ECHOInterface.R

Connor Brogan

March 10, 2018

## Summary

The purpose of the script ECHOInterface.R is threefold: 1. Generate a list of all facilities (active and inactive) involved in the effluent monitoring regulated by the Environmental Protection Agency (EPA) for a given state 2. Develop a list of active outfalls from the facility reporting discharge monitoring data 3. Compile discharge monitoring statistics for active outfalls This code accesses the EPA Enforcement and Compliance History Online (ECHO) RESTful web services through several primary queries to achieve these primary functions. Using an input state and date range of interest, it queries EHCO databases for a list of statewide facilities and navigates ECHO’s datastructure to extract relevant identifiers that can subsequently drive outfall and discharge montoring record (DMR) development. The output dataframe **a** of active and inactive facilites can be used to show facility position across the state, while output dataframe **FlowFrame** compiles relevant statistics about each outfall. This code also creates a unique outfall id, listed here as the *VPDESID*, to provide users a means of conglomerating info for relevant outfalls. This document serves as a step-by-step analysis of how the code works and identifies regions a user may need to adjust for their own testing. This code is continually checked to match difference in ECHO strucutre so check the Git for the most up to date version and please report errors to the primary contributors.

# Initialization

This portion of ECHOInterface.R serves as the primary inputs that drives ECHO queries. First, the R workspace is cleaned up by erasing objects already in the global environment. In doing so, potential errors are avoided in which users have defined functions with elements common to this script. The *XML* and *RCurl* libraries are called here to assist in data downloads and formatting. ECHO offers multiple formats including JSON, XML, and GeoJSON, but only XML downloads are demonstrated here. In this section, users must define their **state** of interest using it’s two letter code (demonstration is for Virginia, VA) and a date range of interest for DMR statistics. ECHO is limited to the past three years and any date outside of that range will be defaulted by ECHO to available data within the three year period. Optional lines are commented out using the pound sign that will automatically fill the end date with the current computer system date. It is important to note that the quality of ECHO data is not necessarily equal for all states. Data completeness and quality should be reviewed on ECHO’s webpage before using this script, starting with the documents offered [here](https://echo.epa.gov/resources/echo-data/about-the-data#completeness).

rm(list=ls())  
library(XML)  
library(RCurl)  
state<-"VA"  
startDate<-"01/01/2017"  
endDate<-"12/31/2017"  
#endDate<-Sys.Date()  
#endDate<-format(as.Date(endDate), "%m/%d/%Y")

# Facility Download

This portion of the script queries ECHO to develop a data frame **a** containing all active and inactive effluent monitoring facilities across teh state. This list is not exclusive to those monitoring discharge under the Clean Water Act, but instead represent a list of all facilities that are or have previously been moniroted under federal regulations include the CWA, CAA, SWDA, and RCRA. The facilities have a name, ECHO identifier (*SourceID*), and a physical adress.

uri\_query<-paste0("https://ofmpub.epa.gov/echo/cwa\_rest\_services.get\_facilities?output=XML&p\_st=",state,"&p\_tribedist=0")  
ECHO\_xml<-getURL(uri\_query)  
ECHO\_query<-xmlParse(ECHO\_xml)  
QID<-xmlToList(ECHO\_query)  
QID<-QID$QueryID  
uri\_summary<-paste0("https://ofmpub.epa.gov/echo/cwa\_rest\_services.get\_download?output=CSV&qid=",QID)  
a<-read.csv(uri\_summary,stringsAsFactors = F)  
rm(uri\_summary,uri\_query,ECHO\_query,ECHO\_xml,QID)  
glimpse(a)

## Observations: 1,952  
## Variables: 16  
## $ CWPName <chr> "ABB INCORPORATED - BLAND", "ABINGDON...  
## $ SourceID <chr> "VA0086355", "VAL026531", "VA0027162"...  
## $ CWPStreet <chr> "171 INDUSTRY DR", "21436 VANCE MILL ...  
## $ CWPCity <chr> "BLAND", "ABINGDON", "MAPPSVILLE", "M...  
## $ CWPState <chr> "VA", "VA", "VA", "VA", "VA", "VA", "...  
## $ CWPStateDistrict <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N...  
## $ CWPEPARegion <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3...  
## $ FacDerivedHuc <int> 5050002, 6010102, 2080109, 2080109, 2...  
## $ FacIndianSpatialFlg <chr> "N", "N", "N", "N", "N", "N", "N", "N...  
## $ CWPTotalDesignFlowNmbr <dbl> 0.010, NA, NA, NA, NA, NA, NA, NA, NA...  
## $ CWPActualAverageFlowNmbr <dbl> NA, NA, 0.009, 0.009, 0.020, 0.242, N...  
## $ ControlMeasure <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N...  
## $ ControlMeasureSchedule <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N...  
## $ Over80CountUs <int> 0, 0, 6, 2, 0, 0, 0, NA, 0, 0, 0, 0, ...  
## $ PctilePctpre1960Us <dbl> 27.1, 47.4, 83.3, 81.1, 18.6, 42.0, 2...  
## $ Subsector <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N...

No latitude and longitude coordinates are available for the facilities in this code. An optional secondary query can be run to give facility coordinates, as shown below. This will loop through each facility within **a** and add facility coordiantes. It requires the *jsonlite* package, which facilitates the download and formatting of javascript object notation. Note that coordinates are associated with a NAD 83 datum. Please note that this download will take a while to complete and it is recommended that you first narrow the list of facilities to those necessary in the intended geospatial analysis.

library(jsonlite)  
for (i in 1:length(a$CWPName)){  
 json\_file<-paste0("https://ofmpub.epa.gov/echo/dfr\_rest\_services.get\_dfr?output=JSON&p\_id=",a$SourceID[i])  
 json\_data<-fromJSON(txt=json\_file)  
 if(length(json\_data$Results$SpatialMetadata$Latitude83)>0){  
 a$Faclat[i]<-json\_data$Results$SpatialMetadata$Latitude83  
 a$Faclong[i]<-json\_data$Results$SpatialMetadata$Longitude83  
 } else {  
 a$Faclat[i]<-NA  
 a$Faclong[i]<-NA  
 }  
}

# Data downloads and formatting

The next primary section of this code sets up a *for* loop that will step through each facility to download all outfalls that discharge water during the timeframe of interest. By altering the parameter code, this code can be altered to analyze non-discharge metrics, such as effluent pH. This first section creates several vectors that track outfall flow, regulated limi values, identifiers, and reported statistics. These vectors are initialized here, empty and of length zero.

Flow<-0;Flow<-Flow[-1]  
Unit<-"";Unit<-Unit[-1]  
Limit<-0;Limit<-Limit[-1]  
VPDESID<-"";VPDESID<-VPDESID[-1]  
ECHOID<-"";ECHOID<-ECHOID[-1]  
feat\_num<-"";feat\_num<-feat\_num[-1]  
Code<-'';Code<-Code[-1]  
Coded<-'';Coded<-Coded[-1]

The below loop is demonstrated arbitrarily for the 1627th facility of data frame **a**, the Surry Power Station and Gravel Neck, Surry, VA (facility SourceID of VA0004090).No *for* loop is demonstrated here, just a step-by-step analysis of how the loop would run.

The first component of the *for* loop stores the relevant ECHO *SourceID* for that iteration of the loop. It then uses this facility’s identifier in tandem with the date range of interest to download DMR data for the facility from all of its discharging outfalls. It extracts all data that matches the paramter code of interest. Here, code 50050 is used, designating that we are interested in facility discharge. It sets zero values to all discharge records in which no discharge is noted. Other parameter codes should ignore this line. Additional information is available on the EHCO webpage and on [this page](https://echo.epa.gov/tools/data-downloads/icis-npdes-dmr-summary)

sourceID<-a$SourceID[i]  
print(paste("Processing SourceID: ",sourceID," (",i," of ",length(a$SourceID),")", sep=""))

## [1] "Processing SourceID: VA0004090 (1627 of 1952)"

uri\_effluent<-paste0("https://ofmpub.epa.gov/echo/eff\_rest\_services.download\_effluent\_chart?p\_id=",sourceID,"&start\_date=",startDate,"&end\_date=",endDate)  
b<-read.csv(uri\_effluent,stringsAsFactors = F)  
b<-b[b$parameter\_code==50050,]  
b$dmr\_value\_nmbr[b$nodi\_code %in% c('C','7')]<-0  
b$monitoring\_period\_end\_date<-as.Date(as.POSIXct(b$monitoring\_period\_end\_date,"EST",format='%m/%d/%Y'))  
glimpse(b[,1:10])

## Observations: 156  
## Variables: 10  
## $ activity\_id <dbl> 3600660200, 3600660200, 3600660200, 360...  
## $ npdes\_id <chr> "VA0004090", "VA0004090", "VA0004090", ...  
## $ version\_nmbr <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...  
## $ perm\_feature\_id <dbl> 3600092453, 3600092453, 3600092453, 360...  
## $ perm\_feature\_nmbr <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...  
## $ perm\_feature\_type\_code <chr> "EXO", "EXO", "EXO", "EXO", "EXO", "EXO...  
## $ perm\_feature\_type\_desc <chr> "External Outfall", "External Outfall",...  
## $ limit\_set\_id <dbl> 3600114413, 3600114413, 3600114413, 360...  
## $ limit\_set\_schedule\_id <dbl> 3600137132, 3600137132, 3600137132, 360...  
## $ limit\_id <dbl> 3601100798, 3601100798, 3601100798, 360...

Next, the length of DMR is evaluated and checked using the below line. This determines whether the facility is active or inactive by storing the number of records avaialble for discharge. If this value is zero, then all reported metrics are listed as *NA*. Otherwise, it initiates the rest of the loop to check for data through an if-statement not shown here. It should be noted that the *data\_length* variable is typically in units of months, but it largely inconsistent between DMRs as some report annually, monthly, quarterly, etc.

data\_length<-length(unique(b$monitoring\_period\_end\_date)); data\_length

## [1] 12

The next portion of the code extracts outfall identifiers for each active outfall for the given facility. It searches for all unique identifiers and then reformats them to be consistent with VPDES identifiers, combining *SourceID* and a three digit permanent feature number (stored in ECHO as *perm\_feature\_nmbr*). It adds leading zeroes to all numeric identifiers below three characters in length. Otherwise, it keeps the permanent feature number as is due to the few VPDES identifiers based on a lettering scheme instead of numbers. It stores these reformated identifiers under the vectors *features* which is later combined with facility *SourceID*.

featuresID<-unique(b$perm\_feature\_nmbr)#Store all outfalls  
features<-unique(b$perm\_feature\_nmbr)#Create a variable to store reformatted outfall IDs that will have leading zeroes where necessary  
for (j in 1:length(features)){  
 if(!is.na(as.numeric(features[j]))){  
 addedzeroes<-paste(rep(0,3-nchar(features[j])),collapse = '')#If the feature j is numeric, add leading zeroes such that the ID   
 features[j]<-paste0(addedzeroes,as.character(features[j]))#is three places long (for outfall ID like those in VPDES)  
 } else{  
 features[j]<-as.character(features[j])#If the feature j is not numeric, then no formatting is necessary  
 }  
}  
glimpse(features)

## chr [1:28] "001" "002" "050" "051" "052" "053" "101" "102" "103" ...

The bulk of ECHOInterface.R is contained in the following nested *for* loops. Each outfall at a given facility is analyzed by looking at the identifiers stored within *features*. The DMR is subset into *bspec* to shorten the data to just that from the selected outfall. ECHO DMRs are written to create separate entries for each reported statistic regarding discharge characteristics. The DMR subset *bspec* is queried for all unique reported statistics and the ECHO [statistical code](https://echo.epa.gov/system/files/REF_ICIS-NPDES_STATISTICAL_BASE.csv) is stored (i.e. MK for monthly average). Vectors are created to store values associated with each statistic as they are analyzed one by one in a *for* loop: *Codedi* stores the statistical code looked at in a particularly loop iteration (each is anlyzed indiviudally, but this help associate data with a statistic), *Flowi* will hold the median value of the statistic over the lenght of record, *Uniti* will hold the units associated with the discharge, and *Limiti* will hold the limit value associated with the permit (we find these are NOT very accurate). The entire loop is shown below, but sample data output is shown for the statistical codes and final flow values. As each statistical code is analyzed one-by-one, the median flow value is stored as well as any instances of unique units and limits associated with that code. The median value for the statistic was chosen as means were skewed by erroneously high values (likely typos) and no one sum could be developed over the time period without first devising a way to check if data was reported monthly, annually, etc. Median thus served as an easy middle-ground to provide representative data for discharge from the site. Occasionaly, limits present more than one value for a given statistic. This can occur due to a change in permit conditions or a typo. In these instances, the median limit value is stored and a warning message is output stating “More than one real limit found, only using median”. A longer description of the statistical code is stored in *Codedi*. Finally, the code, flow, unit, and limit value are added to larger vectors of similar names (*Flow*, *Unit*,*Limit*,*Coded*,*Code*). The outfall identifier is stored both indiviudally in the vector *feat\_num* as well as in *VPDESID*, where it is combined with the *SourceID*. To provide a facility identifier, *SourceID* is stored separatley as *ECHOID*. These are all colectivley stored in the data frame *FlowFrame*, which is simplified by removing all NA values (or inactive outfals). The output data frame *FlowFrame* can be used with **AnalysisCode.R** to provide summary data frames of facility activity, state-wide statistic counts, and aggregated outfall lists that group outfalls by *VPDESID* to show all relevant statistics at a single outfall.

for (k in 1:length(features)){  
 outfall<-as.character(features[k])  
 bspec<-b[b$perm\_feature\_nmbr==featuresID[k],]  
 codes<-unique(bspec$statistical\_base\_code)  
 Flowi<-numeric(length(codes))  
 Uniti<-numeric(length(codes))  
 Limiti<-numeric(length(codes))  
 Codedi<-unique(bspec$statistical\_base\_code)  
 for (j in 1:length(codes)){  
 Flowi[j]<-median(bspec$dmr\_value\_nmbr[bspec$statistical\_base\_code==codes[j]],na.rm=T)  
 Uniti[j]<-unique(bspec$standard\_unit\_desc[bspec$statistical\_base\_code==codes[j]])  
 LimitswNA<-unique(bspec$limit\_value\_nmbr[bspec$statistical\_base\_code==codes[j]])  
 if(length(LimitswNA)>1){  
 if(length(LimitswNA[!is.na(LimitswNA)])>1){  
 warning("More than one real limit found, only using median")  
 Limiti[j]<-median(bspec$limit\_value\_nmbr[bspec$statistical\_base\_code==codes[j]&bspec$limit\_end\_date==max(bspec$limit\_end\_date[bspec$statistical\_base\_code==codes[j]],na.rm=T)],na.rm=T)  
 }else{  
 Limiti[j]<-LimitswNA[!is.na(LimitswNA)]   
 }  
 }else{  
 Limiti[j]<-LimitswNA  
 }  
 Codedi[j]<-unique(bspec$statistical\_base\_short\_desc[bspec$statistical\_base\_code==codes[j]])  
 }  
 Flow<-c(Flow,Flowi)  
 Unit<-c(Unit,Uniti)  
 Limit<-c(Limit,Limiti)  
 Code<-c(Code,codes)  
 Coded<-c(Coded,Codedi)  
 feat\_num<-c(feat\_num,rep(outfall,length(codes)))  
 VPDESID<-c(VPDESID,paste0(sourceID,rep(outfall,length(codes))))  
 ECHOID<-c(ECHOID,rep(sourceID,length(codes)))  
}  
  
FlowFrame<-data.frame(ECHOID,VPDESID,feat\_num,Flow,Unit,Limit,Code,Coded)  
FlowFrame<-FlowFrame[!is.na(FlowFrame$VPDESID),]  
glimpse(FlowFrame)

## Observations: 56  
## Variables: 8  
## $ ECHOID <fct> VA0004090, VA0004090, VA0004090, VA0004090, VA0004090...  
## $ VPDESID <fct> VA0004090001, VA0004090001, VA0004090002, VA000409000...  
## $ feat\_num <fct> 001, 001, 002, 002, 050, 050, 051, 051, 052, 052, 053...  
## $ Flow <dbl> 1995.760000, 2078.950000, 0.026150, 0.026150, 0.77550...  
## $ Unit <fct> MGD, MGD, MGD, MGD, MGD, MGD, MGD, MGD, MGD, MGD, MGD...  
## $ Limit <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...  
## $ Code <fct> MK, DD, MK, DD, MK, DD, DD, MK, DD, MK, DD, MK, DD, M...  
## $ Coded <fct> MO AVG, DAILY MX, MO AVG, DAILY MX, MO AVG, DAILY MX,...