# The dataRetrieval R package

By Laura De Cicco and Robert Hirsch

August 8, 2014

# Contents

# Figures

# Tables

# 1 Introduction to dataRetrieval

The dataRetrieval package was created to simplify the process of loading hydrologic data into the R environment. It has been specifically designed to work seamlessly with the EGRET R package: Exploration and Graphics for RivEr Trends. See: *https://github.com/USGS-R/EGRET/wiki* for information on EGRET. EGRET is designed to provide analysis of water quality data sets using the Weighted Regressions on Time, Discharge and Season (WRTDS) method as well as analysis of discharge trends using robust time-series smoothing techniques. Both of these capabilities provide both tabular and graphical analyses of long-term data sets.

The dataRetrieval package is designed to retrieve many of the major data types of United States Geological Survey (USGS) hydrologic data that are available on the web. Users may also load data from other sources (text files, spreadsheets) using dataRetrieval. Section 2 provides examples of how one can obtain raw data from USGS sources on the Web and load them into data frames within the R environment. The functionality described in section 2 is for general use and is not tailored for the specific uses of the EGRET package. The functionality described in section 3 is tailored specifically to obtaining input from the Web and structuring it for use in the EGRET package. The functionality described in section 4 is for converting hydrologic data from user-supplied files and structuring it specifically for use in the EGRET package.

For information on getting started in R and installing the package, see (5): Getting Started.

A quick workflow for major dataRetrieval functions:

```r
library(dataRetrieval)
# Site ID for Choptank River near Greensboro, MD
siteNumber <- "01491000"
ChoptankInfo <- getSiteFileData(siteNumber)
parameterCd <- "00060"

#Raw daily data:
rawDailyData <- retrieveNWISdvData(siteNumber,parameterCd,
                    "1980-01-01","2010-01-01")
```

```
# Data compiled for EGRET analysis
Daily <- getDVData(siteNumber,parameterCd,
                   "1980-01-01","2010-01-01")

# Sample data Nitrate:
parameterCd <- "00618"
Sample <- getSampleData(siteNumber,parameterCd,
                        "1980-01-01","2010-01-01")

# Metadata on site and nitrate:
INFO <- getMetaData(siteNumber,parameterCd)

# Merge discharge and nitrate data to one dataframe:
Sample <- mergeReport()
```

## 2 General USGS Web Retrievals

In this section, five examples of Web retrievals document how to get raw data. This data includes site information (2.1), measured parameter information (2.2), historical daily values(2.3), unit values (which include real-time data but can also include other sensor data stored at regular time intervals) (2.4), and water quality data (2.5) or (2.6). We will use the Choptank River near Greensboro, MD as an example. The site-ID for this streamgage is 01491000. Daily discharge measurements are available as far back as 1948. Additionally, nitrate has been measured since 1964.

The USGS organizes hydrologic data in a standard structure. Streamgages are located throughout the United States, and each streamgage has a unique ID. Often (but not always), these ID's are 8 digits. The first step to finding data is discovering this 8-digit ID. There are many ways to do this, one is the National Water Information System: Mapper *http://maps.waterdata.usgs.gov/mapper/index.html*.

Once the site-ID is known, the next required input for USGS data retrievals is the "parameter code". This is a 5-digit code that specifies the measured parameter being requested. For example, parameter code 00631 represents "Nitrate plus nitrite, water, filtered, milligrams per liter as nitrogen", with units of "mg/l as N". A complete list of possible USGS parameter codes can be found at *http:*

*//nwis.waterdata.usgs.gov/usa/nwis/pmcodes?help.*

Not every station will measure all parameters. A short list of commonly measured parameters is shown in Table 1.

**Table 1.** Common USGS Parameter Codes

| pCode | shortName |
|-------|-----------|
| 00060 | Discharge [ft$^3$/s] |
| 00065 | Gage height [ft] |
| 00010 | Temperature [C] |
| 00045 | Precipitation [in] |
| 00400 | pH |

A complete list (as of September 25, 2013) is available as data attached to the package. It is accessed by the following:

```
library(dataRetrieval)
parameterCdFile <-  parameterCdFile
names(parameterCdFile)

[1] "parameter_cd"        "parameter_group_nm"
[3] "parameter_nm"        "casrn"
[5] "srsname"             "parameter_units"

# Sorting out some common values:
subset(parameterCdFile,parameter_cd %in% c("00060","00010","00400"))

     parameter_cd parameter_group_nm
1179        00010             Physical
1206        00060             Physical
1266        00400             Physical
                                parameter_nm casrn
1179         Temperature, water, degrees Celsius
1206             Discharge, cubic feet per second
1266 pH, water, unfiltered, field, standard units
                   srsname parameter_units
1179       Temperature, water          deg C
1206 Stream flow, mean. daily          ft3/s
1266                     pH       std units
```

For unit values data (sensor data measured at regular time intervals such as 15 minutes or hourly), knowing the parameter code and site ID is enough to make a request for data. For most variables that are

measured on a continuous basis, the USGS also stores the historical data as daily values. These daily values are statistical summaries of the continuous data, e.g. maximum, minimum, mean, or median. The different statistics are specified by a 5-digit statistics code. A complete list of statistic codes can be found here:

*http://nwis.waterdata.usgs.gov/nwis/help/?read_file=stat&format=table*

Some common codes are shown in Table 2.

**Table 2.** Commonly used USGS Stat Codes

| StatCode | shortName |
|----------|-----------|
| 00001 | Maximum |
| 00002 | Minimum |
| 00003 | Mean |
| 00008 | Median |

Examples for using these site ID's, parameter codes, and stat codes will be presented in subsequent sections.

## 2.1 Site Information

### 2.1.1 getSiteFileData

Use the `getSiteFileData` function to obtain all of the information available for a particular USGS site such as full station name, drainage area, latitude, and longitude:

```
# Site ID for Choptank River near Greensboro, MD
siteNumber <- "01491000"
ChoptankInfo <- getSiteFileData(siteNumber)
```

A specific example piece of information can be retrieved, in this case a station name, as follows:

```
ChoptankInfo$station.nm

[1] "CHOPTANK RIVER NEAR GREENSBORO, MD"
```

Site information is obtained from *http://waterservices.usgs.gov/rest/Site-Test-Tool.html*

## 2.1.2 getDataAvailability

To discover what data is available for a particular USGS site, including measured parameters, period of record, and number of samples (count), use the `getDataAvailability` function. It is possible to limit the retrieval information to a subset of types (`"dv"`, `"uv"`, or `"qw"`). In the following example, we limit the retrieved Choptank data to only daily data. Leaving the `"type"` argument blank returns all of the available data for that site.

```
# Continuing from the previous example:
# This pulls out just the daily data:

ChoptankDailyData <- getDataAvailability(siteNumber,
                    type="dv")
```

**Table 3.** Daily mean data availabile at the Choptank River near Greensboro, MD. [Some columns deleted for space considerations]

| srsname | startDate | endDate | count | units |
|---|---|---|---|---|
| Temperature, water | 1988-10-01 | 2012-05-09 | 894 | deg C |
| Temperature, water | 2010-10-01 | 2012-05-09 | 529 | deg C |
| Temperature, water | 2010-10-01 | 2012-05-09 | 529 | deg C |
| Stream flow, mean. daily | 1948-01-01 | 2014-08-07 | 24326 | ft$^3$/s |
| Specific conductance | 2010-10-01 | 2012-05-09 | 527 | uS/cm @25C |
| Specific conductance | 2010-10-01 | 2012-05-09 | 527 | uS/cm @25C |
| Specific conductance | 2010-10-01 | 2012-05-09 | 527 | uS/cm @25C |
| Suspended sediment concentration (SSC) | 1980-10-01 | 1991-09-30 | 3651 | mg/l |
| Suspended sediment discharge | 1980-10-01 | 1991-09-30 | 3652 | tons/day |

See Section 6 for instructions on converting an R dataframe to a table in Microsoft Excel or Word to display a data availability table similar to Table 3.

## 2.2 Parameter Information

To obtain all of the available information concerning a measured parameter, use the `getParameterInfo` function:

```
# Using defaults:
parameterCd <- "00618"
parameterINFO <- getParameterInfo(parameterCd)
colnames(parameterINFO)

[1] "parameter_cd"        "parameter_group_nm"
[3] "parameter_nm"        "casrn"
[5] "srsname"             "parameter_units"
```

A specific example piece of information, in this case parameter name, can be obtained as follows:

```
parameterINFO$parameter_nm

[1] "Nitrate, water, filtered, milligrams per liter as nitrogen"
```

Parameter information is obtained from *http://nwis.waterdata.usgs.gov/nwis/pmcodes/*

## 2.3 Daily Values

To obtain daily records of USGS data, use the `retrieveNWISdvData` function. The arguments for this function are siteNumber, parameterCd, startDate, endDate, statCd, and a logical (TRUE/FALSE) interactive. There are 2 default arguments: statCd (defaults to `"00003"`), and interactive (defaults to TRUE). If you want to use the default values, you do not need to list them in the function call. By setting the `"interactive"` option to FALSE, the operation of the function will advance automatically. It might make more sense to run large batch collections with the interactive option set to FALSE.

The dates (start and end) must be in the format `"YYYY-MM-DD"` (note: the user must include the quotes). Setting the start date to `""` (no space) will prompt the program to ask for the earliest date, and setting the end date to `""` (no space) will prompt for the latest available date.

```
# Continuing with our Choptank River example
parameterCd <- "00060"  # Discharge (ft3/s)
startDate <- ""  # Will request earliest date
endDate <- "" # Will request latest date

discharge <- retrieveNWISdvData(siteNumber,
                parameterCd, startDate, endDate)
names(discharge)

[1] "agency_cd"          "site_no"
[3] "datetime"           "X02_00060_00003"
[5] "X02_00060_00003_cd"
```

The column "datetime" in the returned dataframe is automatically imported as a variable of class "Date" in R. Each requested parameter has a value and remark code column. The names of these columns depend on the requested parameter and stat code combinations. USGS remark codes are often "A" (approved for publication) or "P" (provisional data subject to revision). A more complete list of remark codes can be found here: *http://waterdata.usgs.gov/usa/nwis/help?codes_help*

Another example that doesn't use the defaults would be a request for mean and maximum daily temperature and discharge in early 2012:

```
parameterCd <- c("00010","00060")  # Temperature and discharge
statCd <- c("00001","00003")  # Mean and maximum
startDate <- "2012-01-01"
endDate <- "2012-05-01"

temperatureAndFlow <- retrieveNWISdvData(siteNumber, parameterCd,
        startDate, endDate, StatCd=statCd)
```

Daily data is pulled from *http://waterservices.usgs.gov/rest/DV-Test-Tool.html*.

The column names can be automatically adjusted based on the parameter and statistic codes using the renameColumns function. This is not necessary, but may be useful when analyzing the data.

```
names(temperatureAndFlow)

[1] "agency_cd"          "site_no"
[3] "datetime"           "X01_00010_00001"
[5] "X01_00010_00001_cd" "X01_00010_00003"
```

9

```
[7] "X01_00010_00003_cd" "X02_00060_00003"
[9] "X02_00060_00003_cd"

temperatureAndFlow <- renameColumns(temperatureAndFlow)
names(temperatureAndFlow)

[1] "agency_cd"
[2] "site_no"
[3] "datetime"
[4] "Temperature_water_degrees_Celsius_Max_01"
[5] "Temperature_water_degrees_Celsius_Max_01_cd"
[6] "Temperature_water_degrees_Celsius_01"
[7] "Temperature_water_degrees_Celsius_01_cd"
[8] "Discharge_cubic_feet_per_second"
[9] "Discharge_cubic_feet_per_second_cd"
```

An example of plotting the above data (Figure 1):

```
par(mar=c(5,5,5,5)) #sets the size of the plot window

with(temperatureAndFlow, plot(
  datetime, Temperature_water_degrees_Celsius_Max_01,
  xlab="Date",ylab="Max Temperature [C]"
  ))
par(new=TRUE)
with(temperatureAndFlow, plot(
  datetime, Discharge_cubic_feet_per_second,
  col="red",type="l",xaxt="n",yaxt="n",xlab="",ylab="",axes=FALSE
  ))
axis(4,col="red",col.axis="red")
mtext("Mean Discharge [ft3/s]",side=4,line=3,col="red")
title(paste(ChoptankInfo$station.nm,"2012",sep=" "))
legend("topleft", c("Max Temperature", "Mean Discharge"),
       col=c("black","red"),lty=c(NA,1),pch=c(1,NA))
```

There are occasions where NWIS values are not reported as numbers, instead there might be text describing a certain event such as "Ice". Any value that cannot be converted to a number will be reported as NA in this package (not including remark code columns).
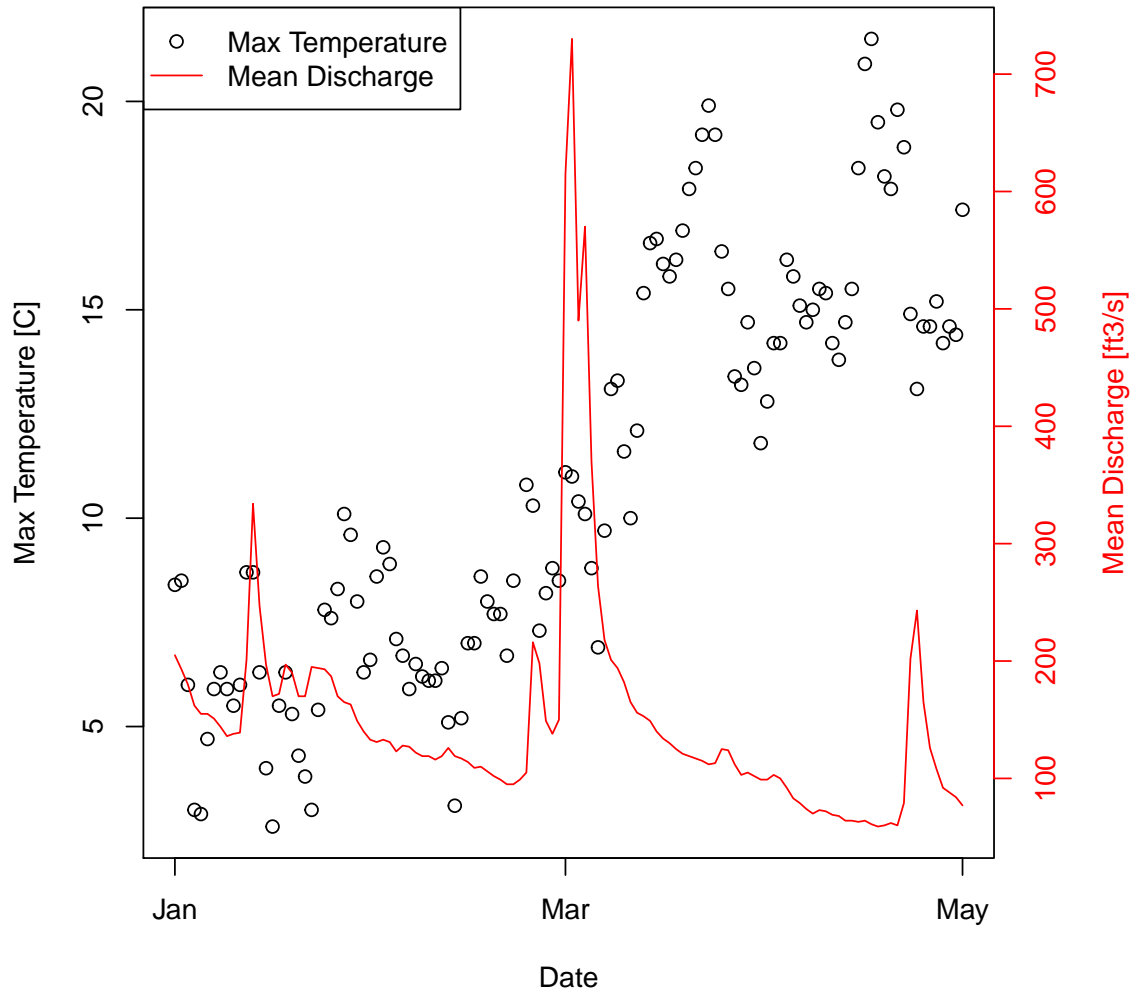
**Figure 1.** Temperature and discharge plot of Choptank River in 2012.

## 2.4 Unit Values

Any data collected at regular time intervals (such as 15-minute or hourly) are known as "unit values". Many of these are delivered on a real time basis and very recent data (even less than an hour old in many cases) are available through the function `retrieveNWISunitData`. Some of these unit values are available for many years, and some are only available for a recent time period such as 120 days. Here is an example of a retrieval of such data.

```
parameterCd <- "00060"  # Discharge (ft3/s)
startDate <- "2012-05-12"
endDate <- "2012-05-13"
dischargeToday <- retrieveNWISunitData(siteNumber, parameterCd,
        startDate, endDate)
```

The retrieval produces the following dataframe:

```
  agency      site              dateTime tz_cd X02_00060_00011
1   USGS 01491000 2012-05-12 00:00:00    EST                83
2   USGS 01491000 2012-05-12 00:15:00    EST                83
3   USGS 01491000 2012-05-12 00:30:00    EST                83
4   USGS 01491000 2012-05-12 00:45:00    EST                83
5   USGS 01491000 2012-05-12 01:00:00    EST                85
6   USGS 01491000 2012-05-12 01:15:00    EST                83
  X02_00060_00011_cd
1                  A
2                  A
3                  A
4                  A
5                  A
6                  A
```

Note that time now becomes important, so the variable datetime is a POSIXct, and the time zone is included in a separate column. Data is retrieved from *http://waterservices.usgs.gov/rest/IV-Test-Tool.html*. There are occasions where NWIS values are not reported as numbers, instead a common example is "Ice". Any value that cannot be converted to a number will be reported as NA in this package.

## 2.5 Water Quality Values

To get USGS water quality data from water samples collected at the streamgage or other monitoring site (as distinct from unit values collected through some type of automatic monitor) we can use the function `retrieveNWISqwData`, with the input arguments: siteNumber, parameterCd, startDate, endDate, and interactive (similar to `retrieveNWISunitData` and `retrieveNWISdvData`).

```r
# Dissolved Nitrate parameter codes:
parameterCd <- c("00618","71851")
startDate <- "1979-10-11"
endDate <- "2012-12-18"

dissolvedNitrate <- retrieveNWISqwData(siteNumber, parameterCd,
      startDate, endDate)
names(dissolvedNitrate)

[1] "dateTime"        "site"             "qualifier_00618"
[4] "value_00618"     "qualifier_71851" "value_71851"
```

```r
with(dissolvedNitrate, plot(
  dateTime, value_00618,
  xlab="Date",ylab = paste(parameterINFO$srsname,
      "[",parameterINFO$parameter_units,"]")
  ))
title(ChoptankInfo$station.nm)
```

**CHOPTANK RIVER NEAR GREENSBORO, MD**
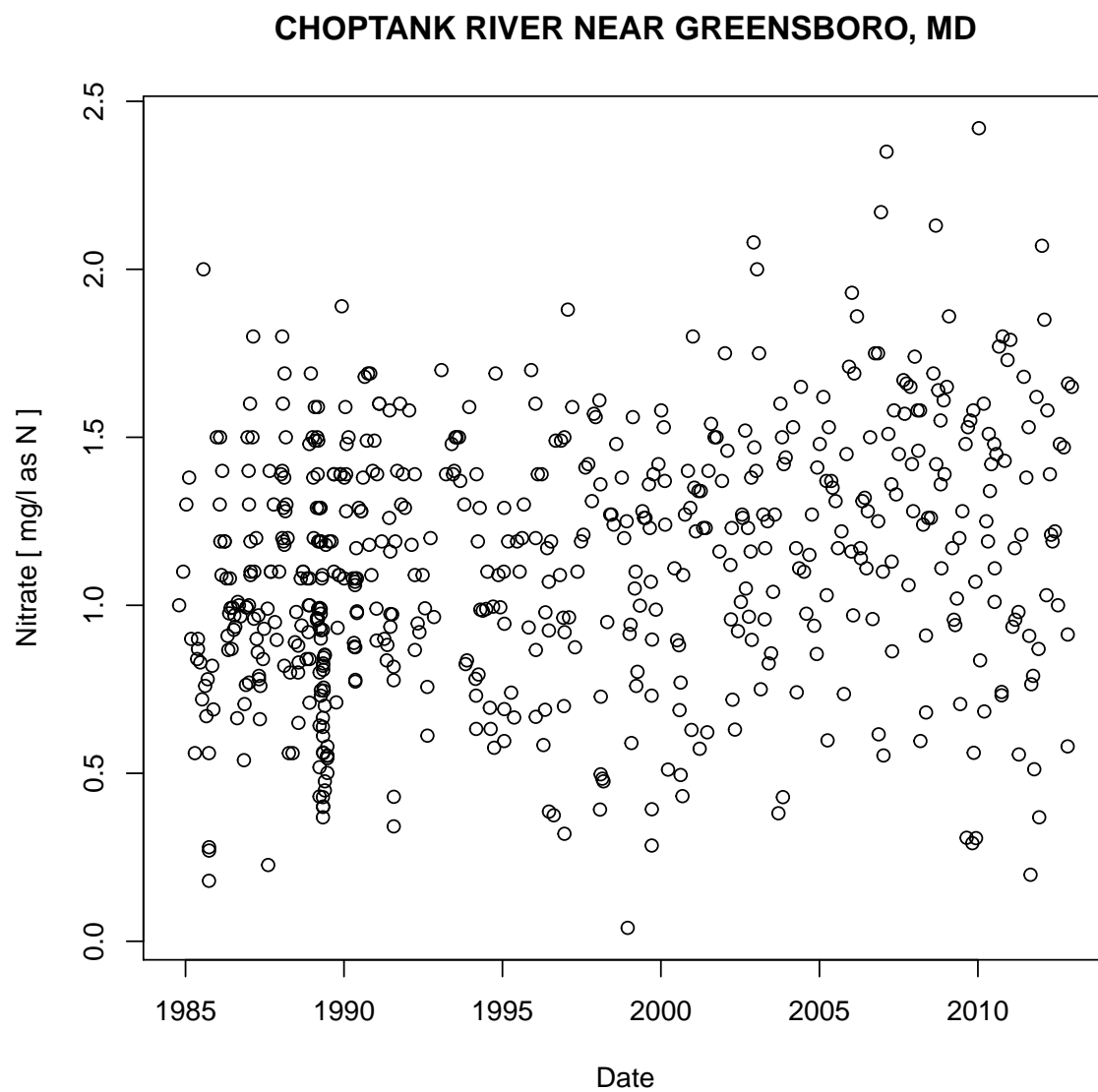


**Figure 2.** Nitrate plot of Choptank River.

## 2.6 STORET Water Quality Retrievals

There are additional water quality data sets available from the Water Quality Data Portal (*http://www.waterqualitydata.us/*). These data sets can be housed in either the STORET (data from EPA) or NWIS database. Since STORET does not use USGS parameter codes, a "characteristic name" must be supplied. The getWQPData function can retrieve either STORET or NWIS, but requires a characteristic name rather than parameter code. The Water Quality Data Portal includes data discovery tools, and information on characteristic names. The following example retrieves specific conductance from a DNR site in Wisconsin.

```
specificCond <- getWQPData('WIDNR_WQX-10032762',
                'Specific conductance','','')
head(specificCond)

    dateTime qualifier. value.
1 2011-02-14             1360
2 2011-02-17             1930
3 2011-03-03             1240
4 2011-03-10             1480
5 2011-03-29             1130
6 2011-04-07             1200
```

There are

## 2.7 URL Construction

There may be times when you might be interested in seeing the URL (web address) that was used to obtain the raw data. The constructNWISURL function returns the URL. In addition to input variables that have been described, there is a new argument "service". The service argument can be "dv" (daily values), "uv" (unit values), "qw" (NWIS water quality values), or "wqp" (general Water Quality Portal values).

```
# Dissolved Nitrate parameter codes:
pCode <- c("00618","71851")
```

```
startDate <- "1964-06-11"
endDate <- "2012-12-18"
url_qw <- constructNWISURL(siteNumber,pCode,startDate,endDate,'qw')
url_dv <- constructNWISURL(siteNumber,"00060",startDate,endDate,
                           'dv',statCd="00003")
url_uv <- constructNWISURL(siteNumber,"00060",startDate,endDate,'uv')
```

# 3 Data Retrievals Structured For Use In The EGRET Package

Rather than using the raw data as retrieved by the web, the dataRetrieval package also includes functions that return the data in a structure that has been designed to work with the EGRET R package (*https://github.com/USGS-R/EGRET/wiki*). In general, these dataframes may be much more 'R-friendly' than the raw data, and will contain additional date information that allows for efficient data analysis.

In this section, we use 3 dataRetrieval functions to get sufficient data to perform an EGRET analysis. We will continue analyzing the Choptank River. We will be retrieving essentially the same data that were retrieved in section 2, but in this case it will be structured into three EGRET-specific dataframes. The daily discharge data will be placed in a dataframe called Daily. The nitrate sample data will be placed in a dataframe called Sample. The data about the site and the parameter will be placed in a dataframe called INFO. Although these dataframes were designed to work with the EGRET R package, they can be very useful for a wide range of hydrology studies that don't use EGRET.

## 3.1 INFO Data

The `getMetaData` function obtains metadata, or data about the streamgage and measured parameters. This function combines `getSiteFileData` and `getParameterInfo`, producing one dataframe called INFO.

```
parameterCd <- "00618"
INFO <-getMetaData(siteNumber,parameterCd, interactive=FALSE)
```

## 3.2 Daily Data

The getDVData function retrieves the daily values (discharge in this case). It requires the inputs

siteNumber, ParameterCd, StartDate, EndDate, interactive, and convert. Most of these arguments are

described in section 2, however "convert" is a new argument (that defaults to TRUE). The convert

argument tells the program to convert the values from cubic feet per second (ft$^3$/s) to cubic meters per

second (m$^3$/s) as shown in the example Daily data frame in Table 4. For EGRET applications with

NWIS web retrieval, do not use this argument (the default is TRUE), EGRET assumes that discharge is

always stored in units of cubic meters per second. If you don't want this conversion and are not using

EGRET, set convert=FALSE in the function call.

```
siteNumber <- "01491000"
startDate <- "2000-01-01"
endDate <- "2013-01-01"
# This call will get NWIS (ft3/s) data , and convert it to m3/s:
Daily <- getDVData(siteNumber, "00060", startDate, endDate)

There are 4750 data points, and 4750 days.
```

If discharge values are negative or zero, the code will set all of these values to zero and then add

a small constant to all of the daily discharge values. This constant is 0.001 times the mean discharge.

The code will also report on the number of zero and negative values and the size of the constant. Use

EGRET analysis only if the number of zero values is a very small fraction of the total days in the record

(say less than 0.1% of the days), and there are no negative discharge values. Columns Q7 and Q30 are

the 7 and 30 day running averages for the 7 or 30 days ending on this specific date. Table 4 lists details

of the Daily data frame.

Notice that the "Day of the year" column can span from 1 to 366. The 366 accounts for leap years.

Every day has a consistent day of the year. This means, February 28th is always the 59th day of the year,

**Table 4.** Daily dataframe

| ColumnName | Type | Description | Units |
|---|---|---|---|
| Date | Date | Date | date |
| Q | number | Discharge in m3/s | $m^3/s$ |
| Julian | number | Number of days since January 1, 1850 | days |
| Month | integer | Month of the year [1-12] | months |
| Day | integer | Day of the year [1-366] | days |
| DecYear | number | Decimal year | years |
| MonthSeq | integer | Number of months since January 1, 1850 | months |
| Qualifier | string | Qualifing code | character |
| i | integer | Index of days, starting with 1 | days |
| LogQ | number | Natural logarithm of Q | numeric |
| Q7 | number | 7 day running average of Q | $m^3/s$ |
| Q30 | number | 30 day running average of Q | $m^3/s$ |

Feb. 29th is always the 60th day of the year, and March 1st is always the 61st day of the year whether or not it is a leap year.

## 3.3 Sample Data

The getSampleData function retrieves USGS sample data from NWIS. The arguments for this function are also siteNumber, ParameterCd, StartDate, EndDate, interactive. These are the same inputs as retrieveWQPqwData or getWQPData as described in the previous section.

```
parameterCd <- "00618"
Sample <-getSampleData(siteNumber,parameterCd,
      startDate, endDate)
```

The getSTORETSampleData function retrieves STORET sample data (or other non-NWIS data) from the water quality portal. The arguments for this function are siteNumber, characteristic-Name, StartDate, EndDate, interactive. Table 5 lists details of the Sample data frame.

```
site <- 'WIDNR_WQX-10032762'
characteristicName <- 'Specific conductance'
Sample <-getSTORETSampleData(site,characteristicName,
      startDate, endDate)
```

**Table 5.** Sample dataframe

| ColumnName | Type | Description | Units |
|---|---|---|---|
| Date | Date | Date | date |
| ConcLow | number | Lower limit of concentration | mg/L |
| ConcHigh | number | Upper limit of concentration | mg/L |
| Uncen | integer | Uncensored data (1=true, 0=false) | integer |
| ConcAve | number | Average of ConcLow and ConcHigh | mg/L |
| Julian | number | Number of days since January 1, 1850 | days |
| Month | integer | Month of the year [1-12] | months |
| Day | integer | Day of the year [1-366] | days |
| DecYear | number | Decimal year | years |
| MonthSeq | integer | Number of months since January 1, 1850 | months |
| SinDY | number | Sine of DecYear | numeric |
| CosDY | number | Cosine of DecYear | numeric |
| Q [1] | number | Discharge | $m^3/s$ |
| LogQ [1] | number | Natural logarithm of discharge | numeric |

[1] Discharge columns are populated from data in the Daily dataframe after calling the `mergeReport` function.

Notice that the "Day of the year" column can span from 1 to 366. The 366 accounts for leap years. Every day has a consistent day of the year. This means, February 28th is always the 59th day of the year, Feb. 29th is always the 60th day of the year, and March 1st is always the 61st day of the year whether or not it is a leap year.

Section 3.4 will talk about summing multiple constituents, including how interval censoring is used. Since the Sample data frame is structured to only contain one constituent, when more than one parameter codes are requested, the `getSampleData` function will sum the values of each constituent as described below.

## 3.4 Censored Values: Summation Explanation

In the typical case where none of the data are censored (that is, no values are reported as "less-than" values) the ConcLow = ConcHigh = ConcAve and Uncen = 1 are equal to the reported value. For the most common type of censoring, where a value is reported as less than the reporting limit, then

ConcLow = NA, ConcHigh = reporting limit, ConcAve = 0.5 * reporting limit, and Uncen = 0.

To illustrate how the dataRetrieval package handles a more complex censoring problem, let us say that in 2004 and earlier, we computed total phosphorus (tp) as the sum of dissolved phosphorus (dp) and particulate phosphorus (pp). From 2005 and onward, we have direct measurements of total phosphorus (tp). A small subset of this fictional data looks like Table 6.

**Table 6.** Example data

| cdate | rdp | dp | rpp | pp | rtp | tp |
|---|---|---|---|---|---|---|
| 2003-02-15 | | 0.020 | | 0.500 | | |
| 2003-06-30 | < | 0.010 | | 0.300 | | |
| 2004-09-15 | < | 0.005 | < | 0.200 | | |
| 2005-01-30 | | | | | | 0.430 |
| 2005-05-30 | | | | | < | 0.050 |
| 2005-10-30 | | | | | < | 0.020 |

The dataRetrieval package will "add up" all the values in a given row to form the total for that sample when using the Sample dataframe. Thus, you only want to enter data that should be added together. If you want a dataframe with multiple constituents that are not summed, do not use getSampleData, getSTORETSampleData, or getSampleDataFromFile. The raw data functions: `getWQPData`, `retrieveNWISqwData`, `retrieveWQPqwData`, `getWQPData` will not sum constituents, but leave them in their individual columns.

For example, we might know the value for dp on 5/30/2005, but we don't want to put it in the table because under the rules of this data set, we are not supposed to add it in to the values in 2005.

For every sample, the EGRET package requires a pair of numbers to define an interval in which the true value lies (ConcLow and ConcHigh). In a simple uncensored case (the reported value is above the detection limit), ConcLow equals ConcHigh and the interval collapses down to a single point. In a simple censored case, the value might be reported as <0.2, then ConcLow=NA and ConcHigh=0.2. We use NA instead of 0 as a way to elegantly handle future logarithm calculations.

For the more complex example case, let us say dp is reported as <0.01 and pp is reported as 0.3. We know that the total must be at least 0.3 and could be as much as 0.31. Therefore, ConcLow=0.3 and

ConcHigh=0.31. Another case would be if dp is reported as <0.005 and pp is reported <0.2. We know in this case that the true value could be as low as zero, but could be as high as 0.205. Therefore, in this case, ConcLow=NA and ConcHigh=0.205. The Sample dataframe for the example data would be:

```
  Sample

        Date ConcLow ConcHigh Uncen ConcAve Julian Month
1 2003-02-15    0.52    0.520     1  0.5200  55927     2
2 2003-06-30    0.30    0.310     0  0.3050  56062     6
3 2004-09-15      NA    0.205     0  0.1025  56505     9
4 2005-01-30    0.43    0.430     1  0.4300  56642     1
5 2005-05-30      NA    0.050     0  0.0250  56762     5
6 2005-10-30      NA    0.020     0  0.0100  56915    10
  Day DecYear MonthSeq    SinDY   CosDY
1  46    2003     1838  0.70407  0.7101
2 182    2003     1842  0.02575 -0.9997
3 259    2005     1857 -0.96251 -0.2712
4  30    2005     1861  0.48506  0.8745
5 151    2005     1865  0.52943 -0.8484
6 304    2006     1870 -0.87861  0.4775
```

Section 3.5 discusses inputting user-generated files. The functions `getSampleDataFromFile` and `getSampleData` assume summation with interval censoring inputs, and are discussed in sections 3.5.1 and 3.5.2.

## 3.5 User-Generated Data Files

In addition to retrieving data from the USGS web services, the dataRetrieval package also includes functions to generate the Daily and Sample data frame from local files.

### 3.5.1 getDailyDataFromFile

The `getDailyDataFromFile` function will load a user-supplied text file and convert it to the Daily dataframe. The file should have two columns, the first dates, the second values. The dates are formatted either mm/dd/yyyy or yyyy-mm-dd. Using a 4-digit year is required. This function has

the following inputs: filePath, fileName,hasHeader (TRUE/FALSE), separator, qUnit, and interactive (TRUE/FALSE). filePath is a string that defines the path to your file, and the string can either be a full path, or path relative to your R working directory. The input fileName is a string that defines the file name (including the extension).

Text files that contain this sort of data require some sort of a separator, for example, a "csv" file (comma-separated value) file uses a comma to separate the date and value column. A tab delimited file would use a tab (`"\t"`) rather than the comma (`","`). Define the type of separator you choose to use in the function call in the `"separator"` argument, the default is `","`. Another function input is a logical variable: hasHeader. The default is TRUE. If your data does not have column names, set this variable to FALSE.

Finally, qUnit is a numeric argument that defines the discharge units used in the input file. The default is qUnit = 1 which assumes discharge is in cubic feet per second. If the discharge in the file is already in cubic meters per second then set qUnit = 2. If it is in some other units (like liters per second or acre-feet per day), the user must pre-process the data with a unit conversion that changes it to either cubic feet per second or cubic meters per second.

So, if you have a file called "ChoptankRiverFlow.txt" located in a folder called "RData" on the C drive (this is a Windows example), and the file is structured as follows (tab-separated):

```
date   Qdaily
10/1/1999   107
10/2/1999   85
10/3/1999 76
10/4/1999 76
10/5/1999 113
10/6/1999 98
...
```

The call to open this file, convert the discharge to cubic meters per second, and populate the Daily data frame would be:

```
fileName <- "ChoptankRiverFlow.txt"
filePath <-  "C:/RData/"
```

```
Daily <- getDailyDataFromFile(filePath,fileName,
                    separator="\t")
```

Microsoft Excel files can be a bit tricky to import into R directly. The simplest way to get Excel data into R is to open the Excel file in Excel, then save it as a .csv file (comma-separated values).

### 3.5.2 getSampleDataFromFile

The `getSampleDataFromFile` function will import a user-generated file and populate the Sample dataframe. The difference between sample data and discharge data is that the code requires a third column that contains a remark code, either blank or `"<"`, which will tell the program that the data was "left-censored" (or, below the detection limit of the sensor). Therefore, the data must be in the form: date, remark, value. An example of a comma-delimited file is:

```
cdate;remarkCode;Nitrate
10/7/1999,,1.4
11/4/1999,<,0.99
12/3/1999,,1.42
1/4/2000,,1.59
2/3/2000,,1.54
...
```

The call to open this file, and populate the Sample dataframe is:

```
fileName <- "ChoptankRiverNitrate.csv"
filePath <-  "C:/RData/"
Sample <- getSampleDataFromFile(filePath,fileName,
                        separator=",")
```

When multiple constituents are to be summed, the format can be date, remark_A, value_A, remark_b, value_b, etc... A tab-separated example might look like the file below, where the columns are date, remark dissolved phosphate (rdp), dissolved phosphate (dp), remark particulate phosphorus (rpp), particulate phosphorus (pp), remark total phosphate (rtp), and total phosphate (tp):

```
date   rdp dp rpp pp rtp tp
2003-02-15 0.020 0.500
```

```
2003-06-30 <0.010 0.300
2004-09-15 <0.005 <0.200
2005-01-30 0.430
2005-05-30 <0.050
2005-10-30 <0.020
...
```

```
fileName <- "ChoptankPhosphorus.txt"
filePath <-  "C:/RData/"
Sample <- getSampleDataFromFile(filePath,fileName,
                                    separator="\t")
```

## 3.6 Merge Report

Finally, there is a function called `mergeReport` that will look at both the Daily and Sample dataframe, and populate Q and LogQ columns into the Sample dataframe. The default arguments are Daily and Sample, however if you want to use other similarly structured dataframes, you can specify localDaily or localSample. Once `mergeReport` has been run, the Sample dataframe will be augmented with the daily discharges for all the days with samples. None of the water quality functions in EGRET will work without first having run the `mergeReport` function.

```
siteNumber <- "01491000"
parameterCd <- "00631"  # Nitrate
startDate <- "2000-01-01"
endDate <- "2013-01-01"

Daily <- getDVData(siteNumber, "00060", startDate, endDate)

There are 4750 data points, and 4750 days.

Sample <- getSampleData(siteNumber,parameterCd, startDate, endDate)
Sample <- mergeReport()

 Discharge Record is 4750 days long, which is 13 years
 First day of the discharge record is 2000-01-01 and last day is 2013-01-01
 The water quality record has 222 samples
 The first sample is from 2000-01-04 and the last sample is from 2012-12-18
 Discharge: Minimum, mean and maximum 0.00991 4.55 246
 Concentration: Minimum, mean and maximum 0.2 1.2 2.4
```

24

```
 Percentage of the sample values that are censored is 0 %

head(Sample)

        Date       Q   LogQ ConcLow ConcHigh Uncen ConcAve
1 2000-01-04  2.747 1.0104    1.59     1.59     1    1.59
2 2000-02-03  3.936 1.3702    1.54     1.54     1    1.54
3 2000-02-15 10.845 2.3837    1.37     1.37     1    1.37
4 2000-02-19 15.518 2.7420    1.24     1.24     1    1.24
5 2000-03-23 56.917 4.0416    0.52     0.52     1    0.52
6 2000-06-05  1.812 0.5946    1.11     1.11     1    1.11
  Julian Month Day DecYear MonthSeq   SinDY   CosDY
1  54789     1   4    2000     1801 0.06005  0.9982
2  54819     2  34    2000     1802 0.54392  0.8391
3  54831     2  46    2000     1802 0.70407  0.7101
4  54835     2  50    2000     1802 0.75113  0.6602
5  54868     3  83    2000     1803 0.98809  0.1539
6  54942     6 157    2000     1806 0.43940 -0.8983
```

## 3.7 EGRET Plots

The Daily, Sample, and INFO data frames (described in Secs. 3.1 - 3.3) are specifically formatted to be used with the EGRET package. The EGRET package has powerful modeling capabilities that use WRTDS, but EGRET also has graphing and tabular tools for exploring the data without using the WRTDS algorithm. See the EGRET vignette, user guide, and/or wiki (*https://github.com/USGS-R/ EGRET/wiki*) for detailed information. Figure 3 shows one of the plotting functions that can be used directly from the dataRetrieval dataframes.

```
# Continuing Choptank example from the previous sections
library(EGRET)
multiPlotDataOverview()
```

# CHOPTANK RIVER NEAR GREENSBORO, MD
## Nitrate
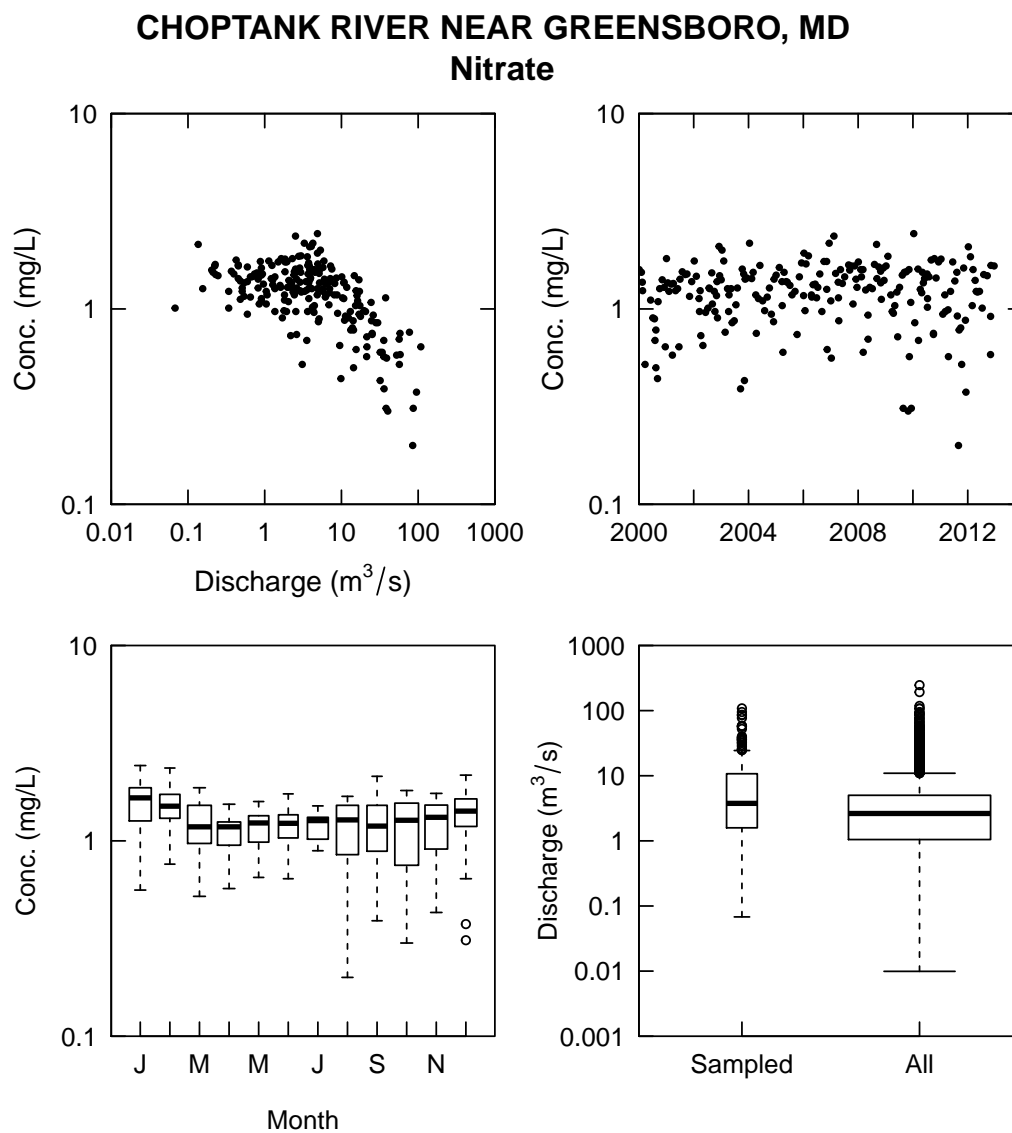


**Figure 3.** Default multiPlotDataOverview

# 4 Summary

Tables 7 and 8 summarize the data retrieval functions:

**Table 7.** dataRetrieval functions

| Data Type | Function Name | Description |
|---|---|---|
| Daily | retrieveNWISdvData | Raw USGS daily data |
| Daily[1] | getDVData | USGS daily values |
| Daily[1] | getDailyDataFromFile | User generated daily data |
| Sample | retrieveNWISqwData | Raw USGS water quality data |
| Sample | retrieveWQPqwData | Raw Water Quality Data Portal data |
| Sample | getQWDataFromFile | Raw user generated water quality data |
| Sample | getWQPData | General Water Quality Portal |
| Sample[1] | getSampleData | USGS water quality data |
| Sample[1] | getSTORETSampleData | STORET Water Quality Data Portal data |
| Sample[1] | getSampleDataFromFile | User generated sample data |
| Unit | retrieveNWISunitData | Raw USGS instantaneous data |
| Information[1] | getMetaData | USGS station and parameter code information |
| Information | getParameterInfo | USGS parameter code information |
| Information | getSiteFileData | USGS station information |
| Information | getDataAvailability | Data available at USGS stations |

[1] Indicates that the function creates a data frame suitable for use in EGRET software

**Table 8.** dataRetrieval miscellaneous functions

| Function Name | Description |
|---|---|
| compressData | Converts value/qualifier into ConcLow, ConcHigh, Uncen |
| getRDB1Data | Retrieves and converts RDB data to dataframe |
| getWaterML1Data | Retrieves and converts WaterML1 data to dataframe |
| getWaterML2Data | Retrieves and converts WaterML2 data to dataframe |
| mergeReport | Merges flow data from the daily record into the sample record |
| populateDateColumns | Generates Julian, Month, Day, DecYear, and MonthSeq columns |
| removeDuplicates | Removes duplicated rows |
| renameColumns | Renames columns from raw data retrievals |

# 5 Getting Started in R

This section describes the options for downloading and installing the dataRetrieval package.

## 5.1 New to R?

If you are new to R, you will need to first install the latest version of R, which can be found here: *http://www.r-project.org/*.

At any time, you can get information about any function in R by typing a question mark before the functions name. This will open a file (in RStudio, in the Help window) that describes the function, the required arguments, and provides working examples.

```
?removeDuplicates
```

This will open a help file similar to Figure 4.

To see the raw code for a particular code, type the name of the function, without parentheses.:

```
removeDuplicates

function (localSample = Sample)
{
    Sample1 <- localSample[!duplicated(localSample[c("DecYear",
        "ConcHigh")]), ]
    return(Sample1)
}
<environment: namespace:dataRetrieval>
```

Additionally, many R packages have vignette files attached (such as this paper). To view the vignette:

```
vignette(dataRetrieval)
```

# Remove Duplicates

## Description

Removes observations from the data frame Sample when the observation has the identical date and value as another observation

## Usage

```
removeDuplicates(localSample = Sample)
```

## Arguments

localSample dataframe with at least DecYear and ConcHigh, default name is Sample

## Value

Sample1 dataframe

## Examples

```
DecYear <- c('1985.01', '1985.01', '1985.02', '1985.02', '1985.03')
ConcHigh <- c(1,2,3,3,5)
dataInput <- data.frame(DecYear, ConcHigh, stringsAsFactors=FALSE)
removeDuplicates(dataInput)
```

[Package *dataRetrieval* version 1.3.0 Index]

**Figure 4.** A simple R help file

## 5.2 R User: Installing dataRetrieval

The following command installs dataRetrieval and subsequent required packages:

```
install.packages("dataRetrieval",
repos=c("http://usgs-r.github.com","http://cran.us.r-project.org"),
dependencies=TRUE,
type="both")
```

After installing the package, you need to open the library each time you re-start R. This is done with the simple command:

```
library(dataRetrieval)
```

# 6 Creating tables in Microsoft from R

There are a few steps that are required in order to create a table in a Microsoft product (Excel, Word, Powerpoint, etc.) from an R dataframe. There are certainly a variety of good methods, one of which is detailed here. The example we will step through here will be to create a table in Microsoft Excel based on the dataframe tableData:

```
availableData <- getDataAvailability(siteNumber)
dailyData <- availableData["dv" == availableData$service,]
dailyData <- dailyData["00003" == dailyData$statCd,]

tableData <- with(dailyData,
      data.frame(
        shortName=srsname,
        Start=startDate,
        End=endDate,
        Count=count,
        Units=parameter_units)
      )
tableData

                          shortName        Start
1                  Temperature, water 2010-10-01
```

```
2                   Stream flow, mean. daily 1948-01-01
3                       Specific conductance 2010-10-01
4 Suspended sediment concentration (SSC) 1980-10-01
5             Suspended sediment discharge 1980-10-01
        End Count      Units
1 2012-05-09   529       deg C
2 2014-08-07 24326      ft3/s
3 2012-05-09   527 uS/cm @25C
4 1991-09-30  3651        mg/l
5 1991-09-30  3652   tons/day
```

First, save the dataframe as a tab delimited file (you don't want to use comma delimited because there are commas in some of the data elements):

```
write.table(tableData, file="tableData.tsv",sep="\t",
            row.names = FALSE,quote=FALSE)
```

This will save a file in your working directory called tableData.tsv. You can see your working directory by typing getwd() in the R console. Opening the file in a general-purpose text editor, you should see the following:

```
shortName   Start   End Count Units

Temperature, water 2010-10-01 2012-06-24 575 deg C

Stream flow, mean. daily 1948-01-01 2013-03-13 23814 ft3/s

Specific conductance 2010-10-01 2012-06-24 551 uS/cm @25C

Suspended sediment concentration (SSC) 1980-10-01 1991-09-30 3651 mg/l

Suspended sediment discharge 1980-10-01 1991-09-30 3652 tons/day
```

Next, follow the steps below to open this file in Excel:

1. Open Excel

2. Click on the File tab

3. Click on the Open option

4. Navigate to the working directory (as shown in the results of getwd())

5. Next to the File name text box, change the dropdown type to All Files (*.*)

6. Double click tableData.tsv

7. A text import wizard will open up, in the first window, choose the Delimited radio button if it is not automatically picked, then click on Next.

8. In the second window, click on the Tab delimiter if it is not automatically checked, then click Finished.

9. Use the many formatting tools within Excel to customize the table

From Excel, it is simple to copy and paste the tables in other Microsoft products. An example using one of the default Excel table formats is here.

| shortName | Start | End | Count | Units |
|---|---|---|---|---|
| Temperature, water | 10/1/2010 | 6/24/2012 | 575 | deg C |
| Stream flow, mean. daily | 1/1/1948 | 3/13/2013 | 23814 | cfs |
| Specific conductance | 10/1/2010 | 6/24/2012 | 551 | uS/cm @25C |
| Suspended sediment concentration (SSC) | 10/1/1980 | 9/30/1991 | 3651 | mg/l |
| Suspended sediment discharge | 10/1/1980 | 9/30/1991 | 3652 | tons/day |

**Figure 5.** A simple table produced in Microsoft Excel