Tech NOTES 10

National Nonpoint Source Monitoring Program

June 2014

A. Bolks[1], A. DeWire, and J.B. Harcum. 2014. Baseline assessment of left-censored environmental data using R. Tech Notes 10, June 2014. Developed for U.S. Environmental Protection Agency by Tetra Tech, Inc., Fairfax, VA, 28 p. Available online at *https://www.epa.gov/polluted-runoff-nonpoint-source-pollution/nonpoint-source-monitoring-technical-notes*.

[1] ORISE Fellow for USEPA Region 5.

Through the National Nonpoint Source Monitoring Program **(NNPSMP)**, states monitor and evaluate a subset of watershed projects funded by the Clean Water Act Section 319 Nonpoint Source Control Program.

The program has two major objectives:

1. To scientifically evaluate the effectiveness of watershed technologies designed to control nonpoint source pollution
2. To improve our understanding of nonpoint source pollution

NNPSMP Tech Notes is a series of publications that shares this unique research and monitoring effort. It offers guidance on data collection, implementation of pollution control technologies, and monitoring design, as well as case studies that illustrate principles in action.

Baseline Assessment of Left-Censored Environmental Data Using R

Introduction

Faced with limited budgets, watershed managers often find it difficult to purchase and use commercial statistical software. This is natural because data analysis, while an important tool to demonstrate environmental results, likely represents a mere fraction of their time in comparison to securing resources, organizing volunteers, implementing best management practices, and so on. Therefore, investing hundreds to more than $1,000 for commercial statistical software that is used less than a couple weeks each year might not be a good option. Nevertheless, the entrepreneurial nature of most watershed managers is to analyze their own data. This can unfortunately lead to data analyses that are not complete or statistically robust. This might occur when a watershed manager is computing summary statistics (e.g., mean, median, quartiles) from a set of water quality samples containing observations reported as less than (<) or not detected (ND). To avoid this trap, watershed managers can turn to R (R Core Team 2013), a freely available software tool from the ***Comprehensive R Archive Network*** (CRAN). R provides a wide array of open source statistical and graphical tools that are collaboratively developed through an international community. Universities are quickly adopting R as an integral coursework component across many areas of study.

This Tech Note provides (1) instructions for downloading and installing R and RStudio; (2) a brief introduction to R; and (3) a method for robustly computing and displaying summary statistics (e.g., mean, median, quartiles) from monitoring program data with less-thans or non-detects using robust regression on order statistics (robust ROS) and maximum likelihood estimation (MLE).[2] We acknowledge that R's command line

[2] The reader is referred to Helsel and Hirsch (1992) for a discussion of hypothesis testing with censored data, which is beyond the scope of this Tech Note.

structure can be difficult to navigate. (See Table 1 for a list of common advantages and disadvantages associated with R.) However, we have provided a sample data set and script to compute summary statistics and generate graphics. The sample data set and script can be modified with your data and used to generate graphics and tabular output for your reports. We hope these scripts will encourage nonpoint source watershed managers to apply these tools to their water quality programs.

Table 1. Common advantages and disadvantages of R

Advantages	Disadvantages
• Free—no initial costs and no maintenance fees. • Can run on several operating systems (Windows, Mac, Linux). • Comprehensive/many packages available for use. • More advanced and robust statistical procedures than many commercial software packages. • Comprehensive online documentation. • Open source.	• No phone support. • Higher learning curve to perform simple graphing tasks. • Lacks convenient spreadsheet-like manipulation of data. • You may be told to "RTM"—read the manual—in response to user forum queries.

Installing R and RStudio

To get started you will need to download and install R (required) and RStudio (Rstudio Team 2012). Installing RStudio is optional; however, we find that the integrated user interface simplifies some aspects of learning how to use R. The instructions in this Tech Note assume that the user will install R and RStudio on a personal workstation with a Windows operating system.[3] This section also includes instructions for copying example data files provided with this Tech Note to a proper location on your computer. Keep in mind that R is updated often, so your specific installation might vary from the instructions provided here. As a result, this section concludes with a list of resources that you can consult for more information.

Initial installation also includes downloading R packages that are needed to implement the instructions in this Tech Note. The initial installation can be completed in less than an hour, discounting the time to download the software and the time to acquire administrative permission to install the software. Once installed, you can open and use the example scripts with the example data set as well as your own data once it is in the correct format.

[3] Instructions are given for a typical single-user Windows workstation installation. Depending on your situation, you might need administrator privileges to install some software. R can also be installed on the Mac or on workstations running Linux.

Initial Installation

1. **Download and Install R.** To get started using R, go to the CRAN website (*http://cran.r-project.org/*). Select the download option (Linux, Mac, or Windows) for your workstation.[4] From the ensuing Web page, select the appropriate file to download—usually the top link. Use the following steps for Windows:
   - Go to *http://cran.r-project.org/*.
   - Click **Download R for Windows**.
   - Select **base**.
   - Click **Download R 3.x.x for Windows**.
   - Click on the downloaded file to install.
   - Click **run** and select the appropriate language.
   - Click **next** through numerous windows. (*We find that the default options work fine for most applications.*)

At the conclusion of a default installation, two shortcuts, **R i386 3.x.x** and **R x64 3.x.x** (see Figure 1), for 32- and 64-bit versions have been installed on your desktop. We recommend using the "native" build (i.e., use the 32-bit version on 32-bit Windows and the 64-bit version on 64-bit Windows). The 64-bit version is faster, but can only work on a 64-bit operating system. Use the following steps to check if your workstation is a 32- or 64-bit operating system:

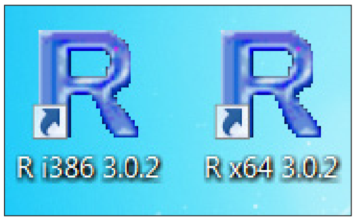


Figure 1. R shortcuts

   - Right-click on **Computer** from your desktop or from Windows Explorer.
   - Click **Properties** (see Figure 2).
   - Under **System type** you will see either 64- or 32-bit operating system listed.

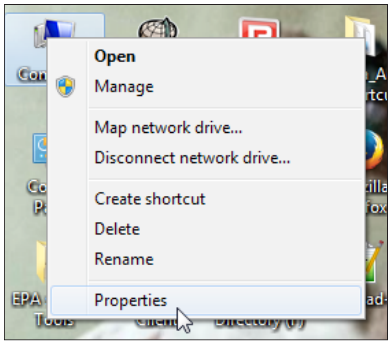


Figure 2. Properties button

2. **Download and Install RStudio** (recommended)**.** Use the following steps to install RStudio:
   - Go to the RStudio, Inc. website, *www.rstudio.com/*.
   - Click **Download now**.
   - Click **Download RStudio Desktop** (*assuming that you are running R on your desktop*).
   - Click on the version of the software recommended for your computer from the next window.
   - Click on the downloaded file to install.
   - Select **next** through a couple of windows. (*As with R, we find the default installation works well.*)

[4] If you already have R on your computer, you will need to have at least version 3.0 (2013) to install the necessary packages.

After installation you can copy the RStudio shortcut to your desktop using the following steps:

- Click the **Start Button** .
- Click **All Programs** > **RStudio** (*i.e., click **All Programs**, and then click the **Rstudio** folder*).
- Right-click on the **RStudio** icon.
- Click **Send To** > **Desktop (create shortcut)** (see Figure 3).

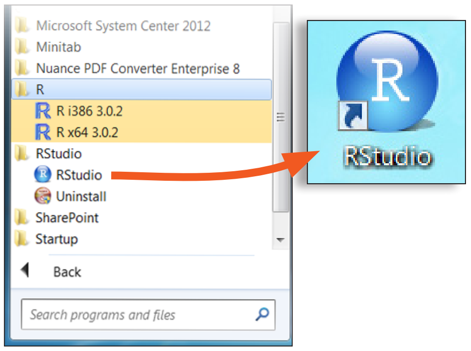
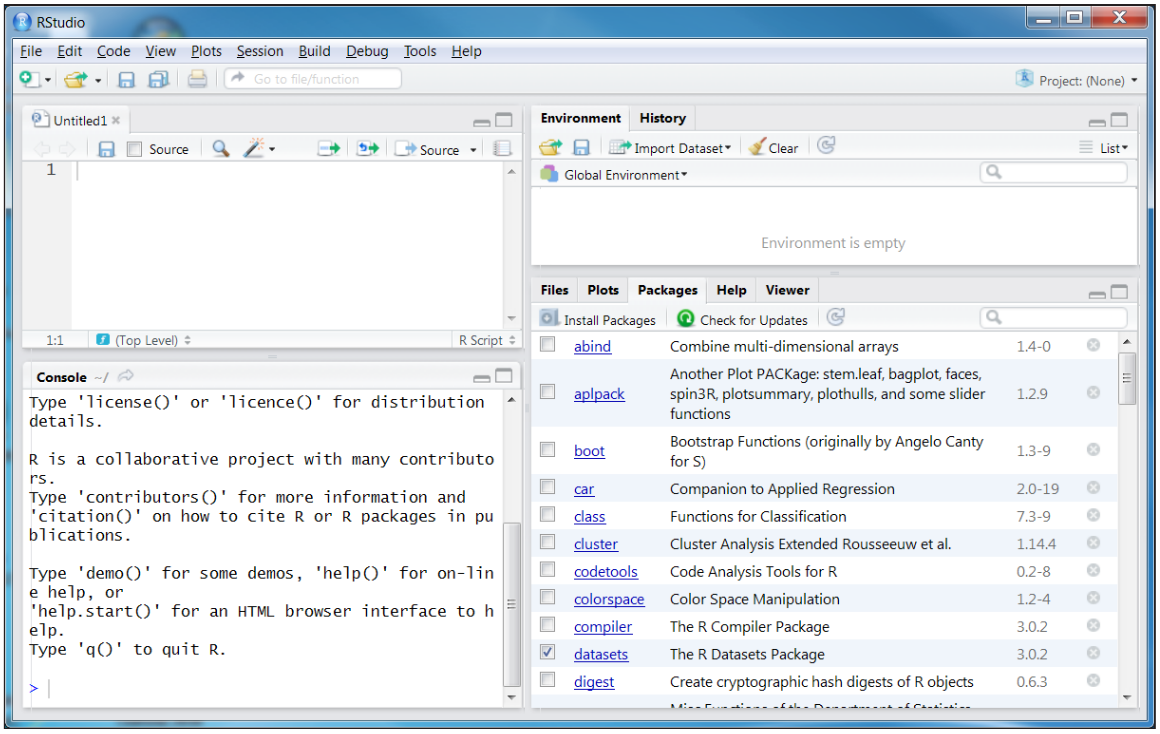

Microsoft System Center 2012
Minitab
Nuance PDF Converter Enterprise 8
R
R i386 3.0.2
R x64 3.0.2
RStudio
RStudio
Uninstall
SharePoint
Startup
Back
Search programs and files
R
RStudio

Figure 3. RStudio shortcut

Use the following instructions to specify which version of R to use within RStudio:

- Double-click on the **RStudio** icon.
- Choose the option that is "native" to your operating system. (*RStudio should only ask you this question the first time you open it.*)
- Click **OK**.

Upon completion, you should see a window similar to Figure 4. The RStudio application has four panes, and some of the panes have multiple tabs. (From within RStudio, you can click **Tools** > **Global Options** > **Pane Layout** to achieve a look similar to Figure 4.)



```
Type 'license()' or 'licence()' for distribution
details.

R is a collaborative project with many contributo
rs.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in pu
blications.

Type 'demo()' for some demos, 'help()' for on-lin
e help, or
'help.start()' for an HTML browser interface to h
elp.
Type 'q()' to quit R.

>
```

☐	abind	Combine multi-dimensional arrays	1.4-0
☐	aplpack	Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, plotsummary, plothulls, and some slider functions	1.2.9
☐	boot	Bootstrap Functions (originally by Angelo Canty for S)	1.3-9
☐	car	Companion to Applied Regression	2.0-19
☐	class	Functions for Classification	7.3-9
☐	cluster	Cluster Analysis Extended Rousseeuw et al.	1.14.4
☐	codetools	Code Analysis Tools for R	0.2-8
☐	colorspace	Color Space Manipulation	1.2-4
☐	compiler	The R Compiler Package	3.0.2
✓	datasets	The R Datasets Package	3.0.2
☐	digest	Create cryptographic hash digests of R objects	0.6.3

Figure 4. RStudio application window

Previous R Versions. If you have previous R versions installed on your computer, you can switch between different versions within RStudio. Click **Tools** > **Global Options**. In the field for **R version** you can specify a version of R you would like RStudio to run. For the instructions in this Tech Note, you need to run version 3.0 or higher. If a lower version is listed, click **Change** and navigate to the folder where the newer version is installed. (*Tip: You can right-click on the RStudio desktop icon or the program in the start menu to see the folder path under **Properties**.*) Chose the R version (and the 32- or 64-bit version as discussed above), and then click **OK**.

3. **Install Additional Packages.** Packages are collections of functions that perform a defined group of analyses and add additional functionality to R. From within RStudio, use the following steps to install the additional packages needed for this Tech Note:
   - Click **Tools** > **Install Packages**.
   - Enter **NADA, Rcmdr, gdata** in the packages dialogue box.
   - Click **Install**.

As you expand your use of R, you might identify other packages that you want beyond those listed here—no problem; just repeat the above process to install the additional packages. (See the *Sources for Help* section for help on how to find more information about other packages.)

Use the following steps to check for updates (on a monthly basis):

- Click **Tools** > **Check for Package Updates**.
- Click the check box for each package.
- Click **Install Updates**.

Setting Up a Test Directory

To follow along with the examples provided with this Tech Note, it will be convenient to set up a test directory that includes the files that accompany this Tech Note. The instructions below provide one approach for accomplishing this task.

1. **Create an R Project directory.** Use the following instructions from within RStudio:
   - Click **File** > **New Project** > **New Directory** > **Empty Project**.
   - Enter **StatsExample** in the directory name dialogue box.
   - Click **Create Project**. (*In our case this would create a directory called C:\Users\abolks\Documents\StatsExample.*)

2. **Copy Example Files.** Outside of RStudio, copy the example files and R scripts (Table 2) that we have provided into the directory created in the previous step.

Table 2. Example files and R script

File Name	Description
Site43.csv	Example data set for Site 43
example1.r	Example R script (reads and analyzes Site43.csv, see Exhibit 1)
LCEB_Data.csv	Nov 2011–Nov 2012 data from Little Calumet East Branch for 46 sites
SummaryStatsROS.r	Production R script (computes summary statistics for LCEB_Data.csv data set)
Parameters.csv	Read by SummaryStatsROS.r (used to identify which parameters to analyze)
cenboxplot2.r	Supporting R script for SummaryStatsROS.r
BootstrapROS.r	Estimates confidence intervals on the mean by implementing ROS using a bootstrapping methodology
Average w Conf_Int.xlsx	Example spreadsheet for creating chart with average concentrations together with confidence intervals

Sources for Help

Several methods exist for obtaining help within RStudio. Some options are:

- Help command on main (top) menu.
- **Help** tab in the bottom-right pane. (Use the accompanying search dialogue box.)
- **Packages** tab in bottom-right pane. (Click on the package name to open the documentation file.)
- At the command prompt enter **??string** to do a global search of string (e.g., **??quartiles**).
- At the command prompt enter **help()** or **help(string)** (e.g., **help(boxplot)**).

Other sources include the following:

- A short (four-page) reference card is available at *http://cran.r-project.org/doc/contrib/Short-refcard.pdf*.
- A (76-page) beginner's guide is available at *http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf*.
- Comprehensive documentation is available at *http://cran.r-project.org/manuals.html*.

Starting Up and Using R

There are many R functions. Our goal in this Tech Note is to provide a few basic commands that build toward functions for estimating summary statistics with data sets that have left-censoring. This section provides beginning level instructions for using R from within RStudio and will explore the first half of the R script shown in Exhibit 1 (Example 1a–1c). This includes opening R script, loading packages, importing data, computing statistics, and generating some simple graphics. We found that beginning R users take about an hour to explore the R script shown in Exhibit 1, Example 1a-1c. (See the *Censored Data* section for a discussion on Example 1d–1f of Exhibit 1).

Exhibit 1. Example workflow (see file **example1.r**)

```
# — load R packages and set working directory                  Example 1a
setwd("C:/Users/abolks/Documents/StatsExample")
library(NADA)
library(Rcmdr, quietly=TRUE);  closeCommander(ask=FALSE)
# — import example data file and summarize total                Example 1b
#     phosphorus data using detection limit substitution
example1 <- read.csv("Site43.csv")
summary(example1$TP)
View(example1)
# — attach example 1; compute summary statistics;               Example 1c
#     prepare boxplots and various x-y bivariate plots
attach(example1)
summary(TP)
numSummary(TP, statistics=c("mean", "sd", "IQR",
   "quantiles", "cv"), quantiles=c(0,.25,.5,.75,1))
boxplot(TP~Season, log="y", range=1.5, xlab="Season",
   ylab="TP, mg/L")
plot(TP~TSS, log="xy", xlab="TSS, mg/L", ylab="TP, mg/L")
scatterplot(TP~TSS | TP_Cen, log="xy", reg.line=lm,
   by.groups=FALSE, xlab="TSS, mg/L", ylab="TP, mg/L",
   legend.title="TP Censoring", legend.coords="bottomright",
   boxplots="xy", smoother=FALSE, cex=1.5)
# — perform robust ROS assuming LOGNORMAL distribution          Example 1d
ros_res = ros(TP, TP_Cen, forwardT="log", reverseT="exp")
plot(ros_res, sub="Lognormal transformation",plot.censored=TRUE)
summary(ros_res)
median(ros_res); mean(ros_res); sd(ros_res)
quantile(ros_res)
# — compute MLE assuming LOGNORMAL distribution                 Example 1e
mle_res = cenmle(TP,TP_Cen, dist="lognormal", conf.int=0.90)
plot(mle_res, sub="Lognormal transformation")
summary(mle_res)
median(mle_res); mean(mle_res); sd(mle_res);
quantile(mle_res)
# — compute Kaplan-Meier                                        Example 1f
km_res = cenfit(TP,TP_Cen, conf.int=0.90)
plot(km_res)
summary(km_res)
median(km_res); mean(km_res); sd(km_res);
quantile(km_res)
```

The shading and font color in Exhibit 1 is used to facilitate viewing. We use **red bold font** to identify functions. Each function can have one or more arguments that instruct R on how to process the function. (Enter **?function** at the > prompt in the **Console** pane (bottom-left pane) to get a list of arguments for a specific function (e.g., **?setwd**). We use a **blue bold font** to identify values that are assigned to arguments, e.g., **boxplot**(TP~Season, log="**y**", range=**1.5**). The lines that begin with **#** are comment lines and are used to describe the R script, but are ignored by R when running the script. The comment lines are shaded in Exhibit 1 for easier identification.

Every Time You Start RStudio

Every time you start RStudio you should set the working directory and load the additional R packages that you anticipate using during that session. From within RStudio, use the instructions provided below. (Notice the information displayed in the **Console** pane as you proceed.)

- Click **Session** > **Set Working Directory** > **Choose Directory**.
- Navigate to the directory where your project files are located.
- Click **Select Folder**.
- Click the **Files** tab in the bottom-right pane to confirm the directory has been set.
- Click the **Packages** tab in the bottom-right pane to load additional R packages.
- Click the check boxes next to **NADA**, **gdata**, and **Rcmdr** for this Tech Note.

The first time the R Commander (Rcmdr) package is turned on there are other packages that it uses, which it will ask you to install. Select **yes** and continue through the dialogues. At the very end another window (the Rcmdr) might come up. Close out of it.

Open example1.r Script

As you went through the above process, you probably noticed commands were entered into the **Console** pane. The **Console** pane is the same window you would see had you opened R directly. All R commands are run from the command prompt signified with "**>**". You could proceed through this Tech Note by simply copying and pasting the code from Exhibit 1 to the command prompt and pressing **Enter**. Although copying and pasting is a common approach, it is easier to open the example script written for this Tech Note using the following instructions from within RStudio:

- Click **File** > **Open File**.
- Select **example1.r**.
- Click **Open**.

These steps open the R script file, **example1.r**, in the top-left panel in RStudio. You could have also clicked on the **Files** tab in the bottom-right pane and clicked on **example1.r**. Compare **example1.r** with Exhibit 1. You will notice that **example1.r** contains additional comment lines for completeness, but the code is the same. The remainder of this section and the *Censored Data* section follow through **example1.r**, assuming that you are using RStudio.

Setting Working Directory and Loading R Packages

Instead of setting a working directory and clicking through each add-on package to turn the package on as described earlier, most R scripts include an initial section of code to complete these steps. Examine the Example 1a code (see Exhibit 2) and find the R code **setwd**(**"C:/Users/abolks/ Documents/StatsExample"**). The function **setwd** sets the working directory for the current session to the directory name in the argument. The **library** function loads add-on packages. Use the following steps to run the R code:

- Change the value inside the parenthesis to the working directory that you set up earlier. (*Note the angle of the "/".*)
- Keep the cursor on this line of code.
- Click **Run** (Run).
- Notice the **setwd** function is run in the **Console** pane (bottom-left pane).
- Your cursor should move to the next line of code that calls the first **library** function. (*Move the cursor to this line of code if necessary.*)
- Click **Run**.
- Notice the **library** function is run in the **Console** pane (bottom-left pane).
- Highlight the next **library** function.
- Click **Run**. (*To save time in the future you can highlight all of these lines of code and click **Run**.*)

Exhibit 2. Example 1a

```
# — load R packages and set working directory                    Example 1a
setwd("C:/Users/abolks/Documents/StatsExample")
library(NADA)
library(Rcmdr, quietly=TRUE);  closeCommander(ask=FALSE)
```

Note that R commands are case sensitive. Neither **Library**(NADA) or **library**(nada) would work. You will also notice that a new window for Rcmdr will open briefly. The package Rcmdr provides a graphical user interface for some basic statistical tools in R. Although there are numerous tools that are not available in Rcmdr, access to the dropdown windows with Rcmdr might be an attractive option for some basic analyses. This Tech Note uses some Rcmdr functions; however, it does not directly use the Rcmdr window. To follow along with the Tech Note, the Rcmdr application is closed with the **closeCommander** function.

Importing Data and Performing Simple Operations

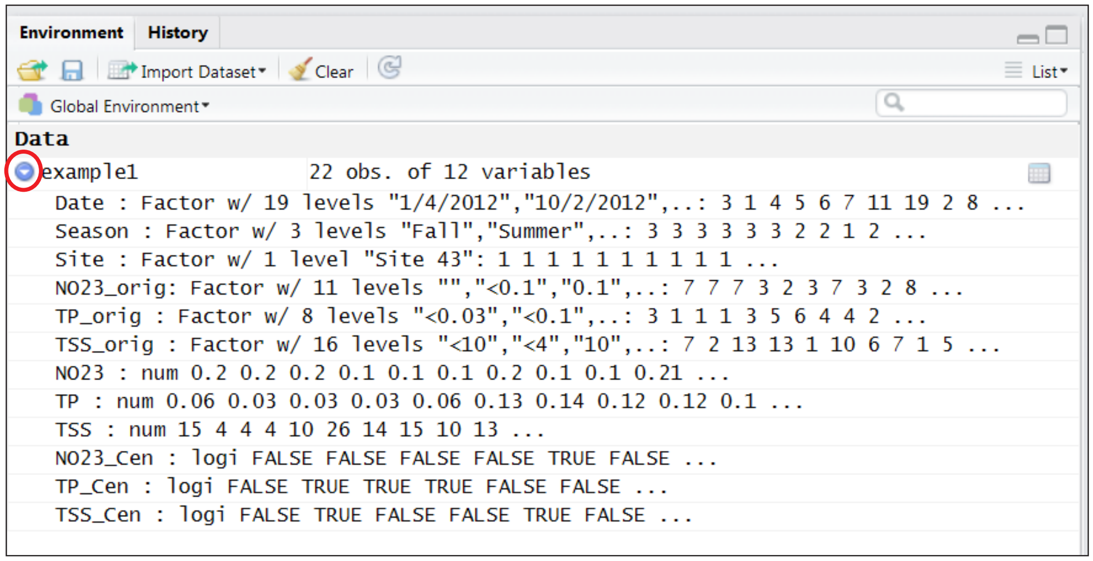
R has tools to import data from numerous other software packages; however, many analysts employ the simplest approach of reading data from comma separated values (*.csv) files, which can be conveniently created using spreadsheet tools. Examine the Example 1b code (see Exhibit 3) and use the following instructions to load data from a *.csv file:

- Place your cursor on the line of code calling the function **read.csv**.
- Click **Run**.

This line of code reads the .csv file into the data frame example1. A data frame is "R-speak" for a table of data organized into columns of variables with the rows containing the values from one observation. R sometimes refers to each column as a vector. In the **Environment** tab[5] (RStudio top-right panel), you can click on the expansion triangle and quickly see the list of variables and summary of information—22 observations and 12 variables; the second variable, season, has three levels: Fall, Summer, and one other value not shown (see Figure 5).

Exhibit 3. Example 1b

```
# — import example data file and summarize total                    Example 1b
#     phosphorus data using detection limit substitution
example1 <- read.csv("Site43.csv")
summary(example1$TP)
View(example1)
```



```
example1                 22 obs. of 12 variables
  Date : Factor w/ 19 levels "1/4/2012","10/2/2012",..: 3 1 4 5 6 7 11 19 2 8 ...
  Season : Factor w/ 3 levels "Fall","Summer",..: 3 3 3 3 3 3 2 2 1 2 ...
  Site : Factor w/ 1 level "Site 43": 1 1 1 1 1 1 1 1 1 1 ...
  NO23_orig: Factor w/ 11 levels "","<0.1","0.1",..: 7 7 7 3 2 3 7 3 2 8 ...
  TP_orig : Factor w/ 8 levels "<0.03","<0.1",..: 3 1 1 1 3 5 6 4 4 2 ...
  TSS_orig : Factor w/ 16 levels "<10","<4","10",..: 7 2 13 13 1 10 6 7 1 5 ...
  NO23 : num 0.2 0.2 0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.21 ...
  TP : num 0.06 0.03 0.03 0.03 0.06 0.13 0.14 0.12 0.12 0.1 ...
  TSS : num 15 4 4 4 10 26 14 15 10 13 ...
  NO23_Cen : logi FALSE FALSE FALSE FALSE TRUE FALSE ...
  TP_Cen : logi FALSE TRUE TRUE TRUE FALSE FALSE ...
  TSS_Cen : logi FALSE TRUE FALSE FALSE TRUE FALSE ...
```

Figure 5. List of variables in data frame example1

Run the next two lines of code (i.e., highlight the two lines of code and click **Run**). The **summary** function responds with a statistical summary in the **Console** pane for the variable total phosphorus (TP). The **View** function provides a table-like view of the data in the top-left panel. You can navigate back to the R script by clicking on the **example1.r** tab in the top-left pane.

In the **summary** function, we used the argument example1$TP to output the summary statistics in the previous R script. This argument instructs R to extract the variable TP from the example1 data frame. An alternative approach is to use the **attach** function to

[5] If you do not see an **Environment** tab, click **Tools** > **Global Options** > **Pane Layout**. Select **Environment** from the list of options and click **OK**.

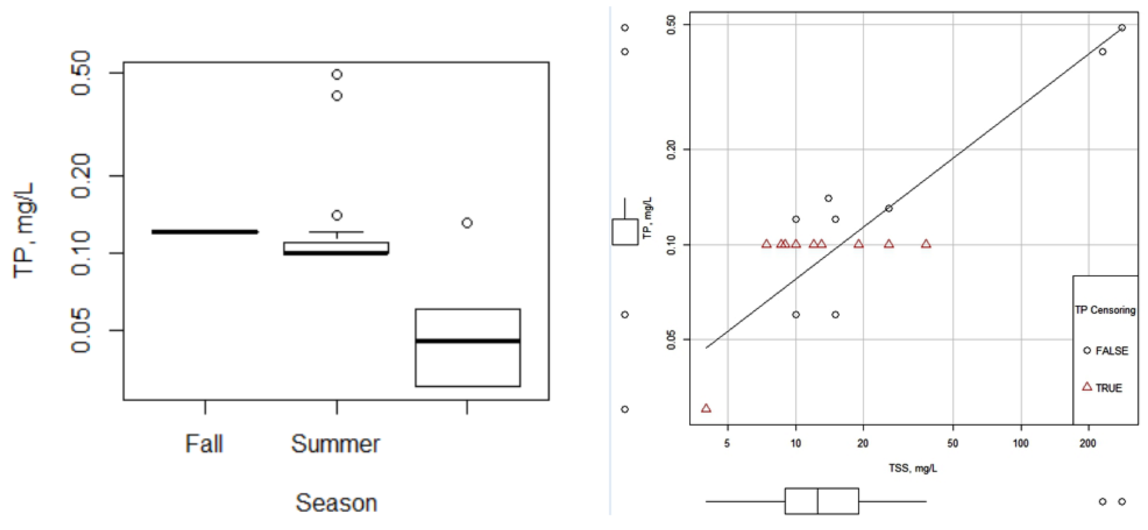
add the data frame to the R search path (see Example 1c code, Exhibit 4). This allows the **summary** function to be shortened as shown in the second line of the Example 1c code.

Exhibit 4. Example 1c

```
# — attach example 1; compute summary statistics;                Example 1c
#     prepare boxplots and various x-y bivariate plots
attach(example1)
summary(TP)
numSummary(TP, statistics=c("mean", "sd", "IQR",
   "quantiles", "cv"), quantiles=c(0,.25,.5,.75,1))
boxplot(TP~Season, log="y", range=1.5, xlab="Season",
   ylab="TP, mg/L")
plot(TP~TSS, log="xy", xlab="TSS, mg/L", ylab="TP, mg/L")
scatterplot(TP~TSS | TP_Cen, log="xy", reg.line=lm,
   by.groups=FALSE, xlab="TSS, mg/L", ylab="TP, mg/L",
   legend.title="TP Censoring", legend.coords="bottomright",
   boxplots="xy", smoother=FALSE, cex=1.5)
```

The **numSummary** function provides an expanded list of summary statistics. The quantiles argument can be expanded to include any list of desired percentiles. Also, notice that the **numSummary** function uses two lines.

The **boxplot** function plots TP grouped by season using a logarithmic scale (see Figure 6). Labels for the x- and y-axis are specified. The **plot** function graphs TP as a function of total suspended solids (TSS) using a logarithmic scale for both x and y (not shown). The **scatterplot** function plots TP as a function of TSS with boxplots; and adds a regression line (see Figure 6). To examine how each function operates in Rstudio, press Run for each line in Exhibit 4 individually or highlight all the code and press Run once. Use the arrows in the plots tab to see previous plots.



Example **boxplot** function

Example **scatterplot** function

Figure 6. Example boxplot and scatterplot figures using total phosphorus (mg/L) data for Site 43, Little Calumet River East Branch

Censored Data

This section provides some background on data censoring and tools that are readily available within R to robustly estimate summary statistics. After the brief background, this section continues with the R script presented in Exhibit 1 (Example 1d–1f) demonstrating three alternative approaches for robustly estimating summary statistics. This section takes about an hour to review and explore the R script shown in Exhibit 1, Example 1d–1f.

What are Censored Data?

Monitoring programs such as those analyzing pesticides, metals, or other constituents may report lab results as below the detection limit (DL) of the analysis. Bacteriological tests may report very high results as "too numerous to count" (TNTC). Such data—typically reported as "<" or ">" some value—are referred to as *censored* data. Censored values are usually associated with limitations of measurement or sample analysis, and are commonly reported as results below or above measurement capacity of the available analytical equipment. See Table 3 for common terms used with censored data. Results that are indistinguishable from a blank sample are normally reported as less than the DL. The true values of these left-censored observations are considered to lie between zero and the DL.[6] Depending on the laboratory, some results greater than the DL may be identified as less than the quantitation limit (QL) or reported as a single value and given a data qualifier to indicate the value is less than the QL. Typically, results reported as less than the QL indicate that the analyte was detected (i.e., greater than the DL), but at a low enough concentration where the precision was deemed too low to reliably report a single value. These interval-censored observations are considered to lie between the DL and QL.

Table 3. Common terms and identifiers associated with various types of censoring

Censoring Type	Left-Censored	Interval-Censored	Uncensored	Right-Censored
Common terms	• Not detected • Not different from a blank, MDL • Zero, "ND", U	• Detected, but not quantified • Method precision is too low for a single number, PQL, LOQ • "Trace", J	• Detected • Positive analytical result • Uncensored	• Too numerous to count • Visible on bottom

As stated above, left and interval censoring is commonly encountered when toxics and pesticides are being analyzed. Left- and interval-censored observations are less commonly encountered when working with sediment and nutrients because they are usually present

[6] In the truest sense of the meaning, *left-censoring* implies no knowledge about the lower bound; however, in most water resource applications, zero is a common lower bound.

at levels above their QLs. Examples of right-censoring include microbiological analyses with misestimated dilution resulting in TNTC and exceedances of flow gage limits during floods. Right-censoring might also be encountered when lakes and estuaries are monitored for light penetration via Secchi depth and the result is reported as "visible on bottom" (i.e., the Secchi disk is observable on the bottom). Helsel (2012) provides a seminal discussion of varying reporting limits and concerns with some data censoring practices. To avoid loss of information, we recommend that DLs and QLs be stored with the measurements and each result be clearly qualified to indicate its relation to the DL or QL as appropriate.

Procedures for Estimating Summary Statistics

Past methods for accommodating censored observations most notably include simple substitution. This involves the replacement of censored observations with zero, ½DL, or DL. Although simple substitution is commonly used (and even recommended) in some state and federal government reports, as well as some refereed journal articles, there is no real theoretical justification for this procedure. Substitution might perform poorly compared to other more statistically robust procedures, especially where censored data represent a high proportion of the entire data set. Substitution can introduce artificial patterns that were not present in the original data. This can lead to incorrect analysis of correlation by producing poor estimates of regression slope and correlation coefficients along with incorrect conclusions of hypothesis testing. More egregiously, some reports have simply deleted observations less than the DL. Removing censored observations from a data set removes the information they contain within the data set. One approach for reporting censored data is shown in Table 4 where "<" is used. Of the 22 TP observations, three observations are reported as "<0.03 mg/L" and 11 observations are reported as "<0.1 mg/L." In this case, <0.03 and <0.1 indicate the results were not detected and the method detection limits (MDLs) were 0.03 and 0.1, respectively. The different MDLs are the result of different laboratories performing the analyses.

Table 4. Total phosphorus (mg/L) data for Site 43, Little Calumet River East Branch (Goulding 2013)

Date	TP, mg/L	TSS, mg/L	Date	TP, mg/L	TSS, mg/L	Date	TP, mg/L	TSS, mg/L
11/29/2011	0.06	0.06	10/2/2012	0.12	0.06	7/17/2012	<0.1	0.06
1/4/2012	<0.03	<0.03	6/11/2012	<0.1	<0.03	7/19/2012	0.41	<0.03
2/7/2012	<0.03	<0.03	6/19/2012	<0.1	<0.03	7/24/2012	<0.1	<0.03
3/6/2012	<0.03	<0.03	6/26/2012	<0.1	<0.03	7/31/2012	0.49	<0.03
4/3/2012	0.06	0.06	6/26/2012	<0.1	0.06	8/7/2012	<0.1	0.06
5/1/2012	0.13	0.13	7/2/2012	<0.1	0.13	8/14/2012	<0.1	0.13
6/5/2012	0.14	0.14	7/10/2012	<0.1	0.14			
8/7/2012	0.12	0.12	7/10/2012	<0.1	0.12			

Extensive research in water resources as well as other fields of science such as survival analysis (e.g., how long does a cancer patient live after treatment) has considered numerous techniques to improve upon simple substitution. Although we acknowledge that simple substitution might be convenient for initial exploratory analyses using spreadsheet tools, more robust procedures are available and recommended. The primary deficiency within watershed management over the last 20 years has been the lack of readily available tools for widespread use, making many of the more robust procedures out of reach for general use. R eliminates this deficiency without purchasing expensive software.

R currently supports two methods for estimating summary statistics (e.g., mean, median, quartiles) from censored data sets that are common in watershed management projects: MLE and robust ROS. Another method, the Kaplan-Meier (KM) method, is commonly used in other disciplines (Helsel 2012) and is also available in R; however, the KM method is only recommended if there are multiple censoring levels (i.e., multiple DLs in the analyzed data set). Other methods also exist, but are not yet available in R. These include a robust MLE method (Kroll and Stedinger 1996) and multiple imputation (Lubin et al. 2004).

Helsel (2012) provides a relatively comprehensive list of recommendations including methods that are available in R and those from other science fields. To date, we have not seen the KM method actively used in watershed management projects, and KM is not recommended for data sets with single censoring limits. Limiting ourselves to those methods that are readily available in R and have some exposure in watershed management, we simplify a more complex set of recommendations by Helsel (2012) to those summarized in Table 5. For larger data sets (n≥50) and censoring from 50–80 percent the MLE is recommended, while the robust ROS is recommended for smaller data sets (n<50) or large data sets with less than 50 percent censoring. Helsel (2012) recommends the KM method for data sets with less than 50 percent censoring and multiple censoring levels. No method is recommended for data sets with more than 80 percent censoring, in which case Helsel (2012) simply recommends reporting the percentage of observations greater than a meaningful threshold. In the future, it might be appropriate to consider additional methods after they gain more traction in watershed management projects or become readily available in R.

Table 5. Current R recommended methods for estimating summary statistics

Sample Size	Percent of Data Censored		
	<50%	50–80%	>80%
n<50	Robust ROS	Robust ROS	Censoring too high to compute summary statistics
n≥50	Robust ROS	MLE	Censoring too high to compute summary statistics

If the data are not normally distributed, the data need to be transformed to use either method from Table 5. If the data cannot be transformed and have multiple censoring levels, consider using the KM method. In the data set **Site43.csv**, the original TP data shown in Table 4, and re-presented in Table 6, are stored in the field TP_orig. This field is separated into a numerical field (TP) which contains the concentrations or MDLs while the field TP_Cen is a logical field set to TRUE when the TP data were not detected and FALSE otherwise. **Site43.csv** also contains data for nitrite+nitrate and TSS stored in a similar fashion. Note that the example R scripts up to this point have simply analyzed the vector TP (and did not consider TP_Cen)—essentially applying simple DL substitution.

Table 6. Organization of data in example file, **Site43.csv**

Date	TP_orig	TP	TP_Cen
11/29/2011	0.06	0.06	FALSE
1/4/2012	<0.03	0.03	TRUE
2/7/2012	<0.03	0.03	TRUE
3/6/2012	<0.03	0.03	TRUE
4/3/2012	0.06	0.06	FALSE
5/1/2012	0.13	0.13	FALSE
…	…	…	…

Both the robust ROS and MLE rely on distribution assumptions. The MLE uses the uncensored observations, the proportion of censored observations, and a distributional assumption to compute estimates of summary statistics. A lognormal distribution is commonly assumed with water quality data; however, a variety of assumptions could be considered.

Robust ROS for multiple censoring levels was introduced by Helsel and Cohn (1988), extending the work of previous investigators. The readily available procedure in R (written by Lee (2013)) requires an assumption that the censored data follow either a normal or lognormal distribution and there must be a minimum of three uncensored observations. Users can apply the method using other transformations but would need to add additional code to do those calculations (i.e., transformations and back transformations). The robust ROS method is based on regressing raw or transformed uncensored concentrations versus their normal score (i.e., develop a linear regression of the raw or transformed concentrations on a normal probability plot using only the detected observations). The censored observations are then imputed based on this regression. If transformations were used, the imputed values are back-transformed. Summary statistics are then computed from the uncensored data and the imputed values (in the original scale) for the censored data.

Robust ROS in R

If you just re-entered RStudio, be sure to run the code that set the working directory, load the additional R packages, read in the example data, and attach the data (see previous sections of the R script). (*Tip: You can highlight all of Example 1a-1c code and press* **Run** *once.*)

To implement the robust ROS using R, highlight and run all the code shown in Example 1d (see Exhibit 5). The remainder of this section summarizes each function.

Exhibit 5. Example 1d

```
# — perform robust ROS assuming LOGNORMAL distribution          Example 1d
ros_res = ros(TP, TP_Cen, forwardT="log", reverseT="exp")
plot(ros_res, sub="Lognormal transformation",plot.censored=TRUE)
summary(ros_res)
median(ros_res); mean(ros_res); sd(ros_res)
quantile(ros_res)
```

The **ros** function uses the arguments TP and TP_Cen as the input data for analysis. The last two arguments (forwardT=**"log"**, reverseT=**"exp"**) specify the transformation process of the data, initially log transforming the data before fitting the regression model and then exponentiation of the modeled data back to normal space. The results are stored in ros_res (see the **Environment** pane in RStudio).

The **plot** function creates the probability plot shown in Figure 7, where the eight uncensored TP observations (after log transformation) are plotted as a function of their normal score together with the ROS regression line (symbolized as solid black dots and solid line). In this example, the y-axis is TP expressed as milligrams per liter (mg/L). This regression line is extended to the censored region of the probability plot and values are imputed (open circles). In this case, the imputed values shown in Figure 7 were computed as log-transformed values, but then transformed back to normal space (i.e., exponentiated) for plotting and summary statistic computation. The imputed values are only used collectively to estimate summary statistics and are not considered estimates for specific samples.

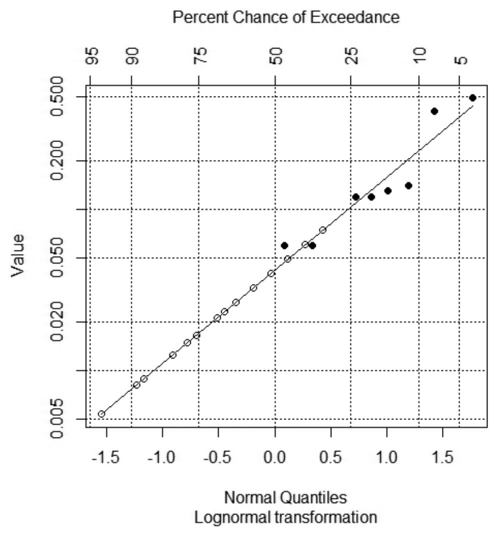


Figure 7. Probability plot of TP (mg/L)

The last three lines of the Example 1d code uses the **summary**, **median**, **mean**, **sd**, and **quantile** functions to extract information from ros_res and present it in the **Console** pane (see Exhibit 6). The **summary** function presents information about the fitted model. In this example, the residuals appear evenly distributed around zero (with a -0.00788 median) and symmetrical (the min/max and 1Q/3Q are of similar magnitude just opposite sign)—both desirable traits. The model fit parameters, including an adjusted R^2 of 0.88 and a p-value <0.001, together with the visual examination of the probability plot (Figure 7) lead us to conclude that log transformation was a good analysis decision. (Additional comments lines with the example script file provide instructions for assuming a normal distribution. It is left to the reader to explore this alternative and contrast with the lognormal assumption results. We found that the larger R^2 value associated with the log-transformed assumption and traditional notion of water quality tended to follow a lognormal distribution compelling.) The **median**, **mean**, **sd**, and **quantile** functions print out the median, mean, standard deviation, and default percentiles.

Exhibit 6. Summary statistics for TP using robust ROS method (lognormal assumption)

```
> summary(ros_res)

Call:
lm(formula = obs.transformed ~ pp.nq)

Residuals:
     Min       1Q   Median       3Q      Max
-0.38693 -0.13178 -0.00788  0.13808  0.38355 ← Residual statistics

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.1637     0.1962 -16.124 3.62e-06 ***
pp.nq         1.3307     0.1855   7.173 0.000371 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2705 on 6 degrees of freedom
Multiple R-squared:  0.8956,     Adjusted R-squared:  0.8782 ← Adjusted R2
F-statistic: 51.45 on 1 and 6 DF,  p-value: 0.0003709

> median(ros_res);  mean(ros_res);    sd(ros_res)
[1] 0.04469706 ← median
[1] 0.08752769 ← mean
[1] 0.1251831  ← standard deviation
> quantile(ros_res)
         5%         10%         25%         50%         75%         90%         95%
0.008246766 0.009292129 0.017834846 0.044697057 0.108745760 0.139000000 0.396500000
```

MLE in R

The **cenmle** function uses the arguments TP and TP_Cen as the input data for analysis (see Example 1e, Exhibit 7). The last two arguments (dist and conf.int) direct the **cenmle** function to assume a lognormal distribution and return the 90 percent confidence intervals for the mean. The results are stored in mle_res. The **plot** function returns a probability plot that is similar to the plot for robust ROS but doesn't include the extrapolation to the censored portion of the data. The **summary** function returns various MLE model fitting parameters. The last two lines of commands interrogate mle_res for various summary statistics (see Exhibit 8). (Additional comment lines with the example script file provide instructions for assuming a normal distribution.)

Exhibit 7. Example 1e

```
# — compute MLE assuming LOGNORMAL distribution                 Example 1e
mle_res = cenmle(TP,TP_Cen, dist="lognormal", conf.int=0.90)
plot(mle_res, sub="Lognormal transformation")
summary(mle_res)
median(mle_res); mean(mle_res); sd(mle_res);
quantile(mle_res)
```

Exhibit 8. Summary statistics for TP using MLE method (lognormal assumption)

```
> median(mle_res);  mean(mle_res);   sd(mle_res)
[1] 0.04393384 ← median
      mean          se      0.9LCL      0.9UCL
0.09408662 0.03798714 0.05326251 0.16620119 ← mean, standard error and 90% CI
[1] 0.1781755 ← standard deviation
> quantile(mle_res)
         5%         10%         25%         50%         75%         90%         95%
0.005770345 0.009034870 0.019111324 0.043933844 0.100996805 0.213637013 0.334500417
```

To implement the MLE using R, highlight and run all the code shown in Example 1e (see Exhibit 7).

Kaplan-Meier in R

The **cenfit** function uses the arguments TP and TP_Cen as the input data for analysis (see Example 1f, Exhibit 9). It is appropriate to use the KM method with this data set because there is more than one DL. The last argument (conf.int) directs the **cenfit** function to return the 90 percent confidence intervals for the mean. KM is nonparametric; therefore, there is no argument regarding distribution. The results are stored in km_res. The **plot** function returns an empirical cumulative distribution function (see Figure 8). In the empirical cumulative distribution function, the data are ranked from smallest to largest and converted to plotting percentiles in which the estimated percentile is the probability of being less or equal to the corresponding observation (symbolized by the solid line). The two-sided confidence interval is symbolized by the dashed lines.

Exhibit 9. Example 1f

```
# — compute Kaplan-Meier                                        Example 1f
km_res = cenfit(TP,TP_Cen, conf.int=0.90)
plot(km_res)
summary(km_res)
median(km_res); mean(km_res); sd(km_res);
quantile(km_res)
```

The **summary** function prints out a summary table of results depicted in Figure 8 while the **median**, **mean**, **sd**, and **quantile** functions return applicable summary statistics (see Exhibit 10). To implement the KM using R, highlight and run all the code shown in Example 1f (see Exhibit 9).

The mean from the three different methods are 0.088 (robust ROS), 0.094 (MLE), and 0.108 (KM). Given the sample size and amount of censoring (n=22, 64 percent censoring) the robust ROS result is preferred.

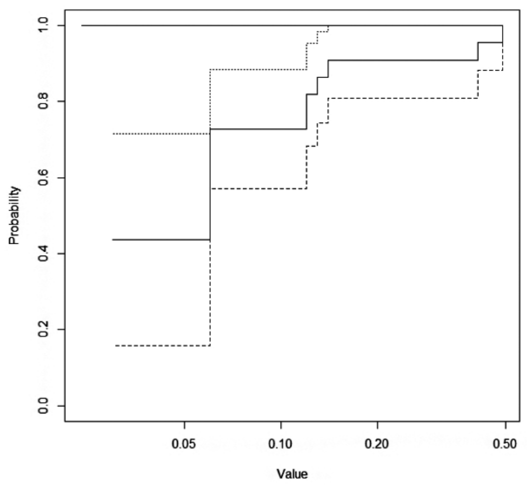


Figure 8. Cumulative distribution function from KM method

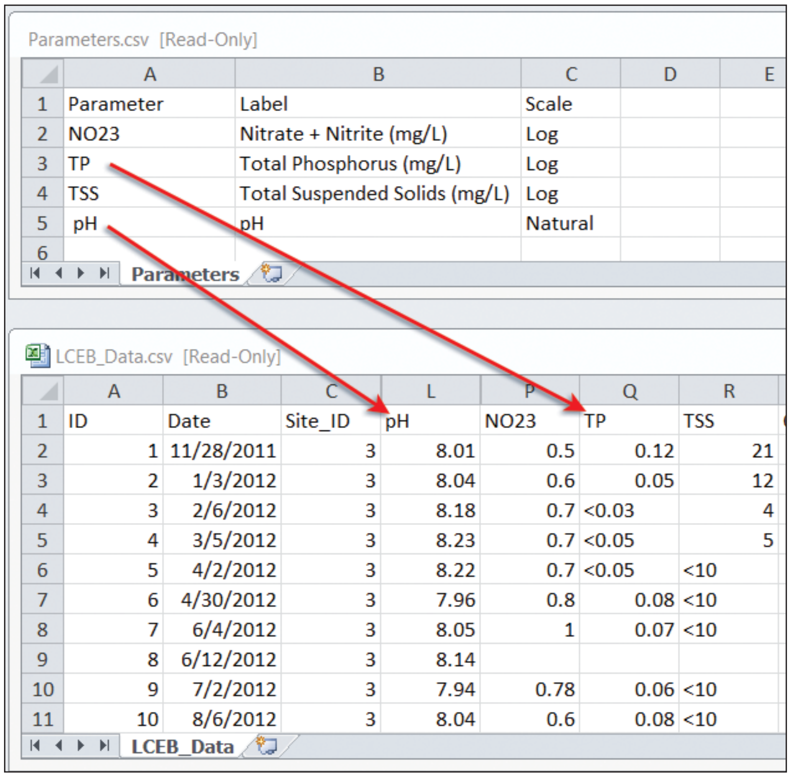
Exhibit 10. Summary statistics for TP using KM method (nonparametric)

```
>  summary(km_res)
   obs n.risk n.event      prob    std.err    0.9LCL    0.9UCL
1 0.03      3       0 0.4363636 0.16921616 0.1580278 0.7146995
2 0.06      5       2 0.4363636 0.16921616 0.1580278 0.7146995
3 0.10     16       0 0.7272727 0.09495145 0.5710915 0.8834540
4 0.12     18       2 0.7272727 0.09495145 0.5710915 0.8834540
5 0.13     19       1 0.8181818 0.08223037 0.6829249 0.9534387
6 0.14     20       1 0.8636364 0.07316500 0.7432906 0.9839821
7 0.41     21       1 0.9090909 0.06129090 0.8082764 1.0000000
8 0.49     22       1 0.9545455 0.04440947 0.8814984 1.0000000
> median(km_res); mean(km_res);  sd(km_res)
[1] 0.06 ← median
      mean         se     0.9LCL     0.9UCL
0.10772727 0.02551877 0.06575264 0.14970191 ← mean, standard error and 90% CI
[1] 0.1196936 ← standard deviation
> quantile(km_res)
  5%  10%  25%  50%  75%  90%  95%
  NA   NA   NA 0.06 0.12 0.14 0.41
```

Production Analyses

Thus far, we have focused on analyzing data for one parameter at one site. This has been a helpful way to introduce R and learn a little bit about how more robust statistical tools can be used to perform summary statistics for left-censored data. Most analysts, however, are interested in more than one site and more than one parameter. Cutting and pasting files and code together to analyze each individual site parameter is cumbersome and has a high likelihood of data entry errors. Included with this Tech Note is the initial release of an R script (**SummaryStatsROS.r**) designed to compute summary statistics using robust ROS, which is demonstrated with November 2011–November 2012 data from Little Calumet East Branch for 46 sites (**LCEB_Data.csv**, provided by Goulding 2013).

The script is designed to analyze multiple sites and parameters in one run. You will notice that the data set **LCEB_Data.csv** only includes one column per parameter (i.e., left-censored data are stored as "<x.x"). Although it is a straightforward process to separate the parameter into two columns, we've chosen to implement that step in the R script. Although the script is designed to skip over missing values, it is not currently designed to process other types of censoring such (e.g., ">", TNTC). There are two supporting files, **Parameters.csv** and **cenboxplot2.r**. The supporting R script (**cenboxplot2.r**) is an updated version of the censored boxplot already available in R, but provides for a more uniform boxplot output. The **Parameters.csv** file (see Figure 9) identifies the parameters to be analyzed. The values in the first column must match the name of the columns to be analyzed in **LCEB_Data.csv** (see red line). The file further specifies parameter labels for the y-axis (Label), and identifies whether log transformation for robust ROS and display of y-scale should be used (Scale).


Parameters.csv [Read-Only]

	A	B	C	D	E
1	Parameter	Label	Scale		
2	NO23	Nitrate + Nitrite (mg/L)	Log		
3	TP	Total Phosphorus (mg/L)	Log		
4	TSS	Total Suspended Solids (mg/L)	Log		
5	pH	pH	Natural		
6					

Parameters

LCEB_Data.csv [Read-Only]

	A	B	C	L	P	Q	R
1	ID	Date	Site_ID	pH	NO23	TP	TSS
2	1	11/28/2011	3	8.01	0.5	0.12	21
3	2	1/3/2012	3	8.04	0.6	0.05	12
4	3	2/6/2012	3	8.18	0.7	<0.03	4
5	4	3/5/2012	3	8.23	0.7	<0.05	5
6	5	4/2/2012	3	8.22	0.7	<0.05	<10
7	6	4/30/2012	3	7.96	0.8	0.08	<10
8	7	6/4/2012	3	8.05	1	0.07	<10
9	8	6/12/2012	3	8.14			
10	9	7/2/2012	3	7.94	0.78	0.06	<10
11	10	8/6/2012	3	8.04	0.6	0.08	<10

LCEB_Data

Figure 9. Relationship between **Parameters.csv** and **LCEB_Data.csv**

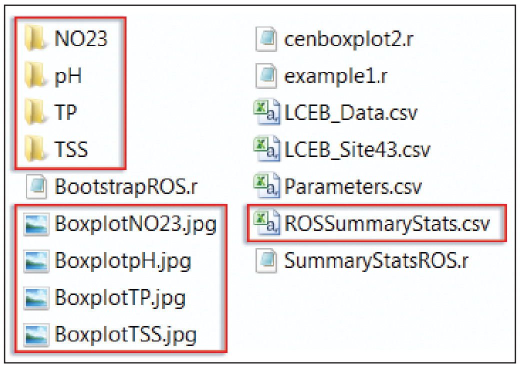
The robust ROS method requires there be at least three uncensored observations and the percent of censoring is not greater than 80 percent. If either of these conditions is not met, the returned summary statistics for the site will be "not applicable" (NA). Within the script there is an option to define a higher number of uncensored data required for generating summary statistics. Although the primary purpose for this script is for left-censored data with multiple censoring levels, it will also work on uncensored data sets.

Use the following steps to run **SummaryStatsROS.r**:

- Open **SummaryStatsROS.r** in RStudio.
- Update the working directory for your computer.
- Highlight all rows in the script.
- Click **Run**. (The program will take about 1–2 minutes to run.)

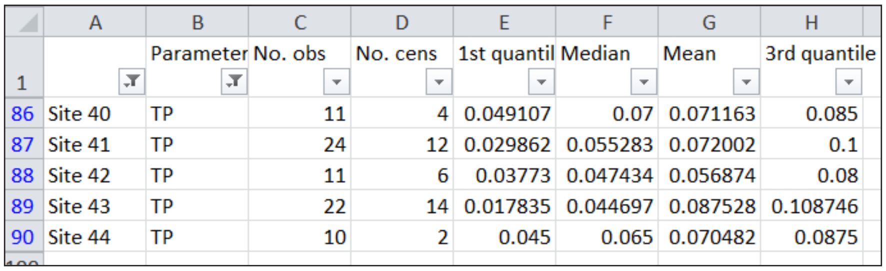
On output, **SummaryStatsROS.r** will store the following to the working directory (see Figure 10):

- Probability plots for each parameter-site combination are stored in separate subdirectories
- Compilation of selected summary statistics (see **ROSSummaryStats.csv**, Figure 11)
- Censored boxplots for each parameter that combine the results for all sites into one figure (see **Boxplotxxxx.jpg**)



NO23	cenboxplot2.r
pH	example1.r
TP	LCEB_Data.csv
TSS	LCEB_Site43.csv
BootstrapROS.r	Parameters.csv
BoxplotNO23.jpg	ROSSummaryStats.csv
BoxplotpH.jpg	SummaryStatsROS.r
BoxplotTP.jpg	
BoxplotTSS.jpg	

Figure 10. Typical directory of files after running **SummaryStatsROS.r**



	A	B	C	D	E	F	G	H
1		Parameter	No. obs	No. cens	1st quantil	Median	Mean	3rd quantile
86	Site 40	TP	11	4	0.049107	0.07	0.071163	0.085
87	Site 41	TP	24	12	0.029862	0.055283	0.072002	0.1
88	Site 42	TP	11	6	0.03773	0.047434	0.056874	0.08
89	Site 43	TP	22	14	0.017835	0.044697	0.087528	0.108746
90	Site 44	TP	10	2	0.045	0.065	0.070482	0.0875

Figure 11. Example output (**ROSSummaryStats.csv**) filtered in a spreadsheet

The probability plots should be reviewed to ensure that the selected normal or lognormal distribution is adequate. Changes can be made to the **Parameters.csv** file as appropriate; however, note that as currently developed the same transformation assumption is made for a given parameter across all sites. The file, **ROSSummaryStats.csv**, contains a compilation of summary statistics (Figure 11). Note that the TP results for Site 43 (as shown in Figure 11) are the same as those for the example ROS results shown earlier (see Exhibit 6).

Figures 12 and 13 show boxplots for the results prepared for TP and TSS, respectively. Sites that could not be estimated because of data constraints are maintained with no information plotted. Like other boxplots, the box represents the 25th, 50th, and 75th percentiles; whiskers extend to the smallest value larger than the 25th percentile - 1.5 x interquartile range and the largest value smaller than the 75th percentile + 1.5 x interquartile range; and open circles represent individual measured values outside the above ranges. Note the gray shading indicating the maximum DL for each site independently. This is particularly useful for the TP data (see Figure 12) where multiple DLs were present even within one year of data. We recommend that figure captions include an explicit statement to ensure that readers are aware that robust ROS was used when preparing the boxplots.

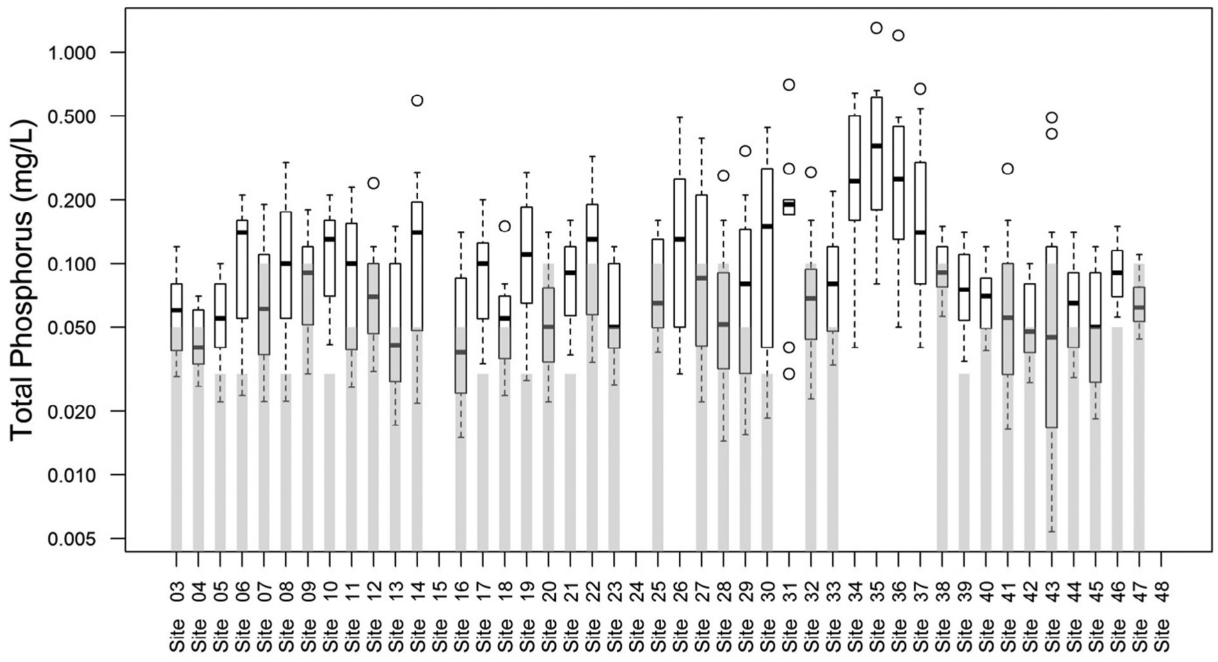


Figure 12. Total phosphorus (mg/L) concentration boxplots by site, Little Calumet East Branch, November 2011–November 2012. (Box represents the 25th, 50th, and 75th percentiles; whiskers extend to the smallest value larger than the 25th percentile - 1.5 x interquartile range and the largest value smaller than the 75th percentile + 1.5 x interquartile range; and open circles represent individual measured values outside the above ranges. Gray shading indicates the maximum detection limit for each boxplot. Portions of boxplot in shaded region were estimated using robust ROS.)

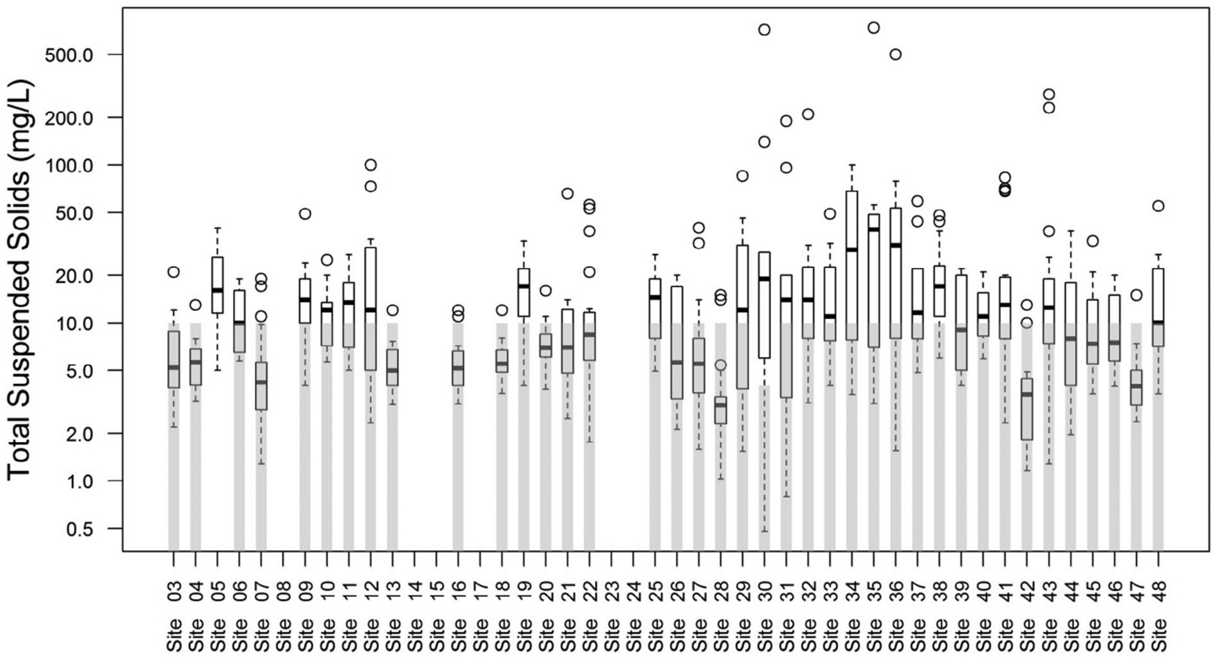


Figure 13. Total suspended solids (mg/L) concentration boxplots by site, Little Calumet East Branch, November 2011–November 2012. (Box represents the 25th, 50th, and 75th percentiles; whiskers extend to the smallest value larger than the 25th percentile - 1.5 x interquartile range and the largest value smaller than the 75th percentile + 1.5 x interquartile range; and open circles represent individual measured values outside the above ranges. Gray shading indicates the maximum detection limit for each boxplot. Portions of boxplot in shaded region were estimated using robust ROS.)

Confidence Interval Estimates on the Estimated Mean

At this point the **SummaryStatsROS.r** script has been used to compute summary statistics using the robust ROS demonstrated with data from Little Calumet East Branch. The output **ROSSummaryStats.csv** includes the number of observations, the number censored, median, first and third quantiles, and the mean.

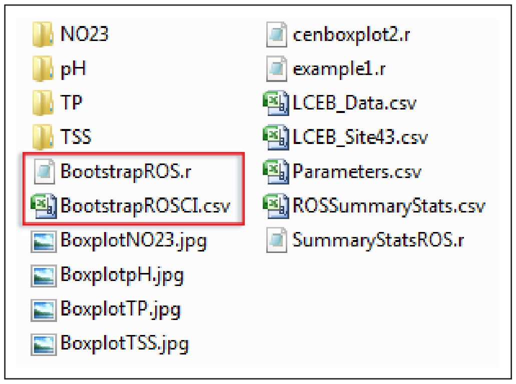
It is also common to provide confidence intervals on the mean. This can be implemented with a bootstrapping method that is a companion to the robust ROS procedure. The R script **BootstrapROS.r** is designed to compute confidence intervals of means for multiple sites and is demonstrated with the Little Calumet East Branch data set (**LCEB_Data.csv**).

Traditional methods of computing confidence intervals that rely on invoking the central limit theorem might not be valid in many instances for this data set because of the small sample size for any particular site. Therefore a bootstrapping method (Helsel 2012) was chosen to obtain unbiased confidence interval estimates for the mean. The bootstrap method selects a random sample (with replacement) from the site data. These data are passed through the robust ROS procedure described above, and a resulting mean is computed. The process of selecting a random sample, implementing the robust ROS procedure, and computing a resulting mean is repeated a large number of, say, 1,000 times. Confidence limits are then empirically selected from this set of 1,000 means. For example, the 10th and 90th percentile of the 1,000 means would correspond to the 80 percent confidence intervals. Also, note that we separated this algorithm from the previous R script (**SummaryStatsROS.r**) because it takes time (5–10 minutes) to run.

Use the following steps to run **BootstrapROS.r**:

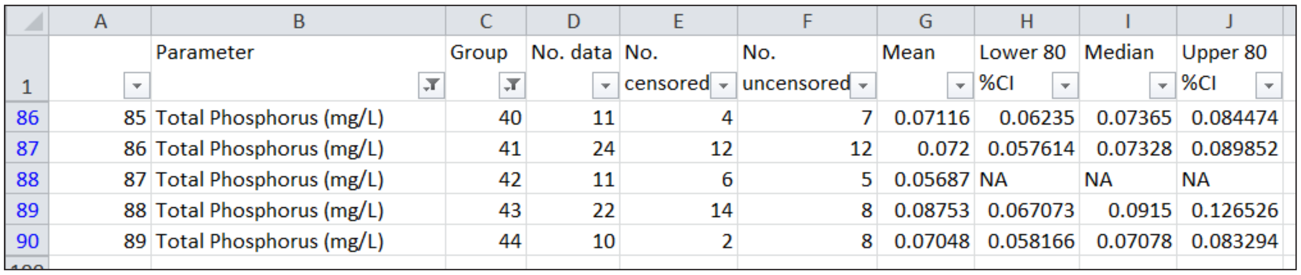
- Open **BootstrapROS.r** in RStudio.
- Update the working directory for your computer.
- Highlight all rows in the script.
- Click **Run**. (The program will take 5–10 minutes to run.)

On output, **BootstrapROS.r** will store the output of the confidence intervals to **BootstrapROSCI.csv** in the working directory (see Figures 14 and 15). Examples of how



```
NO23                  cenboxplot2.r
pH                    example1.r
TP                    LCEB_Data.csv
TSS                   LCEB_Site43.csv
BootstrapROS.r        Parameters.csv
BootstrapROSCI.csv    ROSSummaryStats.csv
BoxplotNO23.jpg       SummaryStatsROS.r
BoxplotpH.jpg
BoxplotTP.jpg
BoxplotTSS.jpg
```

Figure 14. Typical directory of files after running **BootstrapROS.r**



	A	B	C	D	E	F	G	H	I	J
1		Parameter	Group	No. data	No. censored	No. uncensored	Mean	Lower 80 %CI	Median	Upper 80 %CI
86	85	Total Phosphorus (mg/L)	40	11	4	7	0.07116	0.06235	0.07365	0.084474
87	86	Total Phosphorus (mg/L)	41	24	12	12	0.072	0.057614	0.07328	0.089852
88	87	Total Phosphorus (mg/L)	42	11	6	5	0.05687	NA	NA	NA
89	88	Total Phosphorus (mg/L)	43	22	14	8	0.08753	0.067073	0.0915	0.126526
90	89	Total Phosphorus (mg/L)	44	10	2	8	0.07048	0.058166	0.07078	0.083294

Figure 15. Example output (**BootstrapROSCI.csv**) filtered in a spreadsheet

these summary data generated by R can be presented using other software are included below.

- Figures 16 and 18 were created using spreadsheet software such as Excel (see **Average w Conf_Int.xlsx**) and present mean TP (TSS) with 80 percent confidence intervals plotted in comparison to interim and long-term targets set by a local stakeholder group. Plotting means with confidence intervals is a good way to portray the level of uncertainty associated with the mean. Some sites clearly exceed the target while others are clearly below. However, the 80 percent confidence interval for numerous sites overlaps the targets.
- Figures 17 and 19 portray this same information using geographic information systems and "stop light" color coding. Red sites indicate the mean (and confidence intervals) are greater than the long-term target, while green sites are less than the long-term target. Yellow sites indicate those sites where the confidence interval brackets the long-term target.

Tips for Using SummaryStatsROS.r and BootstrapROS.r

As distributed, both R scripts are intended to run with the example data set and should yield the same results. However, **BootstrapROS.r** will yield somewhat different confidence intervals each time it is run because bootstrapping relies on a random selection algorithm. Both R scripts include a section at the beginning of the script to allow the user to make updates for local use. The following is a list of four parameters that the analyst can readily change by following the comment lines associated with the R script:

- **Groupby**. Both R scripts are currently distributed to analyze each parameter by site (i.e., **Site_ID**). The user can change the grouping by selecting another field in the input file (e.g., Year, Month). The user should update **Xlabel** in coordination with **Groupby**.
- **nGreq**. The robust ROS procedure requires a minimum of three uncensored observations. The user might want to increase the minimum number of uncensored observations to a larger number. The effect of a larger **nGreq** will be that fewer sites will be evaluated (i.e., more NAs in the output file). Three uncensored observations were used in this Tech Note, in part, because of the small sample size and because the objective of this document was exploratory analysis.
- **CI**. The confidence interval for the mean in the bootstrapping algorithm can be changed to a user-selected value.
- **Nrep**. As distributed, **BootstrapROS.r** uses a default setting of 1,000 iterations for computing the confidence intervals. The number of bootstrap iterations can be changed. If the data are not highly variable, a smaller number of iterations can be satisfactorily used (say 500). It is possible to evaluate the sensitivity of the results based on **Nrep**. Optimizing the number of iterations, however, tends to be more of an academic exercise that was more important when computers were slower. For production calculations, we recommend running **BootstrapROS.r** twice using 1,000 iterations each time. If the computed confidence intervals are similar, simply use the results from the first run; otherwise, consider increasing the **Nrep**.

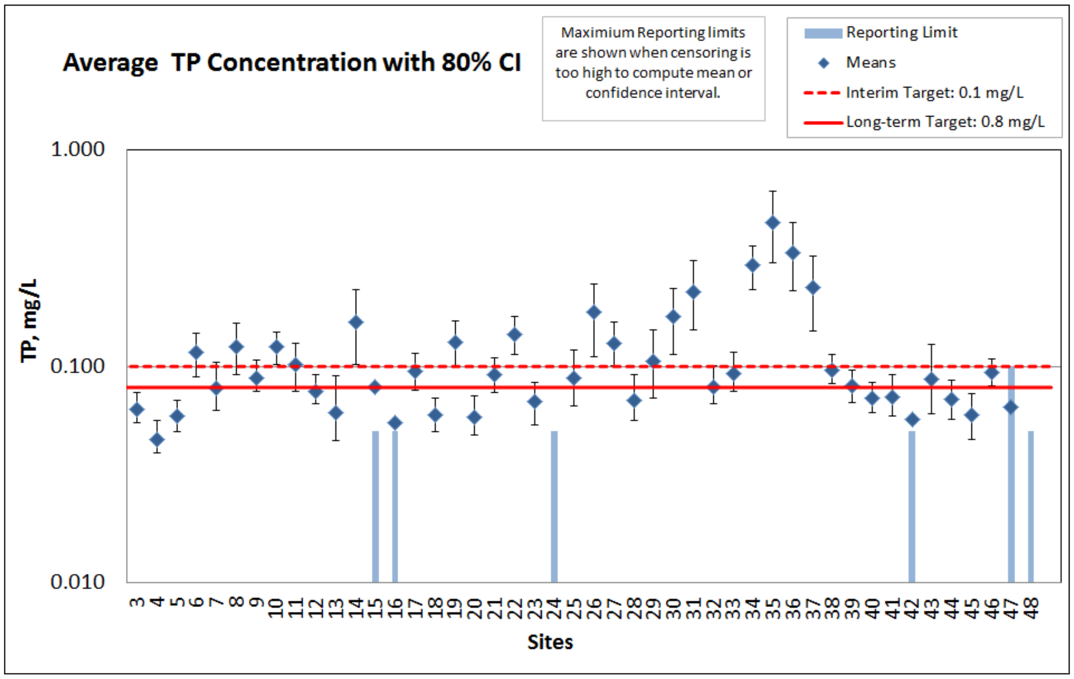


Figure 16. Average total phosphorus (mg/L) concentration by site, Little Calumet East Branch, November 2011–November 2012. (Blue solid diamonds and error bars represent the average concentration and 80 percent confidence intervals estimated using robust ROS with bootstrapping; red dashed and solid red line represent interim and long-term targets set by local stakeholder group.)

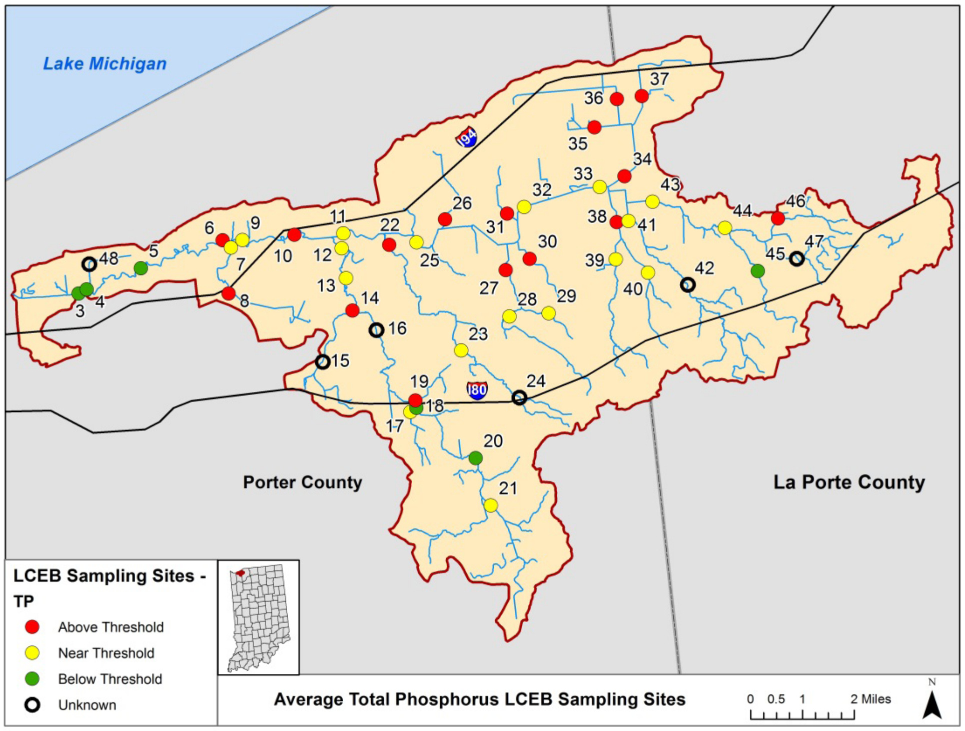


Figure 17. Comparison of average total phosphorus (mg/L) concentration to long-term threshold, Little Calumet East Branch, November 2011–November 2012.

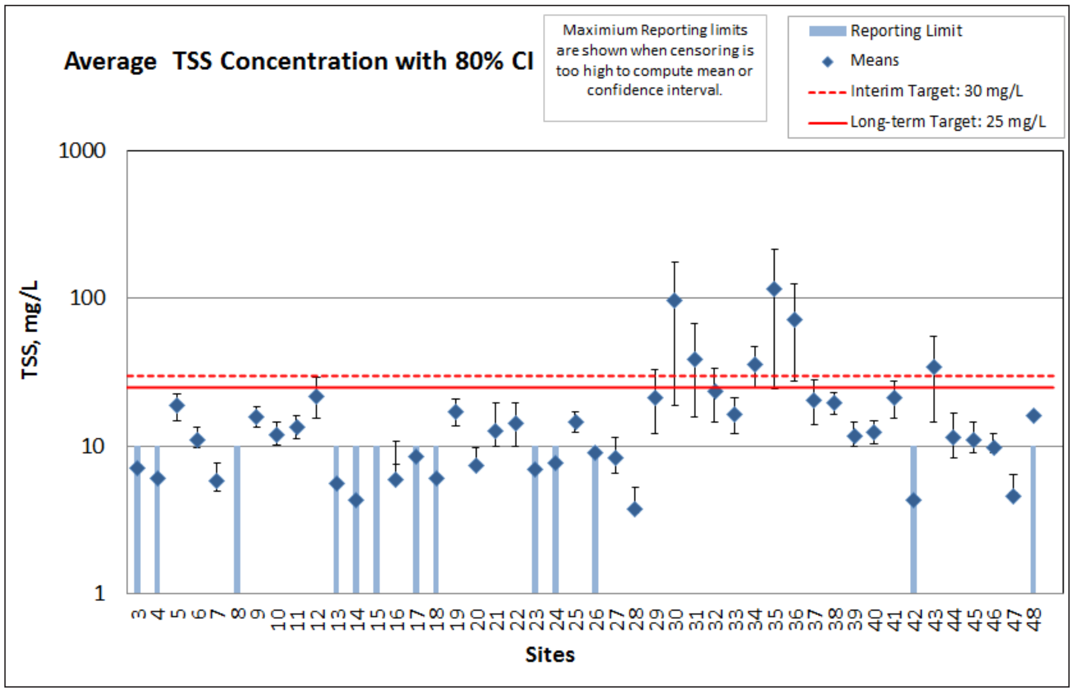


Figure 18. Average total suspended solids (mg/L) concentration by site, Little Calumet East Branch, November 2011–November 2012. (Blue solid diamonds and error bars represent the average concentration and 80 percent confidence intervals estimated using robust ROS with bootstrapping; red dashed and solid red line represent interim and long-term targets set by local stakeholder group.)

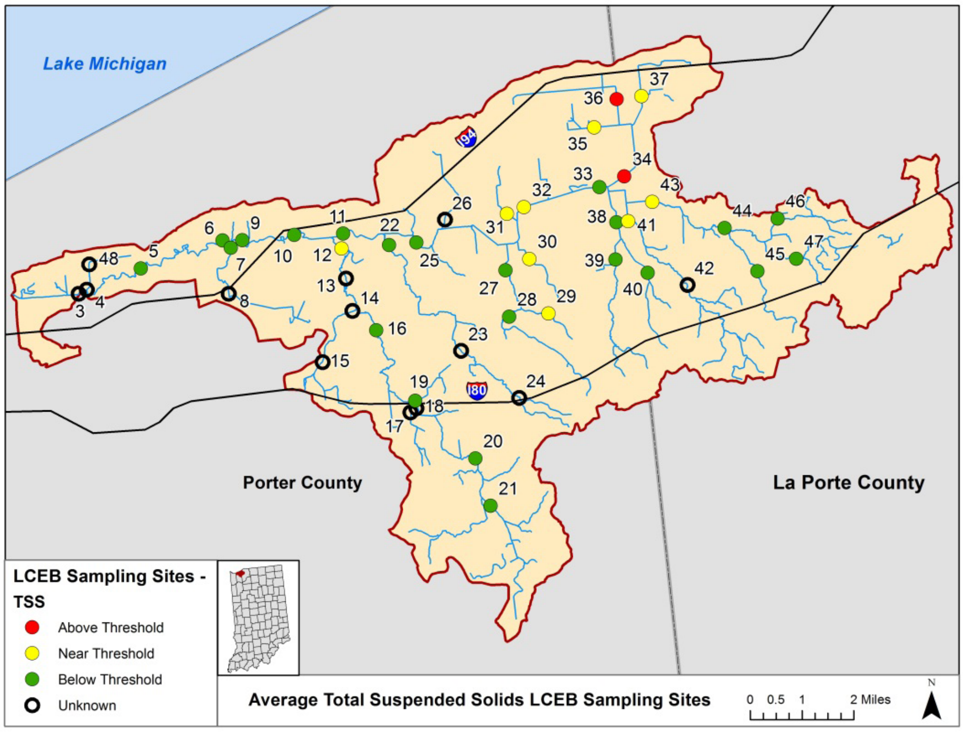


Figure 19. Comparison of average total suspended solids (mg/L) concentration to long-term threshold, Little Calumet East Branch, November 2011–November 2012.

References

Goulding, M. 2013, April 15. Save the Dunes. Email to A. Bolks, ORISE Fellow, EPA Region 5.

Helsel, D.R. 2012. *Statistics for Censored Environmental Data Using Minitab and R*. 2nd ed. Wiley and Sons.

Helsel, D.R., and T.A. Cohn. 1988. Estimation of descriptive statistics for multiply-censored water-quality data. *Water Resources Research* 24(12):1997–2004.

Helsel, D.R., and R.M. Hirsch. 1992. Statistical methods in water resources: U.S. Geological Survey Techniques of Water-Resources Investigations, book 4, chap. A3. ***http://water.usgs.gov/pubs/twri/twri4a3/*** (Accessed 1-22-2013).

Kroll, C.N., and J.R. Stedinger. 1996. Estimation of moments and quantiles using censored data. *Water Resources Research* 32:1005–1012.

Lee, L. 2013. NADA: Nondetects And Data Analysis for environmental data. R package version 1.5–6. ***http://CRAN.R-project.org/package=NADA*** (Accessed 5-21-2014).

Lubin, J.H., J.S. Colt, D. Camann, S. Davis, J.R. Cerhan, R.K. Severson, L. Bernstein, and P. Hartge. 2004. Epidemiologic evaluation of measurement data in the presence of detection limits. *Environmental Health Perspectives* 112:1691–1696.

R Core Team. 2013. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ***http://www.R-project.org/*** (Accessed 6-9-2014).

RStudio Team. 2012. RStudio: Integrated Development for R. RStudio, Inc., Boston, MA. ***http://www.rstudio.com/*** (Accessed 5-21-2014).

This research was supported in part by an appointment to the Postgraduate Research Participation Program to the U.S. Environmental Protection Agency, administered by the Oak Ridge Institute for Science and Engineering through an interagency agreement between the U.S. Department of Energy and EPA.