

Introduction to the EGRET package

Robert Hirsch¹ and Laura De Cicco¹

¹*United States Geological Survey*

July 16, 2014

Contents

1	Introduction to Exploration and Graphics for RivEr Trends (EGRET)	5
2	EGRET Workflow.....	6
3	EGRET Dataframes and Units	9
3.1	Daily	9
3.2	Sample.....	10
3.3	INFO	11
3.4	Units	12
4	Flow History	13
4.1	Plotting Options.....	15
4.2	Table Options.....	23
5	Summary of Water Quality Data (without using WRTDS).....	24
5.1	Plotting Options.....	25
5.2	Table Options.....	30
6	Weighted Regressions on Time, Discharge and Season (WRTDS) Analysis	30
7	WRTDS Results	32
7.1	Plotting Options.....	32
7.2	Table Options.....	45
8	Extending Plots Past Defaults	47
9	Getting Started in R	57
9.1	New to R?.....	57
9.2	R User: Installing EGRET	57
10	Common Function Variables	58
10.1	flowHistory Plotting Input.....	58
10.2	Water Quality Plotting Input.....	59
10.3	WRTDS Estimation Input	60
10.4	WRTDS Plotting Input.....	61
11	Creating tables in Microsoft from R.....	64
12	Saving Plots	66

Figures

Figure 1	Plots of discharge statistics	17
Figure 2	Merced River Winter Trend.....	18
Figure 3	plotFour(qUnit=3)	20
Figure 4	plotFourStats(qUnit=3)	21
Figure 5	Mississippi River at Keokuk Iowa	22
Figure 6	Concentration box plots	26
Figure 7	The relation of concentration vs time or discharge.....	27
Figure 8	The relation of flux vs discharge	28
Figure 9	multiPlotDataOverview(qUnit=1).....	29
Figure 10	Concentration and flux vs time	33
Figure 11	Concentration and flux predictions	34
Figure 12	Residuals	35
Figure 13	Residuals with respect to time.....	36
Figure 14	Default boxConcThree().....	37
Figure 15	Concentration and flux history	38
Figure 16	Concentration vs.....	39
Figure 17	plotConcTimeSmooth	40
Figure 18	fluxBiasMulti(qUnit=1)	41
Figure 19	plotContours(yearStart=2008,yearEnd=2010,qBottom=20,qTop=1000,contourLevels=seq(0,2,0....	43
Figure 20	plotDiffContours(year0=2000,year1=2010,qBottom=20,qTop=1000,maxDiff=0	44
Figure 21	Modifying text and point size	48
Figure 22	Modified plotConcQ.....	49
Figure 23	Serif font	50
Figure 24	Contour plot with modified axis and color scheme	52
Figure 25	Difference contour plot with modified color scheme	53
Figure 26	Custom multipanel plot using tinyPlot.....	54
Figure 27	Custom multipanel plot	56
Figure 28	A simple table produced in Microsoft Excel.....	66

Tables

Table 1	Daily dataframe	9
Table 2	Columns added to Daily dataframe after running <code>modelEstimation</code>	10
Table 3	Sample dataframe	10
Table 4	Columns added to Sample dataframe after running <code>modelEstimation()</code>	11
Table 5	INFO dataframe	11
Table 6	INFO dataframe after running <code>modelEstimation()</code>	12
Table 7	Period of Analysis Information	14
Table 8	Index of discharge statistics information	15
Table 9	Table created from <code>tableChangeSingle</code> function using Microsoft Excel (see section 11)	46
Table 10	Useful plotting parameters to adjust in EGRET plotting functions. For details of any of these see ?par.....	47
Table 11	Useful functions to add on to default plots. [Type ? then the function name to get help on the individual function].	47
Table 12	Variables used in flow history plots (<code>plot15</code> , <code>plotFour</code> , <code>plotFourStats</code> , <code>plotQTimeDaily</code> , <code>plotSDLogQ</code>)	58
Table 13	Selected variables used in water quality analysis plots	59
Table 14	Selected variables in WRTDS	60
Table 15	Selected variables used in plots for analysis of WRTDS results	61
Table 16	Variables used in WRTDS contour plots: <code>plotContours</code> and <code>plotDiffContours</code>	62
Table 17	Variables used in WRTDS <code>plotConcQSmooth</code> and/or <code>plotConcTimeSmooth</code> functions	63

1 Introduction to Exploration and Graphics for RivEr Trends (EGRET)

EGRET includes statistics and graphics for streamflow history, water quality trends, and the modeling algorithm Weighted Regressions on Time, Discharge, and Season (WRTDS). **Please see the official EGRET manual:** (link to download) **for more information on the EGRET package.** For information on getting started in R, downloading and installing the package, see section 9.

The best way to learn about the WRTDS approach and to see examples of its application to multiple large data sets is to read two journal articles. They are available, for free, from the journals in which they were published. The first relates to nitrate and total phosphorus data for 9 rivers draining to Chesapeake Bay. The URL is (1): <http://onlinelibrary.wiley.com/doi/10.1111/j.1752-1688.2010.00482.x/full>. The second is an application to nitrate data for 8 monitoring sites on the Mississippi River or its major tributaries (2). The URL is: <http://pubs.acs.org/doi/abs/10.1021/es201221s>

This vignette assumes that you understand the concepts underlying WRTDS, and reading at least the first of these papers is necessary for that understanding. The method has been enhanced beyond what was previously published, and it now properly handles censored data by using survival regression rather than ordinary regression. The details of this change are in a report on Chesapeake Bay river input trends (3):<http://pubs.usgs.gov/sir/2012/5244/>. The specific enhancements for handling censored data are on pages 9-11 of the report.

This vignette guides you through the major functions provided by the EGRET package. The package `dataRetrieval` is required for importing data in an EGRET-friendly format. The `dataRetrieval` package and installation instructions are at: <https://github.com/USGS-R/dataRetrieval>. Installing `dataRetrieval` will provide a vignette similar to this document, with complete working examples of the main `dataRetrieval` functions. This document assumes you are familiar with the `dataRetrieval` package. Further details are in the user guide available on gitHub: <https://github.com/USGS-R/EGRET/raw/master/inst/doc/EGRET.pdf>

2 EGRET Workflow

Subsequent sections of this vignette discuss the EGRET workflow steps in greater detail. This section provides a handy cheat sheet for diving into an EGRET analysis. The first example is for a flow history analysis:

```
library(dataRetrieval)
library(EGRET)

# Flow history analysis

#####
# Gather discharge data:
siteID <- "01491000" #Choptank River at Greensboro, MD
startDate <- "" # Get earliest date
endDate <- "" # Get latest date
Daily <- getDWDData(siteID, "00060", startDate, endDate)
# Gather site and parameter information:
# Here user must input some values for
# the default (interactive=TRUE)
INFO<- getMetaData(siteID, "00060")
INFO$shortName <- "Choptank River at Greensboro, MD"
#####

#####
# Check flow history data:
annualSeries <- makeAnnualSeries()
plotFlowSingle(istat=7, qUnit="thousandCfs")
plotSDLogQ()
plotQTimeDaily(qLower=1, qUnit=3)
plotFour(qUnit=3)
plotFourStats(qUnit=3)
#####

# modify this for your own computer file structure:
savePath<-"/Users/rhirsch/Desktop/"
saveResults(savePath)
```

The second workflow example is for a water quality analysis. It includes data retrieval, merging of water quality and streamflow data, running the WRTDS estimation, and various plotting functions available in the EGRET package.

```

library(dataRetrieval)
library(EGRET)

#####
# Gather discharge data:
siteID <- "01491000" #Choptank River at Greensboro, MD
startDate <- "" #Gets earliest date
endDate <- "2011-09-30"
# Gather sample data:
parameter_cd<-"00631" #5 digit USGS code
Sample <- getSampleData(siteID,parameter_cd,startDate,endDate)
#Gets earliest date from Sample record:
#This is just one of many ways to assure the Daily record
#spans the Sample record
startDate <- min(as.character(Sample$Date))
# Gather discharge data:
Daily <- getDVData(siteID,"00060",startDate,endDate)
# Gather site and parameter information:

# Here user must input some values for
# the default (interactive=TRUE)
INFO<- getMetaData(siteID,parameter_cd)
INFO$shortName <- "Choptank River at Greensboro, MD"

# Merge discharge with sample data:
Sample <- mergeReport()
#####

#####
# Check sample data:
boxConcMonth()
boxQTwice()
plotConcTime()
plotConcQ()
multiPlotDataOverview()
#####

#####
# Run WRTDS model:
modelEstimation()
#####

#####
#Check model results:

#Require Sample + INFO:
plotConcTimeDaily()
plotFluxTimeDaily()

```

```

plotConcPred()
plotFluxPred()
plotResidPred()
plotResidQ()
plotResidTime()
boxResidMonth()
boxConcThree()

#Require Daily + INFO:
plotConcHist()
plotFluxHist()

# Multi-line plots:
date1 <- "2000-09-01"
date2 <- "2005-09-01"
date3 <- "2009-09-01"
qBottom<-100
qTop<-5000
plotConcQSmooth(date1, date2, date3, qBottom, qTop,
                 concMax=2, qUnit=1)

q1 <- 10
q2 <- 25
q3 <- 75
centerDate <- "07-01"
yearEnd <- 2009
yearStart <- 2000
plotConcTimeSmooth(q1, q2, q3, centerDate, yearStart, yearEnd)

# Multi-plots:
fluxBiasMulti()

#Contour plots:
clevel<-seq(0,2,0.5)
maxDiff<-0.8
yearStart <- 2000
yearEnd <- 2010

plotContours(yearStart, yearEnd, qBottom, qTop,
             contourLevels = clevel, qUnit=1)
plotDiffContours(yearStart, yearEnd,
                 qBottom, qTop, maxDiff, qUnit=1)

# modify this for your own computer file structure:
savePath<-"/Users/rhirsch/Desktop/"
saveResults(savePath)

```


3 EGRET Dataframes and Units

The EGRET package uses 3 default dataframes throughout the calculations, analysis, and graphing. These dataframes are Daily (3.1), Sample (3.2), and INFO (3.3). EGRET uses entirely SI units to store the data, but for purposes of output, it can report results in a wide variety of units, which will be discussed in (3.4). To start our exploration, you must install the packages (check Section 9 for detailed instructions), and then open EGRET with the following command:

```
library(dataRetrieval)
library(EGRET)
```

3.1 Daily

The Daily dataframe initially is populated with columns generated by the dataRetrieval package (Table 1). After you run the WRTDS calculations by using the function `modelEstimation` (as will be described in section 6), additional columns are inserted (Table 2).

Table 1. Daily dataframe

ColumnName	Type	Description	Units
Date	Date	Date	date
Q	number	Discharge in m ³ /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	Qualifying 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 running average of Q	m ³ /s

Table 2. Columns added to Daily dataframe after running `modelEstimation`

ColumnName	Type	Description	Units
yHat	number	The WRTDS estimate of the log of concentration	numeric
SE	number	The WRTDS estimate of the standard error of yHat	numeric
ConcDay	number	The WRTDS estimate of concentration	mg/L
FluxDay	number	The WRTDS estimate of flux	kg/day
FNConc	number	Flow-normalized estimate of concentration	mg/L
FNFlux	number	Flow-normalized estimate of flux	kg/day

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.

3.2 Sample

The Sample dataframe initially is populated with columns generated by the `dataRetrieval` package (Table 3). After you run the WRTDS calculations using the `modelEstimation` function (as described in section 6), additional columns are inserted (Table 4):

Table 3. 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 concentration	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	cms
LogQ ¹	number	Natural logarithm of discharge	numeric

¹ Populated after calling `mergeReport`.

Table 4. Columns added to Sample dataframe after running `modelEstimation()`

ColumnName	Type	Description	Units
yHat ¹	number	estimate of the log of concentration	numeric
SE ¹	number	estimate of the standard error of yHat	numeric
ConcHat ¹	number	unbiased estimate of concentration	mg/L

¹ These estimates are “leave-one-out cross validation” estimates. See the EGRET Manual for more details.

3.3 INFO

The INFO dataframe stores information about the measurements, such as station name, parameter name, drainage area, and so forth. There can be many additional, optional columns, but the columns in Table 5 are required to initiate the EGRET analysis. After you run the WRTDS calculations (as described in section 6), additional columns (Table 6) are automatically inserted into the INFO dataframe (see the EGRET Manual for complete description of each term):

Table 5. INFO dataframe

ColumnName	Type	Description
shortName	string	Name of site, suitable for use in graphical headings
staAbbrev	string	Abbreviation for station name, used in saveResults
paramShortName	string	Name of constituent, suitable for use in graphical headings
constitAbbrev	string	Abbreviation for constituent name, used in saveResults
drainSqKm	numeric	Drainage area in km ²
paStart ¹	integer (1-12)	Starting month of period of analysis
paLong ¹	integer (1-12)	Length of period of analysis in months

¹ Inserted with the `setPA` function.

Table 6. INFO dataframe after running `modelEstimation()`

ColumnName	Description	Units
bottomLogQ	Lowest discharge in prediction surfaces	numeric
stepLogQ	Step size in log discharge in prediction surfaces	numeric
nVectorLogQ	Number of steps in discharge, prediction surfaces	numeric
bottomYear	Starting year in prediction surfaces	numeric
stepYear	Step size in years in prediction surfaces	numeric
nVectorYear	Number of steps in years in prediction surfaces	numeric
windowY	Half-window width in the time dimension	years
windowQ	Half-window width in the log discharge dimension	numeric
windowS	Half-window width in the seasonal dimension	years
minNumObs	Minimum number of observations for regression	integer
minNumUncen	Minimum number of uncensored observations	integer

3.4 Units

EGRET uses entirely SI units to store the data, but for purposes of output, it can report results in a wide variety of units. The defaults are mg/L for concentration, cubic meters per second (m^3/s) for discharge, kg/day for flux, and km^2 for drainage area. When discharge values are imported from USGS web services (using the `dataRetrieval` package), they are automatically converted from cubic feet per second (cfs) to cms unless the argument "convert" in function `getDVDData` is set to FALSE. This can cause confusion if you are not careful.

For all functions that provide output, you can define two arguments to set the output units: `qUnit` and `fluxUnit`. `qUnit` and `fluxUnit` are defined by either a numeric code or name. You can call two functions that can be called to see the options: `printqUnitCheatSheet` and `printFluxUnitCheatSheet`.

`printqUnitCheatSheet()`

The following codes apply to the `qUnit` list:

```
1 = cfs   ( Cubic Feet per Second )
2 = cms   ( Cubic Meters per Second )
3 = thousandCfs ( Thousand Cubic Feet per Second )
4 = thousandCms ( Thousand Cubic Meters per Second )
```

When a function has an input argument `qUnit`, you can define the discharge units that will be used in the figure or table that is generated by the function with the index (1-4) as shown above. Base your choice on the units that are customary for your intended audience, but also so that the discharge values don't have

too many digits to the right or left of the decimal point.

```
printFluxUnitCheatSheet()
```

The following codes apply to the fluxUnit list:

```
1 = poundsDay ( pounds/day )
2 = tonsDay ( tons/day )
3 = kgDay ( kg/day )
4 = thousandKgDay ( thousands of kg/day )
5 = tonsYear ( tons/year )
6 = thousandTonsYear ( thousands of tons/year )
7 = millionTonsYear ( millions of tons/year )
8 = thousandKgYear ( thousands of kg/year )
9 = millionKgYear ( millions of kg/year )
10 = billionKgYear ( billions of kg/year )
11 = thousandTonsDay ( thousands of tons/day )
12 = millionKgDay ( millions of kg/day )
```

When a function has an input argument fluxUnit, you can define the flux units with the index (1-12) as shown above. Base the choice on the units that are customary for your intended audience, but also so that the flux values don't have too many digits to the right or left of the decimal point. Tons are always "short tons" and not "metric tons".

4 Flow History

This section describes functions included in the EGRET package that provide a variety of table and graphical outputs for examining discharge statistics based on time-series smoothing. These functions are designed for studies of long-term change and work best for daily discharge data sets of 50 years or longer. This type of analysis might be useful for studying issues such as the influence of land use change, water management change, or climate change on discharge conditions. This includes potential impacts on average discharges, high discharges, and low discharges, at annual time scales as well as seasonal or monthly time scales.

At this point it is assumed that you can load the daily discharge record into R, create the Daily dataframe, and enter the required meta-data into the INFO dataframe. If not, see the dataRetrieval vignette:

```
vignette("dataRetrieval")
```

We will walk through an example from Columbia River at The Dalles, OR.

```
siteID <- "14105700"
startDate <- ""
endDate <- ""

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

There are 49717 data points, and 49718 days.

discharge data jumps from 2014-05-12 to 2014-05-14

INFO <- getMetaData(siteID, "", interactive=FALSE)
INFO$shortName <- "Columbia River at The Dalles, OR"
```

You first must determine the period of analysis to use (PA). What is the period of analysis? If you want to examine your data set as a time series of water years, then the period of analysis is October through September. If you want to examine the data set as calendar years then the period of analysis is January through December. You might want to examine the winter season, which you could define as December through February, then those 3 months become the period of analysis. The only constraints on the definition of a period of analysis are these: it must be defined in terms of whole months; it must be a set of contiguous months (like March-April-May), and have a length that is no less than 1 month and no more than 12 months. Define the PA by using two arguments: `paLong` and `paStart`. `paLong` is the length of the PA, and `paStart` is the first month of the PA. Table 7 summarizes `paLong` and `paStart`.

Table 7. Period of Analysis Information

Period of Analysis	paStart	paLong
Calendar Year	1	12
Water Year	10	12
Winter	12	3
September	9	1

To set a period running from December through February:

```
INFO <- setPA(paStart=12, paLong=3)
```

To set the default value (water year):

```
INFO <- setPA()
```

The next step is to create the annual series of discharge statistics. These will be stored in a matrix called `annualSeries` that contain the statistics described in table 8. The statistics are based on the period of analysis set with the `setPA` function.

Table 8. Index of discharge statistics information

istat	Name
1	1-day minimum discharge
2	7-day minimum discharge
3	30-day minimum discharge
4	median discharge
5	mean discharge
6	30-day maximum discharge
7	7-day maximum discharge
8	1-day maximum discharge

To create the `annualSeries` matrix, using the function `makeAnnualSeries`:

```
annualSeries <- makeAnnualSeries()
```

After you create the `annualSeries` matrix, you can generate the plots of any of the stored statistics by using the `plotFlowSingle` function.

4.1 Plotting Options

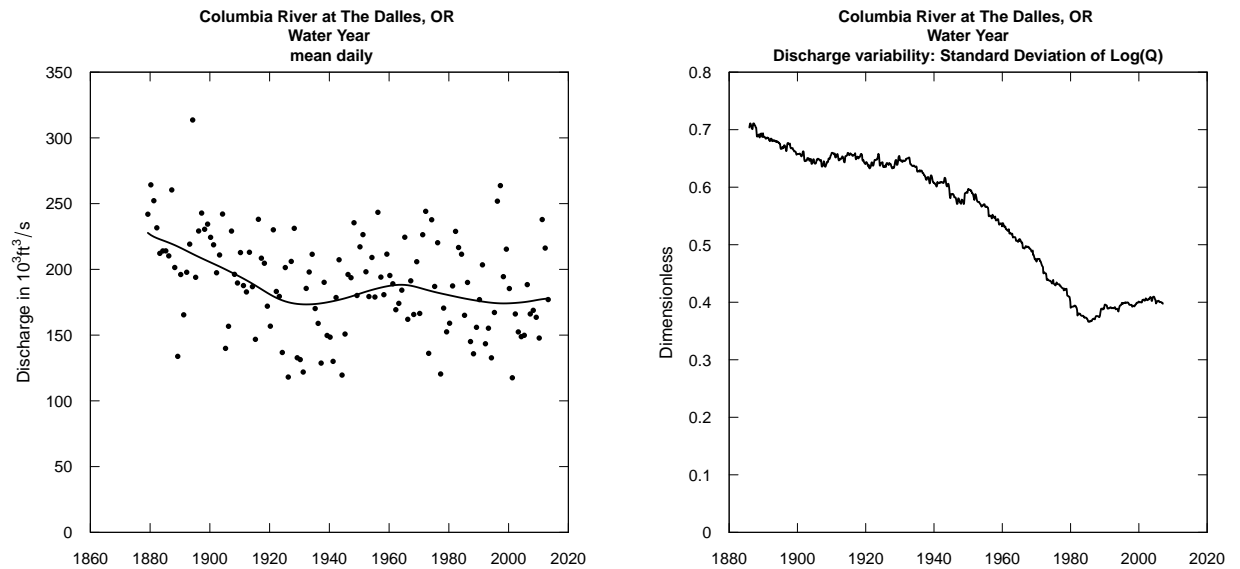
This section shows examples of the available plots appropriate for studying discharge history once the `annualSeries` has been created. The plots here use the default variable input options. For any function, you can get a complete list of input variables (as described in section 10.1) in a help file by typing a `?` before the function name in the R console. The EGRET manual has more detailed information for each plot type ([link to download](#)). Finally, see section 12 for information on saving plots.

The simplest way to look at these time series is with the function `plotFlowSingle`. The statistic index (`istat`) must be defined by the user, but for all other arguments there are default values so the user isn't required to specify anything else. To see a list of these optional arguments and other information about the function, type `?plotFlowSingle` in the R console. All of the graphs in `plotFlowSingle`,

`plotFourStats`, and all but one of the graphs in `plotFour`, show both the individual annual values of the selected discharge statistic (e.g. the annual mean or 7-day minimum), but they also show a curve that is a smooth fit to those data. The curve is a LOWESS (locally weighted scatterplot smooth). The algorithm for computing it is provided in (4):<http://pubs.usgs.gov/sir/2012/5151/> (pages 6 and 7). The default is that the annual values of the selected discharge statistics are smoothed with a “half-window width” of 30 years. The smoothing window is an optional user-defined option.

`plotSDLogQ` produces a graphic of the running standard deviation of the log of daily discharge over time to visualize how variability of daily discharge is changing over time. By using the standard deviation of the log discharge the statistic becomes dimensionless. The standard deviation plot is a way of looking at variability quite aside from average values, so, in the case of a system where discharge might be increasing over a period of years, this graphic provides a way of looking at the variability relative to that changing mean value. The standard deviation is much like a coefficient of variation, but it has sample properties that make it a smoother measure of variability. People often comments about how things like urbanization or enhanced greenhouse gases in the atmosphere are bringing about an increase in variability, and this analysis is one way to explore that idea. `plotFour`, `plotFourStats`, and `plot15` are all designed to plot several graphs from the other functions in a single figure.


```
plotFlowSingle(istat=5,qUnit="thousandCfs")
plotSDLogQ()
```



(a) `plotFlowSingle(istat=5,qUnit='thousandCfs')`

(b) `plotSDLogQ()`

Figure 1. Plots of discharge statistics

Here is an example of looking at mean daily discharge for the full water year and then looking at mean daily discharge for the winter season only for the Merced River at Happy Isles Bridge in Yosemite National Park in California. First, we look at the mean daily discharge for the full year (after having read in the data and metadata):

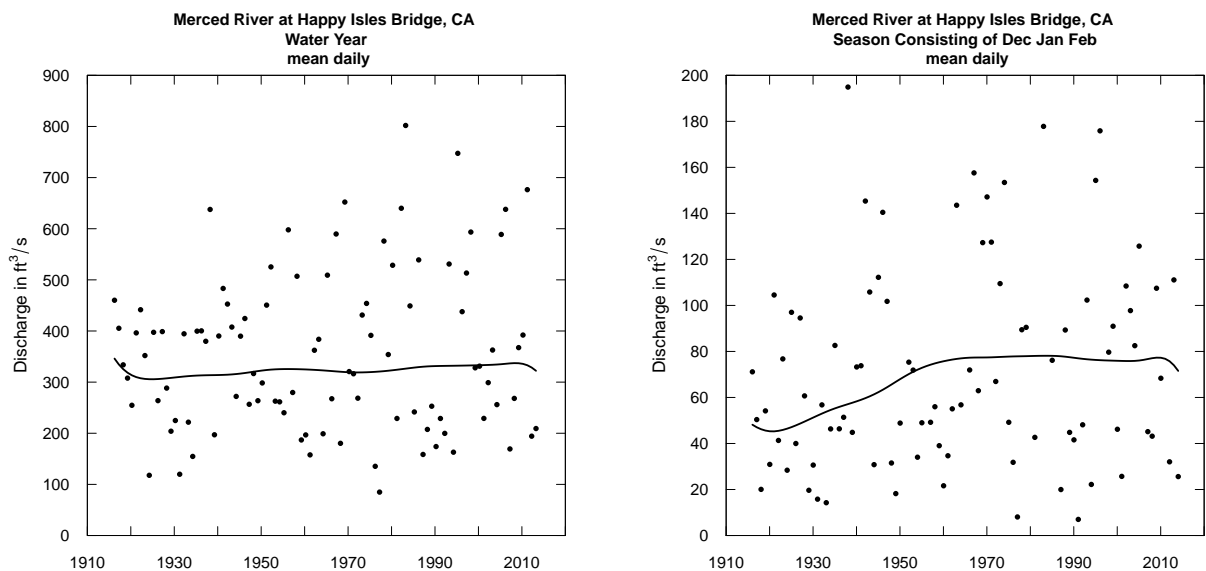
```
sta<-"11264500"
Daily <-getDVDData(sta,"00060",StartDate="",EndDate="")

There are 36122 data points, and 36122 days.

INFO <- getMetaData(sta,"",interactive=FALSE)
INFO$shortName <- "Merced River at Happy Isles Bridge, CA"
INFO <- setPA()
annualSeries <- makeAnnualSeries()
plotFlowSingle(istat=5)

# Then, we can run the same function, but first set
# the pa to start in December and only run for 3 months.

INFO<-setPA(paStart=12,paLong=3)
annualSeries<-makeAnnualSeries()
plotFlowSingle(istat=5,qMax=200)
```



(a) Water Year

(b) Dec-Feb

Figure 2. Merced River Winter Trend

What these figures show us is that on an annual basis there is very little indication of a long-term trend in mean discharge, but for the winter months there is a pretty strong indication of an upward trend. This

could well be related to the climate warming in the Sierra Nevada, resulting in a general increase in the ratio of rain to snow in the winter and more thawing events.

```
plotFour(qUnit=3)
```

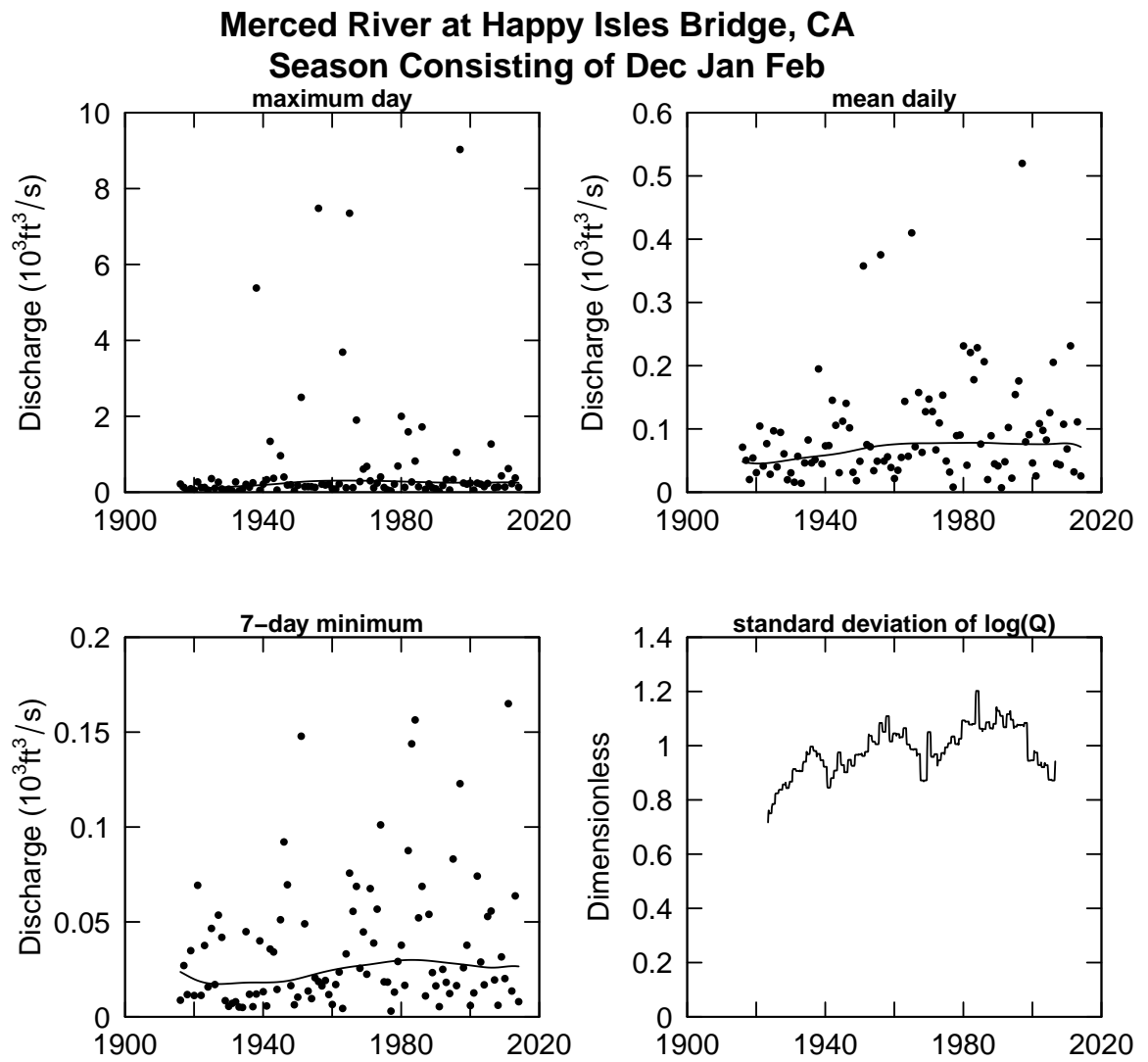


Figure 3. plotFour(qUnit=3)

```
plotFourStats(qUnit=3)
```

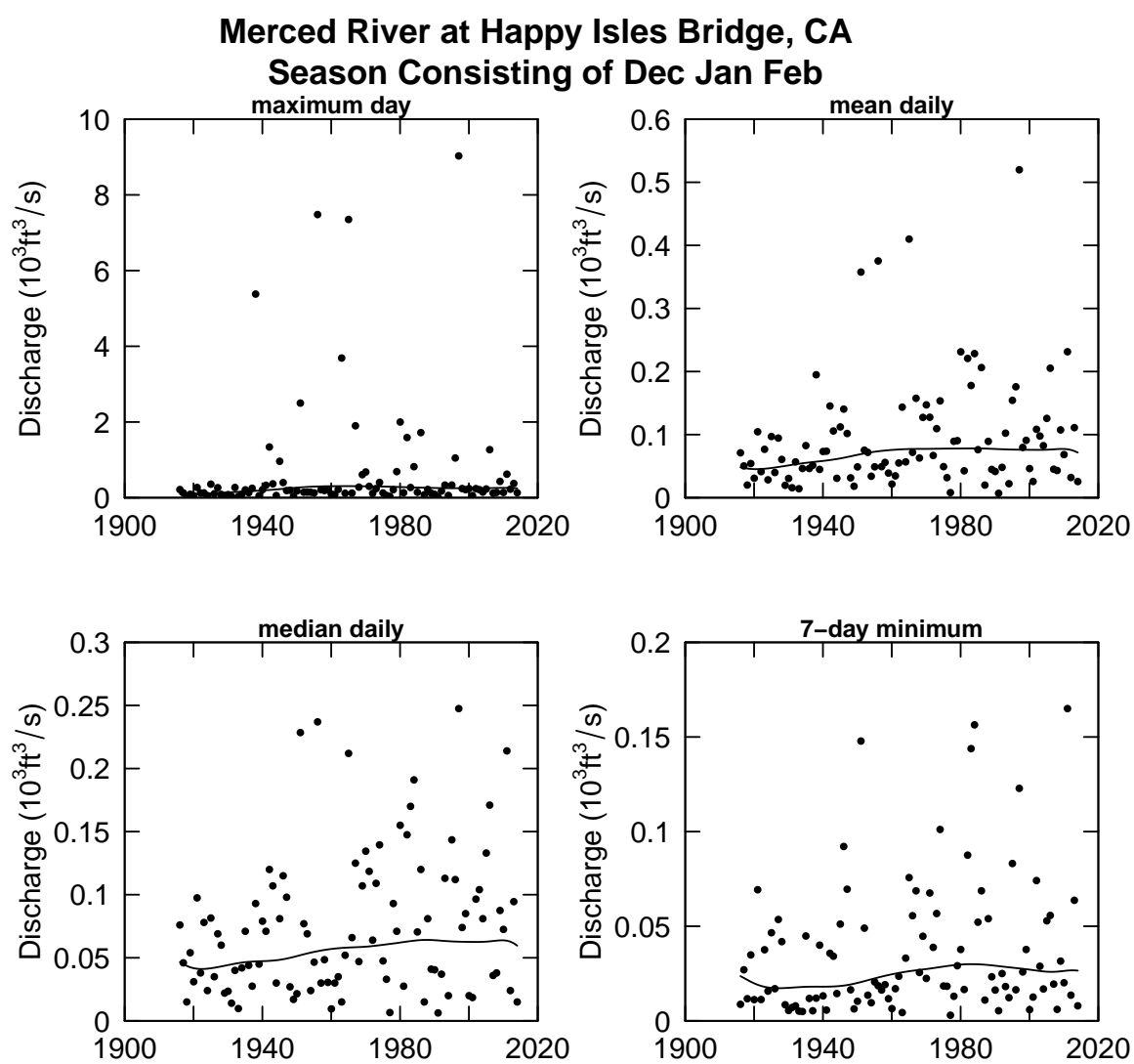


Figure 4. plotFourStats(qUnit=3)

```

sta<-"05474500"
Daily <-getDVData(sta,"00060",StartDate="",EndDate="")

There are 49854 data points, and 49854 days.

INFO <- getMetaData(sta,"",interactive=FALSE)
INFO$shortName <- "Mississippi River at Keokuk Iowa"
INFO <- setPA()

plotQTimeDaily(qUnit=3,qLower=300)

```

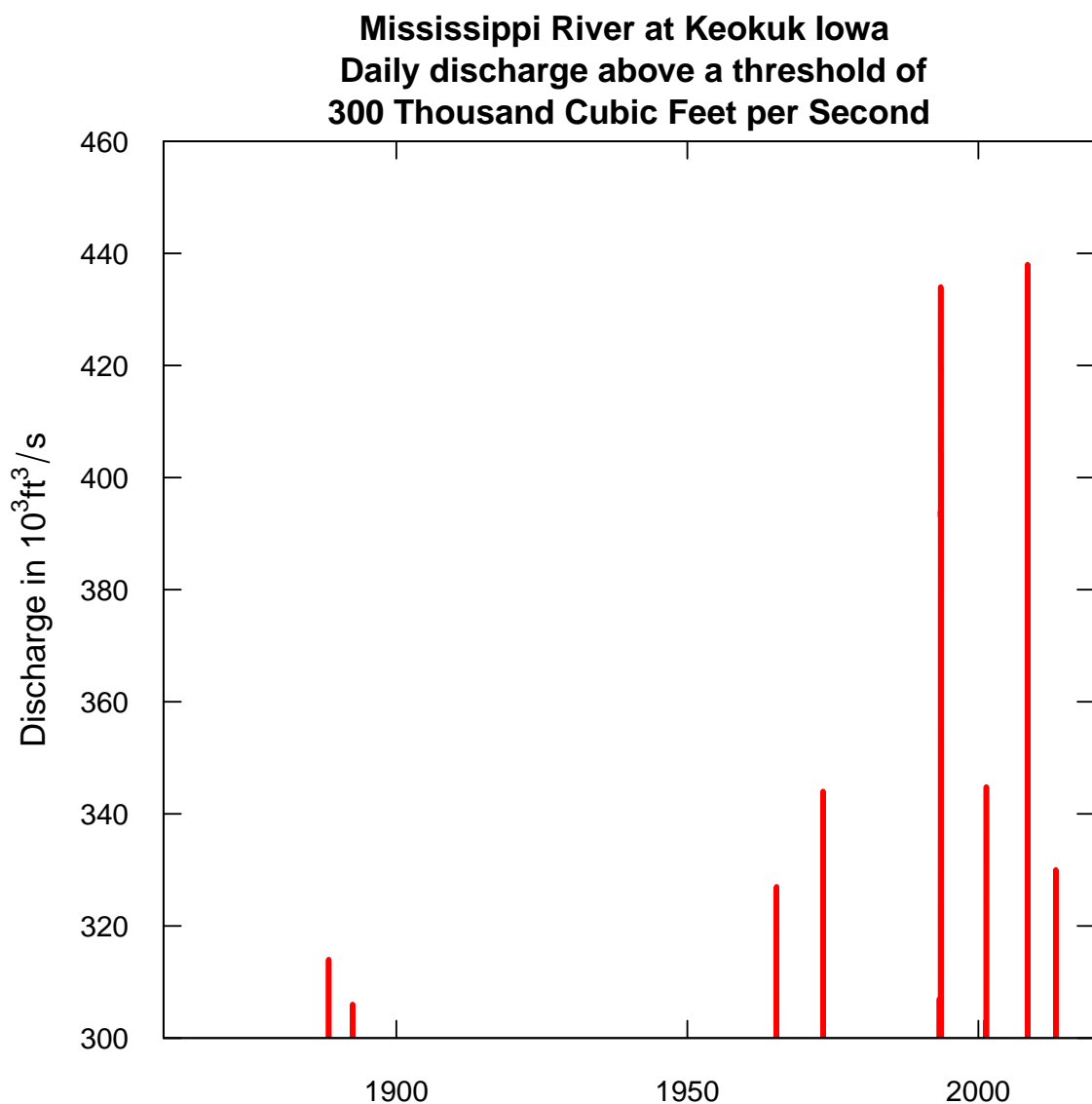


Figure 5. Mississippi River at Keokuk Iowa

`plotQTimeDaily` is simply a time series plot of discharge. But, it is most suited for showing events above some discharge threshold. In the simplest case, it can plot the entire record, but given the line weight and use of an arithmetic scale it will primarily provide a visual focus on the higher values.

The example shown in Figure 5 illustrates a very long record with a long gap of more than 60 years with no discharges above 300,000 cfs, followed by the last 50 years with at least 5 events above that threshold. `plotQTimeDaily` requires `startYear` and `endYear`, along with some other optional arguments (see `?plotQTimeDaily` for more details).

4.2 Table Options

Sometimes it is easier to consider results in table formats rather than graphically. Similar to the function `plotFlowSingle`, the `printSeries` will print the requested discharge statistics (Table 8), as well as return the results in a dataframe. A small sample of the output is printed below.

```
annualSeries<-makeAnnualSeries()  
seriesResult <- printSeries(istat=3, qUnit=3)
```

```
Mississippi River at Keokuk Iowa  
Water Year  
  30-day minimum  
  Thousand Cubic Feet per Second  
year   annual   smoothed  
      value     value  
1879    22.6     30.1  
1880    31.7     28.7  
1881    23.0     27.5  
...  
2011    51.0     32.4  
2012    34.3     32.1  
2013    16.2     31.8
```

Another way to look at the results is to consider how much the smoothed values change between various pairs of years. These changes can be represented in four different ways.

- As a change between the first and last year of the pair, expressed in the discharge units selected.
- As a change between the first and last year of the pair, expressed as a percentage of the value in the first year

- As a slope between the first and last year of the pair, expressed in terms of the discharge units per year.
- As a slope between the first and last year of the pair, expressed as a percentage change per year (a percentage based on the value in the first year).

Another argument can be very useful in this function: `yearPoints`. In the default case, the set of years that are compared are at 5 year intervals along the whole data set. If the data set was quite long this can be a daunting number of comparisons. For example, in an 80 year record, there would be 136 such pairs. Instead, we could look at changes between only 3 year points: 1890, 1950, and 2010:

```
annualSeries <- makeAnnualSeries()
tableFlowChange(istat=3, qUnit=3, yearPoints=c(1890, 1950, 2010))
```

```
Mississippi River at Keokuk Iowa
Water Year
30-day minimum
```

Streamflow Trends						
time span			change	slope	change	slope
			10 ³ cfs	10 ³ cfs/yr	%	%/yr
1890	to	1950	0.54	0.0091	2.4	0.04
1890	to	2010	9.4	0.078	41	0.34
1950	to	2010	8.8	0.15	38	0.63

NULL

See section 11 for instructions on converting an R dataframe to a table in Microsoft.

5 Summary of Water Quality Data (without using WRTDS)

Before you run the WRTDS model, it is helpful to examine the measured water quality data graphically to better understand its behavior, identify possible data errors, and visualize the temporal distribution of the data (identify gaps). It is always best to clear up these issues before moving forward.

The examples below use the Choptank River at Greensboro, MD. The Choptank River is a small tributary of the Chesapeake Bay. Inorganic nitrogen (nitrate and nitrite) has been measured from 1979 onward. First, we need to load the discharge and nitrate data into R. Before we can graph or use it for WRTDS anal-

ysis, we must bring the discharge data into the Sample dataframe. We do this by using the `mergeReport` function which merges the discharge information and also provides a compact report about some major features of the data set.

```
siteID <- "01491000" #Choptank River at Greensboro, MD
startDate <- "1979-10-01"
endDate <- "2011-09-30"
param<-"00631"
Daily <- getDVDData(siteID, "00060", startDate, endDate)
INFO<- getMetaData(siteID, param, interactive=FALSE)
INFO$shortName <- "Choptank River"

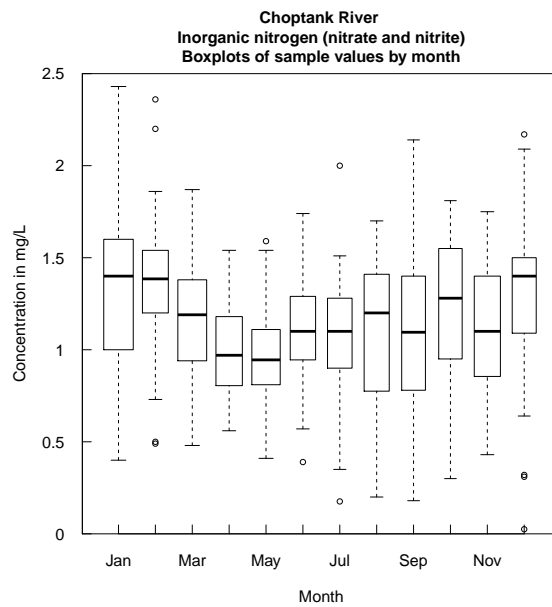
Sample <- getSampleData(siteID, param, startDate, endDate)
Sample <- mergeReport()
```

5.1 Plotting Options

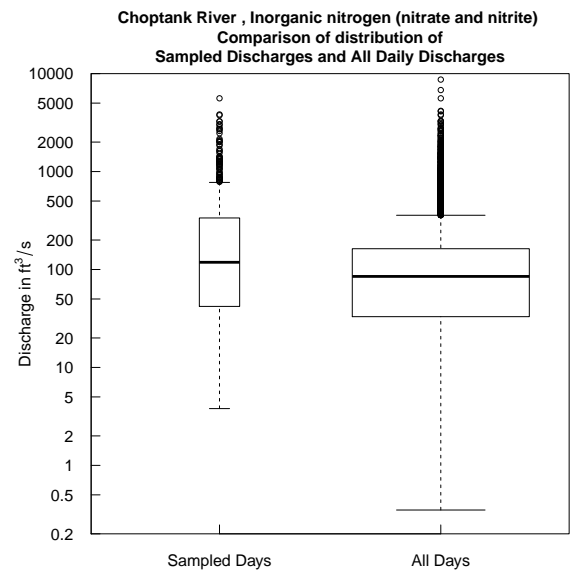
This section shows examples of the available plots appropriate for analyzing data prior to performing a WRTDS analysis. The plots here use the default variable input options. For any function, you can get a complete list of input variables in a help file by typing a `?` before the function name in the R console. See section 10.2 for information on the available input variables for these plotting functions.

Note that for any of the plotting functions that show the sample data, if a value in the data set is a non-detect (censored), it is displayed on the graph as a vertical line. The top of the line is the reporting limit and the bottom is either zero, or if the graph is plotting log concentration values, or the minimum value on the y-axis. This line is an “honest” representation of what we know about that observation and thus we need not become involved in using a statistical model to fill in what we don’t know.

```
boxConcMonth()  
boxQTwice(qUnit=1)
```



(a) boxConcMonth()

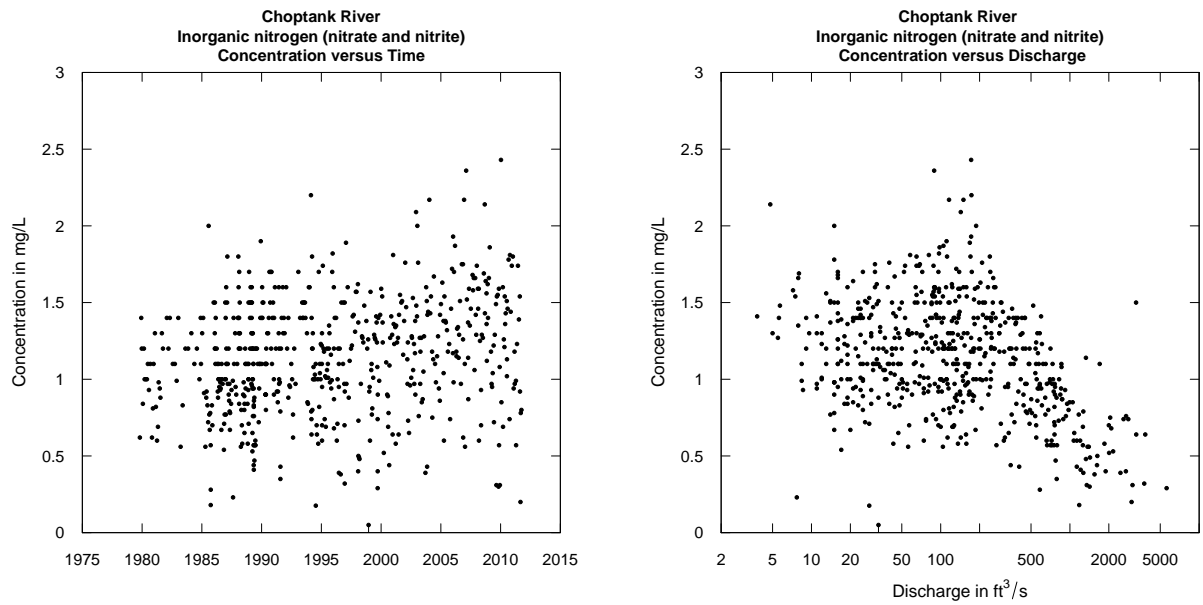


(b) boxQTwice(qUnit=1)

Figure 6. Concentration box plots

Note that the statistics to create the boxplot in `boxQTwice` are performed after the data are log-transformed.

```
plotConcTime()  
plotConcQ(qUnit=1)
```



(a) plotConcTime()

(b) plotConcQ()

Figure 7. The relation of concentration vs time or discharge

It is interesting to note in Figure 7 the change in the convention for rounding of data values that occurred around 1995.

```
plotFluxQ(fluxUnit=4)
```

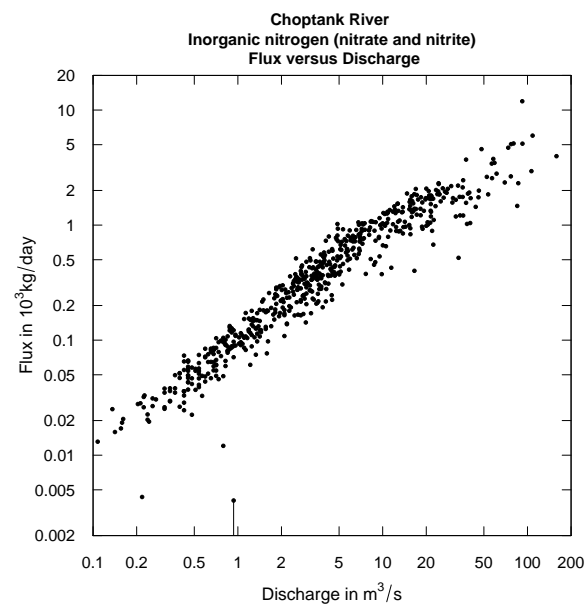


Figure 8. The relation of flux vs discharge

The `plotFluxQ` (Figure 8) function only plots in a log-log scale.

```
multiPlotDataOverview(qUnit=1)
```

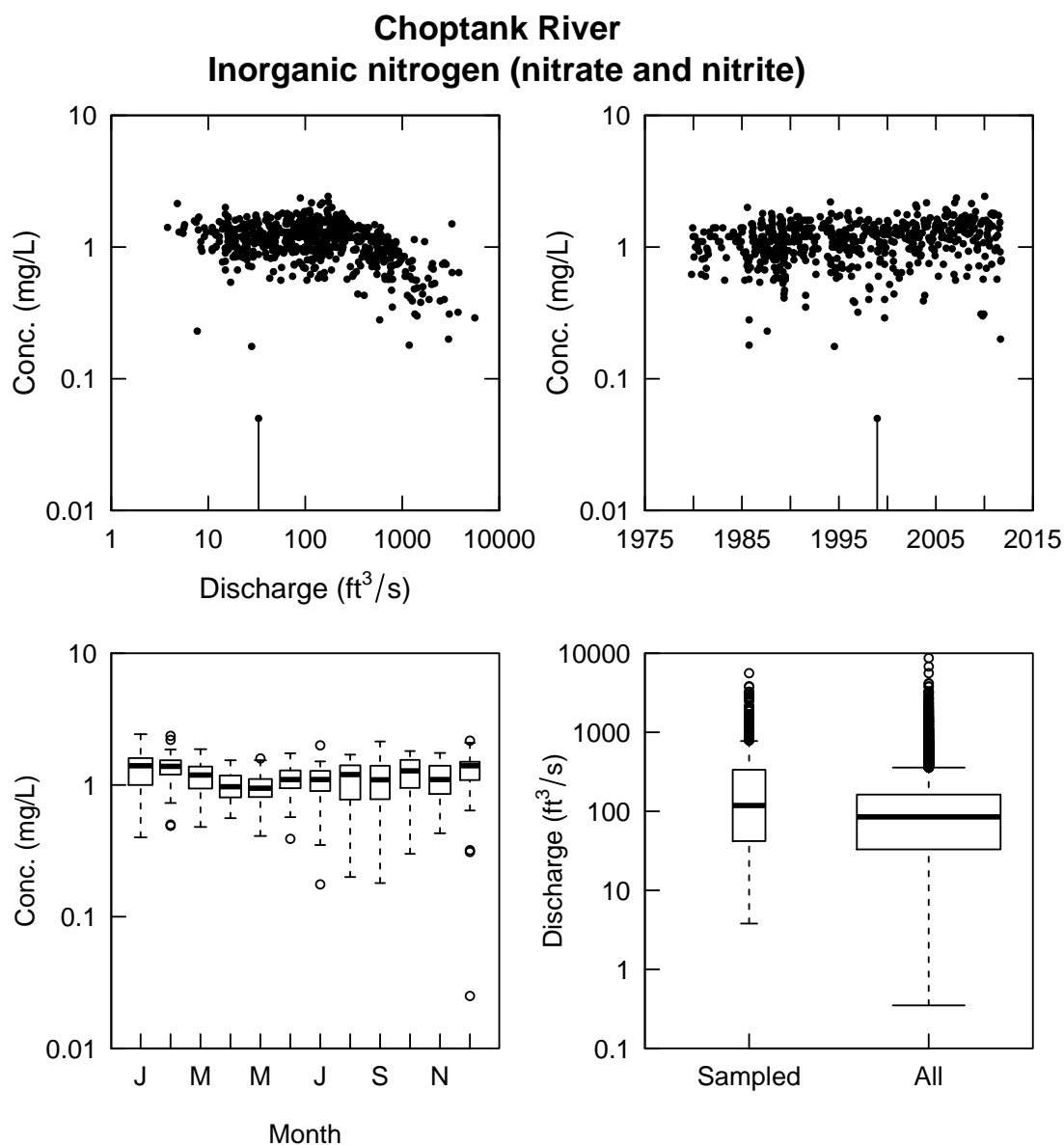


Figure 9. multiPlotDataOverview(qUnit=1)

The multiPlotDataOverview (Figure 9) function uses a log scale as default. To change the concentration axes to an arithmetic scale, use logScaleConc=FALSE in the multiPlotDataOverview function call.

5.2 Table Options

Another useful tool for checking the data before running the WRTDS estimations is `flowDuration`. This is a utility function that can help you define the discharge ranges that we want to explore. It prints out key points on the discharge duration curve. Define the points for a particular part of the year using the "centerDate" and "span" arguments, although the points can be defined for the entire year (default).

```
flowDuration(qUnit=1)
```

```
Flow Duration for Choptank River
```

```
Flow duration is based on full year
```

```
Discharge units are Cubic Feet per Second
```

min	5%	10%	25%	50%	75%	90%
0.35	12.00	16.00	33.00	85.00	163.00	290.00
95%	max					
462.00	8700.00					

```
flowDuration(qUnit=1, centerDate="09-30", span=30)
```

```
Flow Duration for Choptank River
```

```
Flow duration period is centered on September 30
```

```
And spans the period from August 31 To October 30
```

```
Discharge units are Cubic Feet per Second
```

min	5%	10%	25%	50%	75%	90%	95%
2.5	8.8	11.0	15.0	27.0	67.0	138.0	223.0
max							
5600.0							

6 Weighted Regressions on Time, Discharge and Season (WRTDS)

Analysis

WRTDS creates a model of the behavior of concentration as a function of three components: time trend, discharge and season. You can use WRTDS to estimate annual or seasonal mean concentrations and fluxes as well as describe long-term trends in the behavior of the system. In this section, we will step through

the process required for a WRTDS analysis. Section (7) provides details about the available methods for viewing and evaluating the model results.

Once you have looked at your data using the tools described in section 5, and have determined there is sufficient representative data, it is time to run the WRTDS model. Assuming you are using the defaults, with dataframes called Daily, Sample, and INFO, the `modelEstimation` function will run the WRTDS modeling algorithm:

```
modelEstimation()
```

Details of the options available when running `modelEstimation` can be found in Section 10.3. This function is slow, and shows the progress in percent complete. See the references and manual for more information. It's important to understand that this is the one function that will globally change your Daily, Sample, and INFO dataframes. It also creates a new matrix "surfaces", and a new dataframe "AnnualResults". It is unusual R programming behavior to create global variables, but global variables were chosen to make it easy for you.

Finally, it is a good idea to save your results because of the computational time that has been invested in producing these results. The workspace is saved to a directory that you designate `savePath` and the file name is determined by the abbreviations for station and constituent that were required entries when the `getMetaData` function was used. The command for saving the workspace is:

```
savePath <- "C:/Users/egretUser/WRTDS_Output/" #An example directory name  
saveResults(savePath)
```

This will now save all of the objects in your workspace. If you have saved workspaces from R versions earlier than 3.0, a warning will appear when you open them in R 3.0 (or later). Re-saving the workspace using R 3.0 (or later) should get rid of the warning.

Using `saveResults`, the workspace is saved with `INFOstaAbbrevandINFOconstitAbbrev` as the filename (separated by a period), and the extension `.RData`. So, if `staAbbrev` was "Chop" and the `constitAbbrev` was "NO3" the file name would be "Chop.NO3.RData". To load the data in some future session the commands could be:

```
loadPath <- "C:/Users/egretUser/WRTDS_Output/"  
staAbbrev <- "Chop"
```

```
constitAbbrev <- "NO3"  
pathToFile <- paste(loadPath, staAbbrev, ".",  
                    constitAbbrev, ".RData", sep="")  
load(pathToFile)
```

7 WRTDS Results

At this point (after having run `modelEstimation`) we can start considering how to view the annual averages for the variables that have been calculated. See section 10.4 for common input variables for these functions. Additionally, check the help files (in the R console, type `?` followed by the function name).

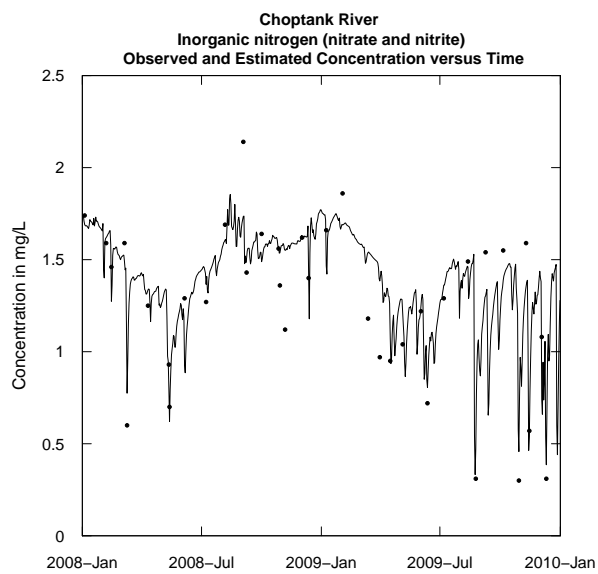
7.1 Plotting Options

Check the help files or manual for more details on the following functions. See section 12 for information on saving plots. In these examples, we will return to looking at the data in the water year by using the `setPA` function. Most plotting functions will use the period of analysis information in the `INFO` dataframe to determine what data is plotted. There are only four graph or table functions that don't allow the user to specify a Period of Analysis (PA). These are: `plotContour`, `plotDiffContour`, `plotConcTimeSmooth`, `plotConcQSmooth`.

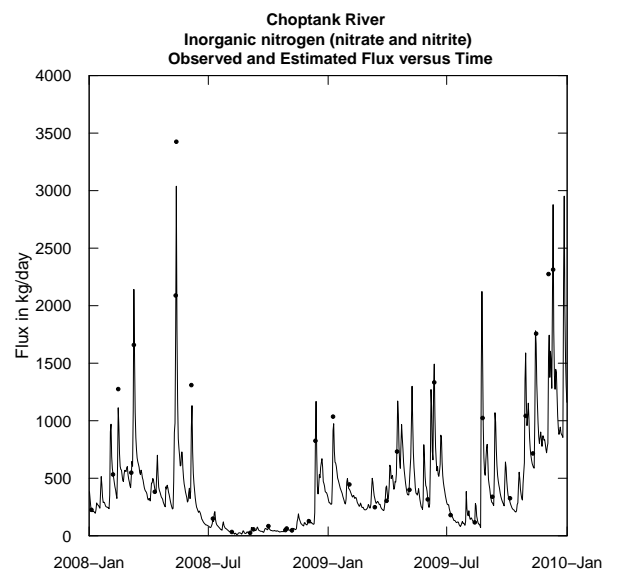

```
# Return to water year:
INFO <- setPA()

yearStart <- 2008
yearEnd <- 2010

plotConcTimeDaily(yearStart, yearEnd)
plotFluxTimeDaily(yearStart, yearEnd)
```



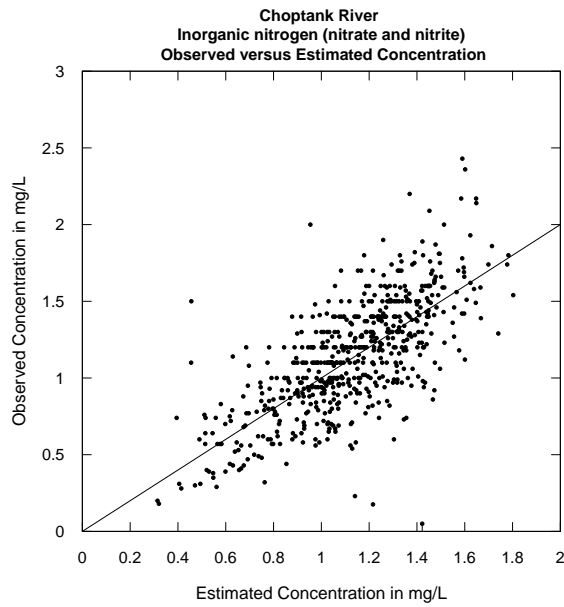
(a) plotConcTimeDaily(startYear=2008, endYear=2010)



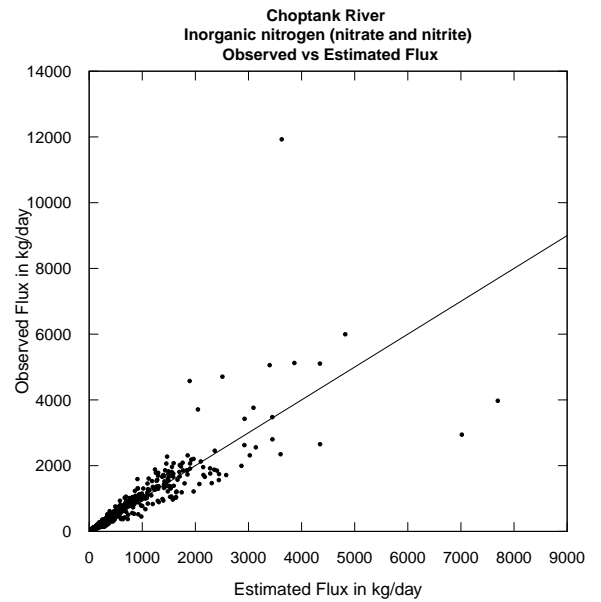
(b) plotFluxTimeDaily(startYear=2008, endYear=2010)

Figure 10. Concentration and flux vs time

```
plotConcPred()  
plotFluxPred()
```



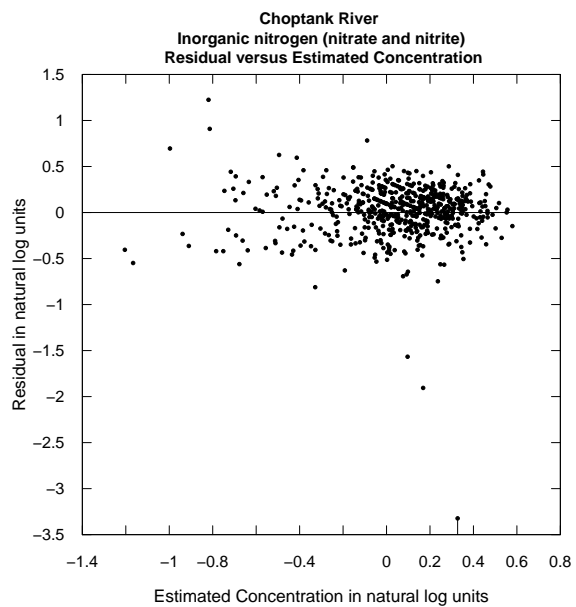
(a) plotConcPred()



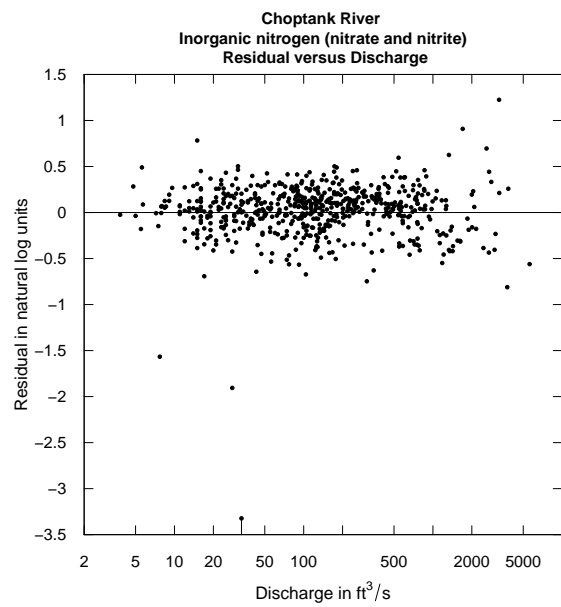
(b) plotFluxPred()

Figure 11. Concentration and flux predictions

```
plotResidPred()  
plotResidQ(qUnit=1)
```



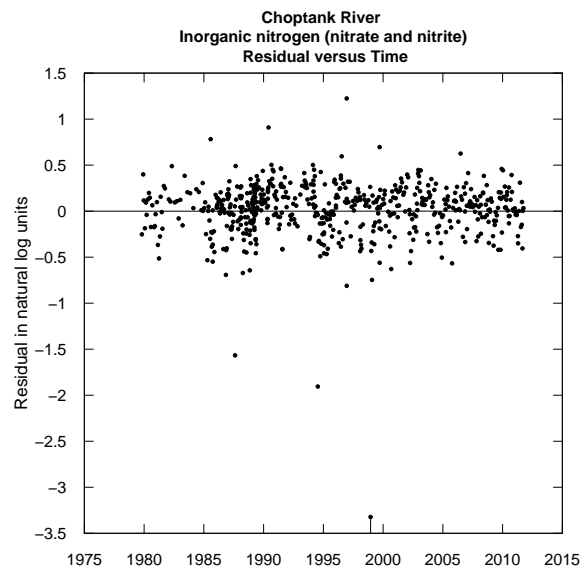
(a) plotResidPred()



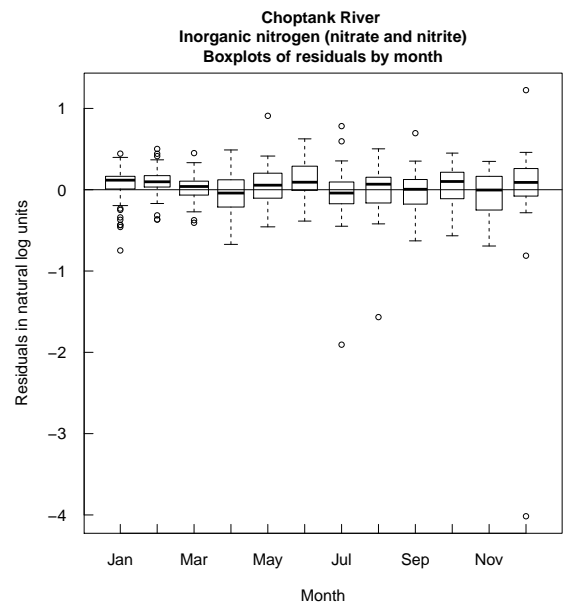
(b) plotResidQ(qUnit=1)

Figure 12. Residuals

```
plotResidTime()  
boxResidMonth()
```



(a) plotResidTime()



(b) boxResidMonth()

Figure 13. Residuals with respect to time

boxConcThree ()

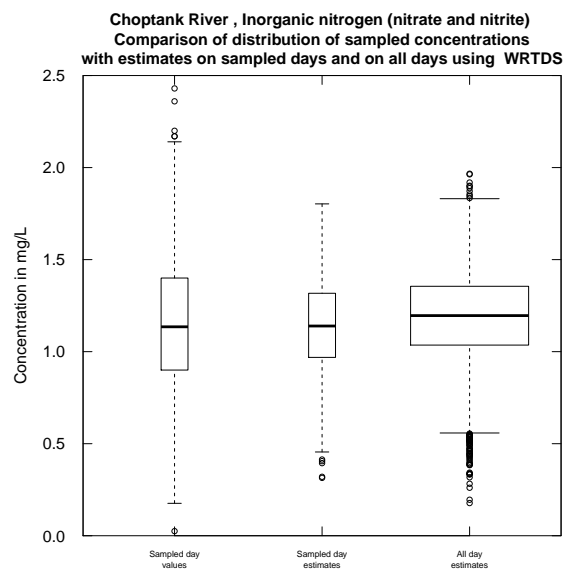
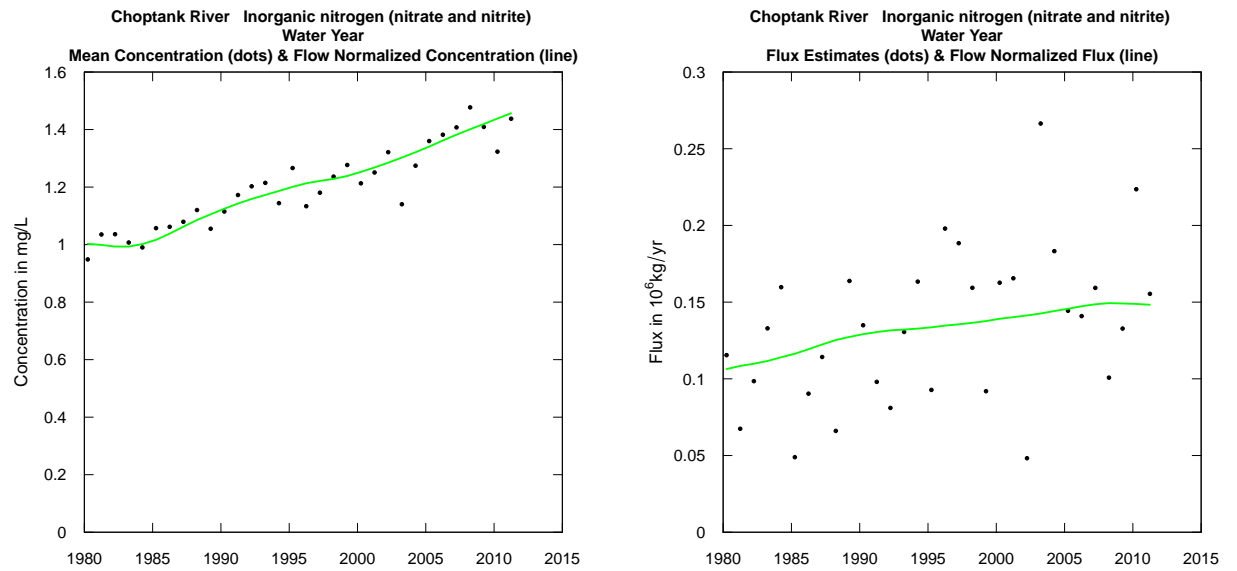


Figure 14. Default boxConcThree()

```
plotConcHist ()
plotFluxHist ()
```



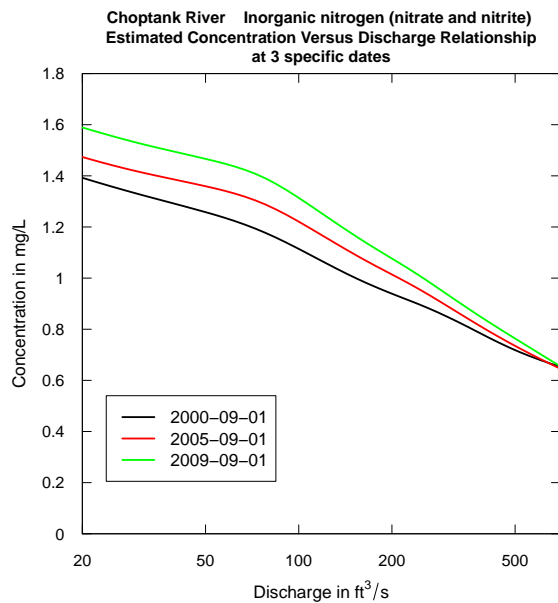
(a) plotConcHist()

(b) plotFluxHist()

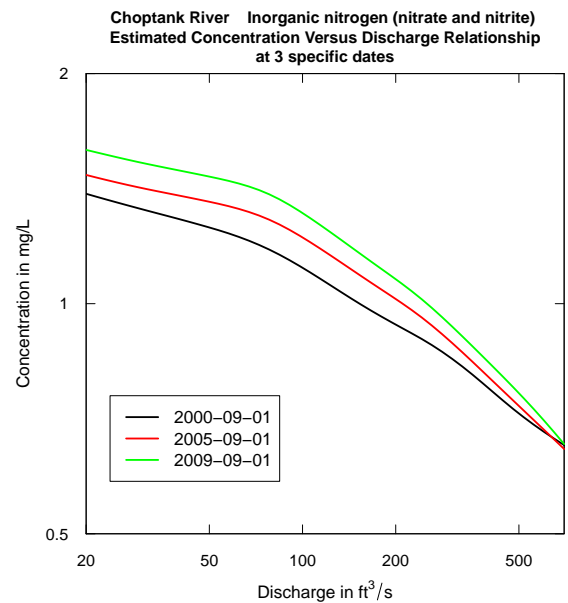
Figure 15. Concentration and flux history

Figures 16 and 17 contain legends. The placement of the legend is controlled by `legendLeft` and `legendTop`. If both are set to 0 (the default values), the legend is placed near the lower left corner of the graphic. Otherwise, the value specified for `legendLeft` places the left edge of the legend, and `legendTop` specifies the top edge of the legend. The units for `legendLeft` and `legendTop` are discharge (in units specified by `qUnit`) and concentration, respectively. The legend can also be turned off with `printLegend=FALSE`. These are also functions that do not recognize the period of analysis in the INFO data frame. However, by choosing centering dates and appropriate half-windows, seasonal behavior can easily be observed in these plots.

```
qBottom<-20
qTop<-700
date1 <- "2000-09-01"
date2 <- "2005-09-01"
date3 <- "2009-09-01"
plotConcQSmooth(date1, date2, date3,
                 qBottom, qTop, qUnit=1)
plotConcQSmooth(date1, date2, date3,
                 qBottom, qTop, qUnit=1, logScale=TRUE)
```



(a) `plotConcQSmooth`



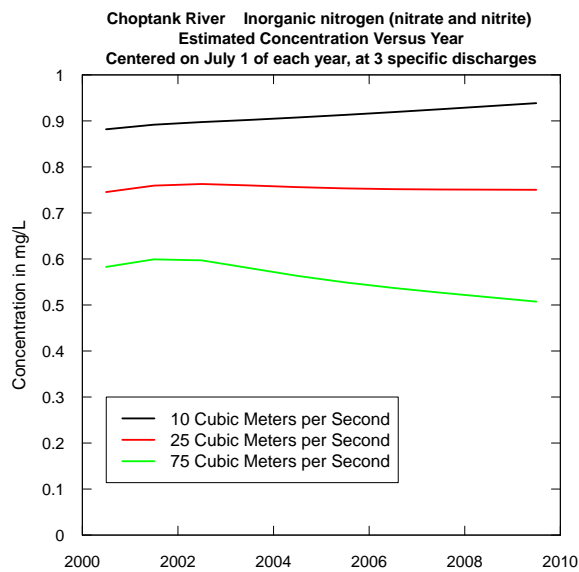
(b) `plotConcQSmooth(logScale=TRUE)`

Figure 16. Concentration vs. discharge

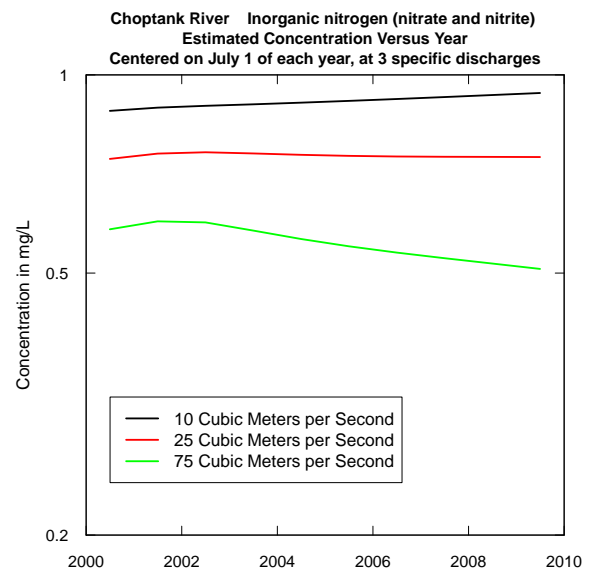
```

q1 <- 10
q2 <- 25
q3 <- 75
centerDate <- "07-01"
plotConcTimeSmooth(q1, q2, q3, centerDate, 2000, 2010)
plotConcTimeSmooth(q1, q2, q3, centerDate,
                    2000, 2010, logScale=TRUE)

```



(a) plotConcTimeSmooth



(b) plotConcTimeSmooth(logScale=TRUE)

Figure 17. plotConcTimeSmooth

Figure 18 shows a predefined multipanel graph using `fluxBiasMulti`.

```

fluxBiasMulti(qUnit=1)

```

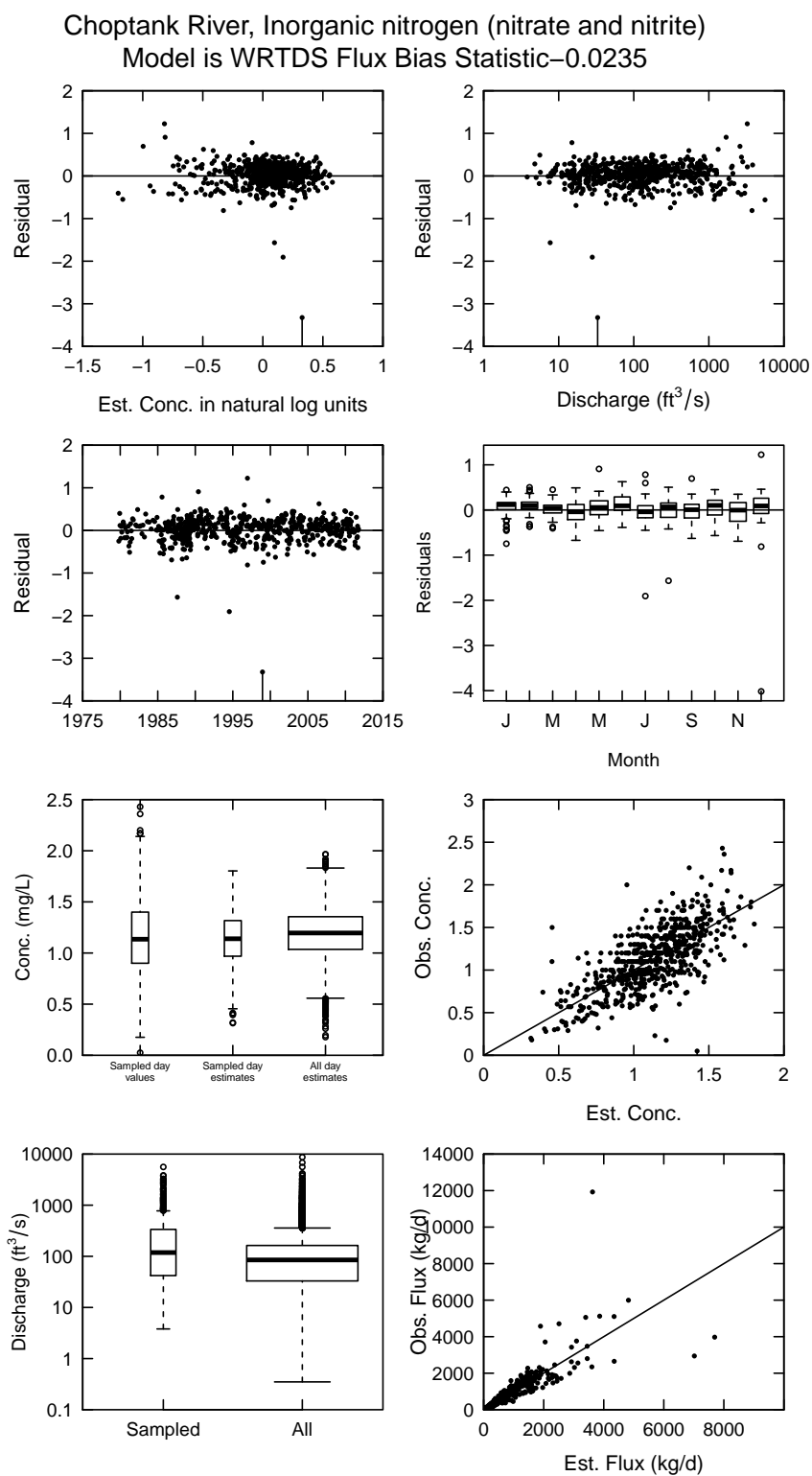



Figure 18. fluxBiasMulti(qUnit=1)

The contour plot functions also do not recognize the PA from the INFO dataframe. They represent the overall results of the WRTDS analysis. To specify contourLevels in the contour plots use the `seq` function (type `?seq` for details). In general, use of the `seq` function would look like this: `contourLevels = seq(from,to,by)`. In the example shown above we are requesting contour levels that run from 0 to 2 in steps of 0.2.

```
clevel<-seq(0,2,0.2)
plotContours(yearStart=2008,yearEnd=2010,qBottom=20,qTop=1000,
             contourLevels = clevel,qUnit=1,
             flowDuration=FALSE)
```

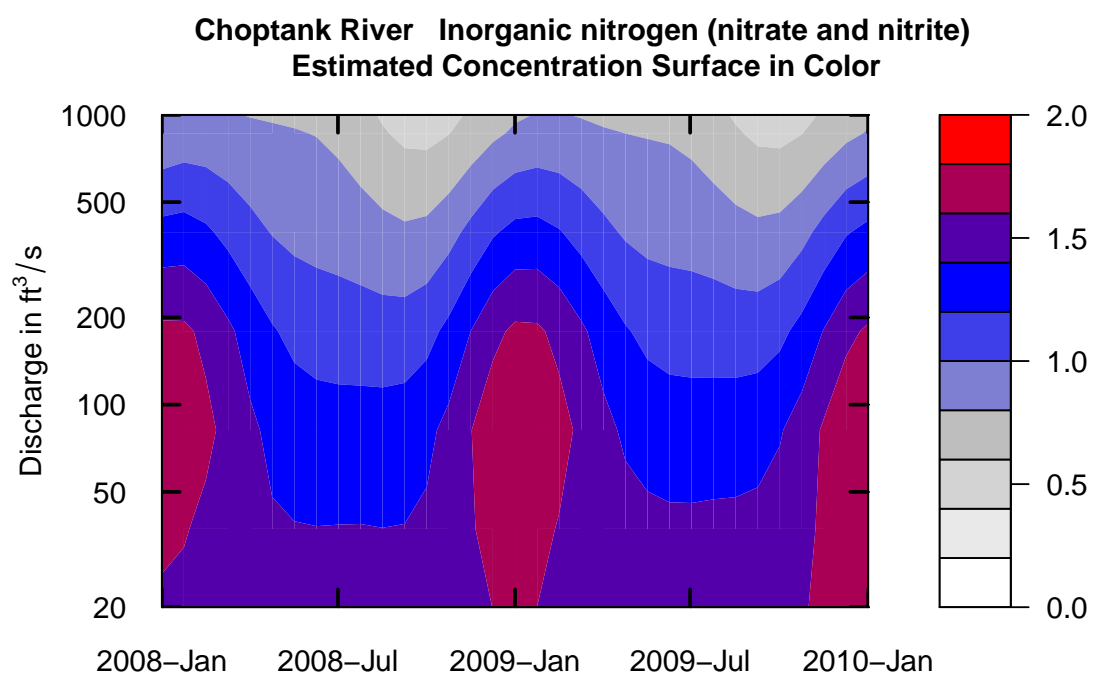


Figure 19. `plotContours(yearStart=2008,yearEnd=2010,qBottom=20,qtop=1000,contourLevels=seq(0,2,0.2),qUnit=1,flowDuration=FALSE)`

The function `plotDiffContours` plots the difference between two selected years (`year0` and `year1`). It can help clarify what combinations of seasons and flow conditions have been showing increases and decreases over the period covered.

```
plotDiffContours(year0=2000,year1=2010,  
                 qBottom=20,qTop=1000,maxDiff=0.6,qUnit=1,  
                 flowDuration=FALSE)
```

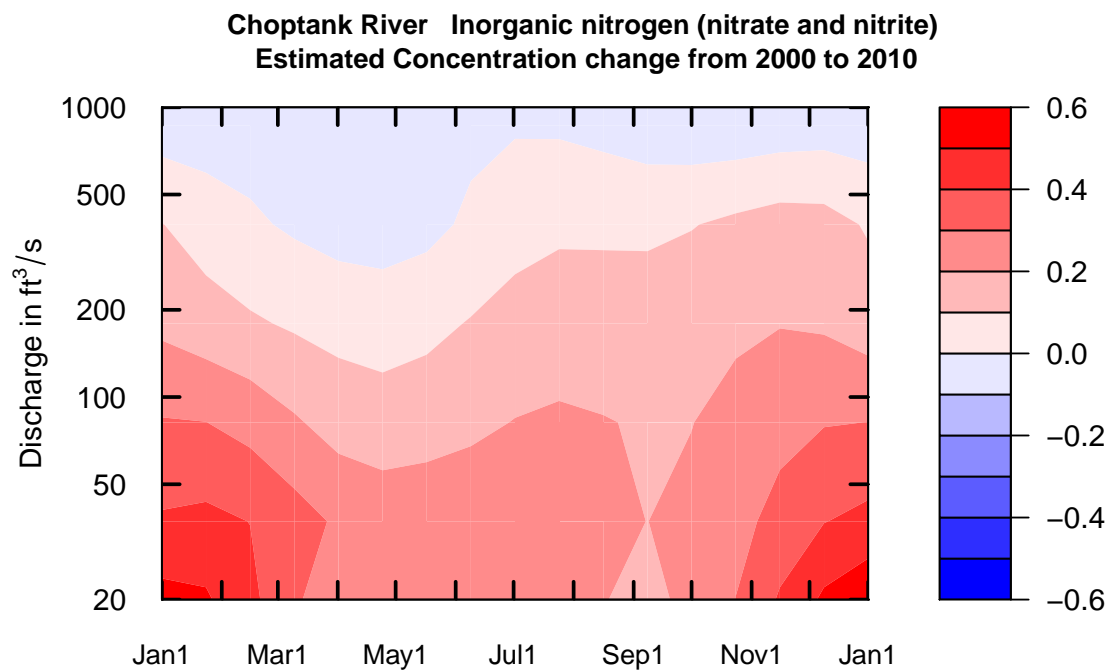


Figure 20. `plotDiffContours(year0=2000,year1=2010,qBottom=20,qTop=1000,maxDiff=0.6,qUnit=1,flowDuration=FALSE)`

7.2 Table Options

Sometimes it is easier to consider the results in table form rather than graphically. The function `tableResults` produces a simple text table that contains the annual values for the results. Each row of the output represents a year and includes: year, average discharge, average concentration, flow-normalized concentration, average flux, and flow-normalized flux. A small sample of the output is printed below. This function can also return a data frame if `returnDataFrame` is set to `TRUE`.

```
tableResults()
returnDF <- tableResults(returnDataFrame=TRUE)
```

```
Choptank River
Inorganic nitrogen (nitrate and nitrite)
Water Year

Year      Discharge      Conc      FN_Conc      Flux      FN_Flux
          cms          mg/L          10^6 kg/yr
1980      4.25          0.949          1.003          0.1154          0.106
1981      2.22          1.035          0.999          0.0675          0.108
...
2010      7.19          1.323          1.438          0.2236          0.149
2011      5.24          1.438          1.457          0.1554          0.148
```

```
head(returnDF)

      Year Discharge [cms] Conc [mg/L] FN_Conc [mg/L]
1    1980          4.25    0.949    1.003
2    1981          2.22    1.035    0.999
3    1982          3.05    1.036    0.993
4    1983          4.99    1.007    0.993
5    1984          5.72    0.990    1.002
6    1985          1.52    1.057    1.017
      Flux [10^6kg/yr] FN_Flux [10^6kg/yr]
1          0.1154          0.106
2          0.0675          0.108
3          0.0985          0.110
4          0.1329          0.112
5          0.1597          0.114
6          0.0489          0.116
```

The other table option is `tableChange`. This is a function that provides for the computation of changes or slopes between any selected pairs of time points. These computations are made only on the flow-normalized results. A detailed explanation of “flow-normalized” result is in the official EGRET manual.

```
tableChange (yearPoints=c(2000,2005,2010))
```

Choptank River
Inorganic nitrogen (nitrate and nitrite)
Water Year

Concentration trends						
time span			change	slope	change	slope
			mg/L	mg/L/yr	%	%/yr
2000	to	2005	0.088	0.018	7	1.4
2000	to	2010	0.19	0.019	15	1.5
2005	to	2010	0.098	0.02	7.3	1.5

Flux Trends						
time span			change	slope	change	slope
			10 ⁶ kg/yr	10 ⁶ kg/yr /yr	%	%/yr
2000	to	2005	0.0065	0.0013	4.7	0.93
2000	to	2010	0.0097	0.00097	6.9	0.69
2005	to	2010	0.0032	0.00063	2.2	0.43

Finally, `tableChangeSingle` (Table 10) operates exactly the same as `tableChange` except for the addition of two arguments: `returnDataFrame` and `flux`. This function provides either concentration results or flux results, but not both. This can be useful when you are producing many output tables for a report that is entirely focused on concentration or one that is entirely focused on flux. The arguments are identical to those for `tableChange`, except for the final two arguments. The first, "returnDataFrame" is a logical argument to indicate if a dataframe of output should be returned (for later manipulation or printing through other programs such as Excel), and its default is FALSE. The second argument is "flux", and the default is TRUE. When `flux=TRUE` the output is only for flux, and when `flux=FALSE` the output is only for concentration. See section 11 for instructions on converting an R dataframe to a table in Microsoft.

```
returnDF <- tableChangeSingle (yearPoints=c(2000,2005,2010) ,  
                                returnDataFrame=TRUE)
```

Year1	Year2	change [mg/L]	slope [mg/L/yr]	change[%]	slope [%/yr]
2000	2005	0.09	0.02	7.00	1.40
2000	2010	0.19	0.02	15.00	1.50
2005	2010	0.10	0.02	7.30	1.50

Table 9. Table created from `tableChangeSingle` function using Microsoft Excel (see section 11)

8 Extending Plots Past Defaults

The basic plotting options were shown in the section 7. This section demonstrates some ways to extend the capabilities of the EGRET plots. EGRET plots use R's basic plotting options. You set many of the formatting details of plotting routines in R by using "Graphical Parameters". To read about all of these graphical parameters see `?par`. When the graphical functions in EGRET are coded, a set of default values for many of these parameters are chosen, but you can override all of these default values. Additionally, you can add features to a plot after calling the plot function. To change the plot margins (`mar`), font, or other graphical parameters initially assigned, set the argument `customPar` to `TRUE`.

A few of R's base graphical parameters are especially useful within the plot functions. These are shown in Table 10.

Argument	Description	Values
<code>cex</code>	Size of data point symbols, relative to default	decimal number
<code>cex.main</code>	Size of font for plot title, relative to default	decimal number
<code>cex.lab</code>	Size of font for axis label text, relative to default	decimal number
<code>cex.axis</code>	Size of font for axis annotation (numbers), relative to default	decimal number
<code>col</code>	Color of data point symbols or lines	color name in " "
<code>lwd</code>	Width of lines, relative to default	decimal number
<code>pch</code>	Type of symbol to use for data points	integer values
<code>lty</code>	Line type number (such as dash or dot)	integer values

Table 10. Useful plotting parameters to adjust in EGRET plotting functions. For details of any of these see `?par`.

After the plot is made, many other functions that might be useful to call, such as to add text, legend, lines, etc. Table 11 lists a few common options.

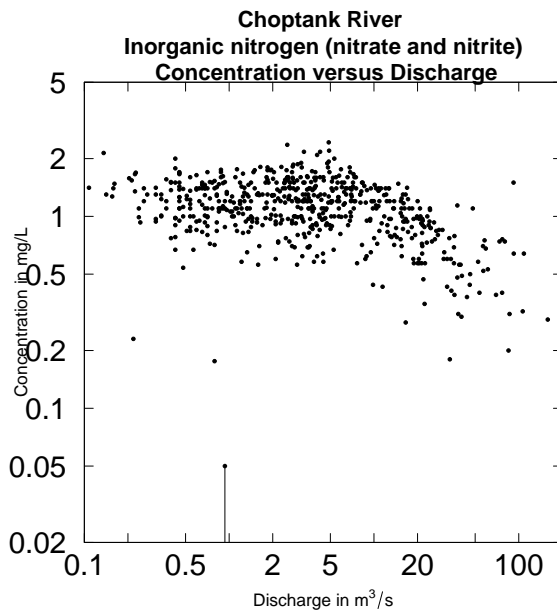
Function	Description
<code>mtext</code>	add text based on specified side of plot
<code>text</code>	add text to a specific point on plot
<code>legend</code>	add a legend
<code>grid</code>	add grid
<code>abline</code>	add line
<code>arrows</code>	add arrow

Table 11. Useful functions to add on to default plots. [Type `?` then the function name to get help on the individual function].

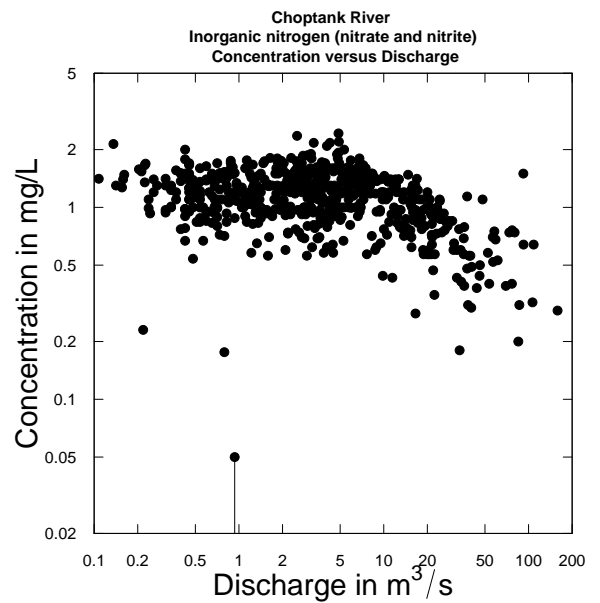
Some basic examples are shown below.

Figure 21 shows a larger title and axis number (left), and larger axis labels and point size (right).

```
plotConcQ(cex.axis=2,cex.main=1.5,logScale=TRUE)
plotConcQ(cex.lab=2,cex=2,logScale=TRUE)
```



(a) `plotConcQ(cex.axis=2,cex.main=1.5,logScale=TRUE)`

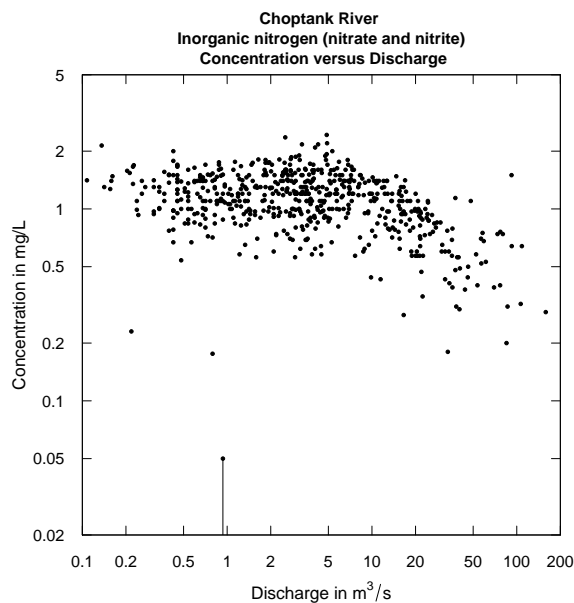


(b) `plotConcQ(cex.lab=2,cex=2,logScale=TRUE)`

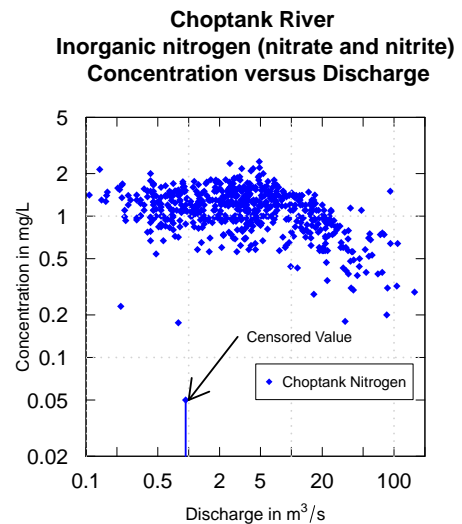
Figure 21. Modifying text and point size

Figure 22 shows the default on the left, and several features on the right. First, the margin is adjusted to `c(8,8,8,8)`, requiring `customPar` set to `TRUE`. The margin vector represents the margin spacing of the 4 "sides" of a plot in the order: bottom, left, top, right. Next, the text labels were adjusted, color set to "blue", point and line size increased, and the point type changed from a solid circle(`pch=20`) to solid diamond (`pch=18`). A grid, legend, arrow, and text are added after the plot is produced.

```
plotConcQ(logScale=TRUE)
par(mar=c(8,8,8,8))
plotConcQ(customPar=TRUE,col="blue",cex=1.1,
          cex.axis=1.4,cex.main=1.5,cex.lab=1.2,
          pch=18,lwd=2,logScale=TRUE)
grid(lwd=2)
legend(4.5,.09,"Choptank Nitrogen",pch=18,col="blue",bg="white")
arrows(3,0.14,1,.05,lwd=2)
text(12,.14,"Censored Value")
```



(a) Default



(b) Modified

Figure 22. Modified `plotConcQ`

Only a few fonts are consistent on all operating systems. Figure 23 shows how to change to the Serif font, as well as how to use the `mtext` function. To see the available fonts for pdf output on your computer, type `names(pdfFonts())`. The available fonts are quite limited in base R. To expand the font choices, a useful R library, “extrafont” can help.

```
# Switching to serif font:  
par(family="serif")  
plotFluxPred(customPar=TRUE)  
mtext(side=3, line=-3, "Serif font example", cex=3)
```

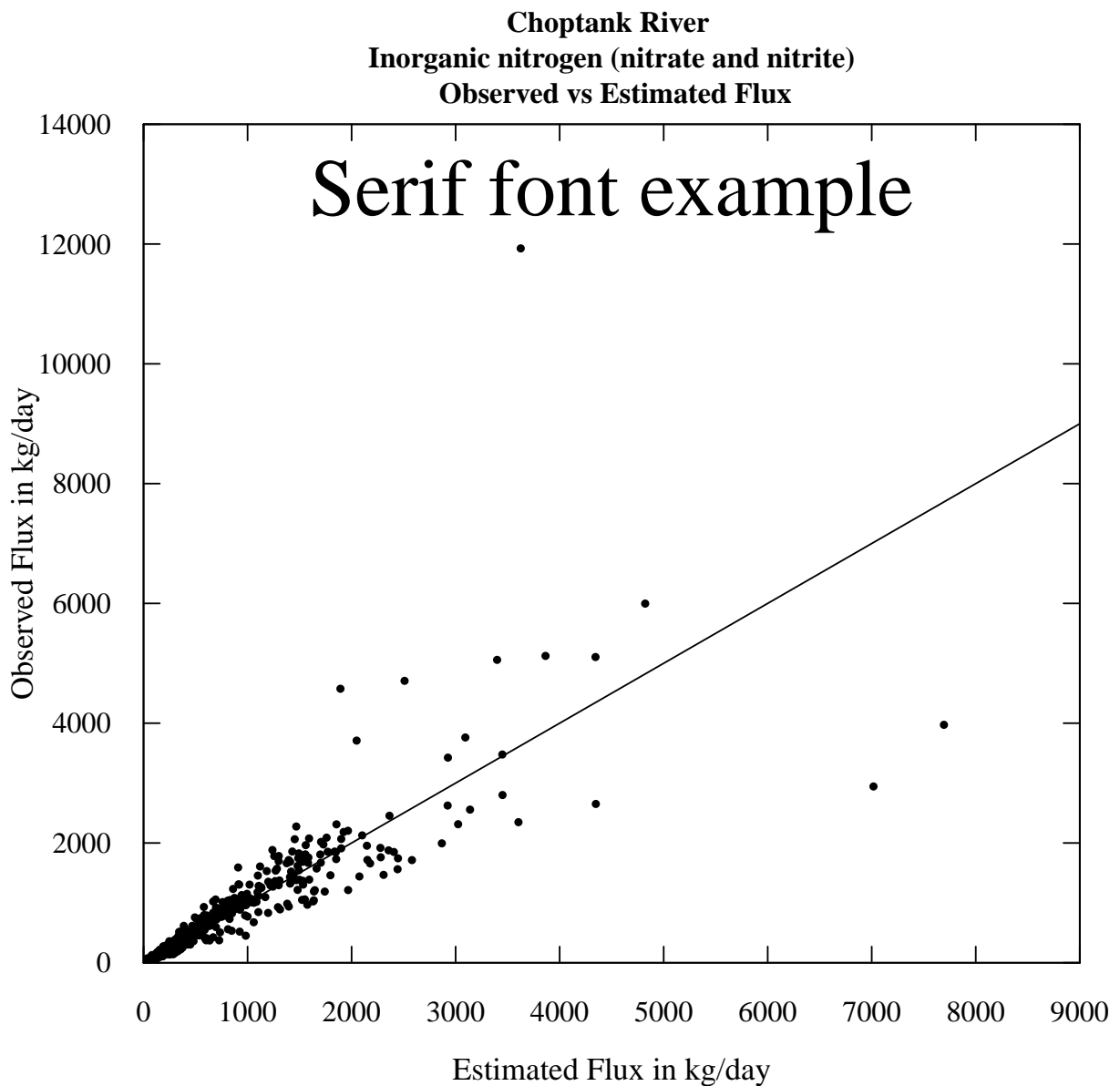


Figure 23. Serif font

You can also extend the contour plots. The default y-axis is determined from `qTop` and `qBottom`. Occasionally, you may need to use a custom axis by specifying `yTicks`. It is also nice to be able to adjust the color scheme of the contour plots. There are some color schemes built into base R such as `heat.colors`, `topo.colors`, `terrain.colors`, and `cm.colors`. Alternatively, you can set colors by using the `colorRampPalette` function. For example, a black and white color scheme might be required. In another example, the `plotDiffContours` might make more sense to go from yellow to white for the negative values, and white to blue for the positive values. Examples are shown below for modifying a contour plot in Figure 24 and modifying a difference contour plot in Figure 25.

```

colors <- colorRampPalette(c("white", "black"))
yTicksModified <- c(.5, 1, 10, 25)
plotContours(2001, 2010, 0.5, 50,
             contourLevels = seq(0, 2.5, 0.5), qUnit=2,
             yTicks=yTicksModified,
             color.palette=colors,
             flowDuration=FALSE,
             tcl=0.2, tick.lwd=2.5)

```

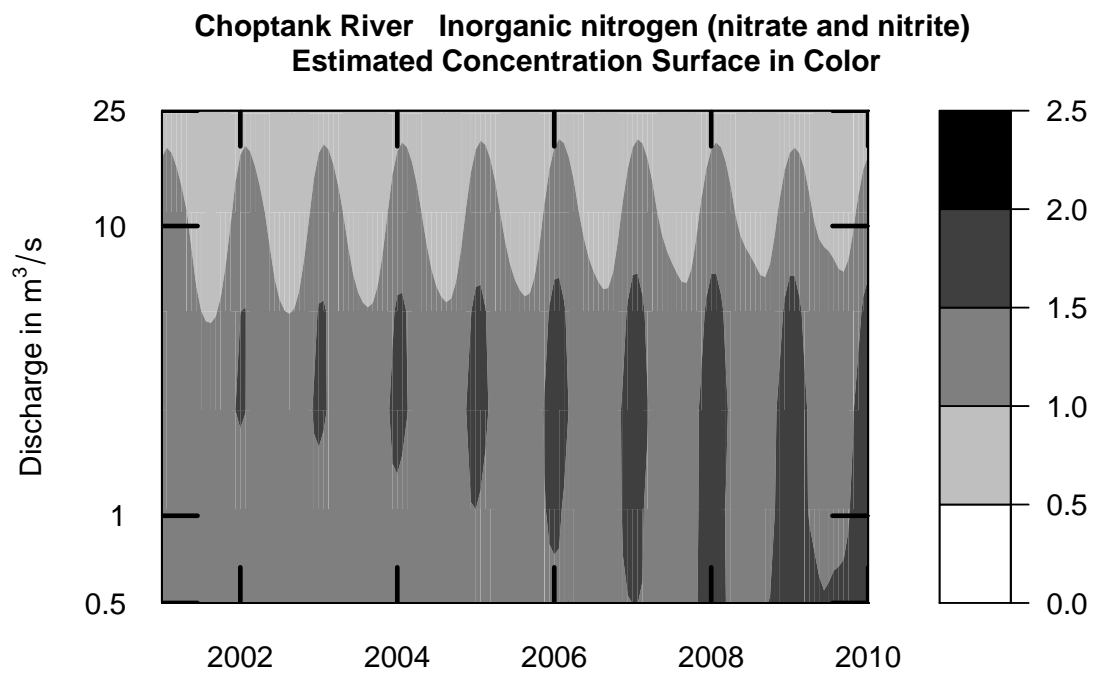


Figure 24. Contour plot with modified axis and color scheme

```

colors <- colorRampPalette(c("yellow", "white", "blue"))
maxDiff<-0.6
par(oma=c(1,1,1,1))
plotDiffContours(year0=2001,year1=2010,qBottom=0.5,qTop=50,
                 maxDiff,lwd=2,qUnit=2,
                 color.palette=colors,
                 flowDuration=FALSE, customPar=TRUE)

```

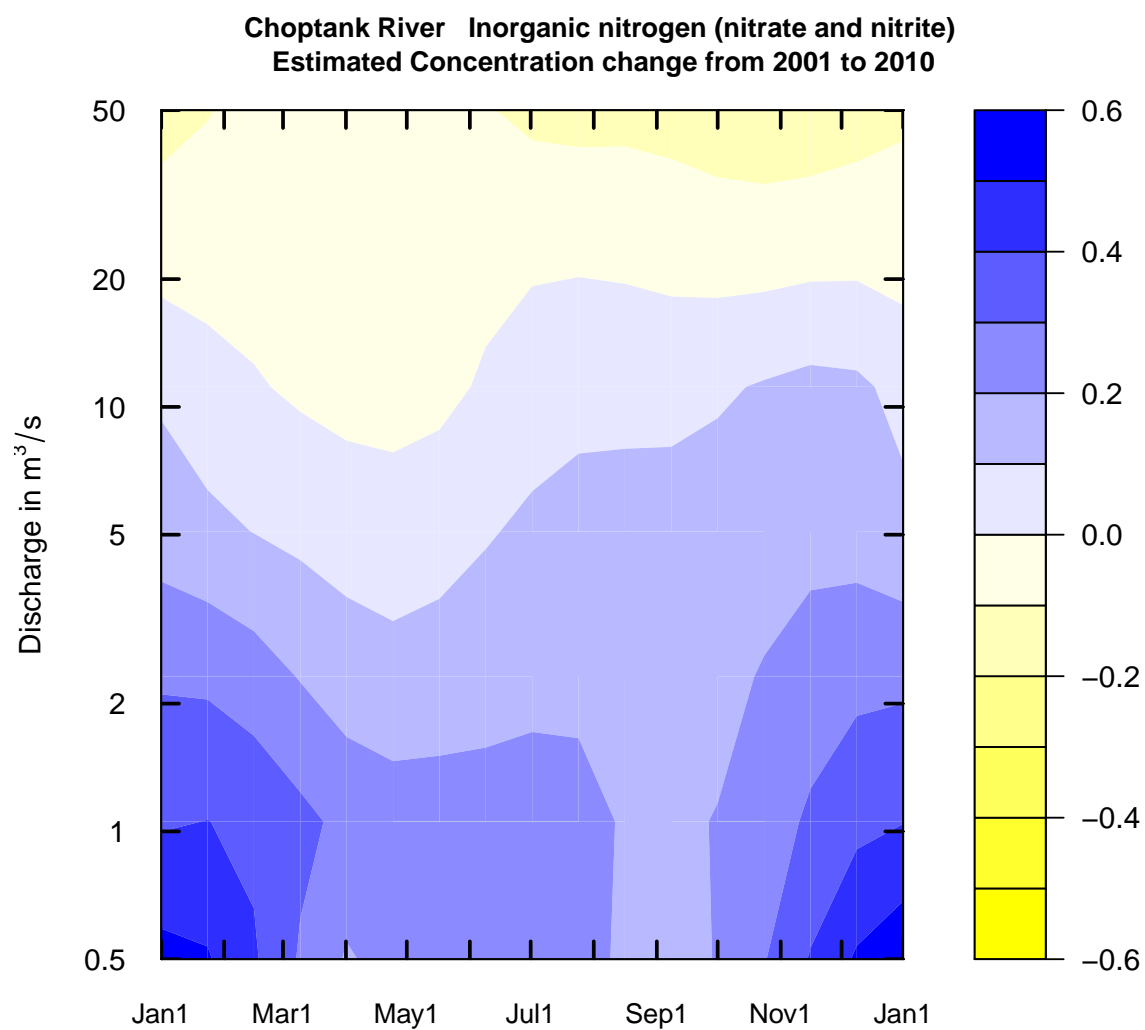


Figure 25. Difference contour plot with modified color scheme

It is also possible to create custom multi-panel plots. In the simplest example (figure 26), you can use the "tinyPlot=TRUE" option.

```
par(mfcol = c(2, 2), oma = c(0, 1.7, 6, 1.7))

plotFluxQ(tinyPlot=TRUE, printTitle=FALSE,
          fluxUnit=9, logScale=FALSE, fluxMax=1)
plotConcQ(tinyPlot=TRUE, printTitle=FALSE)
plotFluxHist(tinyPlot=TRUE, printTitle=FALSE, fluxMax=1)
plotConcHist(tinyPlot=TRUE, printTitle=FALSE, concMax=3)
mtext("Custom multi-pane graph using tinePlot=TRUE", outer=TRUE, font=2)
```

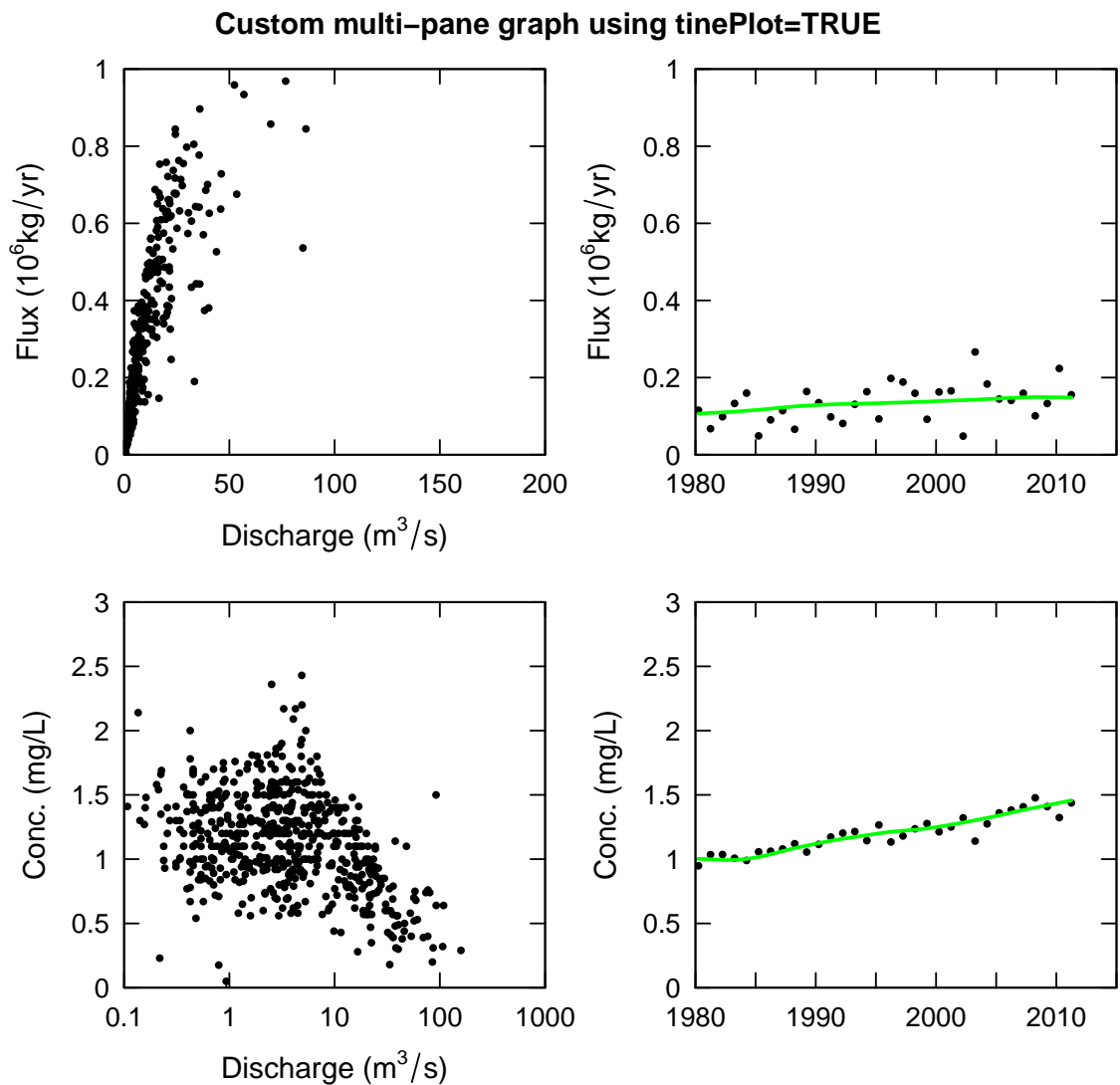


Figure 26. Custom multipanel plot using tinyPlot

Finally, figure 27 shows a method to create a panel of plots with a finer control.

```
par(mar=c(3.5,3.5,0.2,0.2), # This is the whitespace around the plots
    oma=c(1,1,3,1), # This is the outer margin
    mgp=c(2,0.5,0), # This is the spacing between the label numbers and plots
    mfcol = c(2,2)) # rows/columns

plotFluxQ(tinyPlot=TRUE, printTitle=FALSE,
          fluxUnit=9, logScale=FALSE, fluxMax=1,
          showXLabels=FALSE, showXAxis=FALSE,
          showYLabels=TRUE, customPar=TRUE)

plotConcQ(tinyPlot=TRUE, printTitle=FALSE, customPar=TRUE,
          removeLastY=TRUE, removeLastX=TRUE,
          showYLabels=TRUE)

plotFluxHist(tinyPlot=TRUE, printTitle=FALSE, fluxMax=1,
             showYLabels=FALSE, showYAxis=FALSE,
             showXLabels=FALSE, showXAxis=FALSE, customPar=TRUE)
plotConcHist(tinyPlot=TRUE, printTitle=FALSE, concMax=3,
             showYLabels=FALSE, showYAxis=FALSE, customPar=TRUE)
mtext("Custom multi-pane graph using customPar", outer=TRUE, font=2)
```

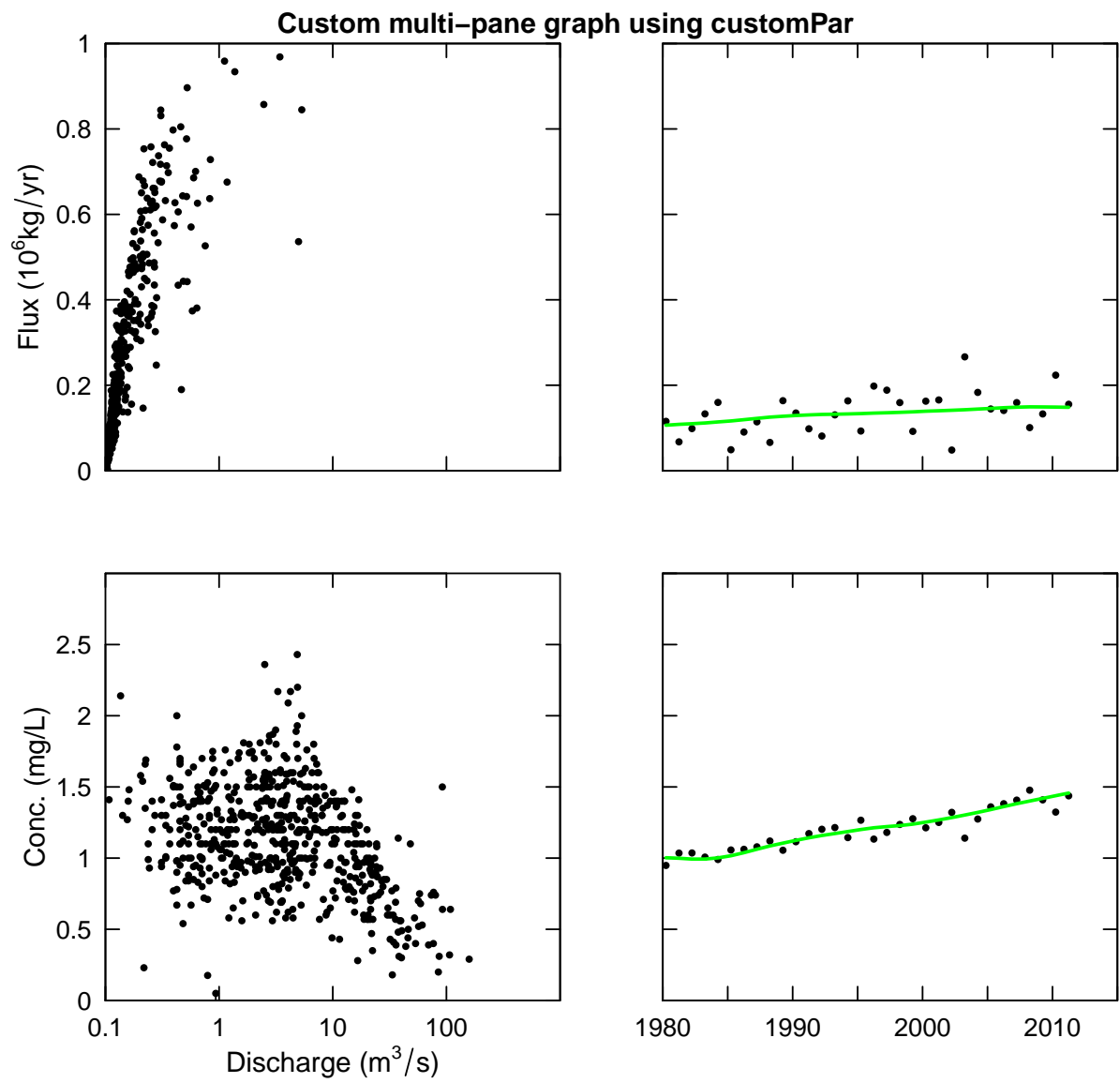


Figure 27. Custom multipanel plot

9 Getting Started in R

This section describes the options for installing the EGRET package.

9.1 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 function's name. This will open a file that describes the function, the required arguments, and provides working examples.

```
?getJulian
```

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

```
getJulian
```

9.2 R User: Installing EGRET

To install the EGRET packages and its dependencies:

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

It is a good idea to re-start R after installing the package if installing an updated version.

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)  
library(EGRET)
```

10 Common Function Variables

This section describes variables that are common for a variety of function types.

10.1 flowHistory Plotting Input

Table 12. Variables used in flow history plots (`plot15`, `plotFour`, `plotFourStats`, `plotQTimeDaily`, `plotSDLogQ`)

Argument	Definition	Default
<code>istat</code>	The discharge statistic to be plotted: 1-8. Must be specified, see Table 8.	
<code>yearStart</code> ¹	The decimal year (<code>decYear</code>) value where you want the graph to start	NA
<code>yearEnd</code> ¹	The decimal year (<code>decYear</code>) value where you want the graph to end	NA
<code>qMax</code>	User specified upper limit on y axis (can be used when we want several graphs to all share the same scale). Value is specified in the discharge units that the user selects.	NA
<code>printTitle</code>	can be TRUE or FALSE, you may want FALSE if it is going to be a figure with a caption or if it is a part of a multipanel plot.	TRUE
<code>tinyPlot</code>	Can be TRUE or FALSE, the TRUE option assures that there will be a small number of tick marks, consistent with printing in a small space	FALSE
<code>runoff</code>	Can be TRUE or FALSE. If true then discharge values are reported as runoff in mm/day. This can be very useful in multi-site analyses.	FALSE
<code>qUnit</code>	An index indicating what discharge units to use. Options run from 1 to 6 (see section 3.4). The choice should be based on the units that are customary for the audience but also, the choice should be made so that the discharge values don't have too many digits to the right or left of the decimal point.	1
<code>printStaName</code> ²	Can be TRUE or FALSE, if TRUE the name of the streamgage is stated in the plot title.	TRUE
<code>printPA</code> ²	Can be TRUE or FALSE, if TRUE the period of analysis is stated in the plot title.	TRUE
<code>printIstat</code> ²	Can be TRUE or FALSE, if TRUE the name of the statistic (e.g. 7-day minimum discharge) is stated in the plot title.	TRUE

¹ Setting `yearStart` and `yearEnd` will determine where the graphs start and end, but they don't determine where the smoothing analysis starts and ends. There are situations, typically where many sites are analyzed together, where you may want to run the smoothing on a consistent period of record across all sites, which requires subsetting the `Daily` data frame before running `makeAnnualSeries` (see `?subset`).

² If the `printTitle` argument is set to FALSE, then it really makes no difference what you do with `printSta`, `printPA`, or `printIstat`. They can all be left as their default values and thus there is no need to include them in the call for the function.

10.2 Water Quality Plotting Input

Table 13. Selected variables used in water quality analysis plots

Argument	Definition	Default
qUnit	Determines what units will be used for discharge, see section 3.4	2
printTitle	If TRUE the plot has a title. If FALSE no title (useful for publications where there will be a caption)	TRUE
qLower	The lower bound on the discharge on the day of sampling that will be used in forming a subset of the sample data set that will be displayed in the graph. It is expressed in the units specified in qUnit. If qLower = NA, then the lower bound is set to zero.	
qUpper	The upper bound on the discharge on the day of sampling that will be used in forming a subset of the sample data set that will be displayed in the graph. It is expressed in the units specified in qUnit. If qUpper = NA, then the upper bound is set to infinity.	
concMax	The upper limit on the vertical axis of graphs showing concentration values in mg/L (NA sets value to just above maximum).	NA
concMin	The lower limit on the vertical axis of graphs showing concentration values in mg/L (NA sets value to just below minimum for log scales, zero for linear).	NA
fluxUnit	Determines what units will be used for flux (see Section 3.4).	9
fluxMax	The upper limit on the vertical axis of graphs showing flux values.	

10.3 WRTDS Estimation Input

Table 14. Selected variables in WRTDS

Argument	Definition	Default
windowY	The half window width for the time weighting, measured in years. Values much shorter than 10 usually result in a good deal of oscillations in the system that are likely not very realistic	10
windowQ	The half window width for the weighting in terms of $\ln(Q)$. For very large rivers (average discharge values in the range of many tens of thousands of cfs) a smaller value than 2 may be appropriate, but probably not less than 1	2
windowS	The half window width for the seasonal weighting, measured in years. Any value >0.5 will make data from all seasons have some weight. Values should probably not be lower than 0.3	0.5
minNumObs	This is the minimum number of observations with non-zero weight that the individual regressions will require before they will be used. If there too few observations the program will iterate, making the windows wider until the number increases above this minimum. The only reason to lower this is in cases where the data set is rather small. It should always be set to a number at least slightly smaller than the sample size. Any value less than about 60 is probably in the “dangerous” range, in terms of the reliability of the regression	100
minNumUncen	This is the minimum number of uncensored observations with non-zero weight that the individual regressions will require before they will be used. If there are too few uncensored observations the program will iterate, making the windows wider until the number increases above this minimum. The only reason to lower this is in cases where the number of uncensored values is rather small. The method has never been tested in situations where there are very few uncensored values	50

10.4 WRTDS Plotting Input

Table 15. Selected variables used in plots for analysis of WRTDS results

Argument	Definition	Default
qUnit	Determines what units will be used for discharge, see section 3.4	2
fluxUnit	An index indicating what flux units will be used , see section 3.4	3
stdResid	This is an option. If FALSE, it prints the regular residuals (they are in ln concentration units). If TRUE, it is the standardized residuals. These are the residuals divided by their estimated standard error (each residual has its own unique standard error). In theory, the standardized residuals should have mean zero and standard deviation of 1	FALSE
printTitle	If TRUE the plot has a title. If FALSE no title (useful for publications where there will be a caption)	TRUE
startYear	The starting date for the graph, expressed as decimal years, for example, 1989	NA
endYear	The ending date for the graph, expressed as decimal years, for example, 1996	NA
moreTitle	A character variable that adds additional information to the graphic title. Typically used to indicate the estimation method.	
fluxMax	The upper limit on the vertical axis of graphs showing flux values.	NA
concMax	The upper limit on the vertical axis of graphs showing concentration values.	NA
plotFlowNorm	If TRUE the graph shows the annual values as circles and the flow-normalized values as a green curve. If false, it only shows the annual values.	TRUE

Table 16. Variables used in WRTDS contour plots: `plotContours` and `plotDiffContours`

Argument	Definition	Defaults
<code>qUnit</code>	Determines what units will be used for discharge, see section 3.4	2
<code>qBottom</code>	The lower limit of the discharge value for the graphs in the units specified by <code>qUnit</code>	
<code>qTop</code>	The upper limit of the discharge value for the graphs in the units specified by <code>qUnit</code>	
<code>printTitle</code>	If TRUE the plot has a title. If FALSE no title (useful for publications where there will be a caption)	TRUE
<code>yearStart</code>	The starting date for the graph, expressed as decimal years, for example, 1989	
<code>yearEnd</code>	The ending date for the graph, expressed as decimal years, for example, 1996	
<code>whatSurface</code>	This should generally be at its default value. At <code>whatSurface = 3</code> , the plotted surface shows the expected value of concentration. For <code>whatSurface = 1</code> , it shows the <code>yHat</code> surface (natural log of concentration). For <code>whatSurface = 2</code> , it shows the SE surface (the standard error in log concentration).	3
<code>contourLevels</code>	With the default value the contour intervals are set automatically, which generally will NOT be a very good choice, but they may provide a starting point. If you want to specify <code>contourLevels</code> , use the <code>seq</code> function. In general it would look like: <code>contourLevels = seq(from,to,by)</code> .	NA
<code>maxDiff</code>	In the <code>plotDiffContours</code> function instead of using <code>contourLevels</code> , the contours are set by <code>maxDiff</code> which is the absolute value of the maximum difference to be plotted. Contour intervals are set to run from <code>-maxDiff</code> to <code>maxDiff</code> .	
<code>span</code>	Specifies the smoothness of the discharge duration information that goes on this graph. A larger value will make it smoother. The default should work well in most cases.	60
<code>pval</code>	The probability value for the discharge frequency information shown on the plot. When <code>flowDuration=TRUE</code> , the plot has two black curves on it. In the default value case these are at the 5 and 95 percent levels on the seasonal discharge duration curve. <code>pval = 0.01</code> would place these at the 1 and 99 percent points. <code>pval = 0.1</code> would place them at 10 and 90.	0.05
<code>vert1</code>	This simply plots a vertical black line on the graph at a particular time (defined in decimal years). It is used to illustrate the idea of a “vertical slice” through the contour plot, which might then be shown in a subsequent use of <code>plotConcQSmooth</code> .	NA
<code>vert2</code>	This gives the location of a second vertical black line on the graph at a particular time (defined in decimal years).	NA
<code>horiz</code>	This simply plots a horizontal black line on the graph at a particular discharge value (defined in the units specified by <code>qUnit</code>). It is used to illustrate the idea of the seasonal cycle in concentrations for a given discharge and the long-term change in this cycle.	NA
<code>flowDuration</code>	If TRUE it draws the discharge duration lines at the specified probabilities. If FALSE, the discharge duration lines are left off.	TRUE

Table 17. Variables used in WRTDS `plotConcQSmooth` and/or `plotConcTimeSmooth` functions

Argument	Definition	Default
date1	This is the date for the first curve to be shown on the <code>plotConcQSmooth</code> graph. It must be in the form "yyyy-mm-dd" (it must be in quotes)	
date2	This is the date for the second curve to be shown on the plot ("yyyy-mm-dd"), If you don't want a second curve then the argument must be <code>date2=NA</code>	
date3	This is the date for the third curve to be shown on the plot ("yyyy-mm-dd"), If you don't want a third curve then the argument must be <code>date3=NA</code>	
q1	This is the discharge for the first curve on the <code>plotConcTimeSmooth</code> graph. It is in units specified by <code>qUnit</code>	
q2	This is the discharge for the second curve. If you don't want a second curve then the argument must be <code>q2=NA</code>	
q3	This is the discharge for the third curve. If you don't want a third curve then the argument must be <code>q3=NA</code>	
qUnit	Determines what units will be used for discharge, see <code>printqUnitCheatSheet</code>	2
qLow	The discharge value that should form the left edge of the <code>plotConcQSmooth</code> graph in the user-selected discharge units.	
qHigh	The discharge value that should form the right edge of the <code>plotConcQSmooth</code> graph in the user-selected discharge units.	
centerDate	This is the month and day at the center of the time window for the <code>plotConcTimeSmooth</code> graph. It must be in the form "mm-dd" in quotes	
yearStart	The starting year for the <code>plotConcTimeSmooth</code> graph	
yearEnd	The ending year for the <code>plotConcTimeSmooth</code> graph	
legendLeft	This determines the placement of the legend on the graph. It establishes the left edge of the legend and is expressed in the discharge units being used. The default (which is NA) will let it be placed automatically. The legend can end up conflicting with one or more of the curves. Once the location of the curves is established then this can be set in a way that avoids conflict.	0
legendTop	This determines the placement of the legend on the graph. It establishes the top edge of the legend and is expressed according to the concentration values on the y-axis. The default (which is NA) will let it be placed automatically. The legend can end up conflicting with one or more of the curves. Once the location of the curves is established then this can be set in a way that avoids conflict.	0
concMax	Maximum value for the vertical axis of the graph. The reason to set <code>concMax</code> is if you want to make several plots that have the same vertical axis.	NA
concMin	[This one is only used when <code>logScale=TRUE</code>]. Minimum value for the vertical axis of the graph. The reason to set <code>concMin</code> is if you want to make several plots that have the same vertical axis.	NA
bw	Default is FALSE, which means we want a color plot. If <code>bw=TRUE</code> that means it should be black and white.	
printTitle	If TRUE the plot has a title. If FALSE no title (useful for publications where there will be a caption).	FALSE
printValues	If TRUE the estimated values that make up the plotted lines are printed on the console. If FALSE they are not printed. This could be useful if you wanted to compute various comparisons across time periods.	FALSE
windowY	This is the half-window width for time in WRTDS. It has units of years.	10
windowQ	This is the half-window width for discharge in WRTDS. It has units of $\ln(\text{discharge})$.	2
windowS	This is the half-window width for seasons in WRTDS. It has units of years.	0.5

11 Creating tables in Microsoft from R

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

```
tableData <- tableResults(returnDataFrame=TRUE)
```

Choptank River					
Inorganic nitrogen (nitrate and nitrite)					
Water Year					
Year	Discharge cms	Conc mg/L	FN_Conc	Flux 10 ⁶	FN_Flux kg/yr
1980	4.25	0.949	1.003	0.1154	0.106
1981	2.22	1.035	0.999	0.0675	0.108
1982	3.05	1.036	0.993	0.0985	0.110
1983	4.99	1.007	0.993	0.1329	0.112
1984	5.72	0.990	1.002	0.1597	0.114
1985	1.52	1.057	1.017	0.0489	0.116
1986	2.63	1.062	1.038	0.0903	0.119
1987	3.37	1.079	1.062	0.1142	0.122
1988	1.87	1.120	1.085	0.0660	0.125
1989	5.61	1.055	1.105	0.1638	0.127
1990	4.01	1.115	1.125	0.1349	0.129
1991	2.75	1.172	1.143	0.0980	0.130
1992	2.19	1.203	1.159	0.0810	0.132
1993	3.73	1.215	1.173	0.1306	0.132
1994	5.48	1.144	1.187	0.1634	0.133
1995	2.41	1.266	1.201	0.0928	0.134
1996	6.24	1.134	1.213	0.1980	0.135
1997	5.83	1.180	1.221	0.1884	0.136
1998	4.88	1.236	1.229	0.1593	0.137
1999	2.90	1.277	1.238	0.0919	0.138
2000	4.72	1.213	1.253	0.1627	0.139
2001	4.88	1.251	1.268	0.1655	0.140
2002	1.24	1.321	1.285	0.0483	0.141
2003	8.64	1.140	1.303	0.2664	0.143
2004	5.28	1.274	1.321	0.1832	0.144
2005	3.81	1.360	1.341	0.1444	0.146
2006	3.59	1.382	1.362	0.1409	0.147
2007	4.28	1.408	1.382	0.1593	0.149
2008	2.56	1.477	1.401	0.1008	0.149

2009	3.68	1.409	1.419	0.1328	0.149
2010	7.19	1.323	1.438	0.2236	0.149
2011	5.24	1.438	1.457	0.1554	0.148

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:

```
Year  Discharge [cms] Conc [mg/L] FN_Conc [mg/L] Flux [10^6kg/yr] FN_Flux [10^6kg/yr]
1980   4.25          0.949      1.003      0.1154          0.106
1981   2.22          1.035      0.999      0.0675          0.108
1982   3.05          1.036      0.993      0.0985          0.110
...
```

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 word processing or presentation software products. An example using one of the default Excel table formats is here.

Year	Discharge [cms]	Conc [mg/L]	FN_Conc [mg/L]	Flux [10^6kg/yr]	FN_Flux [10^6kg/yr]
1980	4.25	0.949	1.003	0.1154	0.106
1981	2.22	1.035	0.999	0.0675	0.108
1982	3.05	1.036	0.993	0.0985	0.11
1983	4.99	1.007	0.993	0.1329	0.112
1984	5.72	0.99	1.002	0.1597	0.114
1985	1.52	1.057	1.017	0.0489	0.116
1986	2.63	1.062	1.038	0.0903	0.119
1987	3.37	1.079	1.062	0.1142	0.122
1988	1.87	1.12	1.085	0.066	0.125
1989	5.61	1.055	1.105	0.1638	0.127
1990	4.01	1.115	1.125	0.1349	0.129
1991	2.75	1.172	1.143	0.098	0.13
1992	2.19	1.203	1.159	0.081	0.132
1993	3.73	1.215	1.173	0.1306	0.132
1994	5.48	1.144	1.187	0.1634	0.133
1995	2.41	1.266	1.201	0.0928	0.134
1996	6.24	1.134	1.213	0.198	0.135
1997	5.83	1.18	1.221	0.1884	0.136
1998	4.88	1.236	1.229	0.1593	0.137
1999	2.9	1.277	1.238	0.0919	0.138
2000	4.72	1.213	1.253	0.1627	0.139
2001	4.88	1.251	1.268	0.1655	0.14
2002	1.24	1.321	1.285	0.0483	0.141
2003	8.64	1.14	1.303	0.2664	0.143
2004	5.28	1.274	1.321	0.1832	0.144
2005	3.81	1.36	1.341	0.1444	0.146
2006	3.59	1.382	1.362	0.1409	0.147
2007	4.28	1.408	1.382	0.1593	0.149
2008	2.56	1.477	1.401	0.1008	0.149
2009	3.68	1.409	1.419	0.1328	0.149
2010	7.19	1.323	1.438	0.2236	0.149

Figure 28. A simple table produced in Microsoft Excel

12 Saving Plots

Plots can be saved from R as JPG, PNG, PDF, and Postscript files. JPG and PNG are easy to use in any number of programs (Microsoft Word or PowerPoint for example), but the images cannot be resized later. PDF and Postscript images are easily re-sizable.

There are three steps to saving plots. The first is to open the “device” (and declare the output type and file name). The second step is to execute the function just as you would when plotting to the screen, but no output will appear. The third step is to turn off the device. It is also possible to put many plots within the

same pdf. Some simple examples should demonstrate this easily:

```
jpeg("plotFlowSingle.jpg")
plotFlowSingle(1)
dev.off()

png("plotFlowSingle.png")
plotFlowSingle(1)
dev.off()

pdf("plotFlowSingle.pdf")
plotFlowSingle(1)
dev.off()

postscript("plotFlowSingle.ps")
plotFlowSingle(1)
dev.off()

#Many plots saved to one pdf:
pdf("manyPlots.pdf")
plotFlowSingle(1)
plotFlowSingle(2)
plotFlowSingle(3)
plotFlowSingle(4)
dev.off()
```

There are many additional options for each of these devices. See the R help files for more information. One useful option for the larger `fluxBiasMulti` graph is to adjust the height and width of the output. The output of `fluxBiasMulti` is larger than the default pdf or postscript devices. Therefore, specifying the height and width eliminates R having to re-size the graphic:

```
postscript("fluxBiasMulti.ps", height=10,width=8)
fluxBiasMulti()
dev.off()
```

References

- [1] Hirsch, R. M., Moyer, D. L. and Archfield, S. A. (2010), Weighted Regressions on Time, Discharge, and Season (WRTDS), with an Application to Chesapeake Bay River Inputs. JAWRA Journal of the American Water Resources Association, 46: 857-880. doi: 10.1111/j.1752-1688.2010.00482.x <http://onlinelibrary.wiley.com/doi/10.1111/j.1752-1688.2010.00482.x/full>
- [2] Sprague, L. A., Hirsch, R. M., and Aulenbach, B. T. (2011), Nitrate in the Mississippi River and Its Tributaries, 1980 to 2008: Are We Making Progress? Environmental Science & Technology, 45 (17): 7209-7216. doi: 10.1021/es201221s
- [3] Moyer, D.L., Hirsch, R.M., and Hyer, K.E. (2012), Comparison of Two Regression-Based Approaches for Determining Nutrient and Sediment Fluxes and Trends in the Chesapeake Bay Watershed: U.S. Geological Survey Scientific Investigations Report: 2012-5244, 118 p. <http://pubs.usgs.gov/sir/2012/5244/>
- [4] Rice, K.C., and Hirsch, R.M. (2012), Spatial and temporal trends in runoff at long-term streamgages within and near the Chesapeake Bay Watershed: U.S. Geological Survey Scientific Investigations Report: 2012-5151, 56 p. <http://pubs.usgs.gov/sir/2012/5151>