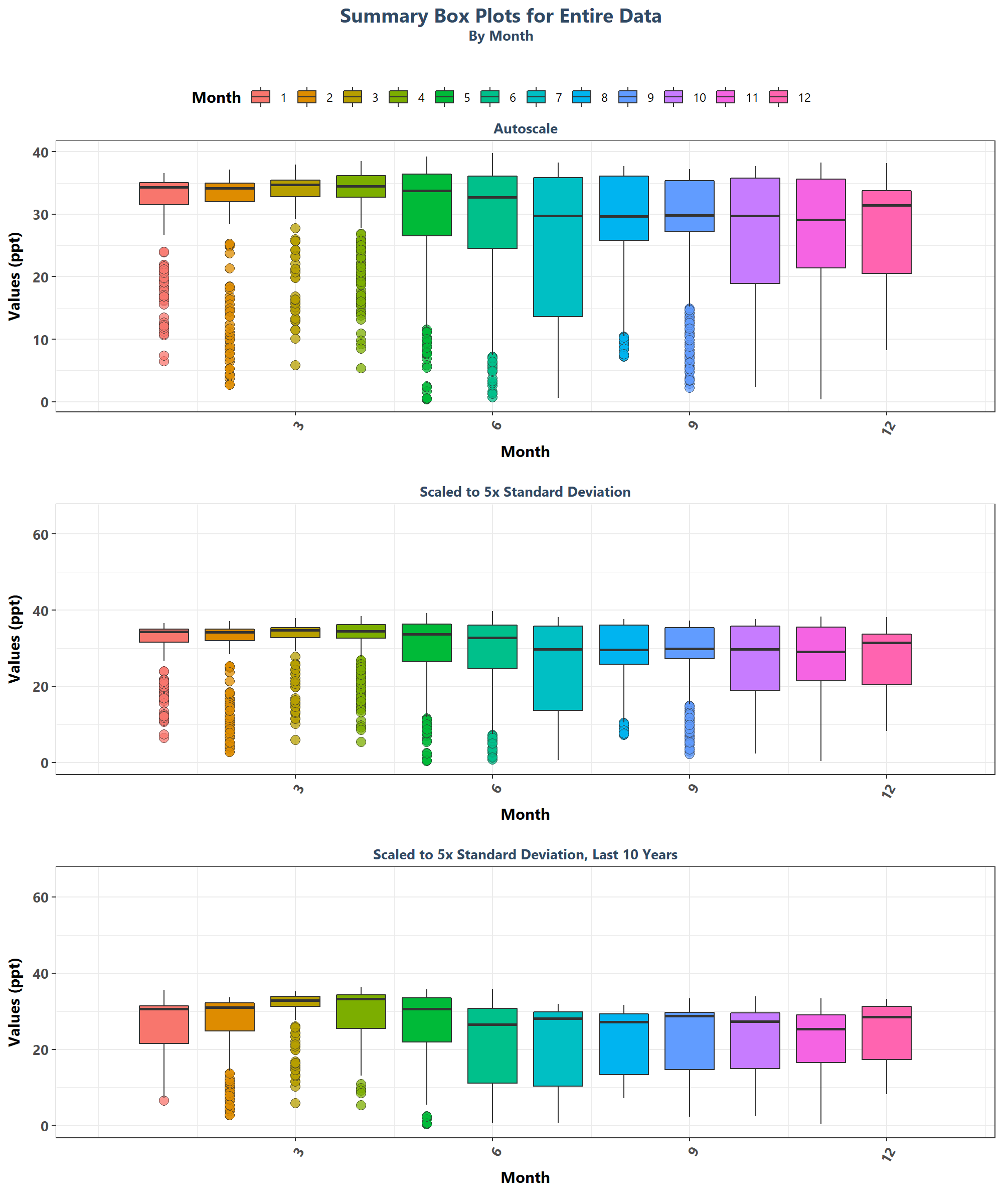
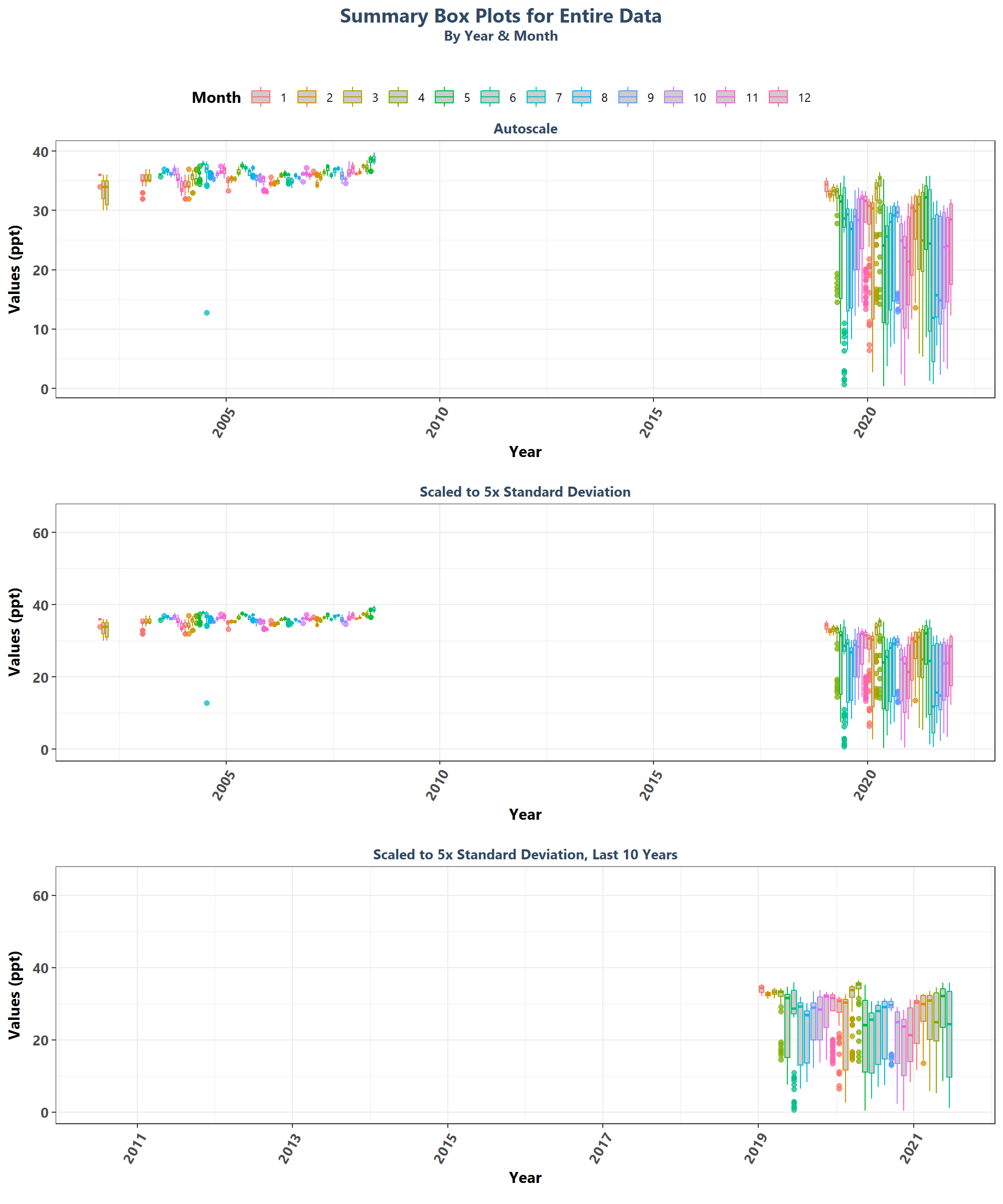
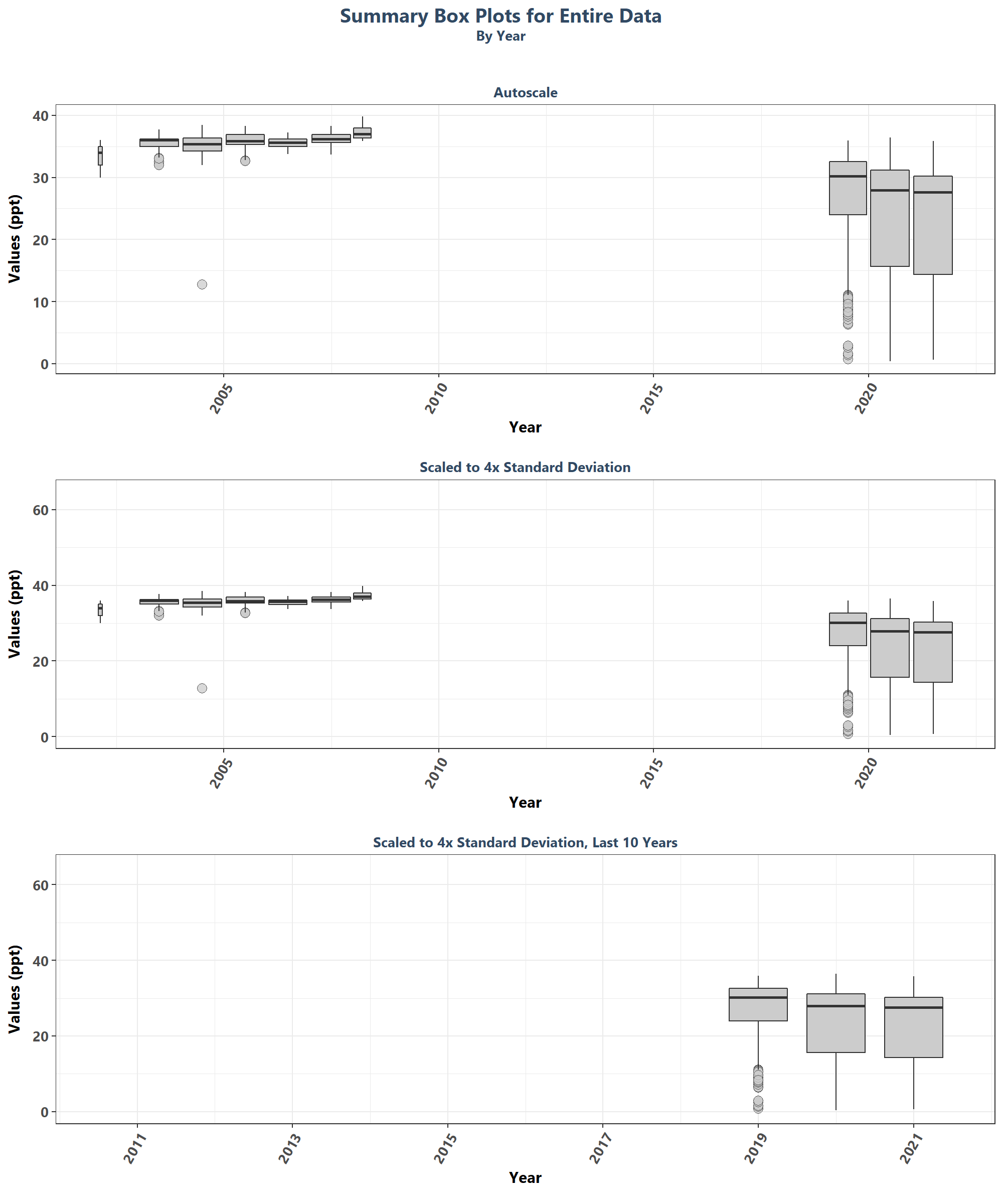
plot\_theme <- theme\_bw() +  
 theme(text=element\_text(family="Segoe UI"),  
 title=element\_text(face="bold"),  
 plot.title=element\_text(hjust=0.5, size=14, color="#314963"),  
 plot.subtitle=element\_text(hjust=0.5, size=10, color="#314963"),  
 axis.title.x = element\_text(margin = margin(t = 5, r = 0,  
 b = 10, l = 0)),  
 axis.title.y = element\_text(margin = margin(t = 0, r = 10,  
 b = 0, l = 0)),  
 axis.text=element\_text(size=10),  
 axis.text.x=element\_text(face="bold", angle = 60, hjust = 1),  
 axis.text.y=element\_text(face="bold"))  
  
min\_RV <- min(data$ResultValue[data$Include==TRUE])  
mn\_RV <- mean(data$ResultValue[data$Include==TRUE &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
sd\_RV <- sd(data$ResultValue[data$Include==TRUE &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
y\_scale <- mn\_RV + 4 \* sd\_RV  
  
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=SampleDate, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")")) +  
 plot\_theme  
  
p2 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=SampleDate, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation", x="Year",  
 y=paste0("Values (", unit, ")")) +  
 ylim(0, y\_scale) +   
 plot\_theme  
  
p3 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(0, y\_scale) +  
 scale\_x\_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+0.5),  
 breaks=seq(max(data$Year) - 10, max(data$Year), 2)) +  
 plot\_theme  
  
set <- ggarrange(p1, p2, p3, ncol=1)  
  
p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",  
 subtitle="By Year") + plot\_theme +  
 theme(panel.border=element\_blank(), panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
  
Yset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))

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

p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")"), color="Month") +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(color=guide\_legend(nrow=1))  
  
p2 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 5x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(0, y\_scale) +  
 plot\_theme +  
 theme(legend.position="none")  
  
p3 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 5x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(0, y\_scale) +  
 scale\_x\_continuous(limits=c(max(data$Year) - 10.5, max(data$Year)+0.5),  
 breaks=seq(max(data$Year) - 10, max(data$Year), 2)) +  
 plot\_theme +  
 theme(legend.position="none")  
leg <- get\_legend(p1)  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
  
p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",  
 subtitle="By Year & Month") + plot\_theme +  
 theme(panel.border=element\_blank(), panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
  
YMset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))

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

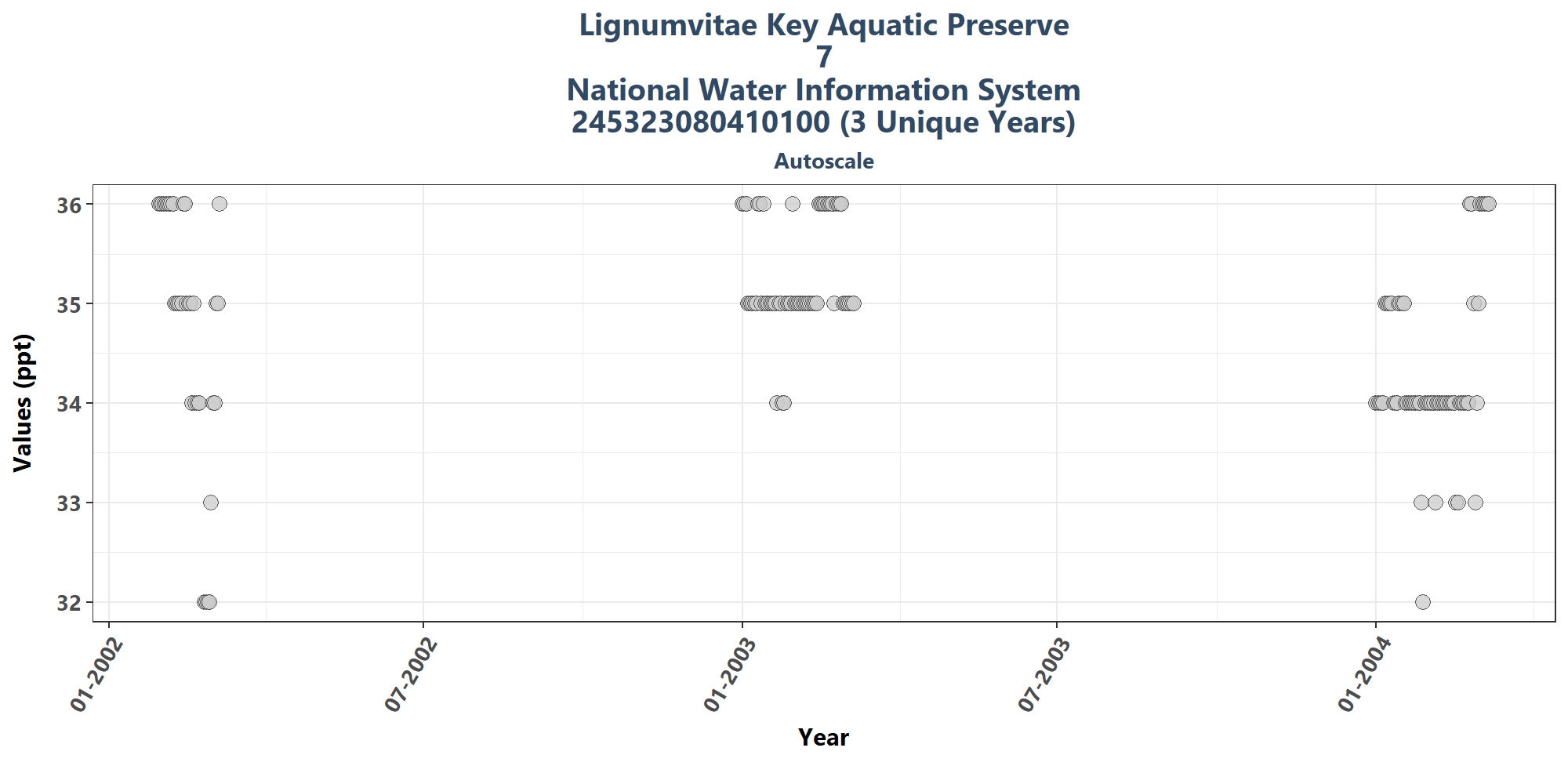
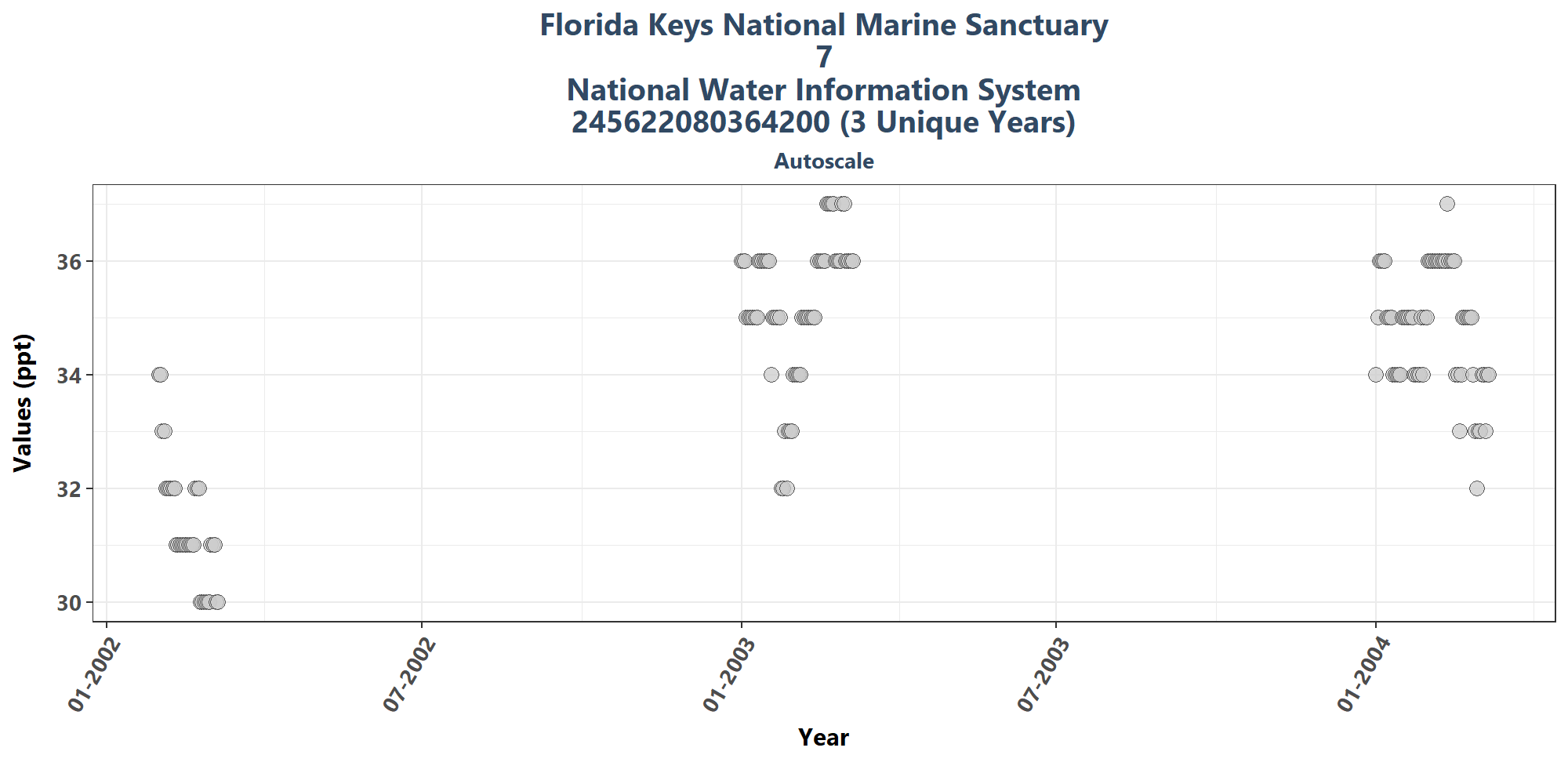
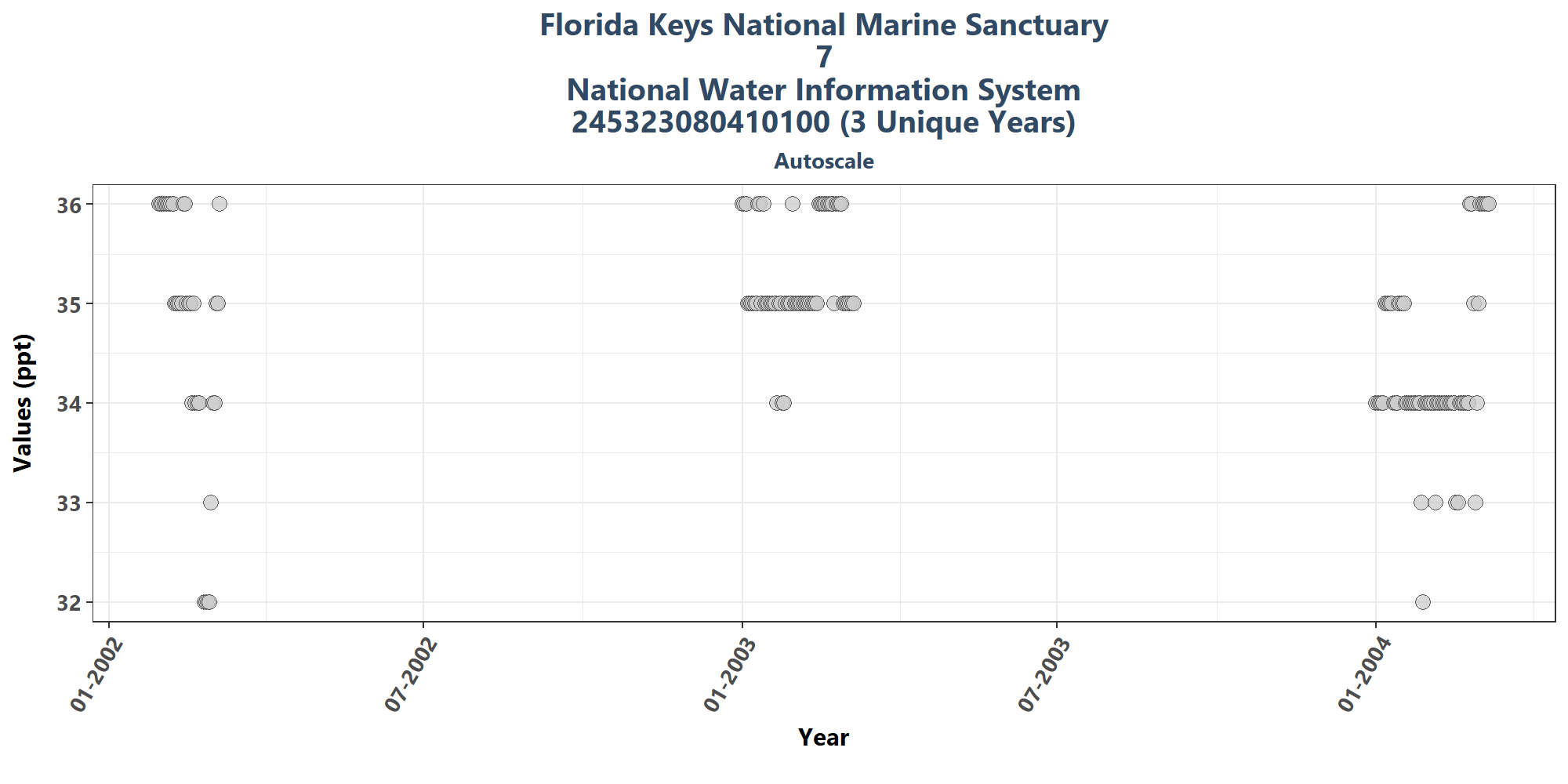
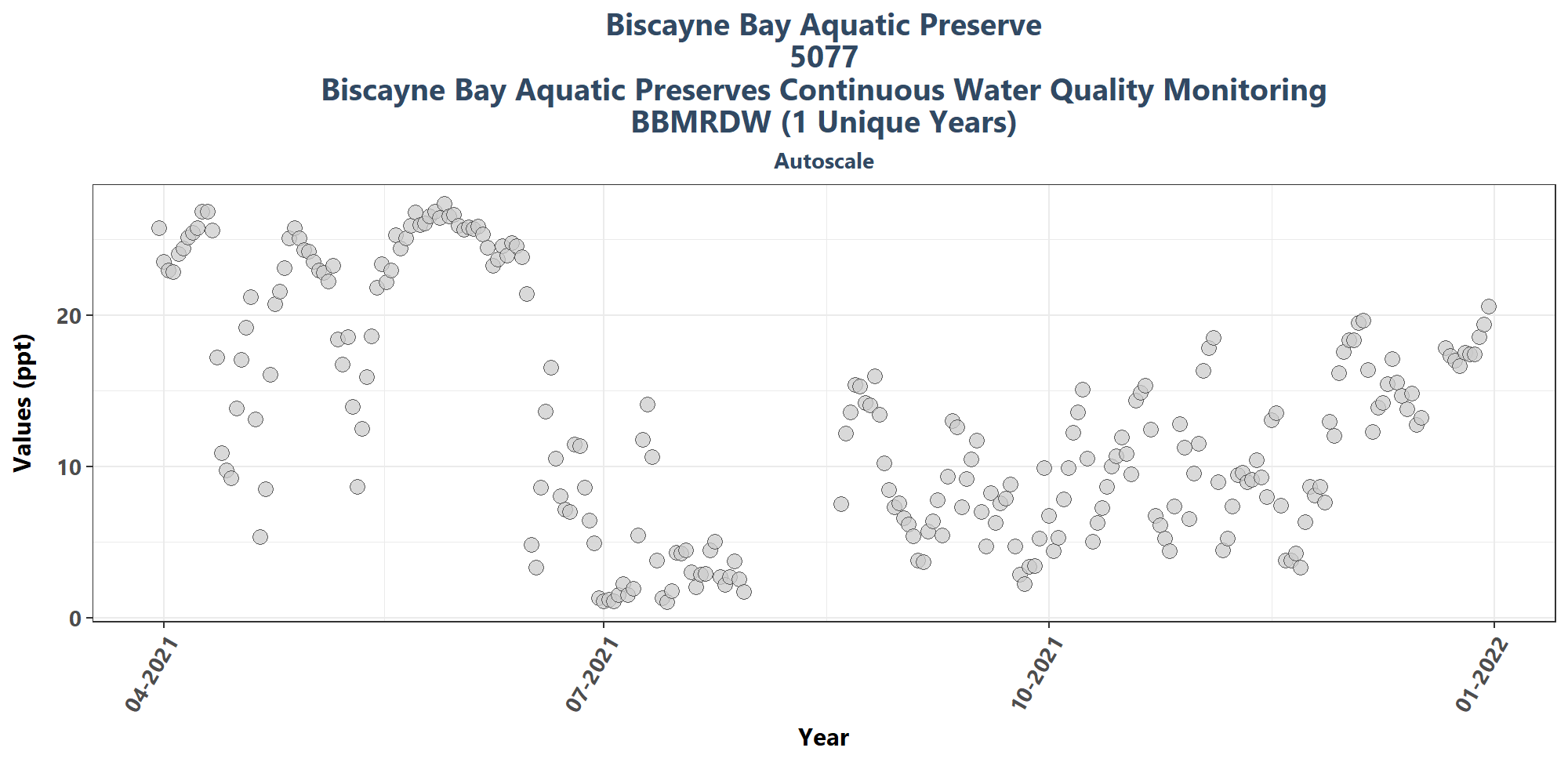
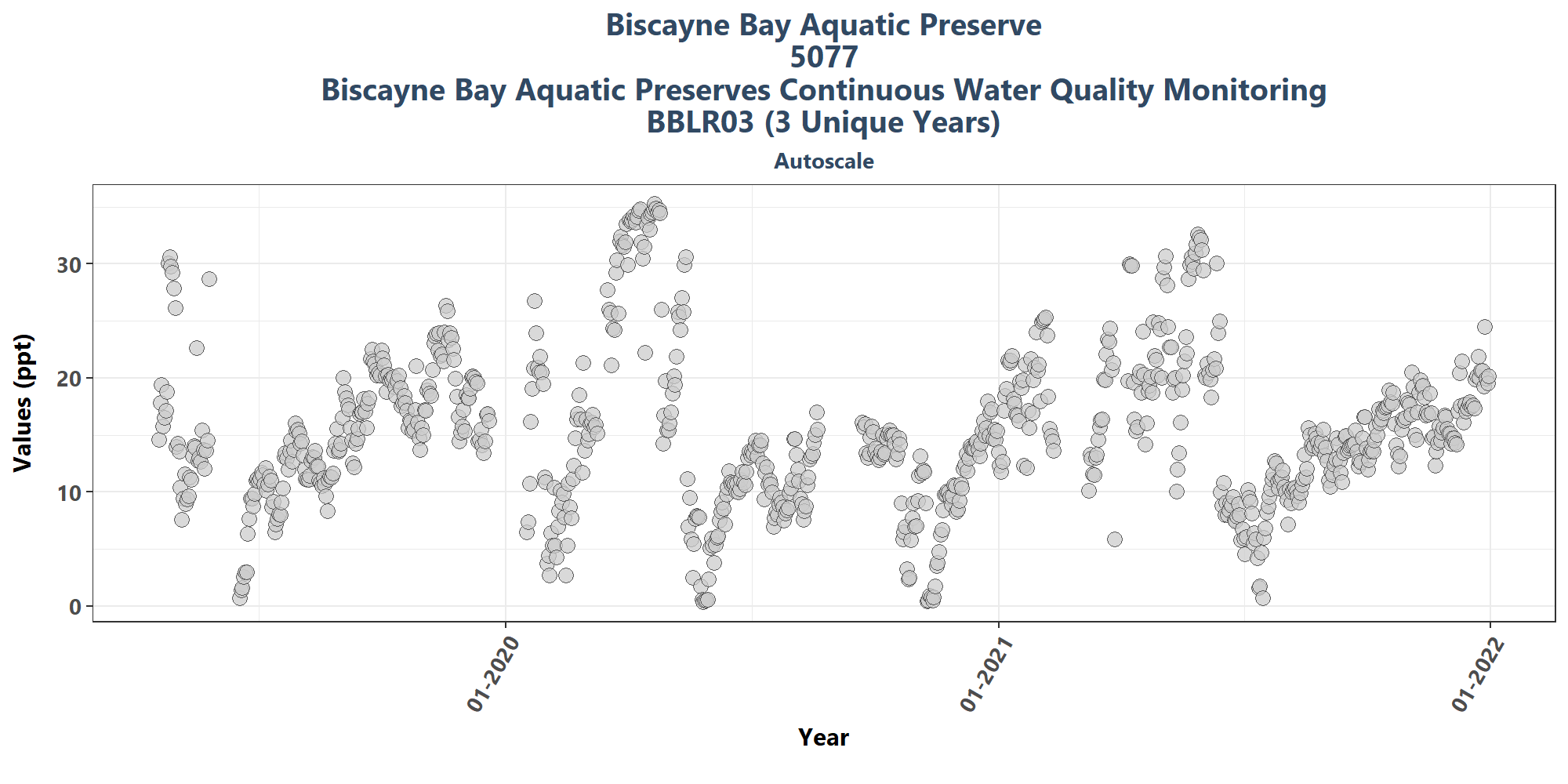
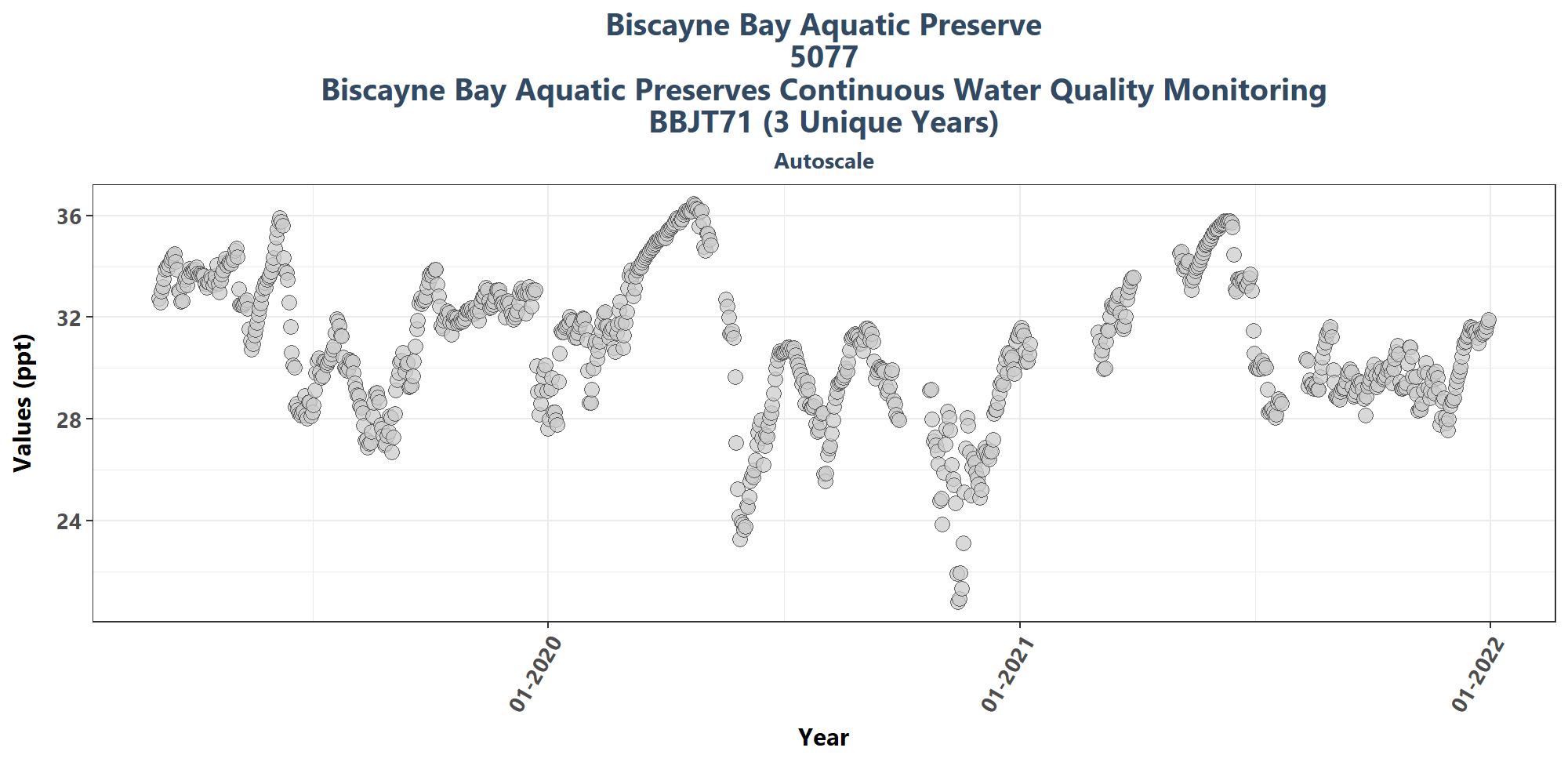
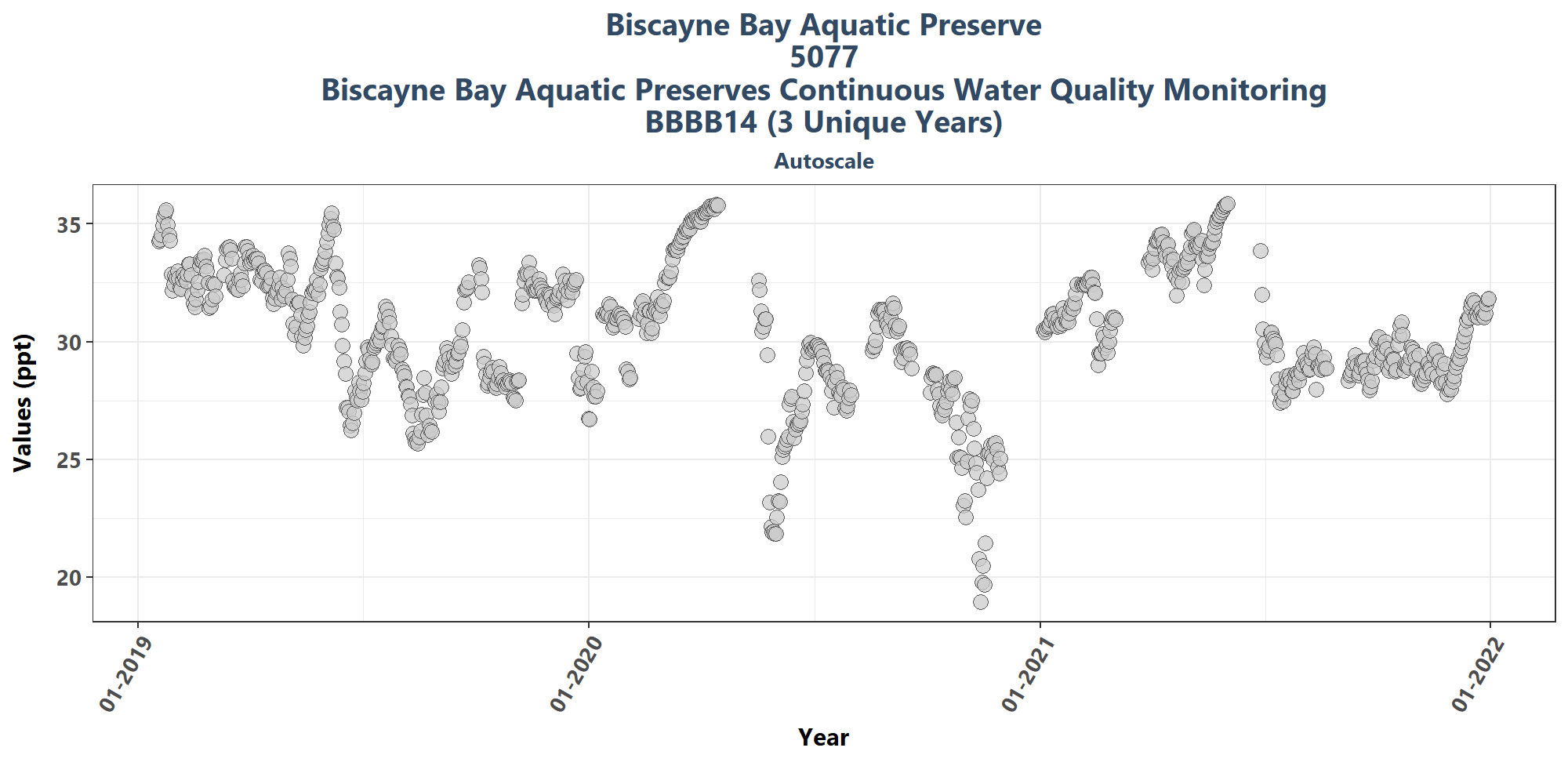
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Month",  
 y=paste0("Values (", unit, ")"), fill="Month") +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(fill=guide\_legend(nrow=1))  
  
p2 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 5x Standard Deviation",  
 x="Month", y=paste0("Values (", unit, ")")) +  
 ylim(0, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
  
p3 <- ggplot(data=data[data$Include==TRUE &  
 data$Year >= max(data$Year) - 10, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 5x Standard Deviation, Last 10 Years",  
 x="Month", y=paste0("Values (", unit, ")")) +  
 ylim(0, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
leg <- get\_legend(p1)  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
  
p0 <- ggplot() + labs(title="Summary Box Plots for Entire Data",  
 subtitle="By Month") + plot\_theme +  
 theme(panel.border=element\_blank(), panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
  
Mset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))



# Appendix II: Excluded Monitoring Locations

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

Mon\_Exclude <- Mon\_Summ[Mon\_Summ$N\_Years<5 & Mon\_Summ$N\_Years>0,]  
Mon\_Exclude <- Mon\_Exclude[order(Mon\_Exclude$MonitoringID),]  
z=nrow(Mon\_Exclude)  
  
if(z==0){  
 print("There are no monitoring locations that qualify.")  
} else {  
 for(i in 1:z){  
 MA\_name <- unique(data$ManagedAreaName[  
 data$MonitoringID==Mon\_Exclude$MonitoringID[i]])  
 Mon\_name <- paste0(unique(data$ProgramID[  
 data$MonitoringID==Mon\_Exclude$MonitoringID[i]]), "\n",  
 unique(data$ProgramName[  
 data$MonitoringID==Mon\_Exclude$MonitoringID[i]]), "\n",  
 unique(data$ProgramLocationID[  
 data$MonitoringID==Mon\_Exclude$MonitoringID[i]]))  
   
 p1<-ggplot(data=data[data$MonitoringID==Mon\_Exclude$MonitoringID[i]&  
 data$Include==TRUE, ],  
 aes(x=SampleDate, y=ResultValue)) +  
 geom\_point(shape=21, size=3, color="#333333", fill="#cccccc",  
 alpha=0.75) +  
 labs(title=paste0(MA\_name, "\n",  
 Mon\_name, " (", Mon\_Exclude$N\_Years[i],  
 " Unique Years)"),  
 subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")")) +  
 plot\_theme +  
 scale\_x\_date(labels=date\_format("%m-%Y"))  
 print(p1)  
 }  
}



# Appendix III: Monitoring Location Trendlines

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

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

if(n==0){  
 print("There are no monitoring locations that qualify.")  
} else {  
 for (i in 1:n) {  
 plot\_data <- data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i],]  
 KT.plot\_data <- KT.Plot[KT.Plot$MonitoringID==Mon\_IDs[i],]  
 year\_lower <- min(plot\_data$Year)  
 year\_upper <- max(plot\_data$Year)  
 min\_RV <- min(plot\_data$ResultValue)  
 mn\_RV <- mean(plot\_data$ResultValue[plot\_data$ResultValue <  
 quantile(plot\_data$ResultValue, 0.98)])  
 sd\_RV <- sd(plot\_data$ResultValue[plot\_data$ResultValue <  
 quantile(plot\_data$ResultValue, 0.98)])  
 x\_scale <- ifelse(year\_upper - year\_lower > 30, 10, 5)  
 y\_scale <- mn\_RV + 4 \* sd\_RV  
   
   
  
 MA\_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID==Mon\_IDs[i]]  
 Mon\_name <- paste0(KT.Stats$ProgramID[KT.Stats$MonitoringID==Mon\_IDs[i]],  
 "\n", KT.Stats$ProgramName[KT.Stats$MonitoringID==Mon\_IDs[i]], "\n",  
 KT.Stats$ProgramLocationID[KT.Stats$MonitoringID==Mon\_IDs[i]])  
   
 p1 <- ggplot(data=plot\_data,  
 aes(x=DecDate, y=ResultValue)) +  
 geom\_point(shape=21, size=3, color="#333333", fill="#cccccc",  
 alpha=0.75) +  
 geom\_line(data=KT.plot\_data, aes(x=x, y=y),  
 color="#000099", size=1.2, alpha=0.7) +  
 labs(title=paste0(MA\_name, "\n", Mon\_name),  
 subtitle=parameter,  
 x="Year", y=paste0("Values (", unit, ")")) +  
 plot\_theme  
   
 # p2 <- ggplot(data=plot\_data,  
 # aes(x=DecDate, y=ResultValue)) +  
 # geom\_point(shape=21, size=3, color="#333333", fill="#cccccc",  
 # alpha=0.75) +  
 # geom\_line(data=KT.plot\_data, aes(x=x, y=y),  
 # color="#000099", size=1.2, alpha=0.7) +  
 # ylim(min\_RV-0.1\*y\_scale, y\_scale) +  
 # labs(subtitle="Scaled to 4x Standard Deviation",  
 # x="Year", y=paste0("Values (", unit, ")")) +  
 # plot\_theme  
   
 # KTset <- ggarrange(p1, p2, ncol=1, heights=c(1, 1))  
 #   
 # p0 <- ggplot() + labs()) +  
 # plot\_theme + theme(panel.border=element\_blank(),  
 # panel.grid.major=element\_blank(),  
 # panel.grid.minor=element\_blank(),  
 # axis.line=element\_blank())  
   
 ResultTable <- KT.Stats[KT.Stats$MonitoringID==Mon\_IDs[i], ] %>%  
 select(RelativeDepth, N\_Data, N\_Years, EarliestYear, LatestYear,  
 Median, Independent, tau, p, SennSlope, SennIntercept)  
   
 t1 <- ggtexttable(ResultTable, rows=NULL,  
 theme=ttheme(base\_size=11)) %>%  
 tab\_add\_footnote(text="p < 0.00005 appear as 0 due to rounding.\n  
 SennIntercept is intercept value at beginning of  
 record for monitoring location",  
 size=10, face="italic")  
   
 print(ggarrange(p1, t1, ncol=1, heights=c(0.85, 0.15)))  
 cat('\n \n \n')  
 rm(plot\_data)  
 rm(KTset, leg)  
 rm(plot\_data)  
 rm(KTset, leg)  
 }  
}

[1] “There are no monitoring locations that qualify.”

# Appendix IV: Monitoring Location Summary Box Plots

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

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

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

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

if(n==0){  
 print("There are no monitoring locations that qualify.")  
} else {  
 for (i in 1:n) {  
 year\_lower <- min(data$Year[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i]])  
 year\_upper <- max(data$Year[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i]])  
 min\_RV <- min(data$ResultValue[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i]])  
 mn\_RV <- mean(data$ResultValue[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i] &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
 sd\_RV <- sd(data$ResultValue[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i] &  
 data$ResultValue <  
 quantile(data$ResultValue, 0.98)])  
 x\_scale <- ifelse(year\_upper - year\_lower > 30, 10, 5)  
 y\_scale <- mn\_RV + 4 \* sd\_RV  
 MA\_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID==Mon\_IDs[i]]  
 Mon\_name <- paste0(KT.Stats$ProgramID[KT.Stats$MonitoringID==Mon\_IDs[i]],  
 "\n", KT.Stats$ProgramName[KT.Stats$MonitoringID==Mon\_IDs[i]], "\n",  
 KT.Stats$ProgramLocationID[KT.Stats$MonitoringID==Mon\_IDs[i]])  
   
 ##Year plots  
 p1 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ],  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme  
   
 p2 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ],  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme  
   
 p3 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i] &  
 data$Year>=year\_upper-10, ],  
 aes(x=Year, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")")) +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_upper - 10.5, year\_upper + 1),  
 breaks=rev(seq(year\_upper, year\_upper - 10,-2))) +  
 plot\_theme  
   
 Yset <- ggarrange(p1, p2, p3, ncol=1)  
   
 p0 <- ggplot() + labs(title=paste0(MA\_name, "\n", Mon\_name),  
 subtitle="By Year") +  
 plot\_theme + theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(),  
 axis.line=element\_blank())  
   
   
 ## Year & Month Plots  
 p4 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Autoscale",  
 x="Year", y=paste0("Values (", unit, ")"), color="Month") +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme +  
 theme(legend.position="none")  
   
 p5 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Year", y=paste0("Values (", unit, ")"), color="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_lower - 1, year\_upper + 1),  
 breaks=rev(seq(year\_upper,  
 year\_lower, -x\_scale))) +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(color=guide\_legend(nrow=1))  
   
 p6 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Year", y=paste0("Values (", unit, ")"), color="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(year\_upper - 10.5, year\_upper + 1),  
 breaks=rev(seq(year\_upper, year\_upper - 10,-2))) +  
 plot\_theme +  
 theme(legend.position="none")  
   
 leg1 <- get\_legend(p5)  
 YMset <- ggarrange(leg1, p4, p5 + theme(legend.position="none"), p6,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
   
 p00 <- ggplot() + labs(title=paste0(MA\_name, "\n", Mon\_name),  
 subtitle="By Year & Month") + plot\_theme +  
 theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
   
 ## Month Plots  
 p7 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale",  
 x="Month", y=paste0("Values (", unit, ")"), fill="Month") +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
   
 p8 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i], ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation",  
 x="Month", y=paste0("Values (", unit, ")"), fill="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(fill=guide\_legend(nrow=1))  
   
 p9 <- ggplot(data=data[data$Use\_In\_Analysis==TRUE &  
 data$MonitoringID==Mon\_IDs[i] &  
 data$Year >= year\_upper - 10, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Scaled to 4x Standard Deviation, Last 10 Years",  
 x="Month", y=paste0("Values (", unit, ")"), fill="Month") +  
 ylim(min\_RV, y\_scale) +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="none")  
   
 leg2 <- get\_legend(p8)  
 Mset <- ggarrange(leg2, p7, p8 + theme(legend.position="none"), p9,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
   
 p000 <- ggplot() + labs(title=paste0(MA\_name, "\n", Mon\_name),  
 subtitle="By Month") + plot\_theme +  
 theme(panel.border=element\_blank(),  
 panel.grid.major=element\_blank(),  
 panel.grid.minor=element\_blank(), axis.line=element\_blank())  
   
 print(ggarrange(p0, Yset, ncol=1, heights=c(0.1, 1)))  
 print(ggarrange(p00, YMset, ncol=1, heights=c(0.1, 1)))  
 print(ggarrange(p000, Mset, ncol=1, heights=c(0.1, 1)))  
   
 rm(plot\_data)  
 rm(p1, p2, p3, p4, p5, p6, p7, p8, p9, p0, p00, p000, leg1, leg2,  
 Yset, YMset, Mset)  
 }  
}

## [1] "There are no monitoring locations that qualify."