SEACAR Continuous Water Quality Analysis: SE Region for Dissolved Oxygen Saturation

Last compiled on 15 May, 2023

Table of Contents

# Important Notes

These scripts were created by [J.E. Panzik](mailto:jepanzik@usf.edu) ([jepanzik@usf.edu](mailto:jepanzik@usf.edu)) for SEACAR.

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

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

This markdown file is designed to be compiled by [SEACAR\_WC\_Continuous\_ReportRender.R](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Coral/SEACAR_WC_Continuous_ReportRender.R) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Coral/SEACAR_WC_Continuous_ReportRender.R>).

# Libraries and Settings

Loads libraries used in the script. The inclusion of scipen option limits how frequently R defaults to scientific notation. Sets default settings for displaying warning and messages in created document, and sets figure dpi.

library(knitr)  
library(data.table)  
library(dplyr)  
library(lubridate)  
library(ggplot2)  
library(ggpubr)  
library(scales)  
library(EnvStats)  
library(tidyr)  
library(kableExtra)  
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.

The latest version of WC Continuous data is available at: <https://usf.box.com/s/7ocbmdsm5bgfz6535t8btrnj3r73ysch>

The file being used for the analysis is: **Combined\_WQ\_WC\_NUT\_cont\_Dissolved\_Oxygen\_Saturation\_SE-2022-Nov-16.txt**

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)  
cat(paste("The data file(s) used:", file\_short, sep="\n"))

## The data file(s) used:  
## Combined\_WQ\_WC\_NUT\_cont\_Dissolved\_Oxygen\_Saturation\_SE-2022-Nov-16.txt

# 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.pdf](https://github.com/FloridaSEACAR/SEACAR_Panzik/blob/main/SEACAR%20Documentation%20-%20Analysis%20Filters%20and%20Calculations.pdf)

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.

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 monitoring location, which is that each monitoring location has 5 year or more of unique year entries and have at least 2 consecutive years of observations with at least 2 repeating months for observations that pass the initial filter. If data passes the first set of filtering criteria and the time criteria, they are used in the analysis.

The function that determines whether a monitoring location has at least 2 consecutive years of observations with at least 2 repeating months takes the data, creates a list of the monitoring IDs and cycles through each monitoring ID. For each monitoring ID cycle:

1. List the unique years and put them in ascending order
2. If there are fewer than 2 unique years, skip to the next area
3. If there are 2 or more unique years, start a loop that compares adjacent year entries for the area
   * Start with the first two year entries
4. See if the year entries are subsequent years (1 year apart)
   * If not, skip to next pair of years
5. For the two years being compared, get the list of months for each
6. Compare the two lists of months to see what months are the same
   * If there are two or more months that are the same, the location passes the criteria and is stored in a variable
7. The list of IDs that pass the 2 consecutive years with at least 2 repeating months is returned and used to determine if there is sufficient data for analysis.

A data frame is created that stores summary information for each monitoring location. This information is stored and combined with the results of the Seasonal Kendall Tau analysis and export to a data file once combined.

The sufficient data qualifier is merged with the original data, and a variable Use\_In\_Analysis is created to indicate what data should be used.

A variable with the monitoring IDs that pass all criteria is created and stored.

# Converts Include to be a logical either TRUE or FALSE  
data$Include <- as.logical(data$Include)  
# Removes any data rows that do not have Include set to TRUE  
data <- data[data$Include==TRUE,]  
# Removes rows that have missing ResultValues  
data <- data[!is.na(data$ResultValue),]  
# Removes rows that have missing RelativeDepth  
data <- data[!is.na(data$RelativeDepth),]  
# Rremoves rows that have an ActivityType with Blank  
data <- data[!grep("Blank", data$ActivityType),]  
  
# Removes any data below threshold value of 0, or 5 for Water Temperature  
if(param\_name=="Water\_Temperature"){  
 data <- data[data$ResultValue>=-5,]  
} else{  
 data <- data[data$ResultValue>=0,]  
}  
  
# Gets list of managed areas for the specific region being looked at  
MA\_All\_Region <- MA\_All[MA\_All$Region==region,]  
  
# Gets AreaID for data by merging data with the managed area list for the region  
data <- merge.data.frame(MA\_All\_Region[,c("AreaID", "ManagedAreaName")],  
 data, by="ManagedAreaName", all=TRUE)  
# Creates MonitoringID to more easily cycle through monitoring locations  
data <- data %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName,  
 ProgramLocationID) %>%  
 mutate(MonitoringID=cur\_group\_id())  
  
# Creates function to checks monitoring location for at least 2 years of  
# continuous consecutive data  
ContinuousConsecutiveCheck <- function(con\_data){  
 # Gets MonitoringIDs  
 IDs <- unique(con\_data$MonitoringID[con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # Loops through each MonitoringID  
 for(i in 1:length(IDs)) {  
 # Gets list of Years for MonitoringID  
 Years <- unique(con\_data$Year[con\_data$MonitoringID==IDs[i] &  
 con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # Puts Years in order  
 Years <- Years[order(Years)]  
 # If there are fewer than 2 years, skip to next MonitoringID  
 if(length(Years)<2) {  
 next  
 }  
 # Starts loop to make sure there are at least 2 consecutive years with  
 # consecutive months of data  
 for(j in 2:length(Years)) {  
 # If adjacent year entries are not 1 year apart, skip to the next set  
 # of year entries  
 if(Years[j]-Years[j-1]!=1) {  
 next  
 }  
 # Gets the list of months from the first year  
 Months1 <- unique(con\_data$Month[con\_data$MonitoringID==IDs[i] &  
 con\_data$Year==Years[j-1] &  
 con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # Gets list of months for the second year  
 Months2 <- unique(con\_data$Month[con\_data$MonitoringID==IDs[i] &  
 con\_data$Year==Years[j] &  
 con\_data$Include==TRUE &  
 !is.na(con\_data$Include)])  
 # If there are more than 2 months shared between the two years, the  
 # MonitoringID passes the check and is stored  
 if(length(intersect(Months1, Months2))>=2) {  
 # Creates variable for stored MonitoringID if it doesn't exist  
 if(exists("consecutive")==FALSE){  
 consecutive <- IDs[i]  
 break  
 } else{  
 # Adds to variable for storing MonitoringID if does exist  
 consecutive <- append(consecutive, IDs[i])  
 break  
 }  
 }  
 }  
 }  
 # After going through all MonitoringID, return variable with list of all  
 # that pass  
 return(consecutive)  
}  
  
# Stores the MonitoringID that pass the consecutive year check  
consMonthIDs <- ContinuousConsecutiveCheck(data)  
  
# Creates data frame with summary for each monitoring location.  
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]),  
 ConsecutiveMonths=ifelse(unique(MonitoringID) %in%  
 consMonthIDs==TRUE, TRUE, FALSE),  
 # Determines if monitoring location is sufficient for analysis  
 # based on having more than 0 data entries, more than the  
 #sufficient number of year, and the consecutive month criteria  
 SufficientData=ifelse(N\_Data>0 & N\_Years>=suff\_years &  
 ConsecutiveMonths==TRUE, TRUE, FALSE),  
 Median=median(ResultValue, na.rm=TRUE))  
Mon\_Summ$ConsecutiveMonths <- NULL  
  
# Puts summary data in order based on MonitoringID  
Mon\_Summ <- as.data.table(Mon\_Summ[order(Mon\_Summ$MonitoringID), ])  
  
# Creates column in data that determines how many years from the start for each  
# Monitoring location  
data <- data %>%  
 group\_by(MonitoringID) %>%  
 mutate(YearFromStart=Year-min(Year))  
# Adds SufficientData column to data table based on MonitoringID  
data <- merge.data.frame(data, Mon\_Summ[,c("MonitoringID", "SufficientData")],  
 by="MonitoringID")  
# Creates Use\_In\_Analysis column for data that is determined if the row has  
# Include value of TRUE and SufficientData value of TRUE  
data$Use\_In\_Analysis <- ifelse(data$Include==TRUE & data$SufficientData==TRUE,  
 TRUE, FALSE)  
# Get list of and number of MonitoringID that are to be used in analysis  
Mon\_IDs <- unique(data$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
   * [WC Continuous Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WC_Continuous/output) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WC_Continuous/output>)

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.

# Create summary statistics for each monitoring location based on Year and Month  
# intervals.  
Mon\_YM\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,  
 ProgramLocationID, Year, Month) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=unique(RelativeDepth),  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate), N=length(ResultValue),  
 Min=min(ResultValue), Max=max(ResultValue),  
 Median=median(ResultValue), Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
# Puts the data in order based on ManagedAreaName, ProgramID, ProgramName,  
# ProgramLocationID, Year, then Month  
Mon\_YM\_Stats <- as.data.table(Mon\_YM\_Stats[order(Mon\_YM\_Stats$ManagedAreaName,  
 Mon\_YM\_Stats$ProgramID,  
 Mon\_YM\_Stats$ProgramName,  
 Mon\_YM\_Stats$ProgramLocationID,  
 Mon\_YM\_Stats$Year,  
 Mon\_YM\_Stats$Month), ])  
# Writes summary statistics to file without MonitoringID  
fwrite(select(Mon\_YM\_Stats, -MonitoringID),  
 paste0(out\_dir\_param,"/WC\_Continuous\_", param\_abrev, "\_", region,  
 "\_MonLoc\_MMYY\_Stats.txt"), sep="|")  
# Get year from start for each monitoring location  
Mon\_YM\_Stats <- Mon\_YM\_Stats %>%  
 group\_by(MonitoringID) %>%  
 mutate(YearFromStart=Year-min(Year))  
# Create decimal value of year and month values  
Mon\_YM\_Stats$YearMonthDec <- Mon\_YM\_Stats$Year + ((Mon\_YM\_Stats$Month-0.5) / 12)  
  
# Create summary statistics for each monitoring location based on Year  
# intervals.  
Mon\_Y\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,  
 Year) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=unique(RelativeDepth),  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate), N=length(ResultValue),  
 Min=min(ResultValue), Max=max(ResultValue),  
 Median=median(ResultValue), Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
# Puts the data in order based on ManagedAreaName, ProgramID, ProgramName,  
# ProgramLocationID, then Year  
Mon\_Y\_Stats <- as.data.table(Mon\_Y\_Stats[order(Mon\_Y\_Stats$ManagedAreaName,  
 Mon\_Y\_Stats$ProgramID,  
 Mon\_Y\_Stats$ProgramName,  
 Mon\_Y\_Stats$ProgramLocationID,  
 Mon\_Y\_Stats$Year), ])  
# Writes summary statistics to file  
fwrite(Mon\_Y\_Stats, paste0(out\_dir\_param,"/WC\_Continuous\_", param\_abrev, "\_",  
 region, "\_MonLoc\_Yr\_Stats.txt"), sep="|")  
  
# Create summary statistics for each monitoring location based on Month  
# intervals.  
Mon\_M\_Stats <- data[data$Use\_In\_Analysis==TRUE, ] %>%  
 group\_by(AreaID, ManagedAreaName, ProgramID, ProgramName, ProgramLocationID,  
 Month) %>%  
 summarize(ParameterName=parameter,  
 RelativeDepth=unique(RelativeDepth),  
 EarliestSampleDate=min(SampleDate),  
 LastSampleDate=max(SampleDate), N=length(ResultValue),  
 Min=min(ResultValue), Max=max(ResultValue),  
 Median=median(ResultValue), Mean=mean(ResultValue),  
 StandardDeviation=sd(ResultValue))  
# Puts the data in order based on ManagedAreaName, ProgramID, ProgramName,  
# ProgramLocationID, then Month  
Mon\_M\_Stats <- as.data.table(Mon\_M\_Stats[order(Mon\_M\_Stats$ManagedAreaName,  
 Mon\_M\_Stats$ProgramID,  
 Mon\_M\_Stats$ProgramName,  
 Mon\_M\_Stats$ProgramLocationID,  
 Mon\_M\_Stats$Month), ])  
# Writes summary statistics to file  
fwrite(Mon\_M\_Stats, paste0(out\_dir\_param,"/WC\_Continuous\_", param\_abrev, "\_",  
 region, "\_MonLoc\_Mo\_Stats.txt"), sep="|")  
# Reduces size of data by getting a daily average  
data <- data %>%  
 group\_by(MonitoringID, AreaID, ManagedAreaName, ProgramID, ProgramName,  
 ProgramLocationID, SampleDate) %>%  
 summarise(Year=unique(Year), Month=unique(Month),  
 RelativeDepth=unique(RelativeDepth),  
 ResultValue=mean(ResultValue), Include=unique(Include),  
 Use\_In\_Analysis=unique(Use\_In\_Analysis))  
# Sets column formats to appropriate types  
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)

# 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
   * [WC Continuous Output Files in SEACAR GitHub](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WC_Continuous/output) (<https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/WC_Continuous/output>)

After the analysis is performed, a variable is created that stores the x & y coordinates of the SKT trend line to be used for plotting

# Creates function to get the Kendall Tau statistics  
tauSeasonal <- function(dat, independent, stats.median, stats.minYear,  
 stats.maxYear) {  
 tau <- NULL  
 # Stores results from seasonal Kendall Tau  
 tryCatch({ken <- kendallSeasonalTrendTest(  
 y=dat$Mean,  
 season=dat$Month,  
 year=dat$YearFromStart,  
 independent.obs=independent)  
 # Gets the values of interest from the trend fit  
 tau <- ken$estimate[1]  
 p <- ken$p.value[2]  
 slope <- ken$estimate[2]  
 intercept <- ken$estimate[3]  
 chi\_sq <- ken$statistic[1]  
 p\_chi\_sq <- ken$p.value[1]  
 trend <- trend\_calculator(slope, stats.median, p)  
 rm(ken)  
 # Prints warnings if a fit does not exist and stores values as NA  
 }, warning=function(w) {  
 print(w)  
 }, error=function(e) {  
 print(e)  
 }, finally={  
 if (!exists("tau")) {  
 tau <- NA  
 }  
 if (!exists("p")) {  
 p <- NA  
 }  
 if (!exists("slope")) {  
 slope <- NA  
 }  
 if (!exists("intercept")) {  
 intercept <- NA  
 }  
 if (!exists("trend")) {  
 trend <- NA  
 }  
 })  
 # Puts variables in a vector for the monitoring location currently being  
 # analyzed  
 KT <-c(unique(dat$MonitoringID),  
 independent,  
 tau,  
 p,  
 slope,  
 intercept,  
 chi\_sq,  
 p\_chi\_sq,  
 trend)  
 # Returns the fit parameters  
 return(KT)  
}  
# Function that determines statistics from data  
runStats <- function(dat, med, minYr, maxYr) {  
 #dat$Index <- as.Date(dat$SampleDate) # , "%Y-%m-%d")  
 dat$Mean <- as.numeric(dat$Mean)  
 # Calculate basic stats  
 stats.median <- med  
 stats.minYear <- minYr  
 stats.maxYear <- maxYr  
 # Calculate Kendall Tau and Slope stats assuming they are serially  
 # independent, then store in variable  
 KT <- tauSeasonal(dat, TRUE, stats.median,  
 stats.minYear, stats.maxYear)  
 # If variable returned is empty, run again assuming they are NOT serially  
 # independent  
 if (is.null(KT[8])) {  
 KT <- tauSeasonal(dat, FALSE, stats.median,  
 stats.minYear, stats.maxYear)  
 }  
 # If KT.Stats does not exist, create it and store values  
 if (is.null(KT.Stats)==TRUE) {  
 KT.Stats <- KT  
 # If KT.Stats does exist, add values to it  
 } else{  
 KT.Stats <- rbind(KT.Stats, KT)  
 }  
 return(KT.Stats)  
}  
# Function to determine trend of Kendal Tau  
trend\_calculator <- function(slope, median\_value, p) {  
 # Trend depends on series of conditions  
 trend <-  
 # If the p value is less than 5% and the slope is greater than 10% of the  
 # median value, the trend is large (2).  
 if (p < .05 & abs(slope) > abs(median\_value) / 10.) {  
 if (slope > 0) {  
 2  
 }  
 else {  
 -2  
 }  
 }  
 # If the p value is less than 5% and the slope is less than 10% of the  
 # median value, there is a trend (1).  
 else if (p < .05 & abs(slope) < abs(median\_value) / 10.) {  
 if (slope > 0) {  
 1  
 }  
 else {  
 -1  
 }  
 }  
 # Otherwise, there is no trend (0)  
 else  
 0  
 return(trend)  
}  
  
KT.Stats <- NULL  
# List for column names  
c\_names <- c("MonitoringID", "Independent", "tau", "p",  
 "SennSlope", "SennIntercept", "ChiSquared", "pChiSquared", "Trend")  
# Determines if there are any monitoring locations to analyze  
if(n==0){  
 # Creates data frame to store analysis values in  
 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{  
 # Starts cycling through Monitoring locations to determine seasonal  
 # Kendall Tau  
 for (i in 1:n) {  
 # Gets the number of rows of data for the monitoring location  
 x <- nrow(Mon\_YM\_Stats[Mon\_YM\_Stats$MonitoringID==Mon\_IDs[i], ])  
 # Perform analysis if there is more than 1 row  
 if (x>0) {  
 # Store the monitoring location summary statistics to be used in  
 # trend analysis  
 SKT.med <- Mon\_Summ$Median[Mon\_Summ$MonitoringID==Mon\_IDs[i]]  
 SKT.minYr <- Mon\_Summ$EarliestYear[Mon\_Summ$MonitoringID==Mon\_IDs[i]]  
 SKT.maxYr <- Mon\_Summ$LatestYear[Mon\_Summ$MonitoringID==Mon\_IDs[i]]  
   
 # Get seasonal Kendall Tau statistics by running data for monitoring  
 # location through the functions  
 KT.Stats <- runStats(Mon\_YM\_Stats[Mon\_YM\_Stats$MonitoringID==  
 Mon\_IDs[i], ],  
 SKT.med, SKT.minYr, SKT.maxYr)  
 }  
 }  
   
 # Stores as data frame  
 KT.Stats <- as.data.frame(KT.Stats)  
   
 # If there was only one location, it is stored as a column vector. Change to  
 # row vector  
 if(dim(KT.Stats)[2]==1){  
 KT.Stats <- as.data.frame(t(KT.Stats))  
 }  
 # Sets column and row names for KT.Stats  
 colnames(KT.Stats) <- c\_names  
 rownames(KT.Stats) <- seq(1:nrow(KT.Stats))  
 # Sets valiables to proper format and rounds values if necessary  
 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$ChiSquared <- round(as.numeric(KT.Stats$ChiSquared), digits=4)  
 KT.Stats$pChiSquared <- round(as.numeric(KT.Stats$pChiSquared), digits=4)  
 KT.Stats$Trend <- as.integer(KT.Stats$Trend)  
}  
  
# Combines the KT.Stats with Mon\_Summ  
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), ])  
  
# Writes combined statistics to file  
fwrite(select(KT.Stats, -MonitoringID), paste0(out\_dir\_param,"/WC\_Continuous\_",  
 param\_abrev, "\_", region,  
 "\_KendallTau\_Stats.txt"),   
 sep="|")  
  
#KT.Stats$MonitoringID <- Mon\_Summ$MonitoringID  
data <- data[!is.na(data$ResultValue),]  
  
# Gets x and y values for starting point for trendline  
KT.Plot <- KT.Stats %>%  
 group\_by(MonitoringID) %>%  
 summarize(x=EarliestYear,  
 y=SennIntercept)  
# Gets x and y values for ending point for trendline  
KT.Plot2 <- KT.Stats %>%  
 group\_by(MonitoringID) %>%  
 summarize(x=decimal\_date(LastSampleDate),  
 y=(x-EarliestYear)\*SennSlope+SennIntercept)  
# Combines the starting and endpoints for plotting the trendline  
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.

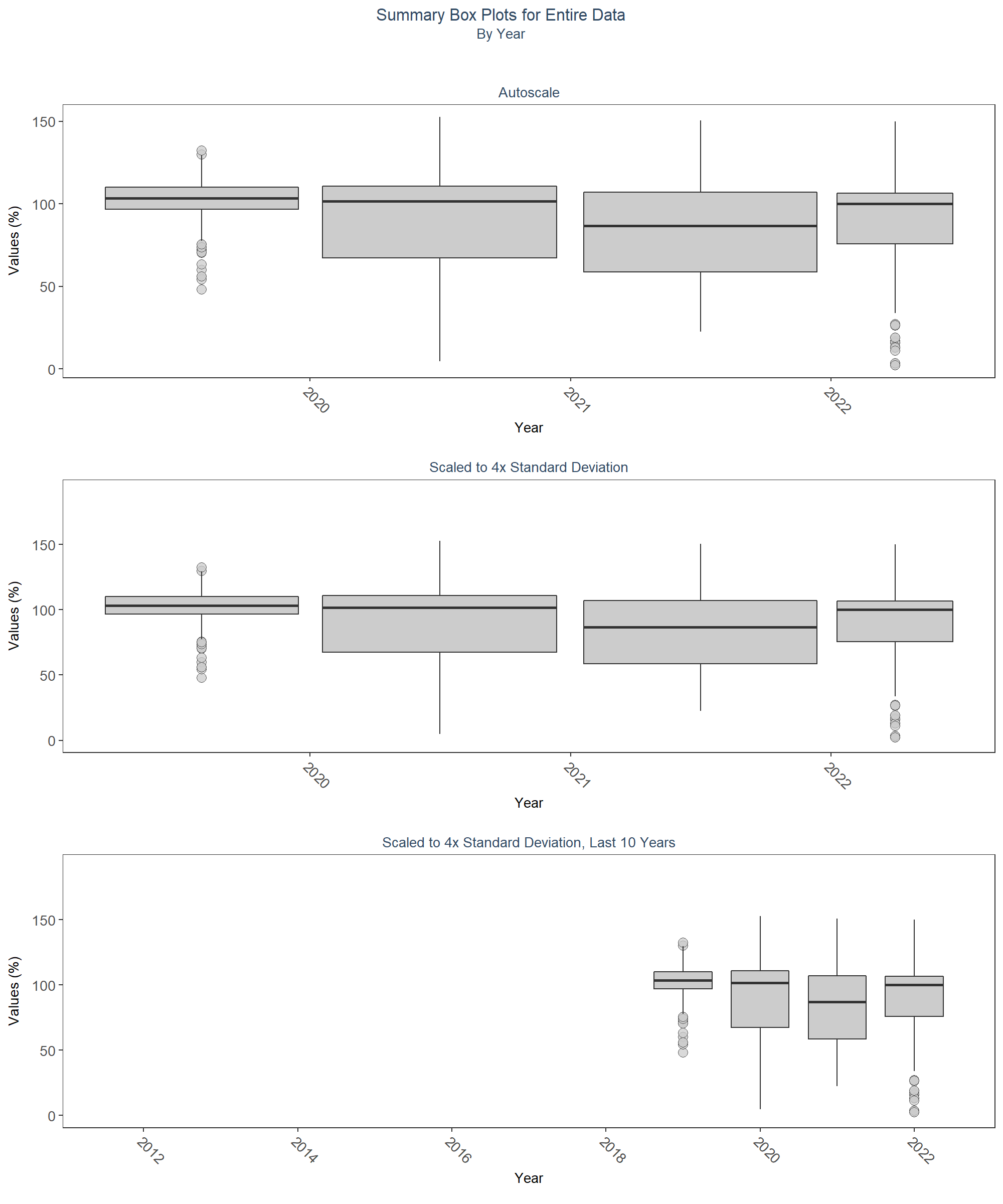
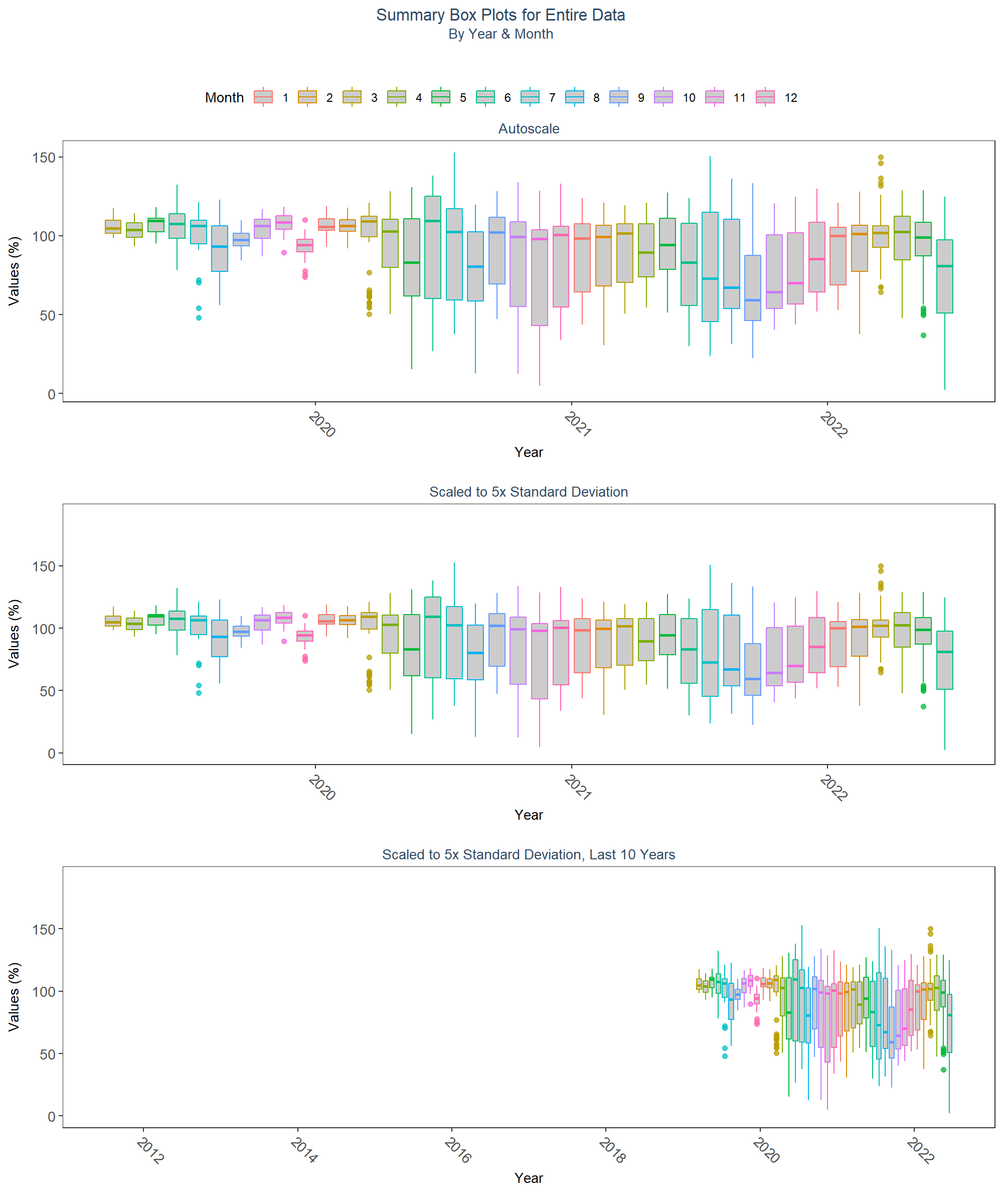
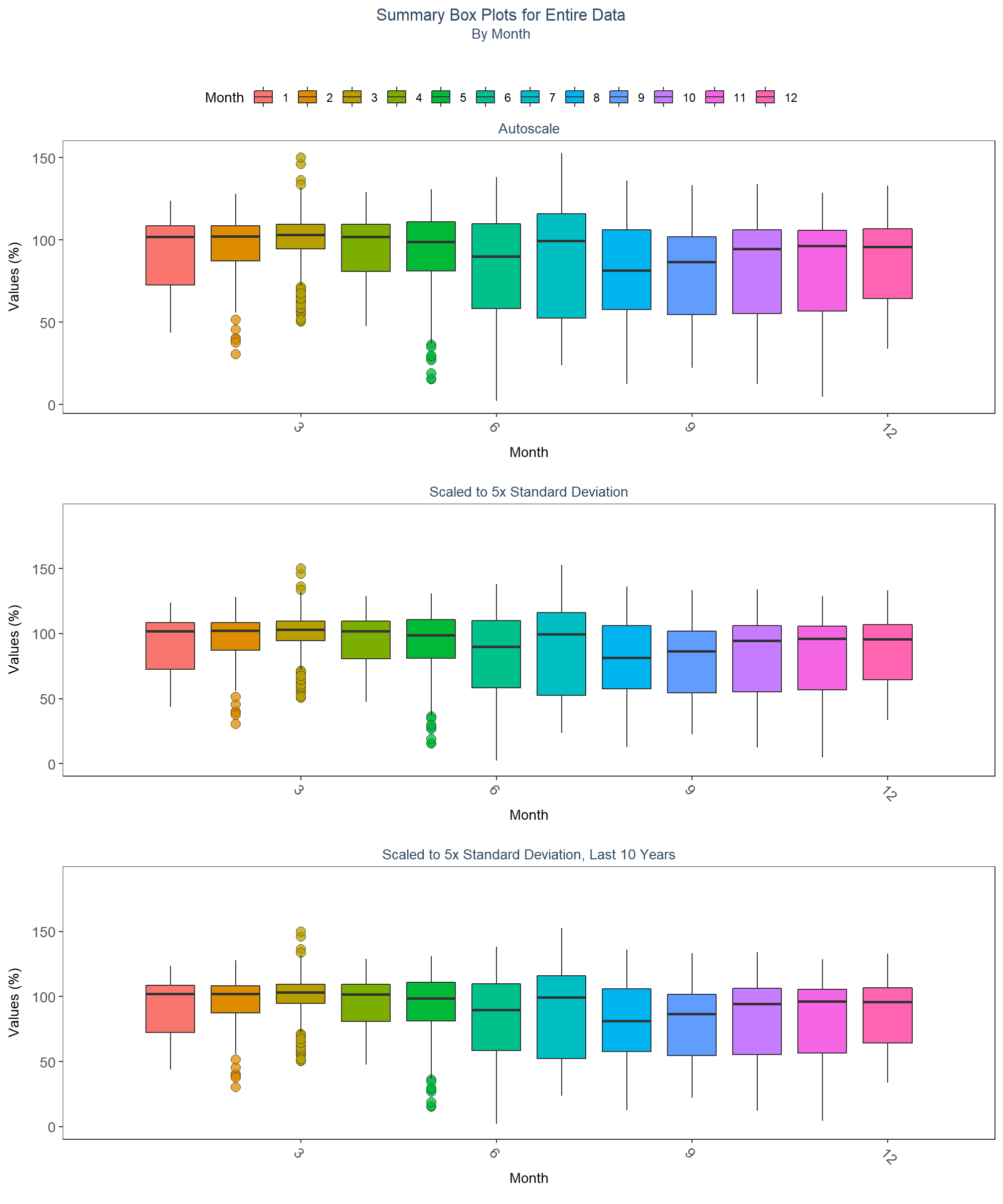
# Defines standard plot theme: black and white, no major or minor grid lines,  
# Arial font. Title is centered, size 12, and blue (hex coded). Subtitle is  
# centered, size 10, and blue (hex coded). Legend title is size 10 and the  
# legend is left-justified. X-axis title is size 10 and the margins are padded  
# at the top and bottom to give more space for angled axis labels. Y-axis title  
# is size 10 and margins are padded on the right side to give more space for  
# axis labels. Axis labels are size 10 and the x-axis labels are rotated -45  
# degrees with a horizontal justification that aligns them with the tick mark  
plot\_theme <- theme\_bw() +  
 theme(panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 text=element\_text(family="Arial"),  
 plot.title=element\_text(hjust=0.5, size=12, color="#314963"),  
 plot.subtitle=element\_text(hjust=0.5, size=10, color="#314963"),  
 legend.title=element\_text(size=10),  
 legend.text.align = 0,  
 axis.title.x = element\_text(size=10, margin = margin(t = 5, r = 0,  
 b = 10, l = 0)),  
 axis.title.y = element\_text(size=10, margin = margin(t = 0, r = 10,  
 b = 0, l = 0)),  
 axis.text=element\_text(size=10),  
 axis.text.x=element\_text(angle = -45, hjust = 0))  
# Get minimum, mean, and standard deviation of the data  
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)])  
# Sets y scale based on data  
y\_scale <- mn\_RV + 4 \* sd\_RV  
  
# Create plot object for auto-scaled y-axis plot  
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=SampleDate, y=ResultValue, group=Year)) +  
 geom\_boxplot(color="#333333", fill="#cccccc", outlier.shape=21,  
 outlier.size=3, outlier.color="#333333",  
 outlier.fill="#cccccc", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")")) +  
 plot\_theme  
# Create plot object for y-axis scaled plot  
p2 <- ggplot(data=data[data$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  
# Create plot object for y-axis scaled plot for past 10 years  
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  
  
# Arrange plot objects  
set <- ggarrange(p1, p2, p3, ncol=1)  
  
# Create title object for plots  
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())  
  
# Arrange title on plots  
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.

# Create plot object for auto-scaled y-axis plot  
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=YearMonthDec, y=ResultValue,  
 group=YearMonth, color=as.factor(Month))) +  
 geom\_boxplot(fill="#cccccc", outlier.size=1.5, outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Year",  
 y=paste0("Values (", unit, ")"), color="Month") +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(color=guide\_legend(nrow=1))  
# Create plot object for y-axis scaled plot  
p2 <- ggplot(data=data[data$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")  
# Create plot object for y-axis scaled plot for past 10 years  
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")  
# Create legend item  
leg <- get\_legend(p1)  
# Arrange plots and legend  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
# Create plot title object  
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())  
# Arrange plots and title  
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.

# Create plot object for auto-scaled y-axis plot  
p1 <- ggplot(data=data[data$Include==TRUE, ],  
 aes(x=Month, y=ResultValue,  
 group=Month, fill=as.factor(Month))) +  
 geom\_boxplot(color="#333333", outlier.shape=21, outlier.size=3,  
 outlier.color="#333333", outlier.alpha=0.75) +  
 labs(subtitle="Autoscale", x="Month",  
 y=paste0("Values (", unit, ")"), fill="Month") +  
 scale\_x\_continuous(limits=c(0, 13), breaks=seq(3, 12, 3)) +  
 plot\_theme +  
 theme(legend.position="top", legend.box="horizontal") +  
 guides(fill=guide\_legend(nrow=1))  
# Create plot object for y-axis scaled plot  
p2 <- ggplot(data=data[data$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")  
# Create plot object for y-axis scaled plot for past 10 years  
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")  
# Create legend object  
leg <- get\_legend(p1)  
# Arrange plots and legend  
set <- ggarrange(leg, p1 + theme(legend.position="none"), p2, p3, ncol=1,  
 heights=c(0.1, 1, 1, 1))  
# Create title object for plots  
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())  
# Arrange plots and title  
Mset <- ggarrange(p0, set, ncol=1, heights=c(0.07, 1))

# Appendix II: 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 averages that have been aggregated by year and month 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 x-axis scale
4. Set the plot type as a line and point plot with the specifics of each
5. Add the linear trend determined form the seasonal Kendall Tau slope and intercept
6. Create the title, x-axis, y-axis, and labels
7. Set the y and x limits
8. Apply the plot theme
9. Set the SKT analysis results as a table figure
10. Combine the plot and table to be displayed together

# Determines whether analyzed monitoring locations exist. If they do, begins  
# looping through them  
if(n==0){  
 print("There are no monitoring locations that qualify.")  
} else {  
 # Begins looping through each monitoring location  
 for (i in 1:n) {  
 # Gets data to be used in plot for monitoring location  
 plot\_data <- Mon\_YM\_Stats[Mon\_YM\_Stats$MonitoringID==Mon\_IDs[i],]  
 # Gets trendline data for monitoring location  
 KT.plot\_data <- KT.Plot[KT.Plot$MonitoringID==Mon\_IDs[i],]  
 #Determine max and min time (Year) for plot x-axis  
 t\_min <- min(plot\_data$Year)  
 t\_max <- max(plot\_data$YearMonthDec)  
 t\_max\_brk <- as.integer(round(t\_max, 0))  
 t <- t\_max-t\_min  
 min\_RV <- min(plot\_data$Mean)  
   
 # Sets break intervals based on the number of years spanned by data  
 if(t>=30){  
 brk <- -10  
 }else if(t<30 & t>=10){  
 brk <- -5  
 }else if(t<10 & t>=4){  
 brk <- -2  
 }else if(t<4){  
 brk <- -1  
 }  
 # Get name of managed area  
 MA\_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID==Mon\_IDs[i]]  
 # Get program location name  
 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]])  
 # Create plot object with data and trendline  
 p1 <- ggplot(data=plot\_data,  
 aes(x=YearMonthDec, y=Mean)) +  
 geom\_line(size=0.75, color="#333333", alpha=0.6) +  
 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, ")")) +  
 scale\_x\_continuous(limits=c(t\_min-0.25, t\_max+0.25),  
 breaks=seq(t\_max\_brk, t\_min, brk)) +  
 plot\_theme  
   
 # Creates ResultTable to display statistics below plot  
 ResultTable <- KT.Stats[KT.Stats$MonitoringID==Mon\_IDs[i], ] %>%  
 select(RelativeDepth, N\_Data, N\_Years, Median, Independent, tau, p,  
 SennSlope, SennIntercept, ChiSquared, pChiSquared, Trend)  
 # Create table object  
 t1 <- ggtexttable(ResultTable, rows=NULL,  
 theme=ttheme(base\_size=10)) %>%  
 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")  
 # Arrange and display plot and statistic table  
 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 III: 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

# Determines whether analyzed monitoring locations exist. If they do, begins  
# looping through them  
if(n==0){  
 print("There are no monitoring locations that qualify.")  
} else {  
 for (i in 1:n) {  
 # Determine upper and lower bounds of time for x-axis  
 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]])  
 # Determine upper and lower bounds of ResultValue for y-axis  
 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$ResultValue, 0.98)])  
 # Sets x- and y-axis scale  
 x\_scale <- ifelse(year\_upper - year\_lower > 30, 10, 5)  
 y\_scale <- mn\_RV + 4 \* sd\_RV  
 # Gets managed area name for title  
 MA\_name <- KT.Stats$ManagedAreaName[KT.Stats$MonitoringID==Mon\_IDs[i]]  
 # Gets program location name for title  
 Mon\_name <- paste0(KT.Stats$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  
 # Create plot object for auto-scaled y-axis plot  
 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  
 # Create plot object for y-axis scaled plot  
 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  
 # Create plot object for y-axis scaled plot for past 10 years  
 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  
 # Arrange plot objects  
 Yset <- ggarrange(p1, p2, p3, ncol=1)  
 # Create plot title object  
 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  
 # Create plot object for auto-scaled y-axis plot  
 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")  
 # Create plot object for y-axis scaled plot  
 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))  
 # Create plot object for y-axis scaled plot for past 10 years  
 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")  
 # Create legend object  
 # Arrange plots and legend  
 leg1 <- get\_legend(p5)  
 YMset <- ggarrange(leg1, p4, p5 + theme(legend.position="none"), p6,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
 # Create plot title object  
 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  
 # Create plot object for auto-scaled y-axis plot  
 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")  
 # Create plot object for y-axis scaled plot  
 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))  
 # Create plot object for y-axis scaled plot for past 10 years  
 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")  
 # Create legend object  
 leg2 <- get\_legend(p8)  
 # Arrange plots and legend  
 Mset <- ggarrange(leg2, p7, p8 + theme(legend.position="none"), p9,  
 ncol=1, heights=c(0.1, 1, 1, 1))  
 # Create title object  
 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())  
 # Arrange and display plots with titles for all combinations  
 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."

# Appendix IV: Excluded Monitoring Locations

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

# Get list of monitoring locations that have data, but without sufficient data  
Mon\_Exclude <- Mon\_Summ[Mon\_Summ$SufficientData==FALSE & N\_Years>0,]  
Mon\_Exclude <- Mon\_Exclude[order(Mon\_Exclude$MonitoringID),]  
z=nrow(Mon\_Exclude)  
# Determines whether excluded monitoring locations exist. If they do, begins  
# looping through them  
if(z==0){  
 print("There are no monitoring locations that qualify.")  
} else {  
 for(i in 1:z){  
 # Get managed area name for title  
 MA\_name <- unique(data$ManagedAreaName[  
 data$MonitoringID==Mon\_Exclude$MonitoringID[i]])  
 # Get program name for title  
 Mon\_name <- paste0(unique(data$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]]))  
 # Create scatter plot with data  
 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)  
 }  
}

