Data Analysis Procedures for Status Assessments, Version 5.0

Guidelines for Florida's Probabilistic Monitoring Network

Division of Environmental Assessment and Restoration
Florida Department of Environmental Protection
June 2024

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

Table of Contents

Introduction

The probabilistic design of the Status Network utilizes an unbiased data set for the purpose of answering water-resource questions with known mathematical confidence. The Network allows DEP to address specific questions regarding water quality and answer those questions within statistical confidence limits. The design is planned with specific questions in mind.

Questions addressed by the Status Network monitoring design comprise three different scales: the state of Florida as a whole, (2) regions of the state, and (3) large drainage basins, or drainage basin complexes (i.e., reporting units). The questions that pertain to the Status Network relate to water quality *on a statewide and regional basis*; they are *not waterbody specific*. The Network is not designed to address questions related to small drainage basins, counties, or individual bodies of water. The Integrated Water Resource Monitoring Design[1] (Copeland, et al., 1999) has determined that questions for these smaller areas be addressed by other monitoring programs. The design of the Status Network is for addressing statewide and coarse-scale questions with a high degree of statistical confidence. The current design allows for annual assessments of seven water resource types: Rivers, Streams, Canals, Large and Small lakes, Confined and Unconfined Aquifers (Florida Department of Environmental Protection. 2024).

The Status Network design provides advantages not offered by other designs. For one, the statistical confidence provided by the methodologies and random sampling design provide the advantage of estimating the proportion of waters not meeting the threshold for an analyte within confidence bounds. Secondly, the random design generates proportion results that not only lend themselves to spatial comparisons (e.g., between one part of the state to another) but also temporal comparisons. Therefore, an objective statement with statistical confidence can be generated that tracks the water quality of the state. Hence, the Network will have a unique ability to provide long-term tracking of statewide water quality, with statistical confidence, for decades to come. Comparisons can be made for the entire state through time, for an individual basin, or between different basins given enough sampling events have been conducted. In summary, the Status Network allows us to ask the questions that both the Department of Environmental Protection, and the Florida public, regularly ask. Simply stated: are conditions getting better, remaining the same, or getting worse?

This document is divided into five sections. Part I outlines the operation of the Status Network and questions that may be addressed. Part II describes quality assurance guidelines. Part III describes the procedure used to extract the data that will be used in the analysis. Part IV describes procedures that are used when the questions being addressed require data from multiple years or resources to be combined. Part V takes an example resource, large lakes, and explains how the computer code used for Status Network analysis works.

[1] The Integrated Water Resource Monitoring Program is sometimes referred to by the acronym, “IWRM” (Copeland et. al. 1999)

Part I – The Status Network

The Status Network has three components. First, standardized protocols are employed in data acquisition, e.g. using standardized sample collection methods to minimize error associated with sampling. Second, the data population must be characterized. This means that the population of resources from which the data were collected must be characterized to statistically describe the variability of the distributions of the indicators sampled. Third, the calculated distributions are used to make inferences on the overall condition of the resources (i.e., the original, overall population that was sampled).

For more information on the Status Monitoring Network please refer to https://floridadep.gov/dear/water-quality-assessment/content/reports-documents-sops-and-links for documentation on the design, sample collection methods, data management procedures, etc.

Questions Addressed by the Status Network

Each year the results are reported for analyses conducted on the data on a statewide basis. After a minimum of three years the data are combined and the results are reported for analyses performed on both the statewide data and the data in each of six reporting zones (based on the five water management districts with South split into an east and west section). For the state and each zone, a number of statistics are calculated. These are performed through extent and continuous variable estimate calculations. There is a minimum sample size needed to report of n=23. Appendix A gives example code (in R) for the streams resource. The following types of statistics can be assessed:

- What is the accessible proportion of a resource (surface or groundwater)?
- How many samples were collected for each resource?
- What is the size of the resource?
- What proportion of the resource is dry?
- What proportion of flowing surface waters (i.e., rivers, streams, and canals) do not meet water quality thresholds?
- What proportion of lake surface waters (i.e., large lakes and small lakes) do not meet water quality thresholds?
- What proportion of groundwater (i.e., as confined and unconfined aquifers) do not meet water quality thresholds?
- For flowing surface water resources, what proportion of dissolved oxygen, total nitrogen, total phosphorus, *Escherichia coli*, pH, total ammonia nitrogen, chlorophyll a, and habitat metrics do not meet water quality thresholds?
- For groundwater resources, what proportion of arsenic, cadmium, chromium, lead, nitrate+ nitrite, sodium, fluoride, and total coliform do not meet water quality thresholds?

Part II – Data Extraction

Water quality data collected for the Status Network 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). Two types of data extractions are performed per water resource. One pulls all site reconnaissance information including the sites which are excluded from sample collection and the reason why each site was excluded. 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 created using one of the exclusion functions from the FDEPgetdata package (getdata_fw_exclusions for flowing surface waters; getdata_lake_exclusions for lake surface waters; getdata_aq_exclusions 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. For surface waters, the data retrieved from the SITE_EVALUATIONS table include applicable total nitrogen (F.A.C. 62-302.531), total phosphorus (F.A.C. 62-302.531), and / or dissolved oxygen (F.A.C. 62-302.533) criteria based on the corresponding nutrient watershed region and bioregion.

The file containing the field and analytical results (from here on in referred to as the 'results file') is created 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).

The Analysis and Reporting Coordinator notifies the data analyst when all data for each water resource and year are complete in GWIS and ready to be extracted using the functions described above. The site evaluation data are used to provide information on the extent of waters determined to be non-target (not part of the water resource being assessed), and the extent of waters excluded due to other reasons. This information is used during data analysis to calculate weighting factors for the sites which were sampled.

Part III – Data Quality Assessments

Status Network data are collected and analyzed in accordance of the quality assurance protocols established for the Watershed Monitoring Section's Status Monitoring Network. Proper field and laboratory protocols are followed prior to their incorporation into the database. Data extracted from the database are then examined by the data analyst. This includes scanning data for outliers and erroneous values. After the analyses are complete, they are independently checked, using the quality assurance checklist in Appendix C as a guide.

The data analyst performs the following checks before data analyses.

Errors in the Data

The data should be screened for anomalies, outliers and questionable results based on proper ranges of values. Data that are orders of magnitude in variance from the central tendency are the most easily identified. Other values may fall outside of the logical range of values (e.g., negative values for NO_3). Continuous variable data (i.e., those not including raw count values) should

have values greater than zero. Unusual values may warrant further investigation, including examination of original field records.

Outliers

One of the most common types of data anomalies are outliers. Outliers should not be discarded or arbitrarily defined. Though arbitrary standards can be set, all data handling is ultimately a matter for best professional judgment. For example, one way of defining outliers is identification of points beyond what are referred to as the upper fence. In order to determine the upper fence of a data distribution, the Inter Quartile Range (IQR) (the value of the upper quartile minus the value of the lower quartile) is determined. The upper fence is then found by adding the quantity 1.5 times the IQR to the upper quartile. However, a point 1.5 times the upper quartile is often a valid data point, and erroneous data values may reside within this range. Ultimately, all decisions are subject to the analyst's best judgment given the data that is provided. Since most Status Network analyses are nonparametric, being based on the rank of the values as opposed to the values themselves, outliers are less problematic than they are for parametric analyses. Removal of data from the dataset should only occur under the most obvious of circumstances. These include measurements that are mathematical impossibilities (e.g., pH values greater than 14), conditions that reflect physical impossibilities of condition (e.g., temperatures of 2700 C), mislabeled data, and laboratory instrumentation errors.

Missing Values

The Status Network sampling design for one cycle (one calendar year) calls for 15 collection points per zone for each surface water resource and 20 for each groundwater resource. When less than these values are present in a specific zone, the reasons must be documented.

Detection Limits

Some reported data are at detection limits. The Status Network employs the laboratory's reported value for the method detection limit (MDL) for incorporation into continuous distribution functions (CDFs) for each indicator of water quality. For indicators with water quality thresholds, measurement values are replaced with 'NA' during Status Network data analysis if the reported values are both above the threshold value (e.g. > 4 MPN/100 mL for total coliform) and are listed as below the method detection limit ('U' qualifier).

Qualifier Codes

Status Network qualifier codes are recorded in Appendix B and adhere to the 2017 QA Rule 62-160.700 Table 1 (Data Qualifier Codes). The type of qualifier, the reason for the particular qualifier, and the overall number of qualifiers should all be considered. It is recommended that qualified data remain in the analysis, except for data with fatal qualifiers (?,O,N,T,X) that are removed during data extraction as described above (Part II – Data Extraction). Clear documentation of the reasons for the qualification of the data should be presented alongside the results. CDFs constructed from heavily qualified data—or of much qualified data near threshold ranges—will have compromised accuracy. Qualified values exceeding thresholds pose the greatest need for additional examination and explanation.

Predetermined guidelines for how to handle various combinations of data qualifiers can be assigned. For example, analysis results could be flagged when the data includes 5 or more qualifiers. Whether, or not, such guidelines are established, data analysts and readers/editors of the reports must ultimately make decisions on a case-by-case basis as to whether, or not, to

accept the findings. In any case, readers should be given enough information to make informed decisions about the resource(s) in question.

Specific issues regarding data qualifiers include the following:

- Q qualified data – samples held beyond holding times. Bacteria data from the lab are submitted only when samples are analyzed within holding times or up to 30 hours outside holding times. All Q qualified bacteria data are currently used based on an in-house study of the Biology Section of the lab showing little degradation of *Escherichia coli* samples up to 30 hours outside holding times (Florida Department of Environmental Protection. 2013).
- J qualified data— J qualified data may have failed QA and QC protocols, whether in the field or the lab.
- V qualified data—data that registers detection in both samples and laboratory method blanks.
- G qualified data—data that registers detection in both samples and field blanks or equipment blanks.

Part IV – Combining Data

Combining Data from Multiple Years

Increased sample size is desirable because it generally has a positive effect on the confidence levels for the reported data, thereby increasing the confidence for statewide reporting. One way to increase the sample size is to combine data collected in different years. For status monitoring reporting, three years of annually collected status monitoring network data are used. The increase in sample size allows statewide reporting with 95% confidence levels at $\pm < 10\%$.

The process of analyzing three years of data requires several steps which are important to maintaining the statistical integrity of the analyses. First, for each water resource, site evaluation data and results files for each of the three years are merged to provide a combined site evaluation R data frame and a combined results R data frame. The FDEPgetdata R package functions described in Part II – Data Extraction perform the data merging as part of the data retrieval process when the user specifies multiple water resources or years. These two combined data sets are checked to ensure that all data fields of interest are populated for all three years. If there are data fields in the site evaluations data that are missing values for one or more years, the site evaluations data may need to be reextracted after the GWIS SITE_EVALUATIONS table is updated by the Data Coordinator. If there are data fields in the results file that are missing values for one or more years (usually due to an analyte being dropped from or added to the analyte list), a decision must be made regarding whether to include those analytes in the analysis. It is desirable to keep as many analytes as possible in the final three-year result file.

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 example, only the water resource coverage for 2017 is used in the 2015-2017 analysis. The location of the sites sampled for 2015 and 2016 are checked to make sure they came from wells or waterbodies present in the 2017 water resource coverage. Any sites that are not a part of the final year's water resource coverage are removed from the combined exclusions and results files before processing. WMS staff have developed several tools to accomplish these checks. An ArcGIS Pro python script

geoprocessing tool (saved on DEP server \\floridadep\data\dear\wqap\sol_z\gis\SurfaceWaterTool_SiteEvaluations) is used to identify sites to be removed when performing multi-year surface water analyses. 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,

The extent of the water resources from the final year's water resource coverage (flowing water length in km or lake area in hectares) are derived during the site selection process from the shapefile used and are recorded in the accompanying design document (initially saved on DEP server \\floridadep\data\dear\wqap\sol z\Status Sample Survey Designs and then copied to the appropriate folder in \\floridadep\data\dear\wqap\sol z\data analysis). The extent of each water resource in each zone is 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 exclusion 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.

Combining Data from Multiple Resources

It is also desirable to report on the condition of combined water resources: (e.g. flowing waters - rivers, streams, and canals, and lakes - large and small). This will increase the sample size and allow a more global look at the condition of the state's waters which also may be tracked over time.

When combining the flowing water resources, the lengths calculated from the coverages for all three resources (rivers in zones 1-6, streams in zones 1-6, and canals in zones 3-6) are summed to obtain the extent (total flowing waters length in kilometers) for the combined rivers and streams resource (LRSS) and the combined rivers, streams, and canals (LRSSCN) resource. These are the lengths used in the R software script as either targetsize or framesize to recalculate the weightings.

When combining the lakes water resources, the areas calculated from the coverages for both resources (large lakes and small lakes) are summed to obtain the extent (total lakes area in hectares) for the combined lakes resource (LLSL). These are the areas used in the R software script as either targetsize or framesize to recalculate the weightings.

Part V – Analysis Procedures

The following is an example of data analysis on Status Network large lakes data from 2021-2023. An R script is used to generate the statistical output. The script was developed from original S-PLUS© code written and supplied by Tony Olsen, a US EPA statistician stationed at the US EPA facility in Corvallis, OR (the original script is saved on a DEP server \\floridadep\data\dear\wqap\sol z\data analysis\2006_305B\Old SPLUS Codes\305b 2006-all codes\HomeDesktop\Suwannee River\Large Lakes). Analyses for other resources use similar script that has been adapted to the resource in question.

Running the R scripts

The following code runs in R Studio version 2023.12.1 build 402, using R Statistical Software version 4.1.3, 32-bit platform.

Start R Studio. Create a new R project, saved to the DEP server data analysis folder (\\floridadep\data\dear\wqap\sol z\data analysis) for the appropriate year and resource. Check that the R workspace is set to the same location as where the R project was created.

Locate the R script file from the previous year's analysis for the same resource, save a copy in the same location as where the R project was created, and rename it to reflect the year and resource being analyzed. Open the renamed R script using R Studio. Read through the script and make minor edits as needed to reflect the year and resource being analyzed.

To run a portion of the script, highlight relevant portions of text code in the script window, and hit the run button (small green arrow near upper right of script window). The console window will show the output, including any error messages received. Run sections of code step-by-step in order to make sure each step runs properly (e.g., no errors). Sections of code are divided by comment lines. (In R these are specified by # symbols preceding text and will not execute with other script. They are descriptive only). Comments are updated each time that the R script is run, to reflect the data analyst's notes about the data or analysis results.

Explanations for Each Portion of Script:

The following illustrates the script and provides commentary. Portions of script code are listed below in indented paragraphs (full code example for streams is in Appendix A), and explanations are beneath in italics.

```
# Code developed using R version 4.1.3 (2022-03-10), spsurvey version 5.4.1,
# FDEPgetdata version 1.11.
# Load libraries for the data analyses
library(FDEPgetdata)
library(spsurvey)
library(sqldf)
```

Determine if the required packages are installed. The FDEPgetdata package can be installed using the install package from Package Archive File tool in the R studio tools menu. The spsurvey and sqldf packages can be downloaded from the Comprehensive R Archive Network (CRAN) contributed packages repository (https://cran.r-project.org/web/packages/index.html) or archives (https://cran.r-project.org/src/contrib/Archive/). After confirming that the required packages are installed load them for analysis.

```
# Run function of FDEPgetdata package which will pull exclusion data.
# Insert variable name between parentheses in function call below.
FDEPgetdata::getdata_lake_exclusions("'LL21','LL22','LL23'")
```

Extract the site evaluation data from GWIS (contact WMS Data Coordinator for access credentials). Function getdata_lake_exclusions creates a data frame named 'Exclusions' from the information provided. Corresponding Nutrient watershed region and bioregion are incorporated for each site.

```
# Determine if any of the sites fall on lakes 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 the R project. The name of the #
file to be imported is LL_sitesNOTintersects.csv.
LL_sitesNOTintersects<-read.csv("LL_sitesNOTintersects.csv")
```

Determine if any of the sites fall on lake segments which are no longer included in the target population. Use ArcGIS Pro python script geoprocessing tool to create a table (.csv file) containing the sites that should be removed from this analysis because they no longer are in the target population. Import this file into this R project using the read.csv function. Name the data frame created during the import step "LL_sitesNOTintersects".

```
# Create a new data frame, named SiteEvaluations, by taking all sites from the
# Exclusions data frame that do not match sites in the LL_sitesNOTintersects
# data frame.
SiteEvaluations <- sqldf('select * from Exclusions
                         where PK_RANDOM_SAMPLE_LOCATION
                         not in (select PK_RANDOM_SAMPLE_LOCATION
                         from LL_sitesNOTintersects)')
```

Note that once data have been loaded into the current R workspace they are then stored within R and are ready to be used in other commands. Once the site removals data (LL_sitesNOTintersects) are loaded, create a new site evaluations data frame by removing the sites contained in LL_sitesNOTintersects.

```
# Create a copy of the SiteEvaluations data frame for use in next steps of
analysis. New data frame is named LL.SITES.
LL.SITES<-SiteEvaluations
names(LL.SITES)

# Convert to Decimal degrees
deg <- floor(LL.SITES$RANDOM_LATITUDE/10000)
min <- floor((LL.SITES$RANDOM_LATITUDE - deg*10000)/100)
sec <- LL.SITES$RANDOM_LATITUDE - deg*10000 - min*100
LL.SITES$latdd <- deg + min/60 + sec/3600
deg <- floor(LL.SITES$RANDOM_LONGITUDE/10000)
min <- floor((LL.SITES$RANDOM_LONGITUDE - deg*10000)/100)
sec <- LL.SITES$RANDOM_LONGITUDE - deg*10000 - min*100
LL.SITES$londd <- deg + min/60 + sec/3600

# Change londd to negative for correct use in sf.
LL.SITES$londd <- -LL.SITES$londd

# Create sf object and transform to Albers projection for analysis
#  This codes 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_LL <- st_as_sf(LL.SITES,
coords = c("londd", "latdd"), remove = FALSE, crs = 4269)
dsgn_sf <- st_transform(dsgn_LL, crs = 3087)

# keep xy coordinatess as variables
tmp <- st_coordinates(dsgn_sf)
dsgn_sf$xcoord <- tmp[, "X"]
dsgn_sf$ycoord <- tmp[, "Y"]
```

In order to calculate variance, an equal area map projection must be used. This set of commands transforms the map projection from NAD83 to Florida Albers, creates a matrix, x and y, and assigns this to the site information data frame. Latitude and longitude should be provided in decimal degree format. If they are provided in a different format, additional commands will be needed to convert to decimal degrees prior to running these commands.

```
# Create a simple features object from a shapefile of polygon features
representing the Zones
# 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

# Use sf to plot the Zone polygons and sites that were evaluated.
jpeg('2021_2023_LL_Evaluated_Sites.jpg', units = 'in',
     width = 7, height = 7, res = 300)
     plot(st_geometry(wms_c3_reporting_units),
     main= '2021-2023 Large Lake 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()
```

A visual inspection of the site location data is performed by creating a simple map that shows the site locations and the reporting units (Zones).

```
# check design variables
addmargins(table(dsgn_sf$EXCLUSION_CATEGORY, dsgn_sf$CAN_BE_SAMPLED,
             useNA = 'ifany'))
addmargins(table(dsgn_sf$EXCLUSION_CRITERIA, useNA = 'ifany'))

# create sampled and target (T) / nontarget (NT) variables
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'))
```

Before proceeding with the remainder of the code, it is helpful to examine the loaded data. Rather than scanning through a data frame, it is easier to look at summary information

Variable TNT represents Target and Non-target samples. The exclusion categories are given but this is an additional simplification. The exclusion categories are subsumed into two categories: T and NT. Non-target (NT) samples are those sites that were excluded as wrong resource / not part of target population. Target (T) samples are sites that were either sampled or excluded for other reasons that are unlikely to be permanent (e.g. unable to access, dry, or no permission from owner).

```
#####  adjust weights for design as implemented
# Save framesize CSV file to R project workspace, then load framesize data into
# R project.
framesize.df <- read.csv('2023 Large Lake Framesize.csv')
```

```
# Reduce framesize to remove total row.
# Change Zone name values to all capital letters.
framesize.df <- framesize.df[framesize.df$Zones !='Sum',]
framesize.df$Zones <- toupper(framesize.df$Zones)

# Convert framesize data frame to named vector
framesize <- as.vector(framesize.df$hectares)
names(framesize) <- framesize.df[,'Zones']

# View framesize and paste results in comments.
print(framesize)
#   ZONE 1   ZONE 2   ZONE 3   ZONE 4   ZONE 5   ZONE 6
#  18077.9   7581.9 121935.8  43095.0  60545.3 129999.5
sum(framesize)
#  Framesize in hectares = 381235.4 for entire data frame.
```

The framesize values are imported from a .CSV file, created from an R software script run for the site selections. The framesize values are the total size of the resource in each reporting unit. The framesize data format is changed to a named vector. This reformatting ensures that the reporting unit names can be used to link the framesize data to the data in the dsgn_sf data frame.

```
# 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$REPORTING_UNIT, framesize=framesize)
dsgn_sf$Combined = rep("All Basins", nr)

# check sum of weights for each reporting unit/basin
addmargins(tapply(dsgn_sf$wgt, dsgn_sf$REPORTING_UNIT, sum))
```

The adjust weights step accounts for oversampling and NT designations. Ideally a selection of 15 lake sites would result in 15 samples: complete accessibility. But what if 15 samples required 20 attempts? Sample size changes (by 20/15). This is the ratio adjustment to account for oversamples. [Entering "help(adjwgt)" in the R command window will display the function description].

Analysis of a different resource will require changing the name of the imported framesize file, as well as data columns names, in order for the code to work. Otherwise, calculations for other resources employ nearly identical code.

```
##### Large Lake area extent estimation
dsgn <- data.frame(PK_RANDOM_SAMPLE_LOCATION =
                   dsgn_sf$PK_RANDOM_SAMPLE_LOCATION,
                   wgt = dsgn_sf$wgt,
                   xcoord = dsgn_sf$xcoord,
                   ycoord = dsgn_sf$ycoord,
                   Basin = dsgn_sf$REPORTING_UNIT,
                   Combined = dsgn_sf$Combined,
                   TNTStatus = dsgn_sf$TNT,
                   EXCLUSION.CATEGORY = dsgn_sf$EXCLUSION_CATEGORY)
```

```
ExtentEst <- cat_analysis(dsgn,
                          vars = c('TNTStatus','EXCLUSION.CATEGORY'),
                          subpops = c('Combined','Basin'),
                          siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                          weight = 'wgt',
                          xcoord = 'xcoord',
                          ycoord = 'ycoord',
                          stratumID = 'Basin',
                          vartype = "Local",
                          conf=95)

# Add data columns to ExtentEst results. These will be used when loading
analysis results to GWIS tables for Status Network analysis results.
analysis_date <- as.character(Sys.Date())
                              ExtentEst <- cbind(ExtentEst,
                              SAMPLE_YEAR = '2021-2023',
                              REPORTING_CYCLE = '15-17',
                              WATER_RESOURCE = 'LL',
                              ANALYSIS_DATE = analysis_date,
                              MATRIX = 'WATER',
                              ESTIMATEU_UNITS = 'HECTARES')

# export results
write.csv(ExtentEst, file='2021_2023_LL_ExtentEst.csv',
          row.names = FALSE)
```

There are different types of data: categorical and continuous. Examples of categorical data are exclusion categories, or whether something was sampled or not. The "cat_analysis" function is used for these types of data. [Entering "help(cat_analysis)" in the R command window will display the function description].

All sites included in the analysis are represented by the data frame "dsgn". The part of the population examined is represented by the cat_analysis function argument "subpops". "Subpops" recombines all the data statewide and can group populations into smaller units (e.g., Reporting Units, Basins, etc.).

Additional cat_analysis function arguments give the survey design used (sites, weights, and stratification). Though x and y coordinates are also specified in the function arguments, ultimately these are not used in making estimates for CDFs. No geographic information goes into a CDF, though geographic information is employed within confidence limits or variance calculations (use of a local neighborhood variance calculation in cat_analysis, is specified by the "vartype" argument).

The cat_analysis function argument "vars" gives the data used in the analysis, which is either 1) an exclusion categorical variable, or 2) a TNT variable.

The data frame "ExtentEst" provides an estimated extent. For example, an estimate of the total area of dry lakes in Zone 2. If 1.6% of lake area is dry, there would be 118.1 ha of dry lake area in a sample size of 7581.9 ha of lakes.

```
#  To estimate the percent of the target population
#  that could be sampled, requires that the analysis be restricted to just sites
#  in the target population, i.e., TNT = "T"
dsgn <- subset(dsgn, dsgn$TNT == "T")
```

```
ExtentEst_Target <- cat_analysis(dsgn,
                                 vars = c('TNTStatus','EXCLUSION.CATEGORY'),
                                 subpops = c('Combined','Basin'),
                                 siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                 weight = 'wgt',
                                 xcoord = 'xcoord',
                                 ycoord = 'ycoord',
                                 stratumID = 'Basin',
                                 vartype = "Local",
                                 conf=95)

# Export results
write.csv(ExtentEst_Target, file = ' 2021_2023_LL_ExtentEst_Target.csv')
```

The analysis is rerun with only sites in the target population included in the analysis. The data frame "ExtentEst_Target" provides an estimated extent of the target population.

```
###### Water Quality Data Analysis
# Run function of FDEPgetdata package which will pull result data.
# Insert variable name between parentheses in function call below.
FDEPgetdata::getdata_results("'LL21','LL22','LL23'")
```

Extract the result data from GWIS. Function getdata_results creates a data frame named 'Results' from the information provided. Replaces measurement values with 'NA' for those measurements having fatal data qualifiers as described in Part II – Data Extraction.

```
# Create new data frame from the one just created.
LL_RSLTS<-Results
names(LL_RSLTS)

# Determine sample types in file.
addmargins(table(LL_RSLTS$SAMPLE_TYPE, LL_RSLTS$MATRIX, useNA = 'ifany'))

# Note that original data spreadsheet may have blank, primary and bottom sample
# types and water and sediment matrices.
# Need to subset before doing water quality analyses.
# Keep Primary Water data only.
keep <- LL_RSLTS$SAMPLE_TYPE == 'PRIMARY' & LL_RSLTS$MATRIX == 'WATER'

# Merge subset of result data with site evaluation data
LL_WQ <- merge(as.data.frame(dsgn_sf)[, c("PK_RANDOM_SAMPLE_LOCATION",
                "REPORTING_UNIT", "EXCLUSION_CATEGORY","TNT", "wgt",
                "londd", "latdd", "xcoord", "ycoord",
                "NUTRIENT_WATERSHED_REGION", "DO_Conc")], LL_RSLTS[keep,],
                by.x = 'PK_RANDOM_SAMPLE_LOCATION',
                by.y = 'FK_RANDOM_SAMPLE_LOCATION')

# Check that merged data includes only PRIMARY WATER data
addmargins(table(LL_WQ$SAMPLE_TYPE, LL_WQ$MATRIX, useNA = 'ifany'))

# Create a second data subset for sediment analyses.
# Keep Primary Sediment data only.
keep2 <- LL_RSLTS$SAMPLE_TYPE == 'PRIMARY' & LL_RSLTS$MATRIX == 'SEDIMENT'

# Merge sediment result data with site evaluation data
LL_SED <- merge(as.data.frame(dsgn_sf)[, c("PK_RANDOM_SAMPLE_LOCATION",
                "REPORTING_UNIT", "EXCLUSION_CATEGORY","TNT", "wgt",
                "londd", "latdd", "xcoord", "ycoord",
                "NUTRIENT_WATERSHED_REGION",
                "DO_Conc")], LL_RSLTS[keep2,],
                by.x = 'PK_RANDOM_SAMPLE_LOCATION',
                by.y = 'FK_RANDOM_SAMPLE_LOCATION')
```

```
# Check that merged data includes only PRIMARY SEDIMENT data
addmargins(table(LL_SED$SAMPLE_TYPE, LL_SED$MATRIX, useNA = 'ifany'))
```

It is necessary to subset the results data file because the SAMPLE_TYPE column often contains three types of data: BLANK, PRIMARY, and BOTTOM. For lakes, the MATRIX column often contains two types of data: WATER and SEDIMENT.

Merging the site evaluations data and results data must happen before water quality analyses can be performed. This ensures that result data from sites that are not part of the target population, and result data that have been flagged as inappropriate for analysis due to quality assurance concerns are removed from the data set.

```
##### Example continuous water quality indicator population estimation
nr <- nrow(LL_WQ)
levels(LL_WQ$TNT)

# Data frame LL_WQ is the merged dsgn_sf and LL_RESULTS data.
# Add Combined category with name "All Basins" and added Basin category
# with reporting unit data.
data_cont_WQ <- data.frame(LL_WQ,
                           Combined = rep("All Basins", nr),
                           Basin = LL_WQ$REPORTING_UNIT)
# List all variables for continuous analysis
ContVars <- c('Water_Temperature','pH_Field',
              'Oxygen_Dissolved_Percent_Saturation','Oxygen_Dissolved_Field',
              'Specific_Conductance_Field','Escherichia_Coli_Quanti_Tray',
              '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','Fluoride_Total',
              'Aluminum_Total','Antimony_Total','Arsenic_Total','Barium_Total',
              'Beryllium_Total','Cadmium_Total','Chromium_Total','Copper_Total',
              'Iron_Total','Lead_Total','Manganese_Total','Molybdenum_total',
              'Nickel_Total','Selenium_Total','Silver_Total','Thallium_Total',
              'Zinc_Total')

# Split list of ContVars into two groups.
# 1. ContVars_LowVar = Variables with low variability, defined as all result
#    values from one or more Zones have the same value.
# 2. ContVars_NotLowVar = Variables without low variability, defined as result
#    values for each zone have more than one distinct value.
# Begin by Counting the number of unique values for each parameter in each
# zone, and in all zones combined.
ContVars_Count_Unique <-data.frame()
ZoneList <- unique(data_cont_WQ$Basin)
# loop through list of Zones
for (i in seq_along(ZoneList)){
   ZoneName <- ZoneList[i]
   data_cont_WQ_subset <- subset(data_cont_WQ, data_cont_WQ$Basin == ZoneName)
  # loop through list of variables for each Zone
  for (i in seq_along(ContVars)) {
     VarName <- ContVars[i]
     tempDataFrame1 <- data.frame(data_cont_WQ_subset[,c(VarName)])
     tempDataFrame <- subset(tempDataFrame1, tempDataFrame1[,1] != 'NA')
     tempCount <- length(unique(tempDataFrame[,1]))
     output_df <- data.frame(ZoneName, VarName, tempCount)
     names(output_df) <- c('ZoneName','VarName', 'Count_Unique_Values')
     ContVars_Count_Unique <- rbind(ContVars_Count_Unique,output_df)
  }
```

```
  data_cont_WQ_subset <- subset(data_cont_WQ,
                                data_cont_WQ$Combined == 'All Basins')
  # loop through list of variables for all Zones combined
  for (i in seq_along(ContVars)) {
    VarName <- ContVars[i]
    tempDataFrame1 <- data.frame(data_cont_WQ_subset[,c(VarName)])
    tempDataFrame <- subset(tempDataFrame1, tempDataFrame1[,1] != 'NA')
    tempCount <- length(unique(tempDataFrame[,1]))
    output_df <- data.frame('All Basins', VarName, tempCount)
    names(output_df) <- c('ZoneName','VarName', 'Count_Unique_Values')
    ContVars_Count_Unique <- rbind(ContVars_Count_Unique,output_df)
  }
}
# Create a list of variables that have low variability in one or more zones.
LowVar <- subset(ContVars_Count_Unique,
    (ContVars_Count_Unique$ZoneName != 'All Basins' &
    ContVars_Count_Unique$Count_Unique_Values == 1))
ContVars_LowVar <- unique(LowVar$VarName)
# Create another list with all variables not in the low variability list.
ContVars_NotLowVar <- setdiff(ContVars,ContVars_LowVar)
# Identify subset of analytes that have low vabiability for individual basins,
# but do not have low variability for the combined subpop. (Analytes where not
# all result values in combined subpop are the same.)
NoVar <- subset(ContVars_Count_Unique,
                (ContVars_Count_Unique$ZoneName == 'All Basins' &
                ContVars_Count_Unique$Count_Unique_Values == 1))
ContVars_NoVar <- unique(NoVar$VarName)
ContVars_Combined_NotLowVar <- setdiff(ContVars_LowVar, ContVars_NoVar)

# Run Continuous analysis for all analytes in ContVars_LowVar.
# Remove percentile estimate ('Pct') from list of statistics. Pct results are
# unable to be calculated for subpops where all result values are the same
# value.
Water_quality_Cont_LowVar <- cont_analysis(data_cont_WQ,
                                           vars = ContVars_LowVar,
                                           subpops = c('Combined','Basin'),
                                           siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                           weight = 'wgt',
                                           xcoord = 'xcoord',
                                           ycoord = 'ycoord',
                                           stratumID = 'Basin',
                                           vartype = 'Local',
                                           statistics = c('CDF','Mean','Total'),
                                           conf=95,
                                           popsize = list(Basin = framesize))
# Run Continuous analysis for all analytes in ContVars_NotLowVar.
# Statistics include both CDF and Pct estimates.
Water_quality_Cont_NotLowVar <- cont_analysis(data_cont_WQ,
                                           vars = ContVars_NotLowVar,
                                           subpops = c('Combined','Basin'),
                                           siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                           weight = 'wgt',
                                           xcoord = 'xcoord',
                                           ycoord = 'ycoord',
                                           stratumID = 'Basin',
                                           vartype = 'Local',
                                           conf=95,
                                           popsize = list(Basin = framesize))
# Run Continuous analysis for all analytes in ContVars_Combined_NotLowVar, for
# combined subpopulation only. Only calculate Pct statistic.
# The CDF, Mean, and Total statistics have already been calculated in the
# previous continuous analysis run (Water_quality_Cont_LowVar.)
```

```
Water_quality_Cont_Combined_NotLowVar <- cont_analysis(data_cont_WQ,
                                          vars = ContVars_Combined_NotLowVar,
                                          subpops = 'Combined',
                                          siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                          weight = 'wgt',
                                          xcoord = 'xcoord',
                                          ycoord = 'ycoord',
                                          stratumID = 'Basin',
                                          vartype = 'Local',
                                          statistics = 'Pct',
                                          conf=95,
                                          popsize = list(Basin = framesize))

# Merge Results from Water_quality_Cont_LowVar, Water_quality_Cont_NotLowVar,
# and Water_quality_Cont_Combined_NotLowVar.
Water_quality_Cont <- list()
Water_quality_Cont[["CDF"]] <- rbind(Water_quality_Cont_LowVar[["CDF"]],
                                     Water_quality_Cont_NotLowVar[["CDF"]])
Water_quality_Cont[["Mean"]] <- rbind(Water_quality_Cont_LowVar[["Mean"]],
                                     Water_quality_Cont_NotLowVar[["Mean"]])
Water_quality_Cont[["Total"]] <- rbind(Water_quality_Cont_LowVar[["Total"]],
                                     Water_quality_Cont_NotLowVar[["Total"]])
Water_quality_Cont[["Pct"]] <-
rbind(Water_quality_Cont_Combined_NotLowVar[["Pct"]],
      Water_quality_Cont_NotLowVar[["Pct"]])
# Add data columns to all Water_quality_Cont results data frames. These will be
# used when loading analysis results to GWIS tables for Status Network analysis
# results.
analysis_date <- as.character(Sys.Date())
Water_quality_Cont <- lapply(Water_quality_Cont, function(df)
                      cbind(df,
                            SAMPLE_YEAR = '2021-2023',
                            REPORTING_CYCLE = '15-17',
                            WATER_RESOURCE = 'LARGE LAKE',
                            ANALYSIS_DATE = analysis_date,
                            MATRIX = 'WATER'))

# Merge Pct and Mean Results into single data frame, for consistency with
# format of previous years' results.
Water_quality_Cont[["Mean"]] <- cbind(Water_quality_Cont[["Mean"]],
                                      Statistic="Mean")
Water_quality_Cont[["Pct"]] <- rbind(Water_quality_Cont[["Pct"]],
                                      Water_quality_Cont[["Mean"]])
```

The continuous analysis portion of the code creates percentile and CDF calculations and writes them to tables. The "cont_analysis" function is for use with the continuous variables (e.g., pH, nitrate, etc.). [Entering "help(cont_analysis)" in the R command window will display the function description]. CDFs can only be created using continuous data; count data such as bacteria can theoretically be used, however tied data may generate problems. This is set up similarly to categorical analysis, with sites listed in the "dsgn" dataframe, and arguments to the cont_analysis function to specify the , subpop, weight, stratumID, etc.

The function above creates CDF estimates and percentile estimates for a range of variables and subpopulations at the same time. The CDFs generated are estimated population CDFs. These are what the CDF would look like if all lakes in the listframe were sampled.

The percentile estimates cannot be calculated for subpopulations where all result values for a variable are the same. Therefore it is necessary to split the list of variables into groups, those with low variability (all result values for one or more subpopulations are the same) and those without low variability. The continuous analysis is then run in two batches, where only the CDF

estimates are calculated for the low variability variables, and both the CDF and percentile estimates are calculated for the remaining variables. The continuous analysis results from both groups are then merged.

```
##### Example categorical water quality indicator population estimation
# Set up threshold category variables for specified water quality analytes

# E_Coli category
LL_WQ$E_Coli_Category <- as.factor(cut(LL_WQ$Escherichia_Coli_Quanti_Tray,
breaks=c(0,410,10000000), include.lowest=TRUE))

# pH Category
LL_WQ$pH_Category <- as.factor(cut(LL_WQ$pH_Field, breaks=c(0,5.999,8.5,14),
include.lowest=TRUE))

# Fluoride class III water quality standard exceedances.
LL_WQ$Fluoride_Category <- as.factor(cut(LL_WQ$Fluoride_Total, breaks=c(0,10,100000),
include.lowest=TRUE))

#### Metals exceeding class III water quality standards.
# Antimony Category
LL_WQ$Antimony_Category <- as.factor(cut(LL_WQ$Antimony_Total,
breaks=c(0,4300,100000), include.lowest=TRUE))

# Arsenic Category
LL_WQ$Arsenic_Category <- as.factor(cut(LL_WQ$Arsenic_Total, breaks=c(0,50,100000),
include.lowest=TRUE))

# Beryllium Category
LL_WQ$Beryllium_Category <- as.factor(cut(LL_WQ$Beryllium_Total,
breaks=c(0,0.13,100000), include.lowest=TRUE))

# Iron Category
LL_WQ$Iron_Category <- as.factor(cut(LL_WQ$Iron_Total, breaks=c(0,1000,100000),
include.lowest=TRUE))

# Selenium Category
LL_WQ$Selenium_Category <- as.factor(cut(LL_WQ$Selenium_Total, breaks=c(0,5,100000),
include.lowest=TRUE))

# Silver Category
LL_WQ$Silver_Category <- as.factor(cut(LL_WQ$Silver_Total, breaks=c(0,0.07,100000),
include.lowest=TRUE))

# Thallium Category
LL_WQ$Thallium_Category <- as.factor(cut(LL_WQ$Thallium_Total,
breaks=c(0,6.3,100000), include.lowest=TRUE))

# categorical water quality estimates
# Data frame LL_WQ is the merged dsgn_sf and LL_RESULTS data.
# Add Combined category with name "All Basins" and add Basin category
# with reporting unit data.
data_cat_WQ <- data.frame(LL_WQ,
                          Combined = rep("All Basins", nr),
                          Basin = LL_WQ$REPORTING_UNIT)

# List of all variables for continuous analysis.
CatVars <- c('Ammonia_Category','Chlorophyll_Category','TN_Category','TP_Category',
             'DO_Category','NNCDO_Category','E_Coli_Category','pH_Category',
             'Fluoride_Category','Antimony_Category','Arsenic_Category',
             'Beryllium_Category','Cadmium_Category','Chromium_Category',
             'Copper_Category','Lead_Category','Iron_Category','Nickel_Category',
             'Selenium_Category','Silver_Category','Thallium_Category',
             'Zinc_Category')
```

```
# Run categorical analysis for all analytes in CatVars.
Water_Quality_Cat <- cat_analysis(data_cat_WQ,
                                  vars = CatVars,
                                  subpops = c('Combined','Basin'),
                                  siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                  weight = 'wgt',
                                  xcoord = 'xcoord',
                                  ycoord = 'ycoord',
                                  stratumID = 'Basin',
                                  vartype = 'Local',
                                  conf=95,
                                  popsize = list(Basin = framesize))

# Add data columns to Water_Quality_Cat results. These will be
# used when loading analysis results to GWIS tables for Status Network analysis
# results.
analysis_date <- as.character(Sys.Date())
Water_Quality_Cat <- cbind(Water_Quality_Cat,
                           SAMPLE_YEAR = '2021-2023',
                           REPORTING_CYCLE = '15-17',
                           WATER_RESOURCE = 'LL',
                           ANALYSIS_DATE = analysis_date,
                           MATRIX = 'WATER',
                           ANALYSIS_TYPE = ' TARGET POPULATION',
                           ESTIMATEU_UNITS = 'HECTARES')
```

The categorical analysis portion of the code creates estimates of the percent of the resource in each of the user-generated categories (i.e. meeting water quality threshold, not meeting water quality threshold) and writes them to a table. The cat_analysis function is used for categorical variables.[Entering "help(cat.analysis)" in the R command window will display the function description].

```
# Export the results
write.csv(Water_Quality_Cat, '2021-2023_LL_WQ_Cat.csv', row.names = FALSE)
write.csv(Water_quality_Cont$CDF,
          file = '2021-2023_LL_WQ_Cont_EstCDF.csv', row.names = FALSE)
write.csv(Water_quality_Cont$Pct,
          file = '2021-2023_LL_WQ_Cont_EstPCT.csv', row.names = FALSE)
write.csv(Water_quality_Cont$Total,
          file = '2021-2023_LL_WQ_Cont_EstTotal.csv', row.names = FALSE))
```

The program generates four output files which are saved in the folder designated as the current R workspace. These are comma delimited text files that can be opened in Excel.

```
#####  Set up threshold category columns for specified sediment analytes
# Create categories using Probable Effects Concentration (PEC) for each indicator
# Create categories for combined analysis of all indicators
LL_SED$AsPECcat <- ifelse(LL_SED$Arsenic_Sediments > 33, 1, 0)
LL_SED$AsPECcat[is.na(LL_SED$AsPECcat)] <- 0
LL_SED$CdPECcat <- ifelse(LL_SED$Cadmium_Sediments > 5, 1, 0)
LL_SED$CdPECcat[is.na(LL_SED$CdPECcat)] <- 0
LL_SED$CrPECcat <- ifelse(LL_SED$Chromium_Sediments > 110, 1, 0)
LL_SED$CrPECcat[is.na(LL_SED$CrPECcat)] <- 0
LL_SED$CuPECcat <- ifelse(LL_SED$Copper_Sediments > 150, 1, 0)
LL_SED$CuPECcat[is.na(LL_SED$CuPECcat)] <- 0
LL_SED$AgPECcat <- ifelse(LL_SED$Silver_Sediments > 2.2, 1, 0)
LL_SED$AgPECcat[is.na(LL_SED$AgPECcat)] <- 0
LL_SED$NiPECcat <- ifelse(LL_SED$Nickel_Sediments > 49, 1, 0)
LL_SED$NiPECcat[is.na(LL_SED$NiPECcat)] <- 0
LL_SED$PbPECcat <- ifelse(LL_SED$Lead_Sediments > 130, 1, 0)
LL_SED$PbPECcat[is.na(LL_SED$PbPECcat)] <- 0
```

```
LL_SED$HgPECcat <- ifelse(LL_SED$Mercury_Sediments > 1.1, 1, 0)
LL_SED$HgPECcat[is.na(LL_SED$HgPECcat)] <- 0
LL_SED$ZnPECcat <- ifelse(LL_SED$Zinc_Sediments > 460, 1, 0)
LL_SED$ZnPECcat[is.na(LL_SED$ZnPECcat)] <- 0

# Combined total PEC categories
LL_SED$NumExceedPECcat <- 0
LL_SED$NumExceedPECcat <- ifelse(LL_SED$AsPECcat == 0 | is.na(LL_SED$AsPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$CdPECcat == 0 | is.na(LL_SED$CdPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$CrPECcat == 0 | is.na(LL_SED$CrPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$CuPECcat == 0 | is.na(LL_SED$CuPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$AgPECcat == 0 | is.na(LL_SED$AgPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$NiPECcat == 0 | is.na(LL_SED$NiPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$PbPECcat == 0 | is.na(LL_SED$PbPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$HgPECcat == 0 | is.na(LL_SED$HgPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(LL_SED$ZnPECcat == 0 | is.na(LL_SED$ZnPECcat),
                          LL_SED$NumExceedPECcat, (LL_SED$NumExceedPECcat+1))
LL_SED$NumExceedPECcat <- ifelse(is.na(LL_SED$AsPECcat) & is.na(LL_SED$CdPECcat)
                               & is.na(LL_SED$CrPECcat) & is.na(LL_SED$CuPECcat)
                               & is.na(LL_SED$AgPECcat) & is.na(LL_SED$NiPECcat)
                               & is.na(LL_SED$PbPECcat) & is.na(LL_SED$HgPECcat)
                               & is.na(LL_SED$ZnPECcat), NA,
                               LL_SED$NumExceedPECcat)

# Sites that exceed at least one PEC threshold
LL_SED$Exceed1_PECcat <- LL_SED$NumExceedPECcat
LL_SED$Exceed1_PECcat[LL_SED$Exceed1_PECcat >= 1] <- 1
addmargins(table(NumExceedPECcat = LL_SED$NumExceedPECcat,
          Exceed1_PECcat = LL_SED$Exceed1_PECcat, useNA = 'ifany'))

##### Sediment Category Population Estimation
# Data frame LL_WQ is the merged dsgn_sf and LL_RESULTS data.
# Add Combined category with name "All Basins" and add Basin category
# with reporting unit data.
# Rename sediment category variables as needed.
data_cat_sed <- data.frame(PK_RANDOM_SAMPLE_LOCATION =
                           data_cont_sed$PK_RANDOM_SAMPLE_LOCATION,
                           Combined = data_cont_sed$Combined,
                           Basin = data_cont_sed$Basin,
                           wgt = data_cont_sed$wgt,
                           xcoord = data_cont_sed$xcoord,
                           ycoord = data_cont_sed$ycoord,
                           Num_Exceed_PEC_Category = LL_SED$NumExceedPECcat,
                           Exceed_1_PEC_Category = LL_SED$Exceed1_PECcat,
                           Arsenic_PEC_Category = LL_SED$AsPECcat,
                           Cadmium_PEC_Category = LL_SED$CdPECcat,
                           Chromium_PEC_Category = LL_SED$CrPECcat,
                           Copper_PEC_Category = LL_SED$CuPECcat,
                           Silver_PEC_Category = LL_SED$AgPECcat,
                           Nickel_PEC_Category = LL_SED$NiPECcat,
                           Lead_PEC_Category = LL_SED$PbPECcat,
                           Mercury_PEC_Category = LL_SED$HgPECcat,
                           Zinc_PEC_Category = LL_SED$ZnPECcat)
```

```
# List all variables for categorical analysis here.
CatVars <- c('Num_Exceed_PEC_Category',
             'Exceed_1_PEC_Category',
             'Arsenic_PEC_Category',
             'Cadmium_PEC_Category',
             'Chromium_PEC_Category',
             'Copper_PEC_Category',
             'Silver_PEC_Category',
             'Nickel_PEC_Category',
             'Lead_PEC_Category',
             'Mercury_PEC_Category',
             'Zinc_PEC_Category')

# Run categorical analysis for all analytes in CatVars.
Sediment_Quality_Cat <- cat_analysis(data_cat_sed,
                                vars = CatVars,
                                subpops = c('Combined','Basin'),
                                siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                weight = 'wgt',
                                xcoord = 'xcoord',
                                ycoord = 'ycoord',
                                stratumID = 'Basin',
                                vartype = 'Local',
                                conf=95,
                                popsize = list(Basin = framesize))

# Add data columns to Sediment_Quality_Cat results. These will be
# used when loading analysis results to GWIS tables for Status Network analysis
# results.
analysis_date <- as.character(Sys.Date())
Sediment_Quality_Cat <- cbind(Sediment_Quality_Cat,
                              SAMPLE_YEAR = '2021-2023',
                              REPORTING_CYCLE = '15-17',
                              WATER_RESOURCE = 'LL',
                              ANALYSIS_DATE = analysis_date,
                              MATRIX = 'SEDIMENT',
                              ANALYSIS_TYPE = ' TARGET POPULATION',
                              ESTIMATEU_UNITS = 'HECTARES')

##### Sediment continuous distribution estimation
# LL_SED is the merged dsgn_sf and LR_RESULTS data.
# Add Combined category with name "All Basins" and added Basin category
# with reporting unit data.
nr <- nrow(LL_SED)
data_cont_sed <- data.frame(LL_SED,
                            Combined = rep("All Basins", nr),
                            Basin = LL_SED$REPORTING_UNIT)

# Create data frame with only sediment analytes to be used for CDFs
ContVars_Sed <- c('Arsenic_Sediments','Cadmium_Sediments','Chromium_Sediments',
                  'Aluminum_Sediments','Antimony_Sediments','Beryllium_Sediments',
                  'Iron_Sediments','Manganese_Sediments','Molybdenum_Sediments',
                  'Selenium_Sediments','Copper_Sediments', 'Silver_Sediments',
                  'Nickel_Sediments','Lead_Sediments','Mercury_Sediments',
                  'Zinc_Sediments')

# Split list of ContVars_Sed into two groups.
# 1. ContVars_LowVar = Variables with low variability, defined as all result
#    values from one or more Zones have the same value.
# 2. ContVars_NotLowVar = Variables without low variability, defined as result
#    values for each zone have more than one distinct value.
# Begin by Counting the number of unique values for each parameter in each
# zone, and in all zones combined.
ContVars_Sed_Count_Unique <-data.frame()
ZoneList <- unique(data_cont_sed$Basin)
```

```
# loop through list of Zones
for (i in seq_along(ZoneList)){
  ZoneName <- ZoneList[i]
  data_cont_sed_subset <- subset(data_cont_sed, data_cont_sed$Basin == ZoneName)
  # loop through list of variables for each Zone
  for (i in seq_along(ContVars_Sed)) {
     VarName <- ContVars_Sed[i]
     tempDataFrame1 <- data.frame(data_cont_sed_subset[,c(VarName)])
     tempDataFrame <- subset(tempDataFrame1, tempDataFrame1[,1] != 'NA')
     tempCount <- length(unique(tempDataFrame[,1]))
     output_df <- data.frame(ZoneName, VarName, tempCount)
     names(output_df) <- c('ZoneName','VarName', 'Count_Unique_Values')
     ContVars_Sed_Count_Unique <- rbind(ContVars_Sed_Count_Unique,output_df)
   }
  data_cont_sed_subset <- subset(data_cont_sed, data_cont_sed$Combined == 'All
  Basins')
 # loop through list of variables for all Zones combined
 for (i in seq_along(ContVars_Sed)) {
     VarName <- ContVars_Sed[i]
     tempDataFrame1 <- data.frame(data_cont_sed_subset[,c(VarName)])
     tempDataFrame <- subset(tempDataFrame1, tempDataFrame1[,1] != 'NA')
     tempCount <- length(unique(tempDataFrame[,1]))
     output_df <- data.frame('All Basins', VarName, tempCount)
     names(output_df) <- c('ZoneName','VarName', 'Count_Unique_Values')
     ContVars_Sed_Count_Unique <- rbind(ContVars_Sed_Count_Unique,output_df)
   }
}
# Create a list of variables that have low variability in one or more zones.
LowVar <- subset(ContVars_Sed_Count_Unique,
          (ContVars_Sed_Count_Unique$ZoneName != 'All Basins' &
          ContVars_Sed_Count_Unique$Count_Unique_Values == 1))
ContVars_Sed_LowVar <- unique(LowVar$VarName)
# Create another list with all variables not in the low variability list.
ContVars_Sed_NotLowVar <- setdiff(ContVars_Sed,ContVars_Sed_LowVar)
# Identify subset of analytes that have low vabiability for individual basins,
# but do not have low variability for the combined subpop. (Analytes where not
# all result values in combined subpop are the same.)
NoVar <- subset(ContVars_Sed_Count_Unique,
          (ContVars_Sed_Count_Unique$ZoneName == 'All Basins' &
          ContVars_Sed_Count_Unique$Count_Unique_Values == 1))
ContVars_Sed_NoVar <- unique(NoVar$VarName)
ContVars_Sed_Combined_NotLowVar <- setdiff(ContVars_Sed_LowVar,
                                           ContVars_Sed_NoVar)

# Run Continuous analysis for all analytes in ContVars_Sed_LowVar.
# Remove percentile estimate ('Pct') from list of statistics. Pct results are
# unable to be calculated for subpops where all result values are the same
# value.Sediment_quality_Cont_LowVar <- cont_analysis(data_cont_sed,
                                        vars = ContVars_Sed_LowVar,
                                        subpops = c('Combined','Basin'),
                                        siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                        weight = 'wgt',
                                        xcoord = 'xcoord',
                                        ycoord = 'ycoord',
                                        stratumID = 'Basin',
                                        vartype = 'Local',
                                        statistics = c('CDF','Mean','Total'),
                                        conf=95,
                                        popsize = list(Basin = framesize))
# Run Continuous analysis for all analytes in ContVars_Sed_NotLowVar.
# Statistics include both CDF and Pct estimates.
```

```
Sediment_quality_Cont_NotLowVar <- cont_analysis(data_cont_sed,
                                        vars = ContVars_Sed_NotLowVar,
                                        subpops = c('Combined','Basin'),
                                        siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                        weight = 'wgt',
                                        xcoord = 'xcoord',
                                        ycoord = 'ycoord',
                                        stratumID = 'Basin',
                                        vartype = 'Local',
                                        conf=95,
                                        popsize = list(Basin = framesize))
# Run Continuous analysis for all analytes in ContVars_Sed_Combined_NotLowVar,
# for combined subpopulation only. Only calculate Pct statistic.
# The CDF, Mean, and Total statistics have already been calculated in the
# previous continuous analysis run (Sediment_quality_Cont_NotLowVar).
Sediment_quality_Cont_Combined_NotLowVar <- cont_analysis(data_cont_sed,
                                        vars = ContVars_Sed_Combined_NotLowVar,
                                        subpops = 'Combined',
                                        siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                        weight = 'wgt',
                                        xcoord = 'xcoord',
                                        ycoord = 'ycoord',
                                        stratumID = 'Basin',
                                        vartype = 'Local',
                                        statistics = 'Pct',
                                        conf=95,
                                        popsize = list(Basin = framesize))

# Merge Results from Sediment_quality_Cont_LowVar,
#  Sediment_quality_Cont_NotLowVar, and
#  Sediment_quality_Cont_Combined_NotLowVar.
Sediment_quality_Cont <- list()
Sediment_quality_Cont[["CDF"]] <- rbind(Sediment_quality_Cont_LowVar[["CDF"]],
                                Sediment_quality_Cont_NotLowVar[["CDF"]])
Sediment_quality_Cont[["Mean"]] <- rbind(Sediment_quality_Cont_LowVar[["Mean"]],
                                Sediment_quality_Cont_NotLowVar[["Mean"]])
Sediment_quality_Cont[["Total"]] <-
                                rbind(Sediment_quality_Cont_LowVar[["Total"]],
                                Sediment_quality_Cont_NotLowVar[["Total"]])
Sediment_quality_Cont[["Pct"]] <-
                         rbind(Sediment_quality_Cont_Combined_NotLowVar[["Pct"]],
                         Sediment_quality_Cont_NotLowVar[["Pct"]])

# Add data columns to all Sediment_quality_Cont results data frames. These will
# be used when loading analysis results to GWIS tables for Status Network
# analysis results.
analysis_date <- as.character(Sys.Date())
Sediment_quality_Cont <- lapply(Sediment_quality_Cont, function(df)
                               cbind(df,
                               SAMPLE_YEAR = '2021-2023',
                               REPORTING_CYCLE = '15-17',
                               WATER_RESOURCE = 'LARGE LAKE',
                               ANALYSIS_DATE = analysis_date,
                               MATRIX = 'SEDIMENT'))

# Merge Pct and Mean Results into single data frame, for consistency with
# format of previous years' results.
Sediment_quality_Cont[["Mean"]] <- cbind(Sediment_quality_Cont[["Mean"]],
                                        Statistic="Mean")
Sediment_quality_Cont[["Pct"]] <- rbind(Sediment_quality_Cont[["Pct"]],
                                        Sediment_quality_Cont[["Mean"]])

# Export the results
write.csv(Sediment_Quality_Cat, "2021-2023_LL_Sed_Cat.csv", row.names = FALSE)
write.csv(Sediment_quality_Cont$CDF, file = '2021-2023_LL_Sed_Cont_EstCDF.csv',
          row.names = FALSE)
```

```
write.csv(Sediment_quality_Cont$Pct, file = '2021-2023_LL_Sed_Cont_EstPCT.csv',
          row.names = FALSE)
write.csv(Sediment_quality_Cont$Total,
          file = '2021-2023_LL_Sed_Cont_EstTotal.csv', row.names = FALSE)
```

Continuous and categorical analyses can be performed for sediment analytes using the same procedures that were used for the water quality analyses. The program generates four output files which are saved in the folder designated as the current R workspace. These are comma delimited text files that can be opened in Excel.

Analysis for Indicator Thresholds Dependent on Geography and other Variables.

Some water quality indicators applicable to Florida's surface waters have complex thresholds, where the threshold value may vary based on geographic location or based on result values of other parameters. Examples of these indicators include total nitrogen (TN), total phosphorous (TP), dissolved oxygen (DO), chlorophyll (lakes only), total ammonia nitrogen, and several metals standards for class III surface waters (cadmium, chromium, copper, lead, nickel, zinc). To evaluate whether the resultant data from each sampled site meets thresholds that vary based on geographic location, the nutrient region and dissolved oxygen region names and associated threshold values (if applicable) must be included in the site evaluation data.

For flowing waters (rivers, streams, and canals), only geographic information is needed to determine the TN, TP, and DO thresholds. For lakes (large lakes and small lakes), only geographic information is needed to determine the DO thresholds. Additional information, specifically the result values for true color and alkalinity, is needed to determine the TN, TP, and chlorophyll thresholds that apply to lake samples. The result data for each sample is categorized according to its color and alkalinity result values.

```
# True color > 40 PCU assigned 0;
# Ture color <= 40 PCU assigned 1.
LL_WQ$Color_cat<- ifelse((LL_WQ$Color_true > 40) ,0,1)

# Alkalinity > 20 mg/L CaCO3 assigned 0;
# Alkalinity <= 20 mg/L CaCO3 assigned 1.
LL_WQ$Alkalinity_cat<- ifelse((LL_WQ$Alkalinity_Total_as_CaCO3 > 20) ,0,1)

# Combine color and alkalinity categories into single character string
# variable.
LL_WQ$Col_Alk_cat<-paste(LL_WQ$Color_cat, LL_WQ$Alkalinity_cat)
```

A conditional statement is then used to assign the appropriate TN, TP, DO, and chlorophyll thresholds to each site based on the color category, alkalinity category, and NNC Region (F.A.C.62-302.531). TN and TP have two applicable thresholds, based on additional criteria not currently being used in WMS reporting. For WMS reporting purposes, the less stringent ("maximum") threshold is currently being used and reported.

```
## Use new Col_Alk_cat character variable to assign thresholds for TN, TP, and
#  chlorophyll
LL_WQ$TN_Max<- ifelse(LL_WQ$Col_Alk_cat=="0 0",2.23,
               ifelse(LL_WQ$Col_Alk_cat=="0 1", 2.23,
               ifelse(LL_WQ$Col_Alk_cat== "1 0", 1.91,
               ifelse(LL_WQ$Col_Alk_cat=="1 1",0.93,
               NA))))
```

```
LL_WQ$TP_Max<- ifelse((LL_WQ$Color_cat==0 &
                  LL_WQ$NUTRIENT_WATERSHED_REGION=="WEST CENTRAL"),0.49,
                  ifelse(LL_WQ$Col_Alk_cat=="0 0",0.16,
                  ifelse(LL_WQ$Col_Alk_cat=="0 1",0.16,
                  ifelse(LL_WQ$Col_Alk_cat=="1 0", 0.09,
                  ifelse(LL_WQ$Col_Alk_cat=="1 1",0.03,
                  NA)))))
 LL_WQ$Chlorophyll_conc<- ifelse(LL_WQ$Col_Alk_cat=="0 0",20,
                          ifelse(LL_WQ$Col_Alk_cat=="0 1", 20,
                          ifelse(LL_WQ$Col_Alk_cat== "1 0", 20,
                          ifelse(LL_WQ$Col_Alk_cat=="1 1", 6,
                           NA))))
```

After TN, TP, DO, and chlorophyll thresholds have been assigned, the result values for each sample are compared to their respective thresholds. To determine the combined result, it is determined whether all three values (TN, TP, and DO) are meeting their thresholds.

```
# Pass = 1, Fail = 0
# TN
# Calculate TN
LL_WQ$TN<-(LL_WQ$Kjeldahl_Nitrogen_Total_as_N+LL_WQ$NitrateNitrite_Total_as_N
# Pass if TN threshold value is >= TN result value.
# Fail if TN threshold value is < TN result value. LL_WQ$TN_cat<-
ifelse((LL_WQ$TN_Max >= LL_WQ$TN),1,0)
# TP
# Pass if TP threshold value is >= TP result value.
# Fail if TP threshold value is < TP result value.
LL_WQ$TP_cat<-ifelse((LL_WQ$TP_Max >= LL_WQ$Phosphorus_Total_as_P),1,0)
# DO
# Fail if DO threshold value is > DO result value.
# Pass if DO threshold value is =< DO result LL_WQ$DO_cat<-ifelse((LL_WQ$DO_Conc
> LL_WQ$Oxygen_Dissolved_Percent_Saturation),0,1)
# Chlorophyll
LL_WQ$Chlorophyll_cat<-ifelse((LL_WQ$Chlorophyll_conc >=
                               LL_WQ$Chlorophyll_A_Monochromatic),1,0)
# Total (TN, TP, DO)
# Note for combined NNc and DO, Pass = 3, Fail < 3
LL_WQ$NNCDO_cat<-(LL_WQ$TN_cat+LL_WQ$TP_cat+LL_WQ$DO_cat)
```

Total Ammonia Nitrogen (TAN) thresholds are calculated using the single sample criteria equation from the DEP TAN calculator (https://floridadep.gov/dear/water-quality-standards-program/documents/total-ammonia-nitrogen-calculator%C2%A0). Thresholds for cadmium, chromium, copper, lead, nickel, and zinc are calculated using the equations for hardness-based metals criteria from the DEP metals criteria calculator (https://floridadep.gov/dear/water-quality-standards/content/surface-water-quality-support-documents). The result values for pH and water temperature are needed to determine the TAN thresholds, and the result values for total hardness (mg/L as $CaCO_3$) are needed to determine the metals thresholds. A conditional statement is used to assign the appropriate thresholds to each site. After the threshold has been assigned, the resultant values for each sample are compared to their respective threshold.

```
# TAN_pH = pH used in TAN threshold calc.
# If measured pH < 6.5, value used is 6.5.
# If measured pH > 9.0, value used is 9.0.
LL_WQ$TAN_pH <- ifelse(LL_WQ$pH_Field < 6.5, 6.5,
                ifelse(LL_WQ$pH_Field > 9.0, 9.0,LL_WQ$pH_Field))
# TAN_temp = water temperature used in TAN threshold calc.
# If measured temp < 7 degrees C, value used is 7.
LL_WQ$TAN_temp <- ifelse(LL_WQ$Water_Temperature < 7, 7,
LL_WQ$Water_Temperature)
```

```
# calculate single sample Total Ammonia Criteria using TAN_pH and
# LL_WQ$TAN_temp
LL_WQ$TAN_Crit_SingleSamp <- ifelse(is.na(LL_WQ$TAN_pH), NA,
    ifelse(is.na(LL_WQ$TAN_temp), NA,
    (2.5*(0.8876*((0.0278/(1+10^(7.688-LL_WQ$TAN_pH)))+
    (1.1994/(1+10^(LL_WQ$TAN_pH-7.688))))* 2.126*10^(0.028*
    (20-(LL_WQ$TAN_temp)))))))
# Round result to two decimal places, for consistency with thresholds generated
# by TAN calculator spreadsheet
LL_WQ$TAN_Crit_SingleSamp <- round(LL_WQ$TAN_Crit_SingleSamp, digits=2)

# Pass=1 AND Fail=0
# Pass if TAN threshold value is >= TAN result value.
# Fail if TAN threshold value is < TAN result value.
LL_WQ$TAN_Cat<-ifelse((LL_WQ$TAN_Crit_SingleSamp >=
LL_WQ$Ammonia_Total_as_N),1,0)

## Metals_Hardness = Harness used in metals threshold calculation. If measured
#   Hardness_calculated_as_CACO3 < 25, value used is 25. If measured
#   Hardness_calculated_as_CACO3 > 400, value used is 400.
LL_WQ$Metals_Hardness <- ifelse(LL_WQ$Hardness_calculated_as_CACO3 < 25, 25,
    ifelse(LL_WQ$Hardness_calculated_as_CACO3 > 400, 400,
    LL_WQ$Hardness_calculated_as_CACO3))

## For cadmium, chromium, copper, lead, nickel, and zinc categories:
#  Pass=1 AND Fail=0
#  Pass if threshold value is >= result value.
#  Fail if threshold value is < result value.

## Calculate single sample Cadmium_Total Criteria using Metals_Hardness
LL_WQ$TotCadmiumCrit_SingleSamp <- ifelse(is.na(LL_WQ$Metals_Hardness), NA,
    (exp((0.7409*(log(LL_WQ$Metals_Hardness)))-4.719)))
# Round result to four decimal places
LL_WQ$TotCadmiumCrit_SingleSamp <- round(LL_WQ$TotCadmiumCrit_SingleSamp,
digits=4)
# Assign values for Pass (1) and Fail (0).
LL_WQ$Cadmium_Category<-ifelse((LL_WQ$TotCadmiumCrit_SingleSamp >=
LL_WQ$Cadmium_Total),1,0)

## Calculate single sample Chromium_Total Criteria using Metals_Hardness
LL_WQ$TotChromiumCrit_SingleSamp <- ifelse(is.na(LL_WQ$Metals_Hardness), NA,
     (exp((0.819*(log(LL_WQ$Metals_Hardness)))+0.6848)))
# Round result to four decimal places
LL_WQ$TotChromiumCrit_SingleSamp <- round(LL_WQ$TotChromiumCrit_SingleSamp,
digits=4)
# Assign values for Pass (1) and Fail (0).
LL_WQ$Chromium_Category<-ifelse((LL_WQ$TotChromiumCrit_SingleSamp >=
LL_WQ$Chromium_Total),1,0)

## Calculate single sample Copper_Total Criteria using Metals_Hardness
LL_WQ$TotCopperCrit_SingleSamp <- ifelse(is.na(LL_WQ$Metals_Hardness), NA,
     (exp((0.8545*(log(LL_WQ$Metals_Hardness)))-1.702)))
# Round result to four decimal places
LL_WQ$TotCopperCrit_SingleSamp <- round(LL_WQ$TotCopperCrit_SingleSamp,
digits=4)
# Assign values for Pass (1) and Fail (0).
LL_WQ$Copper_Category<-ifelse((LL_WQ$TotCopperCrit_SingleSamp >=
LL_WQ$Copper_Total),1,0)
```

```
## Calculate single sample Lead_Total Criteria using Metals_Hardness
LL_WQ$TotLeadCrit_SingleSamp <- ifelse(is.na(LL_WQ$Metals_Hardness), NA,
     (exp((1.273*(log(LL_WQ$Metals_Hardness)))-4.705)))
# Round result to four decimal places
LL_WQ$TotLeadCrit_SingleSamp <- round(LL_WQ$TotLeadCrit_SingleSamp, digits=4)
# Assign values for Pass (1) and Fail (0).
LL_WQ$Lead_Category<-ifelse((LL_WQ$TotLeadCrit_SingleSamp >=
LL_WQ$Lead_Total),1,0)

## Calculate single sample Nickel_Total Criteria using Metals_Hardness
LL_WQ$TotNickelCrit_SingleSamp <- ifelse(is.na(LL_WQ$Metals_Hardness), NA,
     (exp((0.846*(log(LL_WQ$Metals_Hardness)))+0.0584)))
# Round result to four decimal places
LL_WQ$TotNickelCrit_SingleSamp <- round(LL_WQ$TotNickelCrit_SingleSamp,
digits=4)
# Assign values for Pass (1) and Fail (0).
LL_WQ$Nickel_Category<-ifelse((LL_WQ$TotNickelCrit_SingleSamp >=
LL_WQ$Nickel_Total),1,0)

## Calculate single sample Zinc_Total Criteria using Metals_Hardness
LL_WQ$TotZincCrit_SingleSamp <- ifelse(is.na(LL_WQ$Metals_Hardness), NA,
     (exp((0.8473*(log(LL_WQ$Metals_Hardness)))+0.884)))
# Round result to four decimal places
LL_WQ$TotZincCrit_SingleSamp <- round(LL_WQ$TotZincCrit_SingleSamp, digits=4)
# Assign values for Pass (1) and Fail (0).
LL_WQ$Zinc_Category<-ifelse((LL_WQ$TotZincCrit_SingleSamp >=
LL_WQ$Zinc_Total),1,0)
```

The percent of the resource meeting the thresholds for TN, TP, DO, chlorophyll, TAN, chromium, copper, lead, nickel, and zinc are then estimated using the cat_analysis function, as described in the previous subsection of this document. For the combined total, the percentage of samples with TN, TP, and DO values meeting their three respective thresholds is reported.

References

Copeland, R., Upchurch, S., Summers, K., Janicki, P., Hansard, P., Paulic, P., Maddox, G., Silvanima, J., and Craig, P.. 1999. Overview of the Florida Department of Environmental Protection's Integrated Water Resource Monitoring Efforts and the Design Plan of the Status Network: Florida Department of Environmental Protection, Ambient Monitoring Section.

Florida Department of Environmental Protection. 2013. Holding Time Study for Water Quality Assessments. Tallahassee, FL: Bureau of Laboratories, Biology Section. 2013.

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.

Appendix A – Example Streams Code

```
# File: 2021-2023 SS Script.R
# Purpose: Analysis of Florida Stream data generated by the Status
#  Monitoring Program. Created by Jay Silvanima using code developed
#  by Tony Olsen, Jay Silvanima, Chris Sedlacek, Stephanie Sunderman-Barnes,
#  and Liz Miller
# Code developed using R version 4.1.3 (2022-03-10), spsurvey version
#   5.4.1, and FDEPgetdata version 1.11.

# Check working directory location.
getwd()

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

# Run function of FDEPgetdata package which pulls exclusion data.
# Insert variable name between parentheses in function call below.
#
# Insert the string of characters which apply to the names for all stream projects.
# For instance for 2018-2020 small stream projects enter "'SS18','SS19','SS20'".
# This will pull the exclusion data for projects Z1SS1805, Z2SS1805, Z3SS1805,
# Z4SS1805, Z5SS1805, Z6SS1805, Z1SS1905, Z2S1905, Z3SS1905, Z4SS1905, Z5SS1905,
# Z6SS1905, Z1SS2005, Z2SS2005, Z3SS2005, Z4S2005, Z5SS2005, and Z6SS2005
# from the FDEP Oracle GWIS database table SITE_EVALUATIONs.
#
# Be sure to enclose character string in double and then single quotes.
# Example "'SS18','SS19','SS20'"
FDEPgetdata::getdata_fw_exclusions("'SS21','SS22','SS23'")

# Function getdata_fw_exclusions creates a dataframe named 'Exclusions' from the
# information provided. Total nitrogen (F.A.C. 62-302.531), total phosphorus (F.A.C. 62-302.531),
# and dissolved oxygen (F.A.C. 62-302.533) criteria are added for each record
# based on the corresponding nutrient watershed region and bioregion.

# Determine if any of the sites fall on stream segments which are no
# longer included in the target population. Use ArcGIS Pro python script geoprocessing tool to create a
# table (.csv file) containing the sites that should be removed from this analysis because they no
# longer are in the target population. Import this file into this R project.
SS_sitesNOTwithin50meters<- read.csv('SS_sitesNOTwithin50meters.csv')
View(SS_sitesNOTwithin50meters)
```

```
# Create a new data frame, named SiteEvaluations, by taking all sites from the Exclusions data frame that do
# not match sites in the site_removals data frame.
SiteEvaluations <- sqldf('select * from Exclusions
                         where PK_RANDOM_SAMPLE_LOCATION
                         not in (select PK_RANDOM_SAMPLE_LOCATION
                                SS_sitesNOTwithin50meters)')
View(SiteEvaluations)

# Create new data frame from the one just created.
SS.SITES<-SiteEvaluations
names(SS.SITES)

# Convert to Decimal degrees
deg <- floor(SS.SITES$RANDOM_LATITUDE/10000)
min <- floor((SS.SITES$RANDOM_LATITUDE - deg*10000)/100)
sec <- SS.SITES$RANDOM_LATITUDE - deg*10000 - min*100
SS.SITES$latdd <- deg + min/60 + sec/3600
deg <- floor(SS.SITES$RANDOM_LONGITUDE/10000)
min <- floor((SS.SITES$RANDOM_LONGITUDE - deg*10000)/100)
sec <- SS.SITES$RANDOM_LONGITUDE - deg*10000 - min*100
SS.SITES$londd <- deg + min/60 + sec/3600

# Change londd to negative for correct use in sf.
SS.SITES$londd <- -SS.SITES$londd

# Create sf object and transform to Albers projection for analysis. 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_SS <- st_as_sf(SS.SITES, coords = c("londd", "latdd"), remove = FALSE, crs = 4269)
dsgn_sf <- st_transform(dsgn_SS, crs = 3087)

# keep xy coords as variables
tmp <- st_coordinates(dsgn_sf)
dsgn_sf$xcoord <- tmp[, "X"]
dsgn_sf$ycoord <- tmp[, "Y"]

# Inspect the site location data by plotting them on a map.
# Create simple features objects from shapefiles of polygon features representing the 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
```

```
# Use sf to plot the Zone polygons and sites that were evaluated.
jpeg('2021_2023_SS_Evaluated_Sites.jpg', units = 'in', width = 7, height = 7, res = 300)
     plot(st_geometry(wms_c3_reporting_units), main= '2021-2023 Steams 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()

# Site Evaluation
# 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'))

# create sampled and target (T) / nontarget (NT) variables
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
# Note need frame size here found in design doc for stream site selections
# Copy "2023 Stream Framesize.csv" from
# Z:\Status Sample Survey Designs\2023\Small Streams.
# Save CSV file to R project workspace, then load framesize data into R project.
framesize.df <- read.csv('2023 Stream Framesize.csv')

# Reduce framesize to remove total row. Change Zone name values to all capital letters.
framesize.df <- framesize.df[framesize.df$Zones !='Sum',]
framesize.df$Zones <- toupper(framesize.df$Zones)

# Convert framesize data frame to named vector
framesize <- as.vector(framesize.df$length_km)
names(framesize) <- framesize.df[,'Zones']

# View framesize and paste results in comments here.
print(framesize)
#  ZONE 1  ZONE 2  ZONE 3  ZONE 4  ZONE 5  ZONE 6
# 12320.7  2170.8  4457.2  4154.6   888.7   177.8
sum(framesize)
#  Framesize in kilometers = 24169.8 for entire data frame.
```

```
# 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$REPORTING_UNIT, framesize=framesize)
# check sum of weights for each reporting unit/basin
addmargins(tapply(dsgn_sf$wgt, dsgn_sf$REPORTING_UNIT, sum))

# This gives the weights for the stream design as implemented in 2023.
# It must include all evaluated sites as some sites are not in the target population.

# Estimate Extent Stream Area.
# Since the sample frame includes portions of stream object line segments that do not meet the
# definition of a stream, the site evaluation information is used to estimate the stream kilometers in
# the target population for entire state and for each of the reporting units/basins.
dsgn_sf$Combined = rep("All Basins", nr)

dsgn <- data.frame(PK_RANDOM_SAMPLE_LOCATION = dsgn_sf$PK_RANDOM_SAMPLE_LOCATION,
                   wgt = dsgn_sf$wgt,
                   xcoord = dsgn_sf$xcoord,
                   ycoord = dsgn_sf$ycoord,
                   Basin = dsgn_sf$REPORTING_UNIT,
                   Combined = dsgn_sf$Combined,
                   TNTStatus = dsgn_sf$TNT,
                   EXCLUSION.CATEGORY = dsgn_sf$EXCLUSION_CATEGORY)

ExtentEst <- cat_analysis(dsgn,
                          vars = c('TNTStatus','EXCLUSION.CATEGORY'),
                          subpops = c('Combined','Basin'),
                          siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                          weight = 'wgt',
                          xcoord = 'xcoord',
                          ycoord = 'ycoord',
                          stratumID = 'Basin',
                          vartype = "Local",
                          conf=95)

# Add data columns to ExtentEst results. These will be used when loading analysis results to GWIS tables
# for Status Network analysis results.
analysis_date <- as.character(Sys.Date())
ExtentEst <- cbind(ExtentEst,
                   SAMPLE_YEAR = '2021-2023',
                   REPORTING_CYCLE = '15-17',
                   WATER_RESOURCE = 'SS',
                   ANALYSIS_DATE = analysis_date,
                   MATRIX = 'WATER',
                   ESTIMATEU_UNITS = 'KILOMETERS')

# Export results
write.csv(ExtentEst,file = '2021_2023_SS_ExtentEst.csv', row.names = FALSE)
```

```
#  To estimate the percent of the target population
#  that could be sampled, requires that the analysis be restricted to just sites
#  in the target population, i.e., TNT = "T"
dsgn <- subset(dsgn, dsgn$TNT == "T")

ExtentEst_Target <- cat_analysis(dsgn,
                                 vars = c('TNTStatus','EXCLUSION.CATEGORY'),
                                 subpops = c('Combined','Basin'),
                                 siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                 weight = 'wgt',
                                 xcoord = 'xcoord',
                                 ycoord = 'ycoord',
                                 stratumID = 'Basin',
                                 vartype = "Local",
                                 conf=95)
# Export results
write.csv(ExtentEst_Target, file = '2021_2023_SS_ExtentEst_Target.csv')

################################################################################
# Water Quality Data Analysis

# 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 "'SS18','SS19','SS20'" for the variable will
# produce a data frame for FDEP Status Streams sampled 2018 - 2020.
# Be sure to enclose in double and single quotes.
FDEPgetdata::getdata_results("'SS21','SS22','SS23'")

# Function getdata_results creates the table 'Results'.
# Examine the results data frame. If more than two columns are present for each parameter, the data set
# includes samples with multiple results for at least one parameter. Need to locate the affected
# samples and investigate further. Type c(" in the R Studio search bar to search the results data frame
# for the affected samples.
#
# Warnings are generated regarding using type.convert.default. Examine the data to ensure
# that result values are data type number. Redefine data types if needed, and
# proceed if data types are correct.

# Create new data frame from the one just created.
SS_RSLTS<-Results
names(SS_RSLTS)

# Determine sample types and matrices present in results data.
addmargins(table(SS_RSLTS$SAMPLE_TYPE, SS_RSLTS$MATRIX, useNA = 'ifany'))
```

```
# Note that BLANK, BOTTOM and PRIMARY sample types are present. Water is the only matrix present.
# Only want to use PRIMARY results for population estimation.
# Results data must be merged with design information. This merge will remove any result data
# marked as inappropriate for Status Network analysis (i.e. where
# pk_random_sample_location contains "B").
keep <- SS_RSLTS$SAMPLE_TYPE == 'PRIMARY' & SS_RSLTS$MATRIX == 'WATER'

# merge with exclusion file
SS_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")], SS_RSLTS[keep,],
                                    by.x = 'PK_RANDOM_SAMPLE_LOCATION',
                                    by.y = 'FK_RANDOM_SAMPLE_LOCATION')

# Check that only PRIMARY sample type and WATER matrix data are present
addmargins(table(SS_WQ$SAMPLE_TYPE, SS_WQ$MATRIX, useNA = 'ifany'))

################################################################################################
##### Categorical Data Variable Setup

# Total Ammonia Nitrogen (TAN)
# Calculator for total ammonia nitrogen (TAN) single sample criteria.
# Created using TAN calculator spreadsheet as a guide
# (accessed 1/3/2020, https://floridadep.gov/dear/water-quality-standards-program/documents/total-
# ammonia-nitrogen-calculator%C2%A0).

# TAN_pH = pH used in TAN threshold calc.
# If measured pH < 6.5, value used is 6.5.
# If meas. pH > 9.0, value used is 9.0.
SS_WQ$TAN_pH <- ifelse(SS_WQ$pH_Field < 6.5, 6.5,
                ifelse(SS_WQ$pH_Field > 9.0, 9.0,SS_WQ$pH_Field))

# TAN_temp = water temperature used in TAN threshold calc.
# If measured temp < 7 degrees C, value used is 7.
SS_WQ$TAN_temp <- ifelse(SS_WQ$Water_Temperature < 7, 7, SS_WQ$Water_Temperature )

# calculate single sample TAN criteria using TAN_pH and TAN_temp
SS_WQ$TAN_Crit_SingleSamp <- ifelse(is.na(SS_WQ$TAN_pH), NA,
                        ifelse(is.na(SS_WQ$TAN_temp), NA,
                        (2.5*(0.8876*((0.0278/(1+10^(7.688-SS_WQ$TAN_pH)))+(1.1994/(1+10^(SS_WQ$TAN_pH-
                        7.688))))*2.126*10^(0.028*(20-(SS_WQ$TAN_temp)))))))

# Round result to two decimal places, for consistency with thresholds generated using TAN calculator
# spreadsheet
SS_WQ$TAN_Crit_SingleSamp <- round(SS_WQ$TAN_Crit_SingleSamp, digits=2)
```

```
# Compare TAN thresholds to TAN result value for each sample
# Pass=1 AND Fail=0
SS_WQ$TAN_Cat<-ifelse((SS_WQ$TAN_Crit_SingleSamp >= SS_WQ$Ammonia_Total_as_N),1,0)

#####  Set up threshold category columns for other analytes
# E_Coli category
SS_WQ$E_Coli_Category <- as.factor(cut(SS_WQ$Escherichia_Coli_Quanti_Tray, breaks=c(0,410,10000000),
                                  include.lowest=TRUE))

# Old Dissolved Oxygen Category
SS_WQ$DO_cat_old <- as.factor(cut(SS_WQ$Oxygen_Dissolved_Field, breaks=c(0,4.999,1000000),
                                  include.lowest=TRUE))

# pH Category
SS_WQ$pH_Category <- as.factor(cut(SS_WQ$pH_Field, breaks=c(0,5.999,8.5,14), include.lowest=TRUE))

# Chlorophyll Category
SS_WQ$Chlorophyll_Category <- as.factor(cut(SS_WQ$Chlorophyll_A_Monochromatic, breaks=c(0,20,10000),
                                  include.lowest=TRUE))

# Fluoride class III water quality standard exceedances.
SS_WQ$Fluoride_Category <- as.factor(cut(SS_WQ$Fluoride_Total, breaks=c(0,10,100000),
                                  include.lowest=TRUE))

##### Numeric Nutrient and DO Categories
# Note Pass=1 AND Fail=0
# Total NNC category = sum of category results for TN, TP, and DO.
# If Tot_cat = 3, sample passed criteria for these 3 parameters.

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

# TN_NNC, TP_NNC, and DO_Conc are thresholds included in site evaluations data, based on nutrient
# watershed regions and bioregions. DO_Conc is thre
SS_WQ$TN_cat<-ifelse((SS_WQ$TN_NNC >= SS_WQ$TN),1,0)
SS_WQ$TP_cat<-ifelse((SS_WQ$TP_NNC >= SS_WQ$Phosphorus_Total_as_P),1,0)
SS_WQ$DO_cat<-ifelse((SS_WQ$Oxygen_Dissolved_Percent_Saturation >= SS_WQ$DO_Conc ),1,0)
SS_WQ$Tot_cat<-(SS_WQ$TN_cat+SS_WQ$TP_cat+SS_WQ$DO_cat)

#### Categories for metals exceeding class III water quality standards.

# Antimony Category
SS_WQ$Antimony_Category <- as.factor(cut(SS_WQ$Antimony_Total, breaks=c(0,4300,100000),
include.lowest=TRUE))

# Arsenic Category
SS_WQ$Arsenic_Category <- as.factor(cut(SS_WQ$Arsenic_Total, breaks=c(0,50,100000),
include.lowest=TRUE))
```

```
# Beryllium Category
SS_WQ$Beryllium_Category <- as.factor(cut(SS_WQ$Beryllium_Total, breaks=c(0,0.13,100000),
include.lowest=TRUE))

# Iron Category
SS_WQ$Iron_Category <- as.factor(cut(SS_WQ$Iron_Total, breaks=c(0,1000,100000), include.lowest=TRUE))

# Selenium Category
SS_WQ$Selenium_Category <- as.factor(cut(SS_WQ$Selenium_Total, breaks=c(0,5,100000),
include.lowest=TRUE))

# Silver Category
SS_WQ$Silver_Category <- as.factor(cut(SS_WQ$Silver_Total, breaks=c(0,0.07,100000),
include.lowest=TRUE))

# Thallium Category
SS_WQ$Thallium_Category <- as.factor(cut(SS_WQ$Thallium_Total, breaks=c(0,6.3,100000),
include.lowest=TRUE))

### Metals thresholds based on hardness (cadmium, chromium, copper, lead, nickel, zinc) are calculating
#   using the calculator spreadsheet as a guide (accessed 10/13/2023,
#   https://floridadep.gov/sites/default/files/MetalsCriteriaCalculator.xlsm)

##  Metals_Hardness = Harness used in metals threshold calculation.
#   If measured hardness_calculated_as_CACO3 < 25, value used is 25.
#   If measured hardness_calculated_as_CACO3 > 400, value used is 400.
                         SS_WQ$Metals_Hardness <- ifelse(SS_WQ$Hardness_calculated_as_CACO3 < 25, 25,
                         ifelse(SS_WQ$Hardness_calculated_as_CACO3 > 400, 400,
                         SS_WQ$Hardness_calculated_as_CACO3))

##  Calculate single sample Cadmium_Total Criteria using Metals_Hardness
SS_WQ$TotCadmiumCrit_SingleSamp <- ifelse(is.na(SS_WQ$Metals_Hardness), NA,
      (exp((0.7409*(log(SS_WQ$Metals_Hardness)))-4.719)))
# Round result to four decimal places
SS_WQ$TotCadmiumCrit_SingleSamp <- round(SS_WQ$TotCadmiumCrit_SingleSamp, digits=4)
#  Assign values for Pass (1) and Fail (0).
SS_WQ$Cadmium_Category<-ifelse((SS_WQ$TotCadmiumCrit_SingleSamp >= SS_WQ$Cadmium_Total),1,0)

##  Calculate single sample Chromium_Total Criteria using Metals_Hardness
SS_WQ$TotChromiumCrit_SingleSamp <- ifelse(is.na(SS_WQ$Metals_Hardness), NA,
      (exp((0.819*(log(SS_WQ$Metals_Hardness)))+0.6848)))
# Round result to four decimal places
SS_WQ$TotChromiumCrit_SingleSamp <- round(SS_WQ$TotChromiumCrit_SingleSamp, digits=4)
# Assign values for Pass (1) and Fail (0).
SS_WQ$Chromium_Category<-ifelse((SS_WQ$TotChromiumCrit_SingleSamp >= SS_WQ$Chromium_Total),1,0)
```

```
## Calculate single sample Copper_Total Criteria using Metals_Hardness
SS_WQ$TotCopperCrit_SingleSamp <- ifelse(is.na(SS_WQ$Metals_Hardness), NA,
       (exp((0.8545*(log(SS_WQ$Metals_Hardness)))-1.702)))
# Round result to four decimal places
SS_WQ$TotCopperCrit_SingleSamp <- round(SS_WQ$TotCopperCrit_SingleSamp, digits=4)
# Assign values for Pass (1) and Fail (0).
SS_WQ$Copper_Category<-ifelse((SS_WQ$TotCopperCrit_SingleSamp >= SS_WQ$Copper_Total),1,0)

## Calculate single sample Lead_Total Criteria using Metals_Hardness
SS_WQ$TotLeadCrit_SingleSamp <- ifelse(is.na(SS_WQ$Metals_Hardness), NA,
       (exp((1.273*(log(SS_WQ$Metals_Hardness)))-4.705)))
# Round result to four decimal places
SS_WQ$TotLeadCrit_SingleSamp <- round(SS_WQ$TotLeadCrit_SingleSamp, digits=4)
# Assign values for Pass (1) and Fail (0).
SS_WQ$Lead_Category<-ifelse((SS_WQ$TotLeadCrit_SingleSamp >= SS_WQ$Lead_Total),1,0)

## Calculate single sample Nickel_Total Criteria using Metals_Hardness
SS_WQ$TotNickelCrit_SingleSamp <- ifelse(is.na(SS_WQ$Metals_Hardness), NA,
       (exp((0.846*(log(SS_WQ$Metals_Hardness)))+0.0584)))
# Round result to four decimal places
SS_WQ$TotNickelCrit_SingleSamp <- round(SS_WQ$TotNickelCrit_SingleSamp, digits=4)
# Assign values for Pass (1) and Fail (0).
SS_WQ$Nickel_Category<-ifelse((SS_WQ$TotNickelCrit_SingleSamp >= SS_WQ$Nickel_Total),1,0)

## Calculate single sample Zinc_Total Criteria using Metals_Hardness
SS_WQ$TotZincCrit_SingleSamp <- ifelse(is.na(SS_WQ$Metals_Hardness), NA,
       (exp((0.8473*(log(SS_WQ$Metals_Hardness)))+0.884)))
# Round result to four decimal places
SS_WQ$TotZincCrit_SingleSamp <- round(SS_WQ$TotZincCrit_SingleSamp, digits=4)
# Assign values for Pass (1) and Fail (0).
SS_WQ$Zinc_Category<-ifelse((SS_WQ$TotZincCrit_SingleSamp >= SS_WQ$Zinc_Total),1,0)

## Export dataframe SS_WQ to .csv file format
write.csv(SS_WQ, "SS_WQ.csv", row.names = FALSE)

#########################################################################
# Continuous and categorical water quality indicator population estimation

#### continuous WQ estimates
# SS_WQ is the merged dsgn_sf and SS_RESULTS data.
# Add Combined category with name "All Basins" and added Basin category with reporting unit data.
nr <- nrow(SS_WQ)
levels(SS_WQ$TNT)
data_cont_WQ <- data.frame(SS_WQ,
                           Combined = rep("All Basins", nr),
                           Basin = SS_WQ$REPORTING_UNIT)
```

```
# List all variables for continuous analysis here.
ContVars <- c('Water_Temperature','pH_Field','Oxygen_Dissolved_Percent_Saturation',
              'Oxygen_Dissolved_Field','Specific_Conductance_Field','Escherichia_Coli_Quanti_Tray',
              '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','Sodium_Total','Copper_Total','Organic_Carbon_Total',
              'Turbidity_Lab','Chloride_Total', 'Fluoride_Total','Aluminum_Total',Antimony_Total',
              'Arsenic_Total','Barium_Total','Beryllium_Total','Cadmium_Total','Chromium_Total',
              'Iron_Total','Lead_Total','Manganese_Total','Molybdenum_total','Nickel_Total',
              'Selenium_Total','Silver_Total','Thallium_Total','Zinc_Total')

# Split list of ContVars_Sed into two groups.
# 1. ContVars_LowVar = Variables with low variability, defined as all result values from one or more
#    Zones have the same value.
# 2. ContVars_NotLowVar = Variables without low variability, defined as result values for each zone have
#    more than one distinct value.
# Begin by Counting the number of unique values for each parameter in each zone, and in all zones
# combined.
ContVars_Count_Unique <-data.frame()
ZoneList <- unique(data_cont_WQ$Basin)
# loop through list of Zones
for (i in seq_along(ZoneList)){
  ZoneName <- ZoneList[i]
  data_cont_WQ_subset <- subset(data_cont_WQ, data_cont_WQ$Basin == ZoneName)
  # loop through list of variables for each Zone
  for (i in seq_along(ContVars)) {
    VarName <- ContVars[i]
    tempDataFrame1 <- data.frame(data_cont_WQ_subset[,c(VarName)])
    tempDataFrame <- subset(tempDataFrame1, tempDataFrame1[,1] != 'NA')
    tempCount <- length(unique(tempDataFrame[,1]))
    output_df <- data.frame(ZoneName, VarName, tempCount)
    names(output_df) <- c('ZoneName','VarName', 'Count_Unique_Values')
    ContVars_Count_Unique <- rbind(ContVars_Count_Unique,output_df)
  }
  data_cont_WQ_subset <- subset(data_cont_WQ, data_cont_WQ$Combined == 'All Basins')
  # loop through list of variables for all Zones combined
  for (i in seq_along(ContVars)) {
    VarName <- ContVars[i]
    tempDataFrame1 <- data.frame(data_cont_WQ_subset[,c(VarName)])
    tempDataFrame <- subset(tempDataFrame1, tempDataFrame1[,1] != 'NA')
    tempCount <- length(unique(tempDataFrame[,1]))
    output_df <- data.frame('All Basins', VarName, tempCount)
    names(output_df) <- c('ZoneName','VarName', 'Count_Unique_Values')
    ContVars_Count_Unique <- rbind(ContVars_Count_Unique,output_df)
  }
}
```

```
# Create a list of variables that have low variability in one or more zones.
LowVar <- subset(ContVars_Count_Unique, (ContVars_Count_Unique$ZoneName != 'All Basins' &
                                         ContVars_Count_Unique$Count_Unique_Values == 1))
ContVars_LowVar <- unique(LowVar$VarName)
# Create another list with all variables not in the low variability list.
ContVars_NotLowVar <- setdiff(ContVars,ContVars_LowVar)
# Identify subset of analytes that have low vabiability for individual basins,
# but do not have low variability for the combined subpop. (Analytes where not
# all result values in combined subpop are the same.)
NoVar <- subset(ContVars_Count_Unique, (ContVars_Count_Unique$ZoneName == 'All Basins' &
                                         ContVars_Count_Unique$Count_Unique_Values == 1))
ContVars_NoVar <- unique(NoVar$VarName)
ContVars_Combined_NotLowVar <- setdiff(ContVars_LowVar, ContVars_NoVar)

# Run Continuous analysis for all analytes in ContVars_LowVar.
# Remove percentile estimate ('Pct') from list of statistics. Pct results are
# unable to be calculated for subpops where all result values are the same
# value.
Water_quality_Cont_LowVar <- cont_analysis(data_cont_WQ,
                                           vars = ContVars_LowVar,
                                           subpops = c('Combined','Basin'),
                                           siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                           weight = 'wgt',
                                           xcoord = 'xcoord',
                                           ycoord = 'ycoord',
                                           stratumID = 'Basin',
                                           vartype = 'Local',
                                           statistics = c('CDF','Mean','Total'),
                                           conf=95,
                                           popsize = list(Basin = framesize))
# Run Continuous analysis for all analytes in ContVars_NotLowVar.
# Statistics include both CDF and Pct estimates.
Water_quality_Cont_NotLowVar <- cont_analysis(data_cont_WQ,
                                           vars = ContVars_NotLowVar,
                                           subpops = c('Combined','Basin'),
                                           siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                           weight = 'wgt',
                                           xcoord = 'xcoord',
                                           ycoord = 'ycoord',
                                           stratumID = 'Basin',
                                           vartype = 'Local',
                                           conf=95,
                                           popsize = list(Basin = framesize))
# Run Continuous analysis for all analytes in ContVars_Combined_NotLowVar, for
# combined subpopulation only. Only calculate Pct statistic.
# The CDF, Mean, and Total statistics have already been calculated in the
# previous continuous analysis run (Water_quality_Cont_LowVar.)
```

```
Water_quality_Cont_Combined_NotLowVar <- cont_analysis(data_cont_WQ,
                                                       vars = ContVars_Combined_NotLowVar,
                                                       subpops = 'Combined',
                                                       siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                                       weight = 'wgt',
                                                       xcoord = 'xcoord',
                                                       ycoord = 'ycoord',
                                                       stratumID = 'Basin',
                                                       vartype = 'Local',
                                                       statistics = 'Pct',
                                                       conf=95,
                                                       popsize = list(Basin = framesize))

# Merge Results from Water_quality_Cont_LowVar, Water_quality_Cont_NotLowVar,
# and Water_quality_Cont_Combined_NotLowVar.
Water_quality_Cont <- list()
Water_quality_Cont[["CDF"]] <- rbind(Water_quality_Cont_LowVar[["CDF"]],
                                     Water_quality_Cont_NotLowVar[["CDF"]])
Water_quality_Cont[["Mean"]] <- rbind(Water_quality_Cont_LowVar[["Mean"]],
                                     Water_quality_Cont_NotLowVar[["Mean"]])
Water_quality_Cont[["Total"]] <- rbind(Water_quality_Cont_LowVar[["Total"]],
                                     Water_quality_Cont_NotLowVar[["Total"]])
Water_quality_Cont[["Pct"]] <- rbind(Water_quality_Cont_Combined_NotLowVar[["Pct"]],
                                     Water_quality_Cont_NotLowVar[["Pct"]])

# Add data columns to all Water_quality_Cont results data frames. These will be
# used when loading analysis results to GWIS tables for Status Network analysis results.
analysis_date <- as.character(Sys.Date())
Water_quality_Cont <- lapply(Water_quality_Cont,
                             function(df)
                             cbind(df,
                                   SAMPLE_YEAR = '2021-2023',
                                   REPORTING_CYCLE = '15-17',
                                   WATER_RESOURCE = 'STREAM',
                                   ANALYSIS_DATE = analysis_date,
                                   MATRIX = 'WATER'))

# Merge Pct and Mean Results into single data frame, for consistency with format of previous years'
# results.
Water_quality_Cont[["Mean"]] <- cbind(Water_quality_Cont[["Mean"]], Statistic="Mean")
Water_quality_Cont[["Pct"]] <- rbind(Water_quality_Cont[["Pct"]],Water_quality_Cont[["Mean"]])

#### categorical WQ estimates
# Data frame LL_WQ is the merged dsgn_sf and LL_RESULTS data.
# Add Combined category with name "All Basins" and add Basin category
# with reporting unit data.
data_cat_WQ <- data.frame(SS_WQ,
                          Combined = rep("All Basins", nr),
                          Basin = SS_WQ$REPORTING_UNIT)
```

```
# List all variables for categorical analysis here.
CatVars <- c('Ammonia_Category', 'Chlorophyll_Category','TN_Category','TP_Category','DO_Category',
             'TN_TP_DO_Category','E_Coli_Category', 'Fluoride_Category','pH_Category',
             'Antimony_Category','Arsenic_Category','Beryllium_Category','Cadmium_Category',
             'Chromium_Category','Copper_Category','Lead_Category','Iron_Category','Nickel_Category',
             'Selenium_Category','Silver_Category','Thallium_Category','Zinc_Category')

# Run categorical analysis for all analytes in CatVars.
Water_Quality_Cat <- cat_analysis(data_cat_WQ,
                                  vars = CatVars,
                                  subpops = c('Combined','Basin'),
                                  siteID = 'PK_RANDOM_SAMPLE_LOCATION',
                                  weight = 'wgt',
                                  xcoord = 'xcoord',
                                  ycoord = 'ycoord',
                                  stratumID = 'Basin',
                                  vartype = 'Local',
                                  conf=95,
                                  popsize = list(Basin = framesize))

# Add data columns to Water_Quality_Cat results. These will be
# used when loading analysis results to GWIS tables for Status Network analysis results.
analysis_date <- as.character(Sys.Date())
Water_Quality_Cat <- cbind(Water_Quality_Cat,
                           SAMPLE_YEAR = '2021-2023',
                           REPORTING_CYCLE = '15-17',
                           WATER_RESOURCE = 'SS',
                           ANALYSIS_DATE = analysis_date,
                           MATRIX = 'WATER',
                           ANALYSIS_TYPE = 'TARGET POPULATION',
                           ESTIMATEU_UNITS = 'KILOMETERS')

# Export the results
write.csv(Water_Quality_Cat, "2021_2023_SS_Cat.csv", row.names = FALSE)
write.csv(Water_quality_Cont$CDF, file = '2021_2023_SS_Cont_EstCDF.csv', row.names = FALSE)
write.csv(Water_quality_Cont$Pct, file = '2021_2023_SS_Cont_EstPCT.csv', row.names = FALSE)
write.csv(Water_quality_Cont$Total, file = '2021_2023_SS_Cont_EstTotal.csv', row.names = FALSE)
```

Appendix B - Data Qualifiers

Value Qualifiers [from 2017 QA Rule 62-160.700 Table 1 (Data Qualifier Codes)]

Symbol	*Meaning*
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.*

Symbol	*Meaning*
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.*
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.*

Note: italicized descriptions deviate from EPA and/or DEP QAS descriptions.

Missing Values: blank

*Notes: Historically, the W qualifier was used in the following ways. 1) If turbidity is greater than 100 NTU, all analytes coming from that well will be qualified with a W. 2) If the well currently has, or historically had, a water level recording device employing a lead weight, all lead values coming from that well will be qualified with a W. 3) All VOC's will be qualified for each glued PVC well. 4) The following detections of analytes coming from galvanized steel wells will be qualified with a W: iron, manganese, zinc, cadmium. 5) The following detections of analytes coming from stainless steel wells will be qualified with a W: nickel, chromium. 6) All detections of trace metals coming from any type of iron well will be qualified with a W

Appendix C – Quality Assurance Checklist

This checklist is used to independently review the results of Status Network analyses. The example checklist below is for the 2021-2023 combined analyses. The reviewer must fill in the resource and date that the checklist is being completed. Any items of concern noted during the independent review are communicated to the data analyst. The data analyst investigates the items noted, applies any necessary corrections, and performs the analyses again if needed.

2021-2023 Status Network Analysis– QA Checks – (List Water Resource) – (List Date Checklist Last Updated)

Exclusion / Site Evaluation Data

- Check that weights seem reasonable (same order of magnitude, etc.).
- Check that lat / long are in DDMMSS.THM format.
- Check that all "Can be sampled = N" have exclusion category & criteria listed.
- Check that all "Can be sampled = Y" do not have exclusion category or criteria.
- Check that there are no sites missing (e.g. "Can be sampled = NA") up through highest selection sampled in each zone.
- Check that site evaluation data from individual years were completely transferred to combined site evaluation file.

Resource	# Sites 2021	# Sites 2022	# Sites 2023	Sum of # Sites from 3 Individual Years	# Sites 2021-2023 in Combined File

- Check GIS analysis and confirm that sites were removed from analysis if they were located on portions of features not included in coverage used for most recent year's site selections. Manually inspect sites that were removed to understand reason for permanent removal from coverage used for site selections. (Applies to multi-year analysis only.)
- Check that TN, TP, & DO thresholds in site evaluations data match thresholds in WMS Design Document. (Applies to surface water resources only.)

Result Data

- Check that data was provided for all sampled sites.

Year	Total Sites Sampled from Combined Site Evaluations Data	Total Sites in Combined Result Data
2021		
2022		
2023		

- Check that result data from individual years were completely transferred to combined result file.

Resource	# Sampled Sites 2021	# Sampled Sites 2022	# Sampled Sites 2023	Sum of # Sampled Sites from 3 Individual Years	# Sampled Sites in 2021-2023 Combined File

- Check that merged site evaluation and result data has correct number of samples (# sites from result data minus samples marked as inappropriate for analysis (random_site_location ends in "B") minus samples from sites removed in GIS analysis step).

R Script and Analysis Results

- Check that framesize information was correctly transferred from site selection documentation to analysis R script.
- Check that weights used in analysis seem reasonable (same order of magnitude, etc.)
- Check that fatal qualifiers (O, ?, N, X, T) and non-detects with MDL above water quality threshold are handled correctly in analysis.
- Check that thresholds in R script & output files match thresholds in WMS Design Document.
- Check that list of parameters in R script & output files for continuous analysis matches reference list (\\floridadep\data\dear\wqap\sol_z\data analysis\Data Protocols\Parameters for Status Network Continuous Analysis.docx).
- Check that list of parameters in R script & output files for categorical analysis matches reference list (\\floridadep\data\dear\wqap\sol_z\data analysis\Data Protocols\Parameters for Status Network Categorical Analysis.docx).
- Check that number of values (N) used in analysis matches N in merged data file.
- Check that population size for categorical and continuous analysis results matches total extent of resource (framesize total).
- Check that total N in ExtentExt.csv match N from site evaluation data.

Zone	Total N from combined Site Evaluation data	Total N from ExtentExt.csv output file
All Zones (Statewide Analysis)		

- Check that analysis is reproducible. Confirm that R script can be rerun without encountering errors. Inspect any warning encountered and ensure that warnings do not affect analysis results.

Misc. Notes

Recommend performing script edits starting at bottom of list and working towards top, to preserve references to line numbers.