Status and Trend Monitoring Networks
Trend Analysis Protocols

Division of Environmental Assessment and Restoration
Watershed Monitoring Section
Florida Department of Environmental Protection

June 2024

2600 Blair Stone Rd, MS 3560
Tallahassee, Florida 32399-2400
floridadep.gov

Table of Contents

Introduction

This document describes the methods used by the Watershed Monitoring Section (WMS) for the determination of statewide and region-wide water quality trends utilizing data generated by the Status and Trends Monitoring Networks. Complete design and operation details for these networks are provided in the Florida Watershed Monitoring Status and Trend Program Design Document (Florida Department of Environmental Protection, 2024). A brief summary of each network follows.

Initiated in 1998, the Trend Monitoring Network (Trend Network) was designed specifically to track water quality trends at individual river, stream, canal, and aquifer (via wells) stations over time. To achieve this goal, fixed locations are sampled at fixed intervals (monthly or quarterly). The Network consists of 129 stations across the state; 78 surface water, and 51 groundwater sites. The Trend Network complements the Status Network by providing spatial and temporal information about water resources and potential changes from anthropogenic or natural influences, including extreme events (e.g., droughts and hurricanes).

The Status Monitoring Network (Status Network) was designed specifically to provide a ‘snapshot’ of current water quality. It was initiated in 2000 and provides an unbiased, cost-effective sampling of the state's water resources by utilizing an EPA-developed probabilistic design. It is used to provide information on regional and statewide conditions for seven water resources: large and small lakes, rivers, streams, canals, and unconfined and confined groundwater (via wells). Since 2009, a statewide sample survey (cycle) for all water resources is completed annually. Historically two cycles were completed using a rotating basin approach. The first cycle took four years with sampling during 2000-2003; the second took five years with sampling during 2004-2008. Conditions from one statewide sampling cycle may be compared to another for the determination of statewide or region-wide water quality trends.

Both of these monitoring network designs provide estimates of water quality indicators with known statistical confidence. Data produced by the Status and Trend Networks fulfill CWA 305(b) reporting needs and complement CWA 303(d) reporting.

Trend Tests Utilized by WMS

Helsel and Hirsch (2002) categorize trend tests into those using data collected throughout a single period (monotonic trends) and those comparing data collected in two or more non-overlapping periods (step trends). DEP uses Trend Network (monotonic) data for trend determination at individual stations and for region-wide trends. Additionally, Status Network data collected in an early and a late statewide sample cycle (step) can be evaluated for determination of region-wide and statewide trends.

The following methods are used as needed to identify water quality changes over time (trend detection):

1. Seasonal Kendall (SK) test for individual station water quality indicator trend detection. This analysis is run every four years for all trend network stations having sufficient data.
2. Regional Kendal (RK) test for statewide evaluation of non-flow adjusted surface water stations. This test aggregates the results of the individual SK tests into a single result for the region of interest. This analysis is run periodically for all trend network stations having sufficient data, as management needs dictate. Regional Kendal analysis supplements results derived from the Change Analysis described below.

3. Change Analysis (CHAN) for region-wide water quality indicator trend detection. This is a step trend analysis comparing early cycles to later cycles. This analysis is run every four years for flowing surface waters, lakes, and groundwater.

Data Extraction

Water quality data collected for the Status and Trend Networks are stored in the Generalized Water Information System (GWIS), an enterprise Oracle database. Once in GWIS, the data are reviewed by DEP Watershed Monitoring Section (WMS) staff before being deemed suitable for distribution and/or use in analysis. Data extractions from GWIS are performed using the FDEPgetdata R package. This R package (version 1.11 developed 2023-08-03) was developed by WMS staff and is saved on a DEP server (\\floridadep\data\dear\wqap\sol_z\R software\Packages).

For analyses of Trend Network data, files containing the field and analytical results are created using the getdata_trend_results function from the FDEPgetdata package. This function creates a set of four R data frames via an oracle data pull from GWIS for the water resource(s) and range of dates specified by the user.

1. The first data frame, named "Trend_All_Data", contains all data retrieved. These data include metadata elements (Sample ID, Station No, Station Name, Collection Date, Sample Type, Matrix) along with all the results (field and laboratory measurements and data qualifiers) for the specified resource(s) and time period. In addition to retrieving the data, this function replaces measurement values in the R data frame with 'NA' for those measurements having any of the following fatal data qualifiers: '?,O,N,T,X'. Definitions for these codes may be found in FS 62-160.700 Table 1 (Data Qualifier Codes)(**Appendix A**. Data Qualifiers).
2. The second data frame, named "duplicates", contains a subset of the records in "Trend_All_Data" that are duplicated in the GWIS database. Duplicated data are defined as having more than one value for a given combination of station, collection date, and parameter.
3. The third data frame, named "Results_Stacked" contains a copy of "Trend_All_Data" with all records in "duplicates" removed.
4. The fourth data frame, named "Results_Pivoted" contains the same data as "Results_Stacked", but in pivot_wider format.

For analyses of Status Network data, two types of data extractions are performed per water resource. The first pulls all site reconnaissance information. The other pulls the field and analytical data generated from the sites which were sampled.

- The reconnaissance information (also referred to as the site evaluation data) is retrieved using one of the exclusion functions from the FDEPgetdata package (getdata_fw_exclusions for flowing surface waters; getdata_lake_exclusions for lake surface waters; or getdata_aq_exclusions_multi_yr for groundwater). These functions create an R data frame via an oracle data pull from the SITE_EVALUATIONS table of GWIS.
- The field and analytical results information is retrieved using the getdata_results function from the FDEPgetdata package. This function creates an R data frame via an oracle data pull from GWIS for the water resource(s) and year(s) specified by the user. The data retrieved includes certain metadata elements (Sample ID, Station No, Station Name, Collection Date, Sample Type, Matrix) along with all the results (field and laboratory measurements and data qualifiers) for a resource. In addition to retrieving the data, this function replaces measurement values in the R

data frame with 'NA' for those measurements having any of the following fatal data qualifiers: '?,O,N,T,X'. Definitions for these codes may be found in FS 62-160.700 Table 1 (Data Qualifier Codes)(**Appendix A**. Data Qualifiers).

Data Examination / Preparation Prior to Trend Analysis

The WMS conducts a series of procedural steps on the data prior to any of the trend analyses identified above are conducted. The steps are:

1. Examine data qualifiers

Data qualifiers are used to determine the suitability of indicator values for analyses. These qualifiers provide additional information on measurement values and are referenced by codes. Data may be qualified for a variety of reasons described in **Appendix A** (from 2018 revision of Chapter 62-160.700, F.A.C.). **Note:** Prior to October 1, 2017 the qualifier definitions and codes used were those listed in the 2014 revision of Chapter 62-160.700 F.A.C. For the analyses presented in this document any data qualified with any of the fatal qualifiers '?,O,N,T,X' are removed from analysis. Documenting the data analyst's use of data qualifiers in corresponding data analysis reports is required to understand the results.

2. Examine data distribution and identify spurious data

WMS relies on nonparametric statistics for most water quality data analyses. Nonparametric statistics do not rely on data belonging to a particular distribution. These techniques rank data from smallest to highest values and then use these ranks to describe and analyze the data. This process is insensitive to spurious outliers and missing values, which are common in environmental data. Nevertheless, outliers should be examined to determine if they represent data errors. If there are errant data, a decision must be made to include or exclude them from analysis.

One method of determining outliers is the Tukey method (Agresti and Franklin 2013). This method is a nonparametric method which utilizes the first (Q1) and third (Q3) quartiles of a dataset to determine the interquartile (Q3 - Q1) range. For Trend Network analyses, upper outliers are identified as values greater than [(Q3) + (1.5 * IQR)], lower outliers are values less than [(Q1) - (1.5 * IQR)], extreme upper outliers are values greater than [(Q3) + (3 * IQR)], and extreme lower outliers are values less than [(Q1) - (3 * IQR)]. Each outlier and extreme outlier may then be considered a point in need of evaluation (PINE). These PINES are shared with the WMS's Analysis and Reporting Coordinator and Data Manager to determine if the PINES are affected by quality assurance issues such as data entry errors. Once the PINES are examined, the Data Analyst (DA) is notified and determines whether to include or exclude the data.

The DA may create scatter plots, time series plots and boxplots for each station's analytes (**Appendix B**). These graphics allow the analyst to examine data distributions prior to analysis.

For each analyte, the reported minimum concentration value reflects the minimum detection level reported by the laboratory. Limitations of laboratory instruments can cause low concentrations of analytes to be undetected. This is referred to as data censorship. When the analyte level in a sample is lower than the minimum detection level of an instrument, the measurement is marked as BDL (below detection limit). The DEP laboratory marks censored / BDL measurements with the U-qualifier (**Appendix A**). During trend analyses, the DA uses the laboratory's reported value for the method detection limit (MDL).

If a period of record is missing a data point, the statistics (median, quartiles, ranks) are based on the number of actual values in the record. For example, if the December sample is missing from a monthly dataset, the median for the year is based only on the January-November values.

3. Adjust for seasonality in the time series data

For time-series data, such as those obtained from the Trend Monitoring network, seasonal cycles, such as wet and dry periods, can mask trends in water quality indicators. Water constituents commonly show seasonal cycles (seasonality); therefore, an effort should be made to collect at least one sample in each season, or four per year at a minimum. Statistically valid trends may be determined only after adjusting the data for seasonality. This adjustment is referred to as de-seasonalizing the data.

The Seasonal Kendall (SK) and Regional Kendall (RK) tests (see below,) remove effects of seasonal cycles. For Trend Network data analysis, each month is assumed to be a season.

Analyses - Trend Network

Determine if linear time series trends exist for each indicator at each site

During the conceptual development of the Trend Monitoring Network, there was a concern that a linear trend in a time series might be found for one period, while another trend might be found for a different period at the same station. For example, suppose an upward trend was found for the first half of a time series, while a downward trend was found for the second half, or vice versa. Detectable trends depend on the period for which they are analyzed. Thus, if the linear, monotonic trend tests indicate the existence of a trend, it is only representative of the time series period of record.

Seasonal Kendall (SK) Test

The statistical analyses used to evaluate data from the Trend Monitoring Network require a long period of record to be meaningful. WMS uses these sampling frequencies to determine trends: 1) monthly field data collection and laboratory analyses for rivers, streams, and canals; 2) monthly field data collection for unconfined aquifers; and 3) quarterly (once every three months) laboratory analyses for confined and unconfined aquifers, and field data collection for confined aquifers.

Gilbert (1987) stated that when testing for trends using time series data, variations added by regularly spaced cycles make it more difficult to detect trends if they exist. Regarding environmental data, Gilbert mentioned that major cycles are often referred to as seasonality. To address this issue, Hirsch and Slack (1984) developed the SK test. It removes the effect of the seasonal cycles prior to analysis. DEP uses the SK test to look for trends for each indicator at each surface water and groundwater Trend Network site. R software (R Core Team 2021) and the kendallSeasonalTrendTest function in the EnvStats R package (Millard 2013) are used to perform these analyses. An example script is provided in **Appendix C**. Example R Script for SK test**.**

As with seasonality, it is more difficult to detect trends in flowing surface waters with highly variable flow rates. Where available, mean daily flow rate data from nearby USGS, SFWMD, or SJRWMD gauging stations are associated with the surface water samples collected on the same date. USGS flow data are retrieved from the USGS National Water Information System (NWIS) database using the dataRetrieval R package (DeCicco et. al. 2024). SFWMD and SJRWMD flow data are manually downloaded from the districts' respective online data portals. With these added measurements, DEP can adjust surface water quality data for flow before conducting the SK trend analyses. Flow-adjusted data values are calculated for each analyte at each station. The flow-adjusted values reflect the residuals from the Loess regression of water quality analyte values and flow values. In contrast, groundwater flow rates generally are much slower, and DEP makes no flow adjustments prior to performing the SK analyses for groundwater.

If a trend exists for either flow-adjusted or nonflow-adjusted data, DEP determines the corresponding direction of the trend by using the Sen Slope (SS) estimator which measures the median difference between all observations over the time series (Gilbert 1987). The SS provides an estimate of the magnitude of change for a water quality indicator over the period of record. Reporting a trend as increasing or decreasing indicates the direction of the slope and does not necessarily indicate impairment or improvement of the analyte being measured.

The SK test allows for missing values, while BDLs are assigned a value as previously discussed. Each test is run as a two-sided test with the null hypothesis indicating no monotonic linear slope present in the analyte value. The α-level for determining the significance of the results is 0.05.

If there are insufficient data (ISD), trend analyses are not conducted. A time series is considered to have ISD if one or more of the following reasons apply:

1. If there are less than 40 monthly observations in the entire time series.
2. If a data gap in the time series represents ≥ 10% of the total number of observations in the series.
3. If observations are not present in at least two years, for at least one season.

For an individual analyte, the results of a time-series analysis can result in a positive trend (numerical values increase over time), a negative trend (numerical values decrease over time), and insufficient evidence of a trend (values do not change over time) or no trend present. Insufficient evidence of a trend (ISE) can result from any of the following scenarios:

1. The resulting probability level of the Chi-Square heterogeneity test is < 0.05 and the resulting probability level of the trend analysis test is < 0.05
2. The probability level of either the Chi-Square heterogeneity test or the trend analysis test cannot be computed.

A result of no trend is reported in the following scenarios:

1. The resulting probability level of the trend analysis test is ≥ 0.05.
2. The resulting probability level of the trend analysis test is < 0.05 and the Sen-Slope estimator = 0.

A result of increasing trend is reported if the resulting probability level of the trend analysis test is < 0.05 and the Sen-Slope estimator > 0.

A result of decreasing trend is reported if the resulting probability level of the trend analysis test is < 0.05 and the Sen-Slope estimator < 0.

Smoothing procedures make visual inspections of time series plots easier to interpret. For each trend, WMS evaluates possible reasons for an analyte trend.

Determine if regional time series trends exist for water quality indicators over the region of interest

Regional Kendall (RK) Test

The RK test was developed by Helsel and Frans (2006) as an extension of the SK test. The RK test conducts an SK test and computes a Sen slope estimate (Sen 1968) at each site over an entire region. The test then combines the results of each SK test, determines an overall p-value of the test, and determines an overall Sen slope for the region. However, the regional Sen slope should only be used to indicate direction of the trend and not the magnitude when the RK test indicates a regional trend.

DEP uses R software (R Core Team 2021) and the rkt function in the Regional Kendall R package (Marchetto 2017) to perform these analyses. An example script is found in **Appendix D**. Example R script for RK test.

At each site, the RK test needs a minimum number of four values. If a minimum of 10 years of data are available, adjustments for spatial autocorrelation can be made. The RK test works best if the product of the number of sites times the number of years is greater than 25 (Helsel and Frans 2006). In addition to adjusting for seasonality, the "rkt" package of R has an option to adjust for the effects of spatial autocorrelation. WMS uses this option for each RK test.

If a regional or statewide trend exists, its direction is determined using the Sen-Slope (S-S) estimator. For each trend, WMS evaluates possible reasons for an analyte trend.

Analyses - Status Network

Determine if water quality indicator step trends are present

Change Analysis

Change analysis (CHAN) is a step trend procedure for Status Network data which may be used to compare summarized data from one period with those from another period. Helsel et al. (2020) mention that a step trend compares two non-overlapping data sets, an early and late period of record. Changes between the periods are called step trends as values of *Y* (in this case, indicator values) step up or down from one time period to the next. The CHAN is fully described in Dumelle et. Al. (2023). DEP uses the Change Analysis function in R software's (R Core Team 2021) package spsurvey (Dumelle et. al. 2023). The DA writes individual R scripts for each water resource analyzed.

Prior to running CHAN on Status Network data, several data preparation steps are necessary, as follows:

1. For CHAN analysis the three current flowing water resources and two lake resources are combined into two datasets: flowing waters and lakes. This also increases the sample sizes of the respective datasets, as a relatively large number of sites are removed from the early period because they were in locations excluded from more recent coverages. This is unnecessary for confined and unconfined aquifer wells, as their target population definitions have remained stable since the inception of the Status Network.
2. R software projects are created for each resource for which CHAN is to be run. Data extractions from the GWIS are performed using the FDEPgetdata R package, as described in **Data Extraction**.
3. The next step is to make sure the wells or waterbody segments in the combined site evaluations data exist in the final year's water resource coverage. For groundwater resources, the

getdata_aq_exclusions_multi_yr function in the FDEPgetdata package is used to identify wells to be removed when performing multi-year groundwater analyses. For surface water resources, ESRI's ArcGIS Pro is used to determine site locations from the early period's sampled sites that do not fall onto the extent of water resource coverages used for the late time period. Sites that do not fall on the more current resource coverage are removed from the site evaluation data frame using an R script.

4. The extent of the water resources from the final year of the late period's water resource coverage (flowing water length in km or lake area in hectares) are derived during the site selection process. The extent of each water resource in each reporting unit (zone) is copied from the site selection documentation or imported from a .CSV file, created from an R software script run for the site selections. During data analysis these water resource extents in conjunction with the site evaluation information are used to calculate site weighting factors. Only the water resource extent for the final year is used in order to ensure that the statistics are not based on data generated from waterbodies, or waterbody segments, which do not meet the definition of the water resource's target population.
5. The site evaluation and results data frames are merged and a factor column defining the time period is added and populated.
6. Analyte/indicator value distributions are then examined using the methods described above in **Data Examination / Preparation Prior to Trend Analysis**.
7. R scripts are written for each of the water resources to run the change analyses utilizing the R package spsurvey's CHAN function. The analysis results can be summarized in table format or visualized using graphics created with the R package ggplot2 (Wickham 2016).

The CHAN test is used to generate spatial-weight matrices for each water resource by using the extent of each water resource in each of the six Status Network reporting units. Once done, the change analysis function then generates statewide spatially weighted mean and median values, plus upper and lower confidence bounds (CBs) for water quality indicators from both the early period and the late period. An example script is located in **Appendix E**. Example R script for CHAN test.

The CHAN test calculates the difference in spatially weighted means for each indicator between the early (E) and late (L) periods and a 95 % confidence interval for the difference. These values are referred to as difference estimates. If the CBs for the difference estimates between the two periods (L minus E) do not include 0, there is statistical evidence that the values differ. Note that CHAN does not calculate p-values. However, if 0 does not lie within the 95 % CBs for the difference estimate, the p-value is known to be < 0.05. If the number of samples in either the early period or late period is less than 30, CHAN results are not reported due to insufficient data.

<u>Summarize Major Water Quality Issues</u>

Using appropriate graphics and the results from the trend analyses tests, WMS summarizes the major water quality issues and reports them every four years to the US EPA. These summaries can lead to recommendations on future sampling of both surface waters and groundwaters.

References

Agresti, A. and .C. Franklin. 2013. Statistics: The Art and Science of Learning from Data. 3rd Edition. Pearson Education, Inc. ISBN 0-321-75594-4.

DeCicco L, Hirsch R, Lorenz D, Watkins D, Johnson M (2024). dataRetrieval: R packages for discovering and retrieving water data available from U.S. federal hydrologic web services. doi:10.5066/P9X4L3GE, https://code.usgs.gov/water/dataRetrieval

Dumelle M, Kincaid T, Olsen AR, Weber M. 2023. spsurvey: Spatial Sampling Design and Analysis in R. Journal of Statistical Software;105(3):1–29. doi:10.18637/jss.v105.i03

Gilbert, R.O. 1987. *Statistical methods for environmental pollution monitoring*. New York: John Wiley and Sons, Inc.

Florida Department of Environmental Protection. 2024. Florida Watershed Monitoring Status and Trend Program design document. Tallahassee, FL: Division of Environmental Assessment and Restoration, Watershed Monitoring Program.

Helsel, D.R., and R.M. Hirsch. 2002. *Statistical methods in water resources*, Chapter A3. In: Techniques of water-resources investigations of the United States Geological Survey, Book 4, Hydrologic analysis and interpretation. https://pubs.usgs.gov/twri/twri4a3/twri4a3.pdf, accessed March 4, 2020.

Helsel, D.R., and L. M Frans. 2006. Regional Kendall test for trend. *Environ Sci Technol*. 2006;40(13):4066-4073. doi:10.1021/es051650b.

Helsel, D.R., Hirsch, R.M., Ryberg, K.R., Archfield, S.A., and Gilroy, E.J. 2020. Statistical methods in water resources: U.S. Geological Survey Techniques and Methods, book 4, chapter A3, 458 p., https://doi.org/10.3133/tm4a3. [Supersedes USGS Techniques of Water-Resources Investigations, book 4, chapter A3, version 1.1.]

Hirsch, R.M., and J.R. Slack. 1984. Nonparametric trend test for seasonal data with serial dependence. *Water Resources Research* 20(6): 727–732. https://www.researchgate.net/publication/224839899_Non-Parametric_Trend_Test_for_Seasonal_Data_With_Serial_Dependence.

Marchetto, Aldo. 2017. rkt: Mann-Kendall Test, Seasonal and Regional Kendall Tests. R package version 1.5. https://CRAN.R-project.org/package=rkt.

Millard, S.P. 2013. *EnvStats: An R package for environmental statistics*. New York: Springer. ISBN 978-1-4614-8455-4. http://www.springer.com.

R Core Team. 2021. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

Sen, P.K. 1968. Estimates of the Regression Coefficient based on Kendall's Tau. Journal of the American Statistical Association, 63, 1379-1389. http://dx.doi.org/10.1080/01621459.1968.10480934.

Wickham, H. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

Appendices

Appendix A. Data Qualifiers

Data qualifiers used by the department beginning October 1, 2017

This is a two-column table. Column 1 lists the qualifier code and Column 2 lists a description.

Code	Definition
A	Value reported is the arithmetic mean (average) of two or more determinations.
B	Results based upon colony counts outside the acceptable range. This code applies to microbiological tests and specifically to membrane filter colony counts. The code is to be used if the colony count is generated from a plate in which the total number of coliform colonies is outside the method indicated ideal range.
G	Indicates that the analyte was detected at or above the method detection limit in both the sample and the associated field collected blank, and the value of the blank is greater than 10% of the associated sample value.
I	The reported value is greater than or equal to the laboratory method detection limit but less than the laboratory practical quantification limit.
J	Estimate value. **Shall be accompanied by a detailed explanation** to justify the reason(s) for designating the value as estimated. Examples of situations in which a "J" code must be reported include: instances where a quality control item associated with the reported value failed to meet the established quality control criteria (the specific failure must be identified); instances when the sample matrix interfered with the ability to make any accurate determination; instances when data are questionable because of improper laboratory or field protocols (e.g., composite sample was collected instead of a grab sample); instances when the analyte was detected at or above the method detection limit in an analytical laboratory blank other than the method blank (such as calibration blank) and the value the blank is greater than 10% of the associated sample value; or instances when the field or laboratory calibrations or calibration verifications did not meet calibration acceptance criteria.
K	Off-scale low. The actual value is known to be less than the value given. (Only used for lab analyses.)
L	Off-scale high. The actual value is known to be greater than the value given. (Only used for lab analyses.)
N	Presumptive evidence of presence of material; component tentatively identified based on mass spectral library search or there is an indication that the analyte is present, but quality control requirements for the confirmation were not met.
O	Sampled but analysis lost or not performed.
Q	Sample held beyond the accepted holding time. Value is derived from a sample that was prepared or analyzed after the approved holding time restrictions for sample preparation or analysis.
R	Significant rain (typically in excess of ½ inch) in the past 48 hours, which might contribute to a lower or higher than normal value.
S	Secchi disk visible to bottom of waterbody. The value reported is the depth of the waterbody at the location of the Secchi disk measurement.
T	Value reported is less than the laboratory method detection limit. Value reported for informational purposes only and shall not be used in statistical analysis.
U	Indicates that the compound was analyzed for but not detected. The reported value shall be the method detection limit.
V	Indicates that the analyte was detected at or above the method detection limit in both the sample and the associated method blank and the blank value was greater than 10% of the associated sample value.

Code	Definition
X	Indicates, when reporting results from a Stream Condition Index Analysis (LT 7200 and FS 7420), that insufficient individuals were present in the sample to achieve a minimum of 280 organisms for identification (the method calls for two aliquots of 140-160 organisms), suggesting either extreme environmental stress or a sampling error.
Y	The laboratory analysis was from an unpreserved or improperly preserved sample. The data may not be accurate.
Z	Too many colonies were present for accurate counting. Historically, this condition has been reported as "too numerous to count" (TNTC). The "Z" qualifier code shall be reported when the total number of colonies of all types is more than 200 in all dilutions of the sample tested using a membrane filter technique. When applicable to the observed test results, a numeric value for the colony count for the microorganism tested may be estimated from the highest dilution factor (smallest sample volume) used for the test and reported with the qualifier code.
!	Indicates that the reported value deviates from historically established concentration ranges.
?	Data are rejected and should not be used. Some or all of the quality control data for the analyte were outside criteria, and the presence or absence of the analyte cannot be determined from the data.

Appendix B. Example Plots

The plots below (Figures 1 – 3) show examples of boxplots, scatterplots, and time series plots for a single analyte, at individual Trend Network stations.

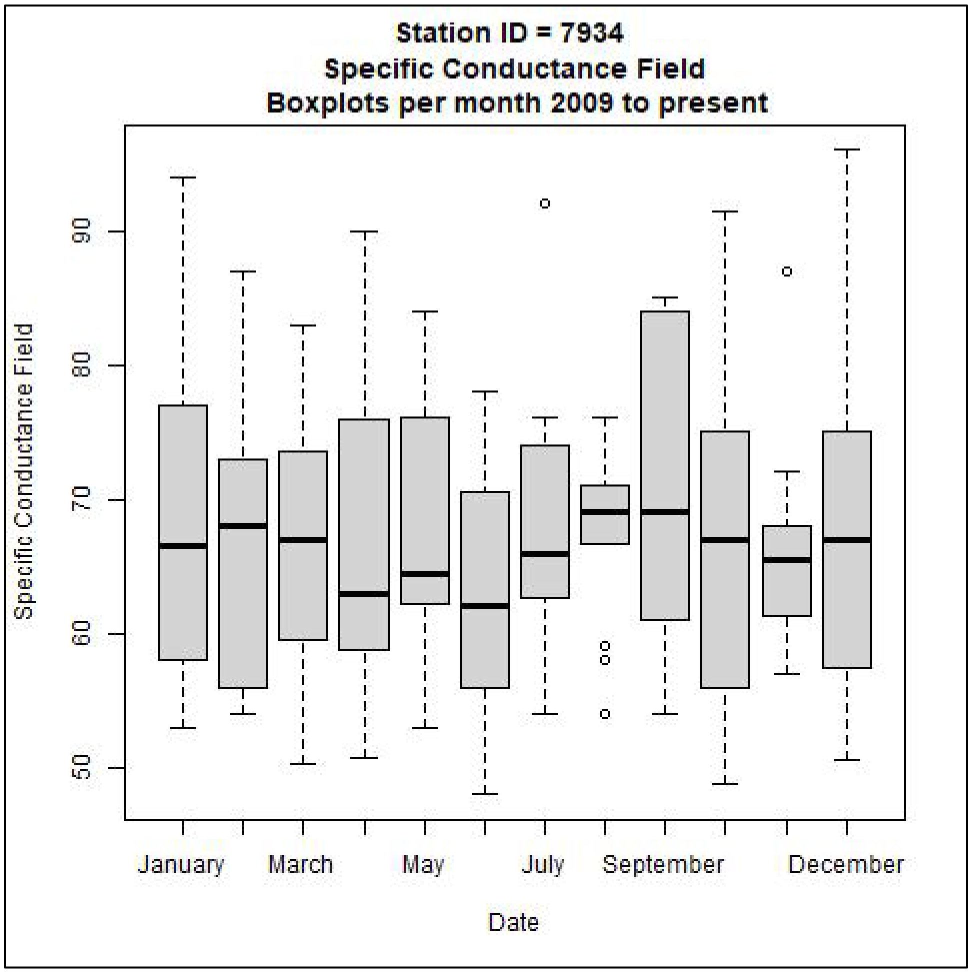

Figure 1. Example of boxplot of specific conductance for a single Groundwater Trend Network station.

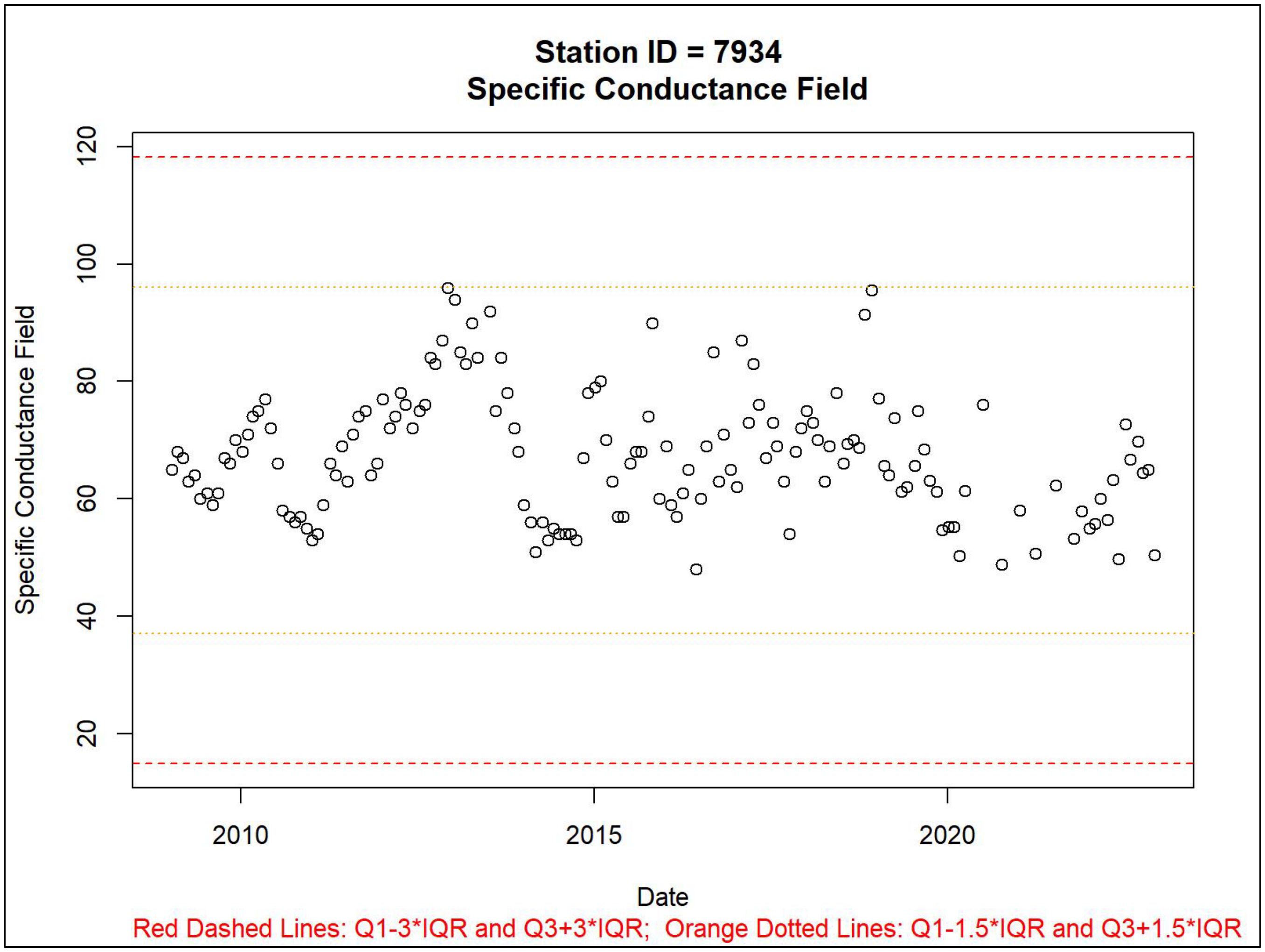

Figure 2. Example of scatterplot of specific conductance for a single Groundwater Trend Network station.

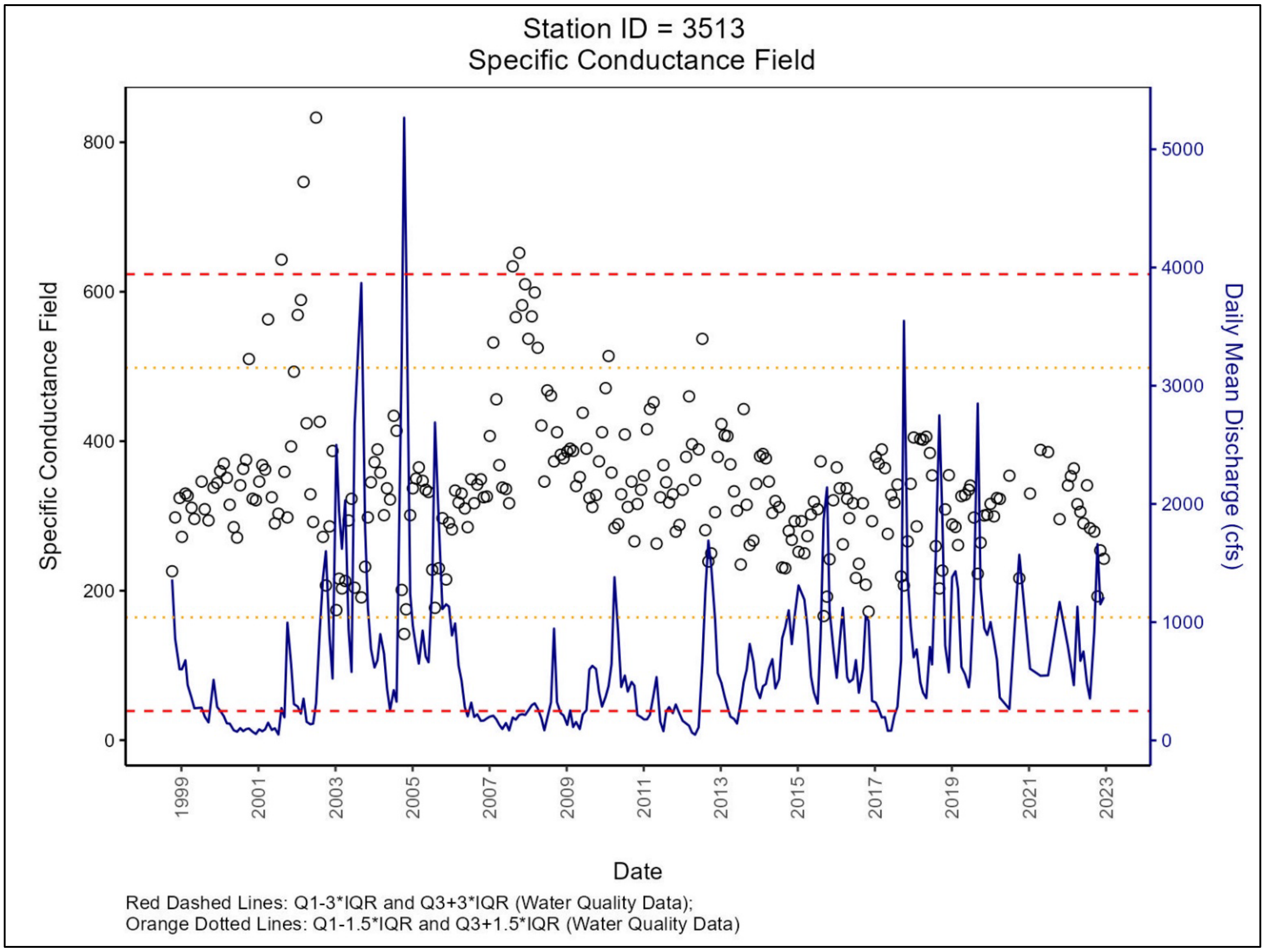

Figure 3. Example of time series plot of specific conductance and mean daily discharge for a single Surface Water Trend Network station.

Appendix C. Example R Script for SK test

R script for non-flow adjusted SK analysis

```
## Non-flow adjusted SK Trend analysis R Script created 6/29/2023
# Run using R version 4.1.3 (2022-03-10) and FDEPgetdata version 1.11.

# This script runs Seasonal Kendall (SK) trend analyses on FDEP Surface Water
#   Trend Network generated data.
# 1) the data are pulled into the R project via the FDEPgetdata package.
# 2) total nitrogen is calculated and added to the dataframe.
# 3) The list of analytes to be analyzed is created.
# 4) The data for trend stations which have been moved from one location
#    on a waterbody to another on the same waterbody are merged.
# 5) A station list is generated from the dataframe and run through a loop
#    which performs the SK analysis for each analyte for each station.
# 6) Output includes
#    a) a folder ("SeasonalKendall_console_logs") containing the SK analysis
#      logs for each station,
#    b) a .csv file containing the statistics generated from the SK analysis,
#    c) A folder ("Plots") containing subfolders for each of the
#       analytes examined, with each subfolder containing plots of the data
#       for each analyte and station used for the SK analysis.
#
# The period of record for the SK analysis is 01-OCT-1998 through 31-DEC-2022.
# The following reductions were made for the data used in the SK analysis.
# 1) Removed E. coil data prior to September 2013 for all stations. E.coli data were
#    not scheduled for collection from July 2001 - September 2013.
# 2) For station 21203, removed all data prior to October 2010.
#    Water quality data were not collected at this station from
#    April 2005 - September 2010

## Open libraries for ODBC Connection to Oracle database GWIS, data manipulation
#   and trend analyses
library(FDEPgetdata)
library(EnvStats)
library(chron)
library(sqldf)

## Use FDEPgetdata package function getdata_trend_results to
#   pull all trend data to be used for the analysis
getdata_trend_results("'CANAL','SPRING RUN','STREAM'","'01-OCT-1998'","'31-DEC-2022'")

## Remove column PK_RESULT as this will cause merging problems and
#  and the data are unnecessary for data analysis.
Results_Stacked <- Results_Stacked[,!(names(Results_Stacked) %in% c("PK_RESULT"))]
```

```
## Determine analyte names in Results_Stacked
unique(Results_Stacked$PARAMETER)

## Calculate Total Nitrogen (TN) from TKN and NOx.
#   Create individual data frames with only results from one analyte.
TN1<- Results_Stacked[sapply(Results_Stacked[,"PARAMETER"],
          function(x)any(x==c('Kjeldahl_Nitrogen_Total__as_N'))),]
TN2<- Results_Stacked[sapply(Results_Stacked[,"PARAMETER"],
          function(x)any(x==c('NitrateNitrite_Total_as_N'))),]
#   Combine TKN and NOx to produce column TN 'total_nitrogen'.
TN3<-merge.data.frame(TN1,TN2,by=1:5)
TN3$TN <- TN3$VALUE.x+TN3$VALUE.y
#   Add data columns for information used to identify each sample.
TN4 <- data.frame('PK_STATION' = TN3$PK_STATION,
                  'FK_PROJECT' = TN3$FK_PROJECT,
                  'COLLECTION_DATE' = TN3$COLLECTION_DATE,
                  'SAMPLE_TYPE' = TN3$SAMPLE_TYPE,
                  'MATRIX' = TN3$MATRIX,
                  'PARAMETER' = "Total_Nitrogen",
                  'VALUE' = TN3$TN,
                  'VALUE_QUALIFIER' = NA)
#   Merge the calculated TN data with the Results_Stacked data that were retrieved from GWIS.
Results_Stacked <- rbind(Results_Stacked, TN4)

# Combine Results from two lab analysis methods for E. coli.
Results_Stacked$PARAMETER <- gsub("Escherichia_coli_Membrane_Filter",
                              "Escherichia_Coli_MF_or_QT",Results_Stacked$PARAMETER)
Results_Stacked$PARAMETER <- gsub("Escherichia_Coli_Quanti_Tray",
                              "Escherichia_Coli_MF_or_QT",Results_Stacked$PARAMETER)

## Preliminary Analysis showed several failures of the data gap QA check.
# Need to make the following reductions to the data used in the SK analysis.
# 1) Removed E. coil data prior to Sep. 2013 for all stations. E.coli data were
#    not scheduled for collection from July 2001 - September 2013.
# 2) For station 21203, removed all data prior to October 2010. Water quality
#    water quality data were not collected at this station from
#    April 2005 - September 2010.
Results_Stacked$Remove <- ifelse(Results_Stacked$PARAMETER == 'Escherichia_Coli_MF_or_QT'
                          & Results_Stacked$COLLECTION_DATE <= '2013-09-01',
                          'Remove - data are before gap',
                               ifelse(Results_Stacked$PK_STATION == '21203' &
                                 Results_Stacked$COLLECTION_DATE <= '2010-10-01',
                                 'Remove - data are before gap', NA))
# Export a copy of the results that were removed for future reference.
Data_Removed_from_SK_Analysis <- subset(Results_Stacked, !is.na(Results_Stacked$Remove))
write.csv(Data_Removed_from_SK_Analysis, 'Data_Removed_from_SK_Analysis.csv',
          row.names = FALSE)
```

```
# Reduce Results_Stacked to only records where Remove is NA. Then drop Remove data field.
Results_Stacked <- subset(Results_Stacked, is.na(Results_Stacked$Remove))
Results_Stacked$Remove <- NULL

## Need to incorporate code to consolidate data from stations that have been moved.
#   PK_STATION for combined data is concatenation of old and new PK_STATION numbers.
# WATERBODY                          OLD PK_STATION          NEW PK_STATION
# CHARLIE CREEK                      3561                    52614
# AEROJET CANAL NUMBER C-111         3570                    37739
# KISSIMMEE RIVER (S-65)             3506                    59629
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 3561] <- 356152614
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 3570] <- 357037739
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 3506] <- 350659629
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 52614] <- 356152614
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 37739] <- 357037739
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 59629] <- 350659629

########### Create PINE_ID Function
# This function will add a column to the data frame called "PINE_Flag" (PINE = Points In Need of
#  Investigation).
# PINE_Flag will be populated with "Upper Outlier Extreme" if the result value is greater than
# [Q3 +  (3*IQR)].
# PINE_Flag will be populated with "Lower Outlier Extreme" if the result value is less than
# [Q1 - (3*IQR)].
# PINE_Flag will be populated with "Upper Outlier" if the result value is greater than [Q3 + (1.5*IQR)].
# PINE_Flag will be populated with "Lower Outlier" if the result value is less than [Q1 - (1.5*IQR)].
# PINE_Flag will be null if none of the conditions stated above are true.
# Function requires input of data frame name. The input data frame must have result values
# in column named "VALUE" and GWIS parameter name in column named "PARAMETER".
# Example:
# EXAMPLE_DATAFRAME_NAME <- PINE_ID(EXAMPLE_DATAFRAME_NAME)
PINE_ID <- function(var1){
   temp_summary <- var1 %>%
   group_by(PARAMETER, PK_STATION) %>%
   summarise(Q1 = quantile(VALUE, (0.25),na.rm = TRUE),
               Q3 = quantile(VALUE, (0.75),na.rm = TRUE),
               IQR = Q3-Q1,
               UpperPINE_Limit = Q3 + (1.5*IQR),
               LowerPINE_Limit = Q1 - (1.5*IQR),
               UpperPINE_Limit_Extreme = Q3 + (3*IQR),
               LowerPINE_Limit_Extreme = Q1 - (3*IQR))
   temp_summary$Q1 <- NULL
   temp_summary$Q3 <- NULL
   temp_summary$IQR <- NULL
   #View(temp_summary)
   var1 <- merge(var1 , temp_summary, by = c("PARAMETER", "PK_STATION"), ALL = TRUE)
   var1$PINE_Flag <- ifelse(var1$VALUE > var1$UpperPINE_Limit_Extreme, 'Upper Outlier
                                    Extreme',
                              ifelse(var1$VALUE < var1$LowerPINE_Limit_Extreme, 'Lower Outlier
                                    Extreme',
                              ifelse(var1$VALUE > var1$UpperPINE_Limit, 'Upper Outlier',
                              ifelse(var1$VALUE < var1$LowerPINE_Limit, 'Lower Outlier', ''))))
   #var1$UpperPINE_Limit <- NULL
   #var1$LowerPINE_Limit <- NULL
   #var1$UpperPINE_Limit_Extreme <- NULL
   #var1$LowerPINE_Limit_Extreme <- NULL
   return(var1)
}
###########End PINE_ID Function Creation. PINE_ID function is now ready for use.

## Run PINE_ID function for Results_Stacked Data Frame
Results_Stacked <- PINE_ID(Results_Stacked)

## Reduce data in Results_Stacked to include only those analytes needed for SK analysis.
input_data <- sqldf('select * from Results_Stacked
                     where Parameter in ("Water_Temperature",
                     "Specific_Conductance_Field",
                     "Oxygen_Dissolved_Field",
                     "pH_Field",
                     "Organic_Carbon_Total",
                     "Total_Nitrogen",
                     "Kjeldahl_Nitrogen_Total__as_N",
                     "NitrateNitrite_Total_as_N",
                     "Phosphorus_Total_as_P",
                     "Chlorophyll_A_Monochromatic",
                     "Escherichia_Coli_MF_or_QT",
                     "Ammonia_Total_as_N",
                     "Alkalinity_Total_as_CaCO3",
                     "Turbidity_Lab",
                     "Chloride_Total",
                     "Sodium_Total",
                     "Calcium_Total",
                     "Magnesium_Total",
                     "Potassium_Total",
                     "Sulfate_Total",
                     "Total_Suspended_Solids_TSS")')

## Date conversions for SK
#   Create column with month of year only
input_data$COLLECTION_DATE<-as.Date(input_data$COLLECTION_DATE, "%m/%d/%Y")
input_data <- as.data.frame(input_data) %>%
   mutate(year = lubridate::year(COLLECTION_DATE),
          Month = month.name[lubridate::month(COLLECTION_DATE, label = TRUE)],
          Moy = lubridate::month(COLLECTION_DATE, label = FALSE))
```

```
# Export a copy of the data that will be used in the SK analysis.
# This data table will include the PINE_Flag information.
write.csv(input_data, 'Results_Input_for_SK.csv', row.names = FALSE)

# Create sub folders for R console logs and plots.
# Note this step will return a warning if these subfolders already exist.
# Proceed with analysis if this warning is received.
dir.create(file.path(getwd(), "SeasonalKendall_console_logs"))
dir.create(file.path(getwd(), "Plots"))
# Within Plots folder, create sub folders for each analyte.
FolderList <- unique(input_data$PARAMETER)
for (i in seq_along(FolderList)) {
   FolderName <- FolderList[i]
   dir.create(file.path(getwd(), "Plots",FolderName))
}

## Use loops to run SK analysis for each analyte at each station. Outer loop runs through list of stations.
#   A text file log of the R console messages is saved for each station. This log contains the SK analysis
#   summary for each analyte at the specified station.
#   Create temporary empty data frame to store statistic results.
SK_combined = data.frame()
# Loop through list of Stations
StationList <- unique(input_data$PK_STATION)
   for (I in seq_along(StationList)) {
   StationID <- StationList[i]
   print(paste0('Station ',StationID))
   # Use a text variable to describe test being performed. This description can be added to text files and
   # tables as needed.
   # ***Remember to update test_description with meaningful information before running the loop.***
   test_description <- paste0('Station ',StationID,' Seasonal Kendall')
   # Within Loop, subset input data frame to only include station of interest.
   Temp_data <- subset(input_data, input_data$PK_STATION == StationID)
   # Begin .TXT file log for each station.
   Sink(paste0('./SeasonalKendall_console_logs/Station_',StationID,'_R_Console_Log.txt'))

   # Inner loop runs through list of analytes and performed SK test and associated quality assurance
   (QA)
   # checks.
   ParamList <- unique(temp_data$PARAMETER)
   for (I in seq_along(ParamList)) {
      ParamName <- ParamList[i]
      #    print(paste0(ParamName))
      print(paste0(test_description, '; Analyte = ', ParamName))
      # Within Loop, subset input data frame to only include analyte of interest.
      Temp_data2 <- subset(temp_data, temp_data$PARAMETER == ParamName)
```

```
# ***QA check for SK Trend test data sufficiency. Loop counts number of observations in each
# season (month), for each year. SK test requires  observations in at least 2 separate years for at
# least one season.
#
# Create empty vector to store year counts for each month.
SK_QA_year_check <- c()
# Loop through list of months in temp_data2 (Single Analyte for SIngle Station)
MonthList <- unique(temp_data2$Month)
for (i in seq_along(MonthList)) {
    MonthName <- MonthList[i]
    # Subset data to single month
    temp_data_qa <- subset(temp_data2, temp_data2$Month == MonthName)
    # Count number of unique years.
    n_years <- length(unique(temp_data_qa$year))
    # Add year count to vector SK_QA_year_check.
    SK_QA_year_check <- c(SK_QA_year_check, n_years)
}
### End subsection of code for QA check for SK Trend test data sufficiency

# ***QA check for data gaps.
#    WMS rule for data gaps is that there must not be gaps greater than or
#    equal to 10% of total number of observations
#    This code is used to calculate the data gap for each pair of consecutive observations for each
#    analyte and station combination.
#    The start date and largest data gap information will be added to
#    the SK_Combined data frame later in the R script.
#
# First sort temp_data2 by COLLECTION_DATE
temp_data2_by_date <- temp_data2[order(temp_data2$COLLECTION_DATE),]
# Reset start_date then calculate start_date
start_date <- c()
start_date <- temp_data2_by_date[1,"COLLECTION_DATE"]
# Create empty vector to store count of number of days between each set of observations.
SK_QA_gap_check <- c()
# Loop through list of observation pairs in temp_data2 (Single Analyte for Single Station)
DateList <- unique(temp_data2_by_date$COLLECTION_DATE)
for (i in seq_along(DateList)) {
    Date1 <- DateList[i]
    Date2 <- DateList[i+1]
    # Calculate number of days between two consecutive observations.
    DateDiff <- as.numeric(Date2 - Date1)
    # Add calculated difference of number of days to vector SK_QA_gap_check.
    SK_QA_gap_check <- c(SK_QA_gap_check, DateDiff)
}
### End subsection of code for QA check for data gaps
```

```
# Data sufficiency requirement for Seasonal Kendall test is observations
# in at least 2 separate years for at least one season
# If this requirement is met, run test and save results in output table.
if(max(SK_QA_year_check, na.rm = TRUE)>=2){
    # Perform Seasonal Kendall Test
    SK_Temp <- kendallSeasonalTrendTest(temp_data2$VALUE,
                                        temp_data2$Month,temp_data2$year,
                                        alternative="two.sided",
                                        correct = TRUE,
                                        ci.slope = TRUE,
                                        conf.level = 0.95,
                                        independent.obs=TRUE,
                                        season_name= NULL,
                                        year_name = null)
     # Display test results in R console.
     print(SK_Temp)

     # Store Seasonal Kendall Results in data frame
     output_SK <- data.frame(test_description,
                             ParamName,
                             SK_Temp[["sample.size"]][["Total"]],
                             SK_Temp[["estimate"]][["tau"]],
                             SK_Temp[["estimate"]][["slope"]],
                             SK_Temp[["interval"]][["limits"]][["LCL"]],
                             SK_Temp[["interval"]][["limits"]][["UCL"]],
                             SK_Temp[["estimate"]][["intercept"]],
                             SK_Temp[["p.value"]][["Chi-Square (Het)"]],
                             SK_Temp[["p.value"]][["z (Trend)"]],
                             start_date,
                             max(SK_QA_gap_check, na.rm = TRUE))
     # Name the data columns. Note these names must match the names in the data frame that they
     # will be merged with.
     names(output_SK) <- c("Test_Description",
                           "Analyte",
                           "Sample_Size_Total",
                           "Tau",
                           "Slope",
                           "Slope LCL",
                           "Slope UCL",
                           "Intercept",
                           "P-Value(Het)",
                           "P-Value(Trend)",
                           "Start_Date",
                           "Max_Data_Gap_Days")
}
### End subsection of code to run if maximim SK_QA_year_check >= 2.
```

```
        # Data sufficiency requirement for Seasonal Kendall test is observations
        # in at least 2 separate years for at least one season
        # If this requirement is not met, don’t run test. Print error message and
        # record NA in output table.
        if(max(SK_QA_year_check, na.rm = TRUE)<2){
            print(paste0('Error in kendallSeasonalTrendTest.default(temp_data2$VALUE,
                  temp_data2$Month. There must be observations in at least 2
                  separate years for at least one season.'))
            output_SK <- data.frame(test_description,
                                    ParamName,
                                    NA,
                                    NA,
                                    NA,
                                    NA,
                                    NA,
                                    NA,
                                    NA,
                                    NA,
                                    start_date,
                                    max(SK_QA_gap_check, na.rm = TRUE))

            # Name the data columns. Note these names must match the names in the data frame that they
            # will be merged with.
            names(output_SK) <- c("Test_Description",
                                  "Analyte",
                                  "Sample_Size_Total",
                                  "Tau",
                                  "Slope",
                                  "Slope LCL",
                                  "Slope UCL",
                                  "Intercept",
                                  "P-Value(Het)",
                                  "P-Value(Trend)",
                                  "Start_Date",
                                  "Max_Data_Gap_Days")
        }
        ### End subsection of code to run if maximim SK_QA_year_check < 2

        # Add results for individual analyte to combined table of results for all analytes.
        SK_combined <- rbind(SK_combined,output_SK)
        }
    ### End for loop of analytes
    # Stop logging R console output for individual station
    sink()
}
###End for loop of stations
```

```
# View Combined Seasonal Kendall Result table
View(SK_combined)

# Check for stations that have fewer analysis results than expected (21 results expected based on input
# data)
SK_combined_analyte_check <- SK_combined %>%
   group_by(Test_Description) %>%
   summarise(Number_of_Analytes = n())
SK_combined_analyte_check <-
SK_combined_analyte_check[order(SK_combined_analyte_check$Number_of_Analytes),]
View(SK_combined_analyte_check)

# Check if Max Data Gap is >= 10% of total observations
# This check assumes that typical sample collection interval is 30 days.
SK_combined$Data_Gap_Exceeds_10PCT <- ifelse((SK_combined$Max_Data_Gap_Days / 30) >=
(SK_combined$Sample_Size_Total * 0.1),'Yes','No')

# Add column to Combined Seasonal Kendall Result table for interpretation of analysis results
SK_combined$SK_Interpret <-
   ifelse(SK_combined$Data_Gap_Exceeds_10PCT == 'Yes', 'Insufficient Data',
   ifelse(SK_combined$Sample_Size_Total < 40, 'Insufficient Data',
   ifelse(!is.na(SK_combined$`P-Value(Het)`) & SK_combined$`P-Value(Het)` < 0.05 &
           SK_combined$`P-Value(Trend)` < 0.05, 'Insufficient Evidence of Trend',
   ifelse(SK_combined$`P-Value(Trend)`>= 0.05, 'No Trend',
   ifelse(SK_combined$`P-Value(Trend)`< 0.05 & SK_combined$Slope == 0, 'No Trend',
   ifelse(SK_combined$`P-Value(Trend)`< 0.05 & SK_combined$Slope > 0, 'Increasing Trend',
   ifelse(SK_combined$`P-Value(Trend)`< 0.05 & SK_combined$Slope < 0, 'Decreasing Trend',
   NA)))))))

#export a copy of the combined Seasonal Kendall results summary as a .CSV file
write.csv(SK_combined, 'Seasonal_Kendall_Results_Combined_NON-FLOW_ADJUSTED.csv',
row.names = FALSE)

### Summarize by analyte and SK interpretation
SK_combined_short <- sqldf('select Analyte, SK_interpret
                          from SK_combined')

SK_Analyte_Sum <- SK_combined_short %>%
   group_by(Analyte, SK_Interpret) %>%
   summarise(n = n())
# Pivot the summary table to wider format (one row per analyte with columns for each value of
SK_Interpret)
SK_Analyte_Interpretation <- pivot_wider(SK_Analyte_Sum, id_cols = c(Analyte),
                          names_from = SK_Interpret,
                          values_from = n, values_fill = NULL, values_fn = NULL)
```

```
#export a copy of the combined seasonal trend analyte-interpretation summary
write.csv(SK_Analyte_Interpretation, 'SK_Analyte_Interpretation_NON-FLOW_ADJUSTED.csv',
          row.names = FALSE)

## The following code creates graphics for each station using loops. The outer loop runs through the list
#   of stations. The inner loop runs through the list of analytes.
StationList <- unique(input_data$PK_STATION)
for (i in seq_along(StationList)) {
   StationID <- StationList[i]
   temp_data <- subset(input_data, input_data$PK_STATION == StationID)
   # Loop through list of Analytes
   ParamList <- unique(temp_data$PARAMETER)
   for (i in seq_along(ParamList)) {
      ParamName <- ParamList[i]
      ParamLabel <- gsub("_"," ",ParamName)
      #    print(paste0(ParamName))
      print(paste0('Station ID = ', StationID, '; Analyte = ', ParamName))
      # Within Loop, subset input data frame to only include analyte of interest.
      temp_data2 <- subset(temp_data, temp_data$PARAMETER == ParamName)

      # Create scatter plot of analyte values
      jpeg(paste0("./Plots/",ParamName,"/",StationID,"_",ParamName,"_Scatter_Plot.jpg"),
            width = 8, height = 6, units = "in",res=200)
      plot(temp_data2$COLLECTION_DATE, temp_data2$VALUE,
          main=(paste("Station ID =",StationID, "\n", ParamLabel)),
          xlab="Date", ylab= ParamLabel,
          xlim=c(min(temp_data2$COLLECTION_DATE),
                  max(temp_data2$COLLECTION_DATE)),
          ylim=c(min(min(temp_data2$VALUE,temp_data2$LowerPINE_Limit_Extreme,na.rm =
                  TRUE)),
                  max(max(temp_data2$VALUE,temp_data2$UpperPINE_Limit_Extreme,na.rm =
                  TRUE))))
          # Add horizontal lines for Q1-3*IQR and Q3+3*IQR
          abline(h=c(temp_data2$UpperPINE_Limit_Extreme,temp_data2$LowerPINE_Limit_Extreme
                  ),col='red', lty='dashed')
          # Add horizontal lines for Q1-1.5*IQR and Q3+1.5*IQR
          abline(h=c(temp_data2$UpperPINE_Limit,temp_data2$LowerPINE_Limit),
                 col='orange', lty='dotted')
          # Add text at bottom of plot describing horizontal lines.
          mtext('Red Dashed Lines: Q1-3*IQR and Q3+3*IQR;  Orange Dotted Lines: Q1-1.5*IQR and
                 Q3+1.5*IQR', side=1, line=4, adj=0, col='red')
      dev.off()

      #   Create boxpolt of value data vs sorted months
      #   Prepare data for boxplot of seasonal data
      temp_data2$month_fac = factor(temp_data2$Month, levels = month.name)
      sort(temp_data2$month_fac)
      #  Create monthly boxplot
```

```
        jpeg(paste0("./Plots/",ParamName,"/",StationID,"_",ParamName,'_BoxPlot.jpg'))
        boxplot(temp_data2$VALUE~(temp_data2$month_fac), data=temp_data2,
                main=(paste("Station ID =",StationID, "\n", ParamLabel,
                        "\n Boxplots per month 1998 to present")),
                xlab="Date", ylab=ParamLabel)
        dev.off()
    }
    ### End for loop of analytes
}
###End for loop of stations
```

Appendix D. Example R script for RK test

```
## Non-flow adjusted surface water RK Trend analysis R Script created 4/29/2023
# Run using R version 4.1.3 (2022-03-10) and FDEPgetdata version 1.11.

# This script runs Regional Kendall (RK) trend analyses on FDEP Surface Water
#   Trend Network generated data.
# 1) the data are pulled into the R project via the FDEPgetdata package.
# 2) total nitrogen is calculated and added to the dataframe.
# 3) The list of analytes to be analyzed is created.
# 4) The data for trend stations which have been moved from one location
#    on a waterbody to another on the same waterbody are merged.
# 5) A loop is used to performs the RK analysis for each analyte for each station.
# 6) Output includes
#    a) a folder ("RegionalKendall_console_logs") containing the RK analysis
#       logs for each analyte,
#    b) a .csv file containing the statistics generated from the RK analysis,
#    c) A folder ("RegionalKendall_Plots") containing subfolders for each of the
#       analytes examined, with each subfolder containing plots of the data
#       for each analyte and station used for the RK analysis.

##  Load R packages needed for analysis.
library(FDEPgetdata)
library(sqldf)
library(rkt)
library(lubridate)

### Use FDEPgetdata package function getdata_trend_results to
##   pull all trend data to be used for the analysis
getdata_trend_results("'CANAL','SPRING RUN','STREAM'","'01-OCT-1998'","'31-DEC-2022'")

# Remove column PK_RESULT as this will cause merging problems and
#  and the data are unnecessary for data analysis.
Results_Stacked <- Results_Stacked[,!(names(Results_Stacked) %in% c("PK_RESULT"))]

# Determine analyte names in Results_Stacked
unique(Results_Stacked$PARAMETER)

## Calculate Total Nitrogen (TN) from TKN and NOx.
#   Create individual data frames with only results from one analyte.
TN1<- Results_Stacked[sapply(Results_Stacked[,"PARAMETER"],
function(x)any(x==c('Kjeldahl_Nitrogen_Total__as_N'))),]
TN2<- Results_Stacked[sapply(Results_Stacked[,"PARAMETER"],
function(x)any(x==c('NitrateNitrite_Total_as_N'))),]
#   Combine TKN and NOx to produce column TN 'total_nitrogen'.
TN3<-merge.data.frame(TN1,TN2,by=1:5)
TN3$TN <- TN3$VALUE.x+TN3$VALUE.y
```

```
#   Add data columns for information used to identify each sample.
TN4 <- data.frame('PK_STATION' = TN3$PK_STATION,
                  'FK_PROJECT' = TN3$FK_PROJECT,
                  'COLLECTION_DATE' = TN3$COLLECTION_DATE,
                  'SAMPLE_TYPE' = TN3$SAMPLE_TYPE,
                  'MATRIX' = TN3$MATRIX,
                  'PARAMETER' = "Total_Nitrogen",
                  'VALUE' = TN3$TN,
                  'VALUE_QUALIFIER' = NA)
#   Merge the calculated TN data with the Results_Stacked data that were retrieved from GWIS.
Results_Stacked <- rbind(Results_Stacked, TN4)

# Combine Results from two lab analysis methods for E. coli.
Results_Stacked$PARAMETER <- gsub("Escherichia_coli_Membrane_Filter",
                                  "Escherichia_Coli_MF_or_QT",Results_Stacked$PARAMETER)
Results_Stacked$PARAMETER <- gsub("Escherichia_Coli_Quanti_Tray",
                                  "Escherichia_Coli_MF_or_QT",Results_Stacked$PARAMETER)

## Preliminary Analysis conducted during RK analysis
#   showed several failures of the data gap QA check.
#   Need to make the following reductions to the data used in the RK analysis.
#  1) Removed E. coil data prior to Sep. 2013 for all stations. E.coli data were
#     not scheduled for collection from July 2001 - September 2013.
#  2) For station 21203, removed all data prior to October 2010. Water quality
#     water quality data were not collected at this station from
#     April 2005 - September 2010.
Results_Stacked$Remove <- ifelse(Results_Stacked$PARAMETER == 'Escherichia_Coli_MF_or_QT'
                                 & Results_Stacked$COLLECTION_DATE <= '2013-09-01',
                                 'Remove - data are before gap',
                              ifelse(Results_Stacked$PK_STATION == '21203' &
                                 Results_Stacked$COLLECTION_DATE <= '2010-10-01',
                                 'Remove - data are before gap', NA))
# Export a copy of the results that were removed for future reference.
Data_Removed_from_RK_Analysis <- subset(Results_Stacked, !is.na(Results_Stacked$Remove))
write.csv(Data_Removed_from_RK_Analysis, 'Data_Removed_from_RK_Analysis.csv',
          row.names = FALSE)

# Reduce Results_Stacked to only records where Remove is NA. Then drop Remove data field.
Results_Stacked <- subset(Results_Stacked, is.na(Results_Stacked$Remove))
Results_Stacked$Remove <- NULL

## Need to incorporate code to consolidate data from stations that have been moved.
#   PK_STATION for combined data is concatenation of old and new PK_STATION numbers.
# WATERBODY                      OLD PK_STATION        NEW PK_STATION
# CHARLIE CREEK                  3561                  52614
# AEROJET CANAL NUMBER C-111     3570                  37739
# KISSIMMEE RIVER (S-65)         3506                  59629
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 3561] <- 356152614
```

```
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 3570] <- 357037739
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 3506] <- 350659629
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 52614] <- 356152614
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 37739] <- 357037739
Results_Stacked['PK_STATION'][Results_Stacked['PK_STATION'] == 59629] <- 350659629

########### Create PINE_ID Function  (Added 8/10/2023)
# This function will add a column to the data frame called "PINE_Flag" (PINE = Points In Need of
# Investigation).
# PINE_Flag will be populated with "Upper Outlier Extreme" if the result value is greater than
# [Q3 + (3*IQR)].
# PINE_Flag will be populated with "Lower Outlier Extreme" if the result value is less than
# [Q1 - (3*IQR)].
# PINE_Flag will be populated with "Upper Outlier" if the result value is greater than [Q3 + (1.5*IQR)].
# PINE_Flag will be populated with "Lower Outlier" if the result value is less than [Q1 - (1.5*IQR)].
# PINE_Flag will be null if none of the conditions stated above are true.
# Function requires input of data frame name. The input data frame must have result values
# in column named "VALUE" and GWIS parameter name in column named "PARAMETER".
# Example:
# EXAMPLE_DATAFRAME_NAME <- PINE_ID(EXAMPLE_DATAFRAME_NAME)
PINE_ID <- function(var1){
  temp_summary <- var1 %>%
  group_by(PARAMETER, PK_STATION) %>%
  summarise(Q1 = quantile(VALUE, (0.25),na.rm = TRUE),
            Q3 = quantile(VALUE, (0.75),na.rm = TRUE),
            IQR = Q3-Q1,
            UpperPINE_Limit = Q3 + (1.5*IQR),
            LowerPINE_Limit = Q1 - (1.5*IQR),
            UpperPINE_Limit_Extreme = Q3 + (3*IQR),
            LowerPINE_Limit_Extreme = Q1 - (3*IQR))
  temp_summary$Q1 <- NULL
  temp_summary$Q3 <- NULL
  temp_summary$IQR <- NULL
  #View(temp_summary)
  var1 <- merge(var1 , temp_summary, by = c("PARAMETER", "PK_STATION"), ALL = TRUE)
  var1$PINE_Flag <- ifelse(var1$VALUE > var1$UpperPINE_Limit_Extreme, 'Upper Outlier
                       Extreme',
                   ifelse(var1$VALUE < var1$LowerPINE_Limit_Extreme, 'Lower Outlier
                       Extreme',
                   ifelse(var1$VALUE > var1$UpperPINE_Limit, 'Upper Outlier',
                   ifelse(var1$VALUE < var1$LowerPINE_Limit, 'Lower Outlier', ''))))
  #var1$UpperPINE_Limit <- NULL
  #var1$LowerPINE_Limit <- NULL
  #var1$UpperPINE_Limit_Extreme <- NULL
  #var1$LowerPINE_Limit_Extreme <- NULL
  return(var1)
}
###########End PINE_ID Function Creation. PINE_ID function is now ready for use.
```

```
# Run PINE_ID function for Results_Stacked Data Frame
Results_Stacked <- PINE_ID(Results_Stacked)

## Reduce data in Results_Stacked to include only those analytes needed for RK analysis.
input_data <- sqldf('select * from Results_Stacked
                    where Parameter in ("Water_Temperature",
                    "Specific_Conductance_Field",
                    "Oxygen_Dissolved_Field",
                    "pH_Field",
                    "Organic_Carbon_Total",
                    "Total_Nitrogen",
                    "Kjeldahl_Nitrogen_Total__as_N",
                    "NitrateNitrite_Total_as_N",
                    "Phosphorus_Total_as_P",
                    "Chlorophyll_A_Monochromatic",
                    "Escherichia_Coli_MF_or_QT",
                    "Ammonia_Total_as_N",
                    "Alkalinity_Total_as_CaCO3",
                    "Turbidity_Lab",
                    "Chloride_Total",
                    "Sodium_Total",
                    "Calcium_Total",
                    "Magnesium_Total",
                    "Potassium_Total",
                    "Sulfate_Total",
                    "Total_Suspended_Solids_TSS")')

## Convert date information not format needed for plots and analysis
input_data$YEAR <- substr(input_data$COLLECTION_DATE, 1, 4)
input_data$MONTH <- substr(input_data$COLLECTION_DATE, 6, 7)
input_data$month<-as.character(input_data$MONTH)
input_data$year<-as.numeric(input_data$YEAR)
input_data$COLLECTION_DATE<-as.Date(input_data$COLLECTION_DATE, "%m/%d/%Y")

input_data$month[input_data$MONTH == '01'] <- 'January'
input_data$month[input_data$MONTH == '02'] <- 'February'
input_data$month[input_data$MONTH == '03'] <- 'March'
input_data$month[input_data$MONTH == '04'] <- 'April'
input_data$month[input_data$MONTH == '05'] <- 'May'
input_data$month[input_data$MONTH == '06'] <- 'June'
input_data$month[input_data$MONTH == '07'] <- 'July'
input_data$month[input_data$MONTH == '08'] <- 'August'
input_data$month[input_data$MONTH == '09'] <- 'September'
input_data$month[input_data$MONTH == '10'] <- 'October'
input_data$month[input_data$MONTH == '11'] <- 'November'
input_data$month[input_data$MONTH == '12'] <- 'December'
```

```
## Convert month data to factor for boxplot by month
input_data$month_fac = factor(input_data$month, levels = c('January', 'February', 'March', 'April',
                                    'May', 'June', 'July', 'August', 'September', 'October', 'November',
                                    'December'))
sort(input_data$month_fac)

## need to create column with month of year only
input_data$moy<-months(input_data$COLLECTION_DATE)

## Prepare for blocking by station and month
#   Get list of unique station IDs
unique(input_data$PK_STATION)
#  There are 78 unique station IDs in input data.
#  Generate a list of the first 78 multiples of 13, to use for BLOCK_STATION_ID.
list_of_13s <- seq(0,1001,by=13)
#  Create data frame with list of station ID and list of new BLOCK_STATION_IDs
BLOCK_PREP <- data.frame(c(unique(input_data$PK_STATION)),list_of_13s)
names(BLOCK_PREP) <- c("PK_STATION", "BLOCK_STATION_ID")
#  Merge BLOCK_PREP with input_data
input_data <- merge(input_data, BLOCK_PREP, by = "PK_STATION", all = TRUE)
#  Calculate BLOCK from mBLOCK_STATION_ID and MONTH
input_data$BLOCK <- as.numeric(input_data$BLOCK_STATION_ID) +
as.numeric(input_data$MONTH)

#  Export a copy of the data that will be used in the RK analysis.
#  This data table will include the PINE_Flag information.
write.csv(input_data, 'Results_Input_for_RK.csv', row.names = FALSE)

# Create sub folders for R console logs and plots.
# Note this step will return a warning if these subfolders already exist.
# Proceed with analysis if this warning is received.
dir.create(file.path(getwd(), "RegionalKendall_console_logs"))
dir.create(file.path(getwd(), "RegionalKendall_Plots"))
# Within Plots folder, create sub folders for each analyte.
FolderList <- unique(input_data$PARAMETER)
for (i in seq_along(FolderList)) {
   FolderName <- FolderList[i]
   dir.create(file.path(getwd(), "RegionalKendall_Plots",FolderName))
}

## Use loops to run RK analysis for each analyte at each station. Outer loop runs through list of stations.
#   A text file log of the R console messages is saved for each station. This log contains the RK analysis
#   summary for each analyte at the specified station.
# Create temporary empty data frame to store statistic results.
RK_combined = data.frame()
# Loop through list of Analytes
ParamList <- unique(input_data$PARAMETER)
```

```
for (i in seq_along(ParamList)) {
  ParamName <- ParamList[i]
  ParamLabel <- gsub("_"," ",ParamName)
  print(paste0('Analyte:',ParamLabel))
  # Use a text variable to describe test being performed. This description can be added to text files and
  # tables as needed.
  # ***Remember to update test_description with meaningful information before running the loop.***
  test_description <- paste0(ParamName,' Regional Kendall')
  # Within Loop, subset input data frame to only include station of interest.
  temp_data <- subset(input_data, input_data$PARAMETER == ParamName)
  # Create .TXT file log for each parameter.
  sink(paste0('./RegionalKendall_console_logs/',ParamName,'_R_Console_Log.txt'))

  ## Histogram of value data
  jpeg(paste0('./RegionalKendall_Plots/',ParamName,'/',ParamName,'_Histogram.jpg'),
    width = 8, height = 6, units = "in",res=200)
  hist(as.numeric(temp_data$MONTH),
    main=paste0(ParamLabel,' Histogram'),
    xlab=ParamLabel, ylab='Frequency')
  dev.off()

  ## Boxpolt of value data vs station
  jpeg(paste0('./RegionalKendall_Plots/',ParamName,'/',ParamName,'_BoxPlot.jpg'),
    width = 10, height = 6, units = "in",res=200)
  boxplot(temp_data$VALUE~(temp_data$PK_STATION), data=temp_data,
    main=(paste(ParamLabel)),
    xlab="Station", ylab=ParamLabel, las=2, cex.axis=0.5)
  dev.off()
  ## Boxpolt of value data vs station
  jpeg(paste0('./RegionalKendall_Plots/',ParamName,'/',ParamName,'_BoxPlot_Monthly.jpg'),
    width = 8, height = 6, units = "in",res=200)
  boxplot(temp_data$VALUE~(temp_data$month_fac), data=temp_data,
    main=(paste(ParamLabel)),
    xlab="", ylab=ParamLabel, las = 2)
  dev.off()

  ## Scatter Plot of analyte values
  jpeg(paste0("./RegionalKendall_Plots/",ParamName,"/",ParamName,"_Scatter_Plot.jpg"),
    width = 8, height = 6, units = "in",res=200)
  plot(temp_data$COLLECTION_DATE, temp_data$VALUE,
    main=(paste("SW Trend Stations 1998-2022")),  xlab="Date", ylab=ParamLabel,
    xlim=c(min(temp_data$COLLECTION_DATE),max(temp_data$COLLECTION_DATE)))
  dev.off()

## Print Summary of data to R Console log
print(summary(temp_data))
```

```
    ##  Run rkt using YEAR, VALUE, BLOCK (BLOCK_STATION_ID + MONTH) using correction
    # for spatial autocorrellation and using a median for values sharing the same date.
    RKT_temp <- rkt(date = as.numeric(temp_data$YEAR),
                    y = temp_data$VALUE,
                    block = temp_data$block,
                    correct = TRUE,
                    rep = "m")
    print(RKT_temp)

    ## Create temporary data frame with key outputs from RK test
    output_df <- data.frame(ParamName,
                            RKT_temp$tau,
                            RKT_temp$S,
                            RKT_temp$varS,
                            RKT_temp$sl,
                            RKT_temp$B,
                            RKT_temp$varS.corrected,
                            RKT_temp$sl.corrected)
    names(output_df) <- c("PARAMETER",
                          "RKT_Tau",
                          "RKT_Score",
                          "RKT_Score_Variance",
                          "RKT_p_value",
                          "RKT_slope",
                          "RKT_Score_Variance_Block_Corrected",
                          "RKT_p_value_Block_Corrected")
    RK_combined <- rbind(RK_combined, output_df)
    # Stop logging R console output for individual station
    sink()
    }
### End for loop of analytes

## Export combined RK results
write.csv(RK_combined, 'Regional_Kendall_Results_Combined_NON-FLOW_ADJUSTED.csv',
          row.names = FALSE)
```

Appendix E. Example R script for CHAN test

CHAN for flowing waters from 2012-2022

```
## R script for step trend analysis of Florida flowing waters data generated by the Status
#   Monitoring Network. Comparison of 2012-2014 to 2020-2022 status flowing waters data
#  via change analysis function in R package spsurvey.
#  Created 8/14/2023. Run in R version 4.1.3 (2022-03-10), using spsurvey version 5.4.1,
#  and FDEPgetdata version 1.11.

## Load libraries for the data analyses
library(FDEPgetdata)
library(spsurvey)
library(sqldf)

## Import data from 2012-2014 and 2020-2022 for flowing waters data analysis.
#   Run function of FDEPgetdata package which pull exclusion data.
#   Insert the string of characters which apply to the names for all stream projects.
#   For instance 2020 large river projects enter "'LR20'".  This will pull the
#   exclusion data for projects Z1LR2004, Z2LR2004, Z3LR2004, Z4LR2004, Z5LR2004,
#   and Z6LR2004 from the FDEP Oracle GWIS database table SITE_EVALUATIONS.
#   Be sure to enclose character string in double and then single quotes.
#   Example for single year "'SS21'"
#   Example for multiple years "'SS19','SS20','SS21'"
#   Function getdata_fw_exclusions creates a dataframe named 'Exclusions' from the
#   information provided.
FDEPgetdata::getdata_fw_exclusions("'CN12','CN13','CN14','LR12','LR13','LR14','SS12','SS13','SS14',
                                   'CN20','CN21','CN22','SS20','SS21','SS22','LR20','LR21','LR22'")

## Determine if any of the sites fall on flowing water segments which are no
#   longer included in the target population. Currently using ArcGIS Pro to create
#   a file with the sites to be removed.  This file is then imported into this R project.
#   The data frame created from the imported table of site to be removed is named
#   "FW_Sites_Not_Within_50Meters".
#   Sites identified as not falling on the most recent GIS coverage of the resource must be removed from
#   the Exclusions data frame.
#   Import CSV file containing the sites to be removed.
FW_sitesNOTwithin50meters <- read.csv('FW_20122014_20202122_Sites_NotWithin50M.csv')

##  Create new data frame, named "SiteEvaluations" by removing the sites not falling within 50  meters
#   from the Exclusions data frame created by FDEPgetdata.
#   SiteEvaluations
#   Remove the stream sites not matching to the 2022 small streams coverage
SiteEvaluations <- sqldf('select * from Exclusions
                          where PK_RANDOM_SAMPLE_LOCATION
                          not in (select PK_RANDOM_ from FW_sitesNOTwithin50meters)')
```

```
#  Remove the river sites not matching to the 2022 large rivers coverage
SiteEvaluations <- sqldf('select * from SiteEvaluations
                              where PK_RANDOM_SAMPLE_LOCATION
                              not in (select PK_RANDO_1 from FW_sitesNOTwithin50meters)')
#   Remove the canal sites not matching to the 2022 canals coverage
SiteEvaluations <- sqldf('select * from SiteEvaluations
                              where PK_RANDOM_SAMPLE_LOCATION
                              not in (select PK_RANDO_3 from FW_sitesNOTwithin50meters)')

# Create copy of the SiteEvaluations data frame to use in the remaining analysis steps.
FW.SITES<-SiteEvaluations
names(FW.SITES)

## Convert location data to decimal degrees
deg <- floor(FW.SITES$RANDOM_LATITUDE/10000)
min <- floor((FW.SITES$RANDOM_LATITUDE - deg*10000)/100)
sec <- FW.SITES$RANDOM_LATITUDE - deg*10000 - min*100
        FW.SITES$latdd <- deg + min/60 + sec/3600
deg <- floor(FW.SITES$RANDOM_LONGITUDE/10000)
min <- floor((FW.SITES$RANDOM_LONGITUDE - deg*10000)/100)
sec <- FW.SITES$RANDOM_LONGITUDE - deg*10000 - min*100
        FW.SITES$londd <- deg + min/60 + sec/3600
#   Change londd to negative for correct use in R package sf (simple features). Negative longitude values
#   indicate location is in the western hemisphere.
FW.SITES$londd <- -FW.SITES$londd

## Convert FW,SITES to sf object and transform to Albers projection for analysis.
#   The transformed data are named "dsgn_sf".
#   This code utilizes Coordinate Reference System (CRS/EPSG) Codes.
#   The first crs code (4269) below is for NAD 83 coordinate system the
#   second crs code (3087) is for Florida albers projection.
#   More information on these codes is found here:
#   https://www.nceas.ucsb.edu/sites/default/files/2020-04/OverviewCoordinateReferenceSystems.pdf.
dsgn_FW <- st_as_sf(FW.SITES, coords = c("londd", "latdd"), remove = FALSE, crs = 4269)
dsgn_sf <- st_transform(dsgn_FW, crs = 3087)
#   Add xy coordinates as variables in dsgn_sf data.
tmp <- st_coordinates(dsgn_sf)
dsgn_sf$xcoord <- tmp[, "X"]
dsgn_sf$ycoord <- tmp[, "Y"]

## Create simple features objects from shapefiles of polygon features representing the reporting units
(Zones).
#   (Watershed_Monitoring_Section_(WMS)_Cycle_3_Reporting_Units).
#   Change projection for Zones sf object to Florida Albers HARN(CRS code 3087).
wms_c3_reporting_units <-
st_read(dsn=".",layer="Watershed_Monitoring_Section_(WMS)_Cycle_3_Reporting_Units")
wms_c3_reporting_units <- st_transform(wms_c3_reporting_units, crs = 3087)
wms_c3_reporting_units
```

```
#   Plot the sf objects for the Zone polygons and sites that were evaluated.
jpeg('2012_13_14-2020_21_22_FW_Evaluated_Sites.jpg', units = 'in', width = 7, height = 7, res = 300)
plot(st_geometry(wms_c3_reporting_units), main= '2012-2014 and 2020-2022\nFlowing Water
Evaluated Sites')
plot(st_geometry(dsgn_sf), pch = 21, bg = 'red', add = TRUE)
legend(120000, 400000, legend='Zones', col='black',lty=1)
legend(120000, 300000, legend='Evaluated Sites', col='red',pch=16)
dev.off()

## Examine site evaluation data.
#   The variables CAN_BE_SAMPLED, EXCLUSION_CATEGORY and EXCLUSION_CRITERIA
#   provide information on the site evaluation results for each site.
#   Review the information and create target/nontarget (TNT) variable.
addmargins(table(dsgn_sf$EXCLUSION_CATEGORY, dsgn_sf$CAN_BE_SAMPLED, useNA =
'ifany'))
addmargins(table(dsgn_sf$EXCLUSION_CRITERIA, useNA = 'ifany'))
dsgn_sf$EXCLUSION_CATEGORY <- as.character(dsgn_sf$EXCLUSION_CATEGORY)
dsgn_sf$EXCLUSION_CATEGORY[dsgn_sf$CAN_BE_SAMPLED == 'Y'] <- 'SAMPLED'
dsgn_sf$EXCLUSION_CATEGORY <- as.factor(dsgn_sf$EXCLUSION_CATEGORY)
levels(dsgn_sf$EXCLUSION_CATEGORY)
dsgn_sf$TNT <- dsgn_sf$EXCLUSION_CATEGORY
levels(dsgn_sf$TNT) <- list(T=c('SAMPLED', 'NO PERMISSION FROM OWNER', 'UNABLE TO
                                ACCESS','OTHERWISE UNSAMPLEABLE','DRY'),
                         NT=c('WRONG RESOURCE/NOT PART OF TARGET POPULATION') )
addmargins(table(dsgn_sf$EXCLUSION_CATEGORY, dsgn_sf$TNT, useNA = 'ifany'))

## Adjust weights for Design As Implemented
#   Determine frame size for combined flowing waters by adding framesize information found in site
#   selections design documentation for each resource (Canals, Rivers and Streams)
framesize <- c("ZONE 1"=13781.7,
               "ZONE 2"=3000.8,
               "ZONE 3"=5636.8,
               "ZONE 4"=5503.3,
               "ZONE 5"=2165.2,
               "ZONE 6"=2202.7)
# Use all evaluated sites to adjust weights
nr <- nrow(dsgn_sf)
dsgn_sf$wgt <- adjwgt(rep(TRUE,nr), wgt=dsgn_sf$NEST1_WT,
                 wgtcat=dsgn_sf$REPORTING_UNIT, framesize=framesize)
# Check sum of weights for each reporting unit. These should match the combined framesize
# information provided above.
addmargins(tapply(dsgn_sf$wgt, dsgn_sf$REPORTING_UNIT, sum))

## Run function of FDEPgetdata package to pull result data.
#   Insert variable name between parentheses in function call below. The  function will pull the water
#   resource for the water resource by year.  For example canal projects during year 2018 the entry
#   would be "'CN18'".
#   Entering "'LR20','LR21','LR22'" for the variable will produce a dataframe for  FDEP Status Rivers
#   sampled 2020 - 2022.
#   Water resource acronyms are CN = canals, LR = rivers, SS, = Streams.
#   Be sure to enclose character string in double and single quotes.
FDEPgetdata::getdata_results("'CN12','CN13','CN14','LR12','LR13','LR14','SS12','SS13','SS14','CN20','
CN21','CN22','SS20','SS21','SS22','LR20','LR21','LR22'")

## Create copy of the Results data frame to use in the remaining analysis steps. Examine result data
#   column names.
FW_RSLTS<-Results
names(FW_RSLTS)

## Create time period column using year from collection date column
FW_RSLTS$TIME_PERIOD <- substr(FW_RSLTS$COLLECTION_DATE, 1, 4)
#  Update time period to indicate the two periods of interest
FW_RSLTS$TIME_PERIOD[FW_RSLTS$TIME_PERIOD == '2020'] <- '2'
FW_RSLTS$TIME_PERIOD[FW_RSLTS$TIME_PERIOD == '2021'] <- '2'
FW_RSLTS$TIME_PERIOD[FW_RSLTS$TIME_PERIOD == '2022'] <- '2'
FW_RSLTS$TIME_PERIOD[FW_RSLTS$TIME_PERIOD == '2012'] <- '1'
FW_RSLTS$TIME_PERIOD[FW_RSLTS$TIME_PERIOD == '2013'] <- '1'
FW_RSLTS$TIME_PERIOD[FW_RSLTS$TIME_PERIOD == '2014'] <- '1'

## Determine sample types in file.
#   Note that sample types present may include BLANK, BOTTOM and PRIMARY data. Sample
#   matrices present may include WATER and BIOLOGY.
addmargins(table(FW_RSLTS$SAMPLE_TYPE, FW_RSLTS$MATRIX, useNA = 'ifany'))
# For statistical analyses of water quality data, use only data with sample type PRIMARY and matrix
# WATER.
keep <- FW_RSLTS$SAMPLE_TYPE == 'PRIMARY' & FW_RSLTS$MATRIX == 'WATER'

## Merge water quality results data with design information (dsgn_sf). This merge will remove any
#   result data marked as inappropriate for Status Network analysis (i.e. where
#   pk_random_sample_location contains "B" or "R").
FW_WQ <- merge(as.data.frame(dsgn_sf)[, c("PK_RANDOM_SAMPLE_LOCATION",
                     "REPORTING_UNIT", "EXCLUSION_CATEGORY","TNT", "wgt",
                     "londd", "latdd", "xcoord", "ycoord", "TN_NNC", "TP_NNC","DO_Conc")],
                     FW_RSLTS[keep,],
                     by.x = 'PK_RANDOM_SAMPLE_LOCATION',
                     by.y = 'FK_RANDOM_SAMPLE_LOCATION')

# Check to confirm the merged data have only PRIMARY sample type and matrix WATER.
addmargins(table(FW_WQ$SAMPLE_TYPE, FW_WQ$MATRIX, useNA = 'ifany'))

## Examine the number of sites per zone in each time_period
addmargins(table(FW_WQ$REPORTING_UNIT, FW_WQ$TIME_PERIOD, useNA = 'ifany'))
```

```
## Check the sum of weights between the two time periods.  How much do they Differ?
tmp0 <- tapply(FW_WQ$wgt, list(FW_WQ$TIME_PERIOD, FW_WQ$REPORTING_UNIT), sum)
tmp0[is.na(tmp0)] <- 0
round(addmargins(tmp0), 1)

## Calculate Total Nitrogen (TN) as sum of TKN & NO3NO2.
FW_WQ$TN<-(FW_WQ$Kjeldahl_Nitrogen_Total_as_N+FW_WQ$NitrateNitrite_Total_as_N)

## Create continuous distribution and run change analyses
nr <- nrow(FW_WQ)
data_cont_WQ <- data.frame(FW_WQ,
                           Combined = rep("All Basins", nr),
                           Basin = FW_WQ$REPORTING_UNIT)
names(data_cont_WQ)
MyVars <- c('Water_Temperature','pH_Field','Oxygen_Dissolved_Percent_Saturation',
          'Specific_Conductance_Field',
          'NitrateNitrite_Total_as_N','Kjeldahl_Nitrogen_Total_as_N',
          'Chlorophyll_A_Monochromatic',  'Ammonia_Total_as_N','TN',
          'Phosphorus_Total_as_P','Alkalinity_Total_as_CaCO3',
          'Total_Suspended_Solids_TSS','Organic_Carbon_Total','Turbidity_Lab',
          'Chloride_Total','Sodium_Total','Calcium_Total','Magnesium_Total',
          'Potassium_Total','Sulfate_Total')

##  Run change analysis on the above indicators' continuous distributions
Change_FW <- change_analysis(data_cont_WQ,
          vars_cont = MyVars,
          subpops = c('Combined','Basin'),
          surveyID = c('TIME_PERIOD'),
          survey_names = c('1','2'),
          siteID = "PK_RANDOM_SAMPLE_LOCATION",
          weight = "wgt",
          xcoord = 'xcoord',
          ycoord = 'ycoord',
          stratumID = 'REPORTING_UNIT',
          vartype = 'Local',
          conf = 95,
          popsize = list(Basin = framesize))

#  Export a copy of the CHAN results
write.csv(Change_FW$contsum_mean, file = '2012_13_14-2020_21_22_FW_ChangeAnalysis.csv',
        row.names = FALSE)
```

```
#  Add a column to the CHAN results data frame for interpretation of statistical results
Change_FW_DF <- data.frame(Change_FW$contsum_mean)
      Change_FW_DF$Change_Interpret <- ifelse(Change_FW_DF$nResp_1 < 30 |
               Change_FW_DF$nResp_2 < 30, 'Insufficient Data',
      ifelse(Change_FW_DF$LCB95Pct > 0 & Change_FW_DF$UCB95Pct > 0, 'Positive Step',
      ifelse(Change_FW_DF$LCB95Pct < 0 & Change_FW_DF$UCB95Pct < 0, 'Negative Step',
      ifelse(Change_FW_DF$LCB95Pct <= 0 & Change_FW_DF$UCB95Pct >= 0, 'No Step',
      ifelse(Change_FW_DF$LCB95Pct >= 0 & Change_FW_DF$UCB95Pct <= 0, 'No Step',
      NA)))))

## Export a copy of CHAN results, with only columns needed for use in report tables.
Change_FW_DF_Subset_Allbasins <- sqldf('select Null Resource, Subpopulation, Indicator,
                                                      Estimate_1, Estimate_2, DiffEst, LCB95Pct, nResp_1,
                                                      UCB95Pct, nResp_2, Change_Interpret
                                                      from Change_FW_DF')
Change_FW_DF_Subset_Allbasins$Resource <- 'Flowing Surface Waters'
write.csv(Change_FW_DF_Subset_Allbasins,
            file = '2012_13_14-2020_21_22_FW_ChangeAnalysis_Subset_Allbasins.csv',
            row.names = FALSE)

## Export a copy of CHAN results, showing only significant trends with only columns needed for use in
#   report tables.
Change_FW_DF_Subset_StateSig <- sqldf('select * from Change_FW_DF_Subset_Allbasins
                                            where Subpopulation ="All Basins"
                                            and Change_Interpret in ("Negative Step","Positive Step")')
write.csv(Change_FW_DF_Subset_StateSig,
            file = '2012_13_14-2020_21_22_FW_ChangeAnalysis_Subset_StateSig.csv',
            row.names = FALSE)
```