The USGSwsQWSR R package

Laura De Cicco[1], Steve Corsi[1], and Austin Baldwin[1]

[1] *United States Geological Survey*

January 9, 2014

Contents

1 Introduction to USGSwsQWSR package **2**

2 General Workflow **2**

3 Workflow Details **3**

3.1 Data Retrieval 3

3.2 Data Merging 5

3.3 Data Investigation 5

3.3.1 Narrow down investigation 5

3.3.2 Plot variables 6

3.4 Stepwise Regression 11

3.5 Model Analysis 11

A Getting Started in R **12**

A.1 New to R? 12

A.2 R User: Installing QWSR 12

1 Introduction to USGSwsQWSR package

The USGSwsQWSR package was designed to simplify the process of gathering water quality sample data and unit surrogate data, running a stepwise regression using the USGSwsQW censReg function, and analyzing those results. This vignette will first show a general overview workflow (2), then a more detailed description of the workflow with working examples (3).

2 General Workflow

```
library("USGSwsQWSR")

#Sample data included with package:
DTComplete <- StLouisDT
UV <- StLouisUV
QWcodes <- StLouisQWcodes
siteINFO <- StLouisInfo

investigateResponse <- "SuspSed"
transformResponse <- "lognormal"

DT <- DTComplete[c(investigateResponse,
                   getPredictVariables(names(UV)),
                   "decYear","sinDY","cosDY","datetime")]
DT <- na.omit(DT)

predictVariables <- names(DT)[-which(names(DT)
                  %in% c(investigateResponse,"datetime","decYear"))]


#Check predictor variables
predictVariableScatterPlots(DT,investigateResponse)

# Create 'kitchen sink' formula:
kitchenSink <- createFullFormula(DT,investigateResponse)

#Run stepwise regression with "kitchen sink" as upper bound:
returnPrelim <- prelimModelDev(DT,investigateResponse,kitchenSink,
                               "BIC", #Other option is "AIC"
                               transformResponse)

steps <- returnPrelim$steps
modelResult <- returnPrelim$modelStuff
```

```
modelReturn <- returnPrelim$DT.mod

# Analyze steps found:
plotSteps(steps,DT,transformResponse)
analyzeSteps(steps, investigateResponse,siteINFO)

# Analyze model produced from stepwise regression:
resultPlots(DT,modelReturn,siteINFO)
resultResidPlots(DT,modelReturn,siteINFO)

# Create prediction plots
predictionPlot(UV,DT,modelReturn,siteINFO=siteINFO)
```

3 Workflow Details

In this section, we will step through the basic workflow.

3.1 Data Retrieval

Data retrieval is currently supported by web service calls to the National Water Information Service (NWIS). The first step is to get the discrete sample data that the regressions are modeling. In this example, we will look at the St Louis River at Scanlon (USGS site ID 04024000). If we don't know the sample data that is available, we can use the whatQW function to discover that information.

```
library(USGSwsQWSR)
```

```
site <- "04024000"
QWcodes <- whatQW(site, minCount=20)
head(QWcodes)
```

```
   parameter_cd  startDate    endDate count service
25        00010 1964-10-28 2013-12-30   265      qw
26        00020 1974-10-30 2013-12-30   154      qw
28        00025 1981-10-20 2013-12-30   135      qw
32        00041 2010-10-07 2013-06-11    55      qw
34        00055 2010-10-07 2013-04-27    35      qw
37        00061 1960-07-28 2013-07-17   310      qw
```

Most likely, there will be a known set of parameters that are to be modeled. If the parameter codes for these analytes are known, the data from NWIS can be accessed directly with the function

importNWISqw. The following example shows the process, and then lists the column names returned in the QW dataframe.

```
pCodeQW <- c("00608","00613","00618","00631","00665",
             "00671","00940","62855","80154")
startDate <- "2011-03-17"
endDate <- ""
QW <- importNWISqw(site, params=pCodeQW,
                   begin.date=startDate, end.date=endDate)
```

```
names(QW)
```

```
 [1] "site_no"              "sample_dt"
 [3] "sample_tm"            "tzone_cd"
 [5] "medium_cd"            "Ammonia.N"
 [7] "Nitrite.N"            "Nitrate.N"
 [9] "NO2PlusNO3.N"         "Phosphorus_WW.P"
[11] "OrthoPhosphate.P"     "Chloride"
[13] "NitrogenTotal_WW.sum" "SuspSed"
[15] "datetime"
```

This brings the data in automatically as a 'qw' object. This means that censoring information is embeded within each data point. If any processing needs to be done to the data, it might be easier to import the raw data first, then convert to 'qw' objects with the makeQWObjects function.

```
QWRaw <- retrieveNWISqwData(site,pCodeQW,startDate,
                            endDate,expanded=TRUE)
QW <- makeQWObjects(QWRaw)
```

Next, the unit value data that will be used as surrogates for the analytes should be retrieved. If the parameters are not known, they can be discovered using the getDataAvailability function, filtering just the 'uv' (unit value) data:

```
UVcodes <- getDataAvailability(site)
UVcodes <- UVcodes[UVcodes$service == "uv",]
names(UVcodes)
```

```
 [1] "parameter_cd"       "statCd"
 [3] "startDate"          "endDate"
 [5] "count"              "service"
 [7] "parameter_group_nm" "parameter_nm"
 [9] "casrn"              "srsname"
[11] "parameter_units"
```

```
UVcodes$parameter_cd


[1] "00010" "00021" "00060" "00065" "00095" "00300" "00301"
[8] "00400" "63680"
```

Finally, the unit value data can be retrieved with the getMultipleUV function. Because of the potentially large amount of data being returned, the web service call is automatically split into individual parameter codes.

```
UVpCodes <- c("00010","00060","00095","00300","00400","63680")
UV <- getMultipleUV(site, startDate, endDate, UVpCodes)
```

```
names(UV)


 [1] "agency_cd"    "site_no"      "datetime"     "tz_cd"
 [5] "Wtemp"        "Wtemp_cd"     "Flow"         "Flow_cd"
 [9] "SpecCond"     "SpecCond_cd" "DO"           "DO_cd"
[13] "pH"           "pH_cd"        "Turb"         "Turb_cd"
```

3.2 Data Merging

We now need to merge the sample and continuous data into one dataframe. This is accomplished using the mergeDatasets function.

```
mergeReturn <- mergeDatasets(QW, UV, QWcodes)
DTComplete <- mergeReturn$DTComplete
QWcodes <- mergeReturn$QWcodes
```

The dataframe DTComplete contains a column of each of the discrete samples, and a column of the nearest (temporaly) unit value data. The function mergeDatasets has an argument called 'max.diff'. The default is set to '2 hours', meaning that if the sample and continious data timestamps do not match, the merge will take the closest continious data within 2 hours. This value can be changed, see ?mergeNearest for more options.

3.3 Data Investigation

3.3.1 Narrow down investigation

We now want to narrow our investigation down to one analyte. Let us look at Chloride. First we will want a dataframe DT with just chloride and the unit values. We will call these the 'prediciton values' because they will eventually be used to predict chloride.

```
investigateResponse <- "Chloride"
predictionVariables <- getPredictVariables(names(UV))

DT <- DTComplete[c(investigateResponse,
                   predictionVariables,
                   "decYear","sinDY","cosDY","datetime")]

names(DT)
```

```
 [1] "Chloride" "Wtemp"    "Flow"     "SpecCond" "DO"
 [6] "pH"       "Turb"     "decYear"  "sinDY"    "cosDY"
[11] "datetime"
```

For the regression, there can be no NA values in any of the columns. There are many ways in R to deal with this requirement. The easiest way to do it is remove any row that has any NA. This can be done as follows:

```
DT <- na.omit(DT)
```

There may be other situations where you want to remove a column that contains the majority of the missing data.

3.3.2 Plot variables

There are a few tools included in this package to explore the data before performing the regression.

```
plotQQTransforms(DT,investigateResponse)
```

```
predictVariableScatterPlots(DT,investigateResponse)
```

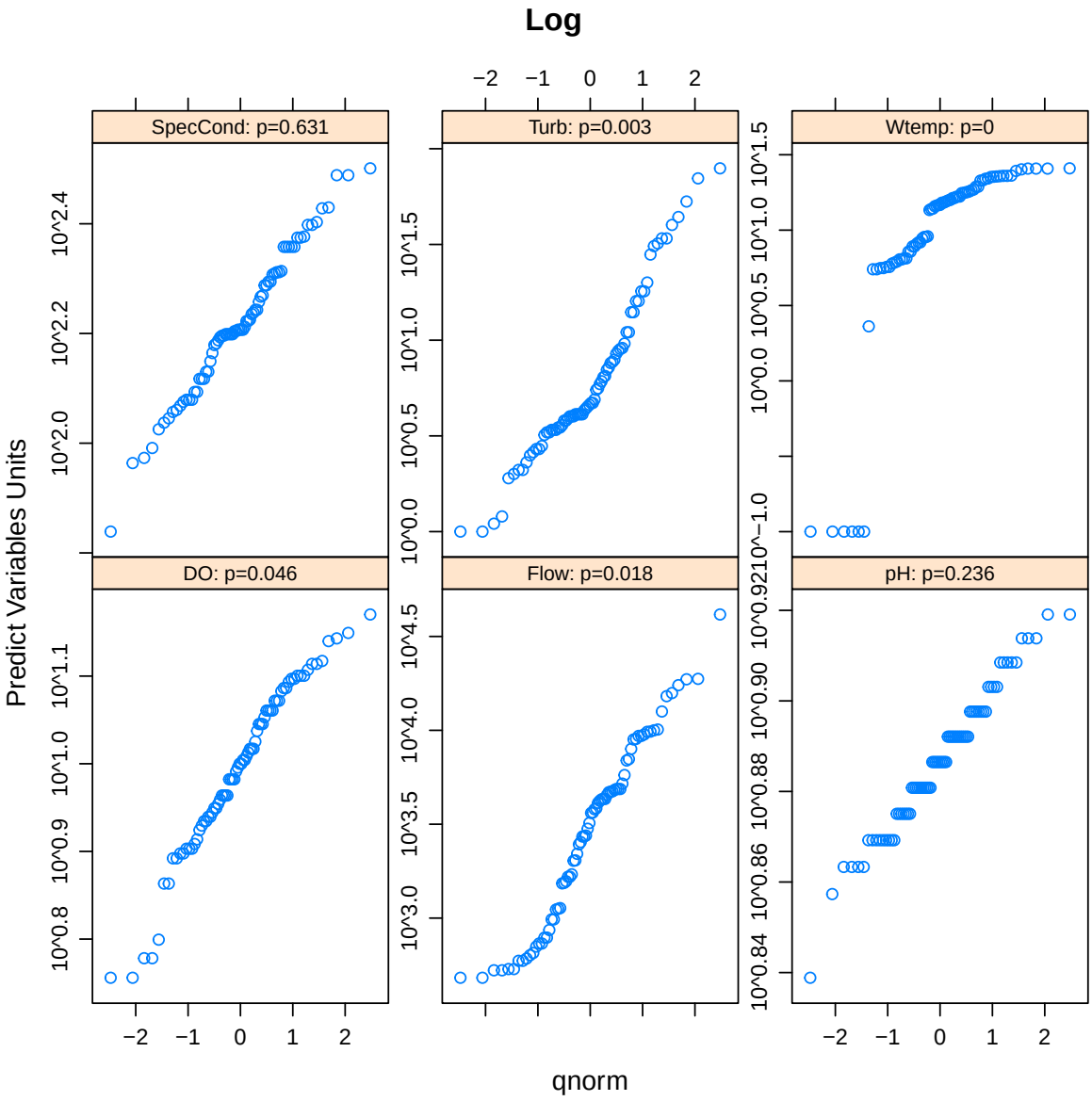

Figure 1: plotQQTransforms

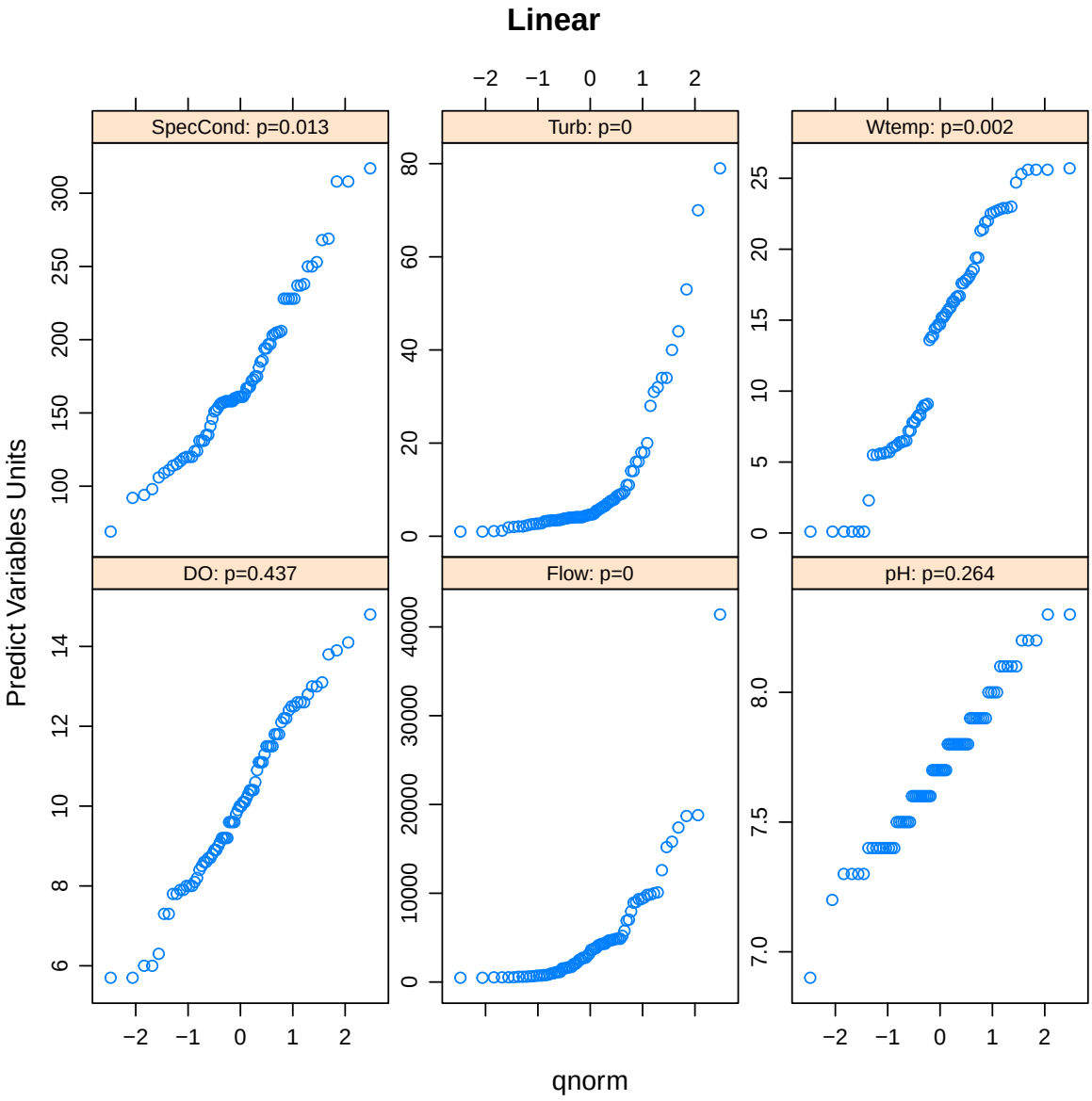

Figure 2: plotQQTransforms

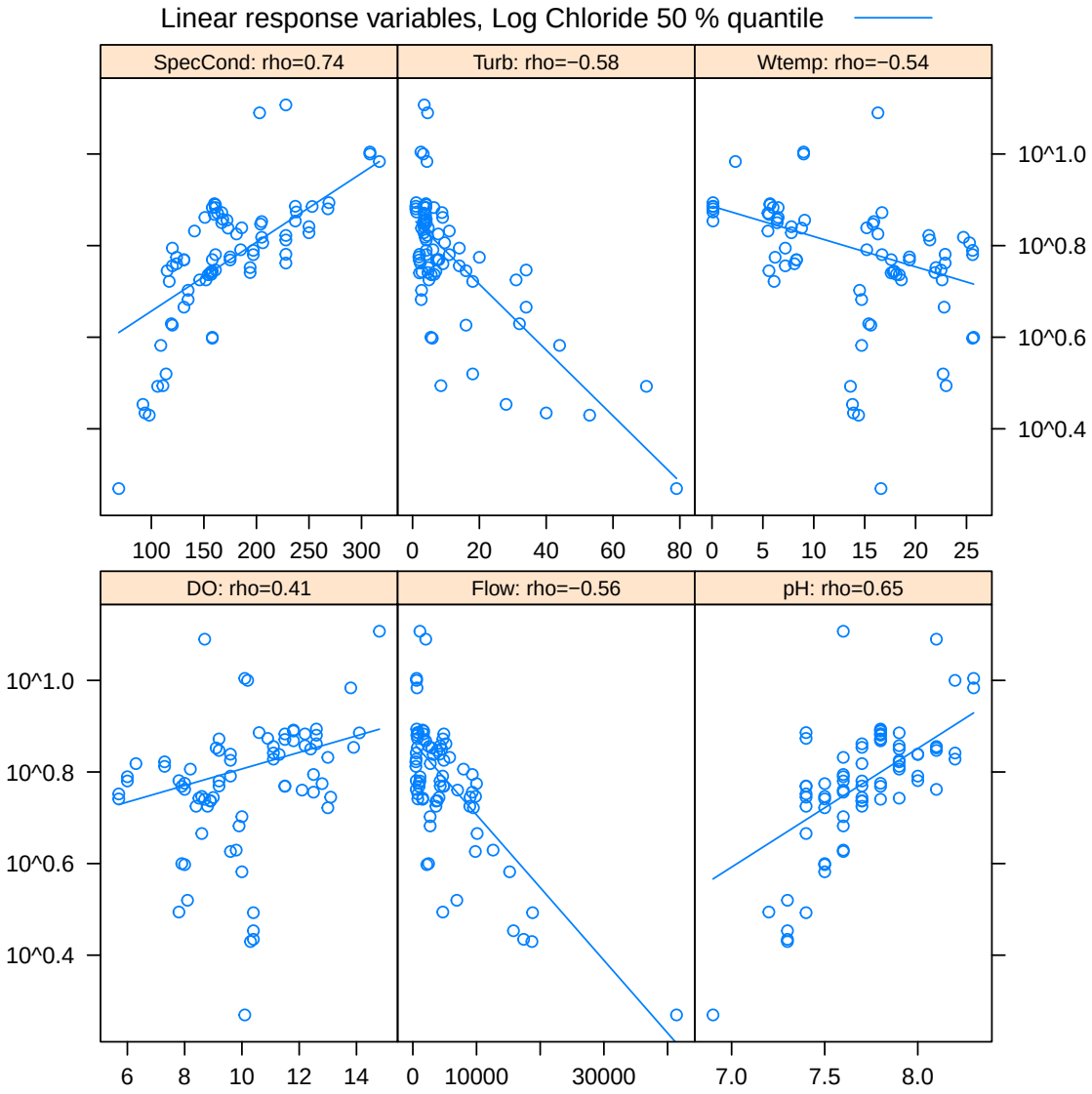

Figure 3: predictVariableScatterPlots

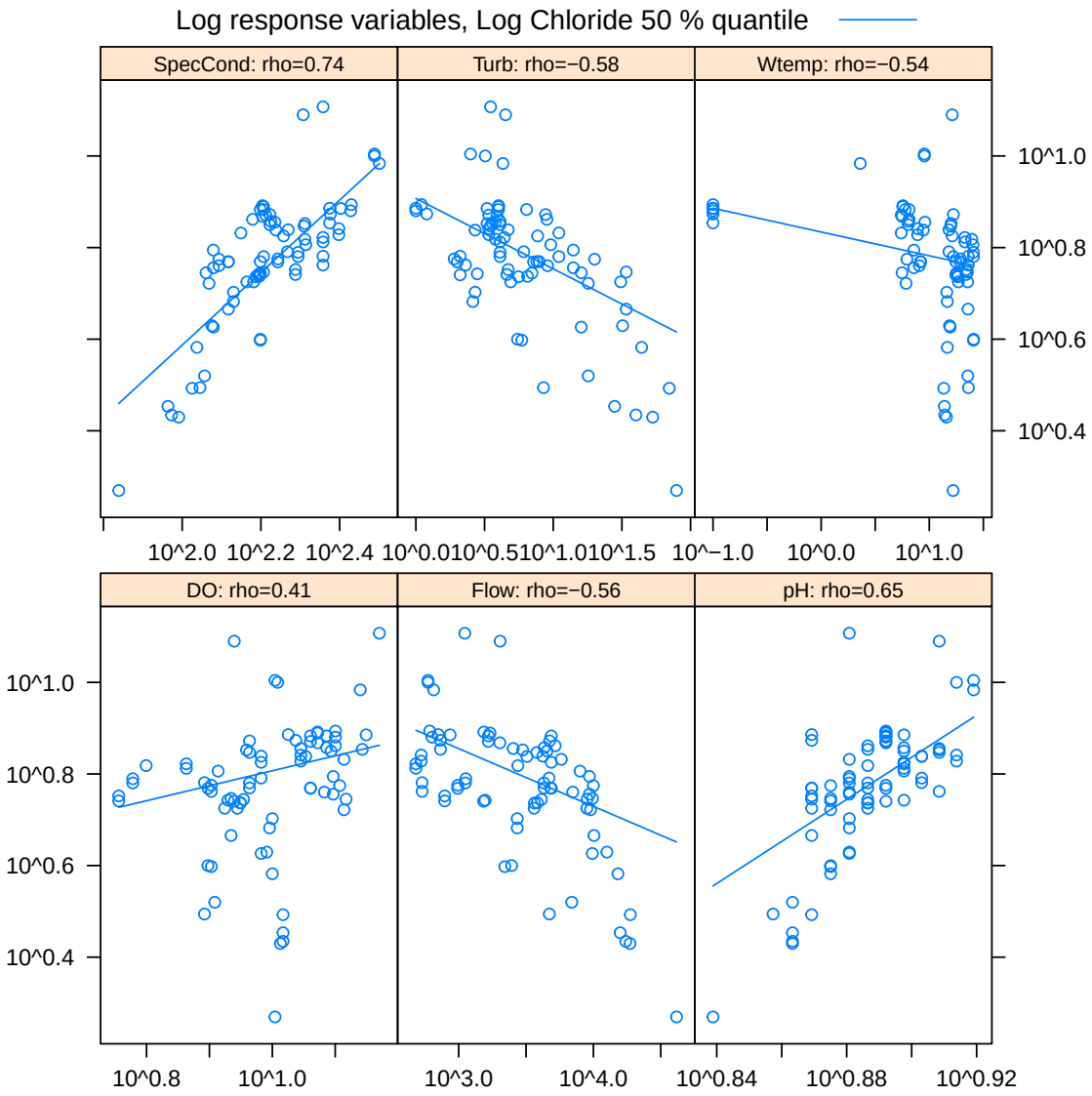

Figure 4: predictVariableScatterPlots

3.4 Stepwise Regression

3.5 Model Analysis

A Getting Started in R

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

A.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/`.

There are many options for running and editing R code, one nice environment to learn R is RStudio. RStudio can be downloaded here: `http://rstudio.org/`. Once R and RStudio are installed, the dataRetrieval package needs to be installed as described in the next section.

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.

```
library(USGSwsQWSR)
?plotSteps
```

To see the raw code for a particular code, type the name of the function:

```
plotSteps
```

A.2 R User: Installing QWSR

Before installing USGSwsQWSR, the dependent packages must be first be installed:

```
install.packages(c("XML", "lubridate", "akima", "KernSmooth",
                   "leaps", "car", "mvtnorm", "digest",
                   "relimp", "BSDA", "RODBC","memoise",
                   "boot","survival","splines","RColorBrewer",
                   "lattice","MASS"),
                 dependencies=TRUE)
install.packages(c("USGSwsBase","USGSwsData","dataRetrieval",
                   "USGSwsGraphs","USGSwsStats",
                   "USGSwsQW","USGSwsQWSR"), repos="http://usgs-r.github.com")
```

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

```
library(USGSwsQWSR)
```