Analysis Code

Connor Brogan

February 8, 2018

## Summary

This code is intended to take inputs from external VPDES databases and combine them with outputs from ECHO, creating merged datasets. It formats and organizes data and searches for lat/long values to assign to VA Hydro MPs/Facs and those from ECHO. Primary outputs are then reformatted to create two primary datasets: VA Hydro facility “**Hydro**” that have corrected geometries and “**AllFacs**” which offers discharge summaries for every state facility. Other data output includes: “*All*” given a merged version of VPDES and ECHO data on an outfall level, reordered for convenience; “*FacSummary*” offers information on the variety of statistics reported by state DMR records and adds a total stat count for each facility onto “**AllFacs**”. The primary purpose of this code, however, is the output data frame “**AllFacs**”. By aggregating outfall DMRs to the facility level, AnalysisCode.R generates a facility summary that shows every available statistic reported in their DMRs. This makes cross-facility comparison possible by more easily giving fields for reference.

# Initialization

The following libraries are called within AnalysisCode. The *foreign* library allows for easy manipulation of \*.dbf file types. It’s ‘read.dbf’ function is called repeatedly in data introduction and writing to retrieve/write data/results. *rgdal* is a package designed to allow users to extract files from file geodatabases, like those create in Esri ArcGIS. It is necessary here to extract the VPDES shapefile, which offers preliminary lat/long data of many outfalls in the state. *dplyr* is a data manipulation package that can speed up group, summarize, and ordering processes used several times here. It’s piping function ‘%>%’ is useful to feed data along a series of functions without needing to call it each time.. *XML* is a package that downloads and reads XML file types from webpages. It is used here to access ECHO rest services to download state facilities. *RCurl* is a package that helps download and interact with web-based content. This code makes use of its ‘getURL’ function to read data from ECHO. *readxl* is a self-explanatory package that enables direct reading of excel files into R used here to read VPDES data. *httr* is similar to RCurl. Both are called here due to the fact

library(foreign)  
library(rgdal)  
library(dplyr)  
library(XML)  
library(RCurl)  
library(readxl)  
library(httr)

After libraries are added, preliminary data is drawn in based on user inputs for the variable *state* (which drives the facility download query). It is necessary to load several data frames into the code manually. *FlowFrame* contains every outfall and every reported value for discharge across the state. It is generated separatley in **ECHOInterface.R** and needs to be loaded here by first writing it from ECHOInterface with ‘write.csv’. *FlowFrameNew* is the flow frame generated in 2017 and is referenced here to show modern values for outfall counts. Finally, the VA Hydro withdrawal facility list *Hydro* should be accessed manually. This can be downloaded from the following site, but downloads within R often timeout or expire: (<http://deq1.bse.vt.edu/d.bet/vahydro_facilities>) The *path* variable should set the data path to an available workspace. AnalysisCode.R will write out several files as well as store downloads from VPDES and ECHO. Manual inputs are seen below, using 2016 as an example year:

state<-"VA"  
path<-"C:/Users/connorb5/Desktop/USGS Testing"  
FlowFrame<-read.csv(paste0(path,"/2016 ECHO/FlowFrameNoDis2016.csv"),stringsAsFactors = F)  
FlowFrameNew<-read.csv(paste0(path,"/2017 ECHO/FlowFrame.csv"),stringsAsFactors = F)  
#Hydro<-read.csv('http://deq1.bse.vt.edu/d.bet/vahydro\_facilities',stringsAsFactors = F)  
Hydro<-read.csv(paste0(path,"/vahydro\_facilities.csv"),stringsAsFactors = F)

First, download facilities from ECHO and store arbitrarily as a after setting state and path. ECHO is queried through its Clean Water Act rest services. Output XML files are downlaoded and reformatted using *XML* and *RCurl* libraries. ECHO generates a query ID *QID* that can be used to find and download the required facility list.

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)  
head(a[,c(1,5,10,12,13)])

## SourceID CWPName FacPopDen FacLat FacLong  
## 1 DC0022004 MIRANT POTOMAC RIVER L.L.C. 6716.34 38.81953 -77.04167  
## 2 DCNOE3034 WFS-DCA 8008.54 38.82436 -77.06965  
## 3 ILG870595 SUMMIT HELICOPTORS NA 37.36209 -79.89973  
## 4 ILR007054 UPS GROUND FREIGHT CO INC 4141.88 37.52595 -77.44713  
## 5 MD0056464 FAIRVIEW BEACH WWTP 165.02 38.32933 -77.23566  
## 6 MD0064556 DUMFRIES TERMINAL 1368.04 38.56603 -77.29891

Now, access VPDES to download a list of active outfalls and separate out the individual permits as these are the only permits that require discharge monitor reporting. Data is downloaded from the VPDES mapper database as a zipped file. This section of code downloads this file as a temporary file before extracting its contents, a geodatabase names “VPDES\_Geodatabase.gdb”. Using *rgdal* the desired layer of VPDES outfalls is extracted. Columns containing data for facility and outfall identifiers are renamed after those in ECHO.

temp<-tempfile()  
download.file("http://www.deq.virginia.gov/mapper\_ext/GIS\_Datasets/VPDES\_Geodatabase.zip",temp)  
unzip(temp,exdir=path)  
VPDES<-as.data.frame(readOGR(paste0(path,"/VPDES\_Geodatabase.gdb"),layer="VPDES\_OUTFALLS"))

## OGR data source with driver: OpenFileGDB   
## Source: "C:/Users/connorb5/Desktop/USGS Testing/VPDES\_Geodatabase.gdb", layer: "VPDES\_OUTFALLS"  
## with 9609 features  
## It has 14 fields

names(VPDES)[names(VPDES)=="OUTFALL\_ID"]<-'VPDESID'  
VPDES\_IP<-VPDES[VPDES$VAP\_TYPE=='VPDES\_IP',]  
names(a)[1]<-"VAP\_PMT\_NO"  
head(VPDES\_IP[,c(1,9,10,11,13,14,15,16)])

## VPDESID OUTFALLNO VAP\_PMT\_NO VAP\_TYPE  
## 1 VA0000248004 VA0000248004 VA0000248 VPDES\_IP  
## 2 VA0000248005 VA0000248005 VA0000248 VPDES\_IP  
## 3 VA0000248006 VA0000248006 VA0000248 VPDES\_IP  
## 4 VA0000248007 VA0000248007 VA0000248 VPDES\_IP  
## 5 VA0000248012 VA0000248012 VA0000248 VPDES\_IP  
## 6 VA0000248014 VA0000248014 VA0000248 VPDES\_IP  
## FAC\_NAME VAP\_MAJOR\_MINOR coords.x1  
## 1 US Army - Radford Army Ammunition Plant M -8967558  
## 2 US Army - Radford Army Ammunition Plant M -8967434  
## 3 US Army - Radford Army Ammunition Plant M -8965703  
## 4 US Army - Radford Army Ammunition Plant M -8964435  
## 5 US Army - Radford Army Ammunition Plant M -8964466  
## 6 US Army - Radford Army Ammunition Plant M -8964404  
## coords.x2  
## 1 4464925  
## 2 4465274  
## 3 4466128  
## 4 4465740  
## 5 4464731  
## 6 4465352

Finally, download a key to statistical codes from ECHO that can help reveal the meaning of various outfall statistics. Also get maximum design flows from routinely published excel files from VPDES. These contain maximum design discharges, facility names, and identifiers but lack any temporal data. Finish by removing uncessary data from the global environment. It should be noted that VPDESFlows also contains facility manager contact information that can be included in NWIS exports as ancillary data

CodeKey<-read.csv("https://echo.epa.gov/system/files/REF\_ICIS-NPDES\_STATISTICAL\_BASE.csv",stringsAsFactors = F,na.strings = 'BLANK')  
GET('http://www.deq.virginia.gov/Portals/0/DEQ/Water/PollutionDischargeElimination/VPDES%20Spreadsheets/VPDES%20Active%20IP%20Nov%202017.xls?ver=2017-11-14-152041-490', write\_disk(temp <- tempfile(fileext = ".xls")))

## Response [http://www.deq.virginia.gov/Portals/0/DEQ/Water/PollutionDischargeElimination/VPDES%20Spreadsheets/VPDES%20Active%20IP%20Nov%202017.xls?ver=2017-11-14-152041-490]  
## Date: 2018-02-09 20:21  
## Status: 200  
## Content-Type: application/vnd.ms-excel  
## Size: 1.08 MB  
## <ON DISK> C:\Users\connorb5\AppData\Local\Temp\RtmpOw3QFC\file1c3877a47b.xls

VPDESFlows <- read\_excel(temp,skip=9)  
VPDESFlows<-VPDESFlows[!is.na(VPDESFlows$Facility),]  
rm(uri\_summary,uri\_query,ECHO\_query,ECHO\_xml,QID,state,temp)  
head(CodeKey[,1:2])

## STATISTICAL\_BASE\_CODE STATISTICAL\_BASE\_SHORT\_DESC  
## 1 1A 1DA GEO  
## 2 1C 12M D AV  
## 3 1D 12MO AVG  
## 4 1E 12MTDFWA  
## 5 1F 120DA AV  
## 6 1G 180DARME

head(VPDESFlows[,c(1,2,13,14)])

## # A tibble: 6 x 4  
## Owner Facility `FIPS City / Cou~ `Design Flow Nu~  
## <chr> <chr> <chr> <chr>   
## 1 US Army US Army - Radford ~ Montgomery County 1.0700   
## 2 Celanese Acetate~ Celanese Acetate L~ Giles County 70.0000   
## 3 American Electri~ APCO - Glen Lyn Giles County 357.0000   
## 4 Lhoist North Ame~ Lhoist North Ameri~ Giles County 5.2500   
## 5 American Electri~ American Electric ~ Russell County 4.8400   
## 6 Goodyear Tire & ~ Goodyear Tire and ~ Danville City 0.1300

# Data formatting

To begin formatting the ECHO data, AnalysisCode first adds the design flows from *VPDESFlows* to the *VPDES\_IP* data frame. With this, a single data frame is created that contains all relevant data from VPDES. This will later be merged with the *FlowFrame* data frames from ECHO. The following code block searches the *VPDESFlows* data frame for flow data pertaining to each facility within *VPDES\_IP* by searching the renamed facility ID field, “VAP\_PMT\_NO”. If it does not find data, it writes out NA for that facility.

for (i in 1:length(VPDES\_IP$VAP\_PMT\_NO)){  
 VPDES\_IP$DesFlow[i]<-NA  
 if (length(VPDESFlows$`Design Flow Null`[VPDESFlows$`Permit Number`==VPDES\_IP$VAP\_PMT\_NO[i]])>0){  
 VPDES\_IP$DesFlow[i]<-VPDESFlows$`Design Flow Null`[VPDESFlows$`Permit Number`==VPDES\_IP$VAP\_PMT\_NO[i]]   
 }  
 VPDES\_IP$TotalFlow[i]<-NA  
 if (length(VPDESFlows$`Total Flow Null`[VPDESFlows$`Permit Number`==VPDES\_IP$VAP\_PMT\_NO[i]])>0){  
 VPDES\_IP$TotalFlow[i]<-VPDESFlows$`Total Flow Null`[VPDESFlows$`Permit Number`==VPDES\_IP$VAP\_PMT\_NO[i]]   
 }  
}  
head(VPDES\_IP[,c(1,9,10,13,17,18)])

## VPDESID OUTFALLNO VAP\_PMT\_NO  
## 1 VA0000248004 VA0000248004 VA0000248  
## 2 VA0000248005 VA0000248005 VA0000248  
## 3 VA0000248006 VA0000248006 VA0000248  
## 4 VA0000248007 VA0000248007 VA0000248  
## 5 VA0000248012 VA0000248012 VA0000248  
## 6 VA0000248014 VA0000248014 VA0000248  
## FAC\_NAME DesFlow TotalFlow  
## 1 US Army - Radford Army Ammunition Plant 1.0700 28.0000  
## 2 US Army - Radford Army Ammunition Plant 1.0700 28.0000  
## 3 US Army - Radford Army Ammunition Plant 1.0700 28.0000  
## 4 US Army - Radford Army Ammunition Plant 1.0700 28.0000  
## 5 US Army - Radford Army Ammunition Plant 1.0700 28.0000  
## 6 US Army - Radford Army Ammunition Plant 1.0700 28.0000

A column can then be added onto the ECHO data to display the number of outfalls from the most recent data frame, *FlowFrameNew*. This looks at each facility in *a* and searches for their equivalent within *FlowFrameNew*. It then counts the number of unique outfall IDs for the facility and stores the result.

for (i in 1:length(a$VAP\_PMT\_NO)){  
 a$ECHO2017Outfalls[i]<-length(unique(FlowFrameNew$VPDESID[FlowFrameNew$ECHOID==a$VAP\_PMT\_NO[i]]))  
}

Now, the ECHO data contained in *FlowFrame* needs to be reformatted. First, AnalysisCode.R extracts the four important data columns of ‘VPDESID’, ‘Flow’, ‘Limit’, and ‘Code’. Following this, the data frame is “reshaped”. In other words, the data frame is transposed such that each type of statistic is given it’s own column within *FlowFrameFlipped*. Each row represents an outfall such that each row shows every statistic reported for a given outfall. A facility ID is added to each outfall by assigning its searching and assigning a corresponding ‘ECHOID’ from each outfall’s ‘VPDESID’.

FlowFrameFlipped<-FlowFrame[,c('VPDESID','Flow','Limit','Code')]  
FlowFrameFlipped<-reshape(FlowFrameFlipped,idvar='VPDESID',timevar = 'Code',direction='wide')  
for (i in 1:length(FlowFrameFlipped$VPDESID)){  
 FlowFrameFlipped$ECHOID[i]<-as.character(FlowFrame$ECHOID[FlowFrame$VPDESID==FlowFrameFlipped$VPDESID[i]][1])  
}  
head(FlowFrameFlipped[1:4,1:5])

## VPDESID Flow.DD Limit.DD Flow.MK Limit.MK  
## 1 DC0022004001 0.00 NA 0 NA  
## 3 DC0022004101 0.00 NA 0 NA  
## 5 DC0022004102 0.00 NA 0 NA  
## 7 MD0064556001 19066.75 NA NA NA

# Merging VPDES and ECHO

Before we can beging merging data from VPDES to ECHO, AnalysisCode.R defines a few functions to better track discharge data. The function *plus* works similarly to *sum()*. However, it is built to treat NA data differently. Vecotrs containing all NA are reported as a sum of NA; otherwise, if the vector is only partially NA, it treats NA as zero and computes the sum (identical to *sum(,r=na.rm=TRUE)*), In this manner, the function can add all statistics for a given outfall and report the sum. If that statistic is not reported, NA is returned ensuring the code does no misrepresent missing discharge data. The second function, *NAcount*, takes in a vector and counts the number of NA values. It takes advantage of R’s dual-storage of binary operators as TRUE/FALSE and 1/0. By summing a series of TRUE/FALSE, a value is returned representing the number of TRUE present. By checking for NAs, TRUE is only retunred when data is missing. Thus, the sum shows the number of NAs within the input vector. This is useful to demonstrate how many outfalls at a given facility are reporting each statistic

plus<-function(x){  
 if(all(is.na(x))){  
 c(NA)  
 }else{  
 sum(x,na.rm = TRUE)}  
}  
NAcount<-function(x){  
 sum(is.na(x))  
}

To complete the merge, Analysis.R adds all columns of *FlowFrameFlipped* to those of *VPDES\_IP* based on their common identifier, the “VPDESID”. After the merge is completed, a column is created to store a central “FacilityID”. This ID is that for the facility given by ECHO or VPDES. AnalysisCode.R assumes this data is present in VPDES and searches for a valid entry of “VAP\_PMT\_NO” for that outfall. If this returns NA, then it looks for an ID from ECHO. Facility IDs are equal across databases so both entries are equally valid. Data is reclassified as numeric or character to prevent treatment as factors.

All<-merge(VPDES\_IP,FlowFrameFlipped,by="VPDESID",all=T)  
for (i in 1:length(All$VPDESID)){  
 if(is.na(All$VAP\_PMT\_NO[i])){  
 All$FacilityID[i]<-as.character(All$ECHOID[i])  
 }else{  
 All$FacilityID[i]<-as.character(All$VAP\_PMT\_NO[i])  
 }  
}  
All$VAP\_PMT\_NO<-as.character(All$VAP\_PMT\_NO)  
All$DesFlow<-as.numeric(All$DesFlow)  
All$TotalFlow<-as.numeric(All$TotalFlow)  
colnames(All)

## [1] "VPDESID" "INSERTED\_BY" "INSERTED\_DATE"   
## [4] "CHANGED\_BY" "CHANGED\_DATE" "REFERENCE\_POINT"   
## [7] "VERIFYDATE" "VERIFIEDBY" "OUTFALLNO"   
## [10] "VAP\_PMT\_NO" "VAP\_TYPE" "VAP\_TYPE\_ALT\_VERSION"  
## [13] "FAC\_NAME" "VAP\_MAJOR\_MINOR" "coords.x1"   
## [16] "coords.x2" "DesFlow" "TotalFlow"   
## [19] "Flow.DD" "Limit.DD" "Flow.MK"   
## [22] "Limit.MK" "Flow.AB" "Limit.AB"   
## [25] "Flow.AF" "Limit.AF" "Flow.MN"   
## [28] "Limit.MN" "Flow.ET" "Limit.ET"   
## [31] "Flow.SA" "Limit.SA" "Flow.WC"   
## [34] "Limit.WC" "Flow.MB" "Limit.MB"   
## [37] "Flow.NB" "Limit.NB" "Flow.MS"   
## [40] "Limit.MS" "Flow.WA" "Limit.WA"   
## [43] "Flow.DB" "Limit.DB" "Flow.DC"   
## [46] "Limit.DC" "Flow.IA" "Limit.IA"   
## [49] "Flow.1I" "Limit.1I" "Flow.MU"   
## [52] "Limit.MU" "Flow.M6" "Limit.M6"   
## [55] "Flow.A1" "Limit.A1" "Flow.MA"   
## [58] "Limit.MA" "Flow.3E" "Limit.3E"   
## [61] "Flow.3C" "Limit.3C" "Flow.AA"   
## [64] "Limit.AA" "ECHOID" "FacilityID"

Now that the data from VPDES and ECHO has been merged, the next step is to create a data frame *AllFacs* that aggregates information for all of the different outfalls up to the facility level. This is accomplished through the use of the *dplyr* library, grouping all entries by facility ID and using the plus function to sum all statistics for each outfall. Once the facility level aggregation is accomplished, the data is reorganized such that all of the statistics appear in the later columns. This makes *AllFacs* easier to read by organizing the data frame to first show facility identifiers and then statistics.

AllFacs<-as.data.frame(All %>% group\_by(FacilityID) %>% summarize\_at(vars(c(17,18,19:64)),funs(plus,NAcount)))

## Warning: package 'bindrcpp' was built under R version 3.3.3

headers<-as.character(unique(FlowFrame$Code))  
order<-numeric(0)  
for (i in 1:length(headers)){  
 orderi<-grep(headers[i],colnames(AllFacs))  
 order<-c(order,orderi)  
}  
allcols<-seq(1,length(colnames(AllFacs)))  
order<-c(allcols[!(allcols %in% order)],order)  
AllFacs<-AllFacs[,order]  
head(AllFacs[,1:5])

## FacilityID DesFlow\_plus TotalFlow\_plus DesFlow\_NAcount TotalFlow\_NAcount  
## 1 DC0022004 NA NA 3 3  
## 2 MD0064556 NA NA 1 1  
## 3 VA0000248 22.47 588.00 0 0  
## 4 VA0000299 980.00 980.00 0 0  
## 5 VA0000370 4641.00 4639.70 0 0  
## 6 VA0000523 26.25 136.83 0 0

To make *AllFacs* a more useful summary data frame, AnalysisCode.R attaches other pieces of summary information to each facility using the folliwng script. First, it uses a facility’s “FacilityID” column to drive a query that searches the list of ECHO facilities in *a* for a facility name. If this fails or returns NA, then it checks VPDES. It creates a column “SourceData” to reveal where information was pulled. Next, it counts the number of outfalls reporting information. The code then looks for the number of outfalls reporting data for the facility within ECHO and VPDES. It stores the total number of outfalls, the number present in *FlowFrameNew*, the number from *FlowFrame* (now in *All*), and the number from *VPDES\_IP* (now in *All*). Finally, it assigns a latitude and lognitude to each facility. It draws these values from matching facilities in ECHO (from *a*) or VPDES (from *All*) based on the “SourceData” column described earlier.

for (i in 1:length(AllFacs$FacilityID)){  
 AllFacs$FacilityName[i]<-NA  
 if(length(a$CWPName[a$VAP\_PMT\_NO==AllFacs$FacilityID[i]])>0){  
 AllFacs$SourceData[i]<-'ECHO'  
 AllFacs$FacilityName[i]<-a$CWPName[a$VAP\_PMT\_NO==AllFacs$FacilityID[i]]  
 }  
 if(is.na(AllFacs$FacilityName[i])){  
 AllFacs$FacilityName[i]<-as.character(All$FAC\_NAME[All$VAP\_PMT\_NO==AllFacs$FacilityID[i]])[1]  
 AllFacs$SourceData[i]<-'VPDES'  
 }  
 AllFacs$ECHOOutfalls[i]<-sum(!(is.na(All$ECHOID[All$FacilityID==AllFacs$FacilityID[i]])))  
 AllFacs$TotalOutfalls[i]<-length(All$FacilityID[All$FacilityID==AllFacs$FacilityID[i]])  
 AllFacs$VPDESOutfalls[i]<-length(VPDES$VAP\_PMT\_NO[VPDES$VAP\_PMT\_NO==AllFacs$FacilityID[i]])  
 AllFacs$ECHO2017Outfalls[i]<-length(unique(FlowFrameNew$VPDESID[FlowFrameNew$ECHOID==AllFacs$FacilityID[i]]))  
}  
for (i in 1:length(AllFacs$FacilityID)){  
 if(AllFacs$SourceData[i]=='ECHO'){  
 AllFacs$lat[i]<-a$FacLat[a$VAP\_PMT\_NO==AllFacs$FacilityID[i]]  
 AllFacs$lon[i]<-a$FacLong[a$VAP\_PMT\_NO==AllFacs$FacilityID[i]]  
 }else{  
 AllFacs$lat[i]<-as.numeric(All$coords.x1[All$VAP\_PMT\_NO==AllFacs$FacilityID[i]])[1]  
 AllFacs$lon[i]<-as.numeric(All$coords.x1[All$VAP\_PMT\_NO==AllFacs$FacilityID[i]])[1]  
 }  
}

Again, AnalysisCode.R reorganizes the *AllFacs* data frame to make sure all of this summary information is now contained at the beginning of the data frame. This makes *AllFacs* easier to read and visualize if opened in text files, Excel, GIS, etc. It is important to note that these may require manual adjustment if the data is amended in the future to prevent certain columns from being deleted or pushed to the back. “order” below is defined as the facility identifier, then those summary columns just created, and finally followed by every other column.

order<-c(1,seq(length(colnames(AllFacs))-7,length(colnames(AllFacs))),seq(2,length(colnames(AllFacs))-8))  
AllFacs<-AllFacs[,order]  
AllFacs<-AllFacs[order(AllFacs$Flow.MK\_plus-AllFacs$DesFlow\_plus,decreasing=T),]  
head(AllFacs[1:5])

## FacilityID FacilityName SourceData  
## 354 VA0057576 DOMINION TERMINAL ASSOCIATES LLP ECHO  
## 71 VA0005720 MOTIVA ENTERPRISES LLC - RICHMOND TERMINAL ECHO  
## 307 VA0029785 LINCOLN TERMINAL COMPANY ECHO  
## 327 VA0052451 DOMINION - NORTH ANNA POWER STATION ECHO  
## 322 VA0050181 MANASSAS CITY WATER TREATMENT PLANT ECHO  
## 334 VA0053317 DOMINION - BATH COUNTY POWER STATION ECHO  
## ECHOOutfalls TotalOutfalls  
## 354 1 1  
## 71 4 4  
## 307 3 5  
## 327 23 23  
## 322 3 3  
## 334 27 27

# Data Analysis

AnalysisCode.R is true to its name. It is designed to create summary data frames that count the number of facilities reporting a certain statistic, show the statewide sum of a given statistic, and show the summed limit value given by ECHO. For convenience, it also splits limits and flows into separate data frames. It extracts all columns containing ECHO data and then analyzes each column one at a time. It takes the statistic code from the column table and references it with the table of codes (*CodeKey*) downloaded from ECHO to list the complete the description. It then sums the statistic for that column and counts the number of facilities reporting it.

order<-grep('plus',colnames(AllFacs))  
PlusFacs<-AllFacs[,order]  
FacSummary<-data.frame(Stat=colnames(PlusFacs),StatCode=character(length(colnames(PlusFacs))),Description=character(length(colnames(PlusFacs))),Present=numeric(length(colnames(PlusFacs))))  
FacSummary$Stat<-as.character(FacSummary$Stat);FacSummary$StatCode<-as.character(FacSummary$StatCode);FacSummary$Description<-as.character(FacSummary$Description)  
for (i in 1:length(colnames(PlusFacs))){  
 column<-as.vector(PlusFacs[,i])  
 FacSummary$StatCode[i]<-gsub(".\*[.]([^\_]+)[\_].\*","\\1",FacSummary$Stat[i])  
 FacSummary$Description[i]<-NA  
 if(FacSummary$StatCode[i] %in% CodeKey$STATISTICAL\_BASE\_CODE){  
 FacSummary$Description[i]<-CodeKey$STATISTICAL\_BASE\_LONG\_DESC[CodeKey$STATISTICAL\_BASE\_CODE==FacSummary$StatCode[i]]  
 }  
 FacSummary$Present[i]<-sum(!(is.na(column)))  
 FacSummary$SumValue[i]<-plus(column)  
}  
order<-grep('Limit',as.character(FacSummary$Stat))  
FacSummaryLimits<-FacSummary[order,]  
FacSummaryFlow<-FacSummary[-order,]  
head(FacSummaryFlow)

## Stat StatCode Description Present SumValue  
## 1 DesFlow\_plus DesFlow\_plus <NA> 858 167354.357  
## 2 TotalFlow\_plus TotalFlow\_plus <NA> 469 130725.341  
## 3 Flow.DD\_plus DD Daily Maximum 440 36039.183  
## 5 Flow.MK\_plus MK Monthly Average 717 64432.585  
## 7 Flow.AB\_plus AB Annual Average 1 5811.500  
## 9 Flow.AF\_plus AF Average 37 536.915

head(FacSummaryLimits)

## Stat StatCode Description Present SumValue  
## 4 Limit.DD\_plus DD Daily Maximum 2 8.600  
## 6 Limit.MK\_plus MK Monthly Average 456 1068.806  
## 8 Limit.AB\_plus AB Annual Average 0 NA  
## 10 Limit.AF\_plus AF Average 17 2165.114  
## 12 Limit.MN\_plus MN Monthly Maximum 0 NA  
## 14 Limit.ET\_plus ET Event Total 0 NA

Finally, a last column is created to count the number of reported statistics for a given facility. *AllFacs* is a long, many columned data frame. “FlowStatTotal” allows for instant assesment of how many statistics are being reported and can be used in conjunction with the number of outfalls to get a sense of how complete the discharge monitoring records are for a given facility. This code chunk operates by first extracting all ECHO columns from *AllFacs*. It then ignores those that deal with limits and focuses on the actual discharge values reported. It creates a reference data frame *test* that is used to compile information about number of avaiable statistics by counting the number of NAs in a given row of *AllFacs*. This is then added to the back end of *AllFacs*

order<-c(1,2,grep('plus',colnames(AllFacs)))  
order2<-grep('Limit',colnames(AllFacs[,order]))  
order<-order[!(order %in% order2)]  
test<-AllFacs[,order]  
test<-test[,-3]  
for (i in 1:length(test$FacilityID)){  
 test$StatTotal[i]<-rowSums(!is.na(test[i,3:length(colnames(test))]))  
}  
for (i in 1:length(AllFacs$FacilityName)){  
 AllFacs$FlowStatTotal[i]<-test$StatTotal[test$FacilityID==AllFacs$FacilityID[i]]  
}  
head(AllFacs[,c(1,2,length(colnames(AllFacs)))])

## FacilityID FacilityName FlowStatTotal  
## 354 VA0057576 DOMINION TERMINAL ASSOCIATES LLP 0  
## 71 VA0005720 MOTIVA ENTERPRISES LLC - RICHMOND TERMINAL 1  
## 307 VA0029785 LINCOLN TERMINAL COMPANY 1  
## 327 VA0052451 DOMINION - NORTH ANNA POWER STATION 2  
## 322 VA0050181 MANASSAS CITY WATER TREATMENT PLANT 1  
## 334 VA0053317 DOMINION - BATH COUNTY POWER STATION 2

# VA Hydro Geometry Correction

The final part of this code deals with bad geometry values found in measuring points and facilities within VA Hydro. It starts by extracting lat/long from the WKT geometry provided by VA Hydro. It then checks rough bounds for Virginia to see if these geometry are valid. If not, it looks for ways to correct them by assuming a mistake was made during data entry. Values are flipped and or reversed to check for a possible valid lat/long. Data is aggregated to the facility level in *HydroFacs* and a similar process is computed. Where needed, facility geomoetry is drawn from MPs and vice versa.

Hydro$FacLat<-as.numeric(gsub(".\*[ ]([^)]+)[)].\*","\\1",Hydro$fac\_geom))  
Hydro$FacLong<-as.numeric(gsub(".\*[(]([^ ]+)[ ].\*","\\1",Hydro$fac\_geom))  
Hydro$tsvalue<-as.numeric(Hydro$tsvalue)  
Hydro$MPLat<-as.numeric(gsub(".\*[ ]([^)]+)[)].\*","\\1",Hydro$mp\_geom))  
Hydro$MPLong<-as.numeric(gsub(".\*[(]([^ ]+)[ ].\*","\\1",Hydro$mp\_geom))  
#A simple search function to check for NAs and replace them for use in boolean code  
NAReplace<-function(x){  
 if(is.na(x)){  
 return(-99)  
 }else{  
 return(x)  
 }  
}  
#Correct lat/long values for possible typos i.e. missing negative signs, switched lat/long, or both  
#Is set to check if data falls in rough state boundaries set by (35.-70) and (45,-90)  
Hydro$Lat<-numeric(length(Hydro$MPLat))  
Hydro$Long<-numeric(length(Hydro$MPLong))  
for (i in 1:length(Hydro$fac\_name)){  
 Hydro$Lat[i]<-Hydro$MPLat[i]  
 Hydro$Long[i]<-Hydro$MPLong[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])  
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$MPLong[i]  
 Hydro$Long[i]<-Hydro$MPLat[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])   
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$MPLat[i]  
 Hydro$Long[i]<-(-1)\*Hydro$MPLong[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])   
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$MPLong[i]  
 Hydro$Long[i]<-(-1)\*Hydro$MPLat[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])   
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$FacLat[i]  
 Hydro$Long[i]<-Hydro$FacLong[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])  
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$FacLong[i]  
 Hydro$Long[i]<-Hydro$FacLat[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])  
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$FacLat[i]  
 Hydro$Long[i]<-(-1)\*Hydro$FacLong[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])  
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$FacLong[i]  
 Hydro$Long[i]<-(-1)\*Hydro$FacLat[i]  
 Hydro$Lat[i]<-NAReplace(Hydro$Lat[i])  
 Hydro$Long[i]<-NAReplace(Hydro$Long[i])  
 if((Hydro$Lat[i]<35|Hydro$Lat[i]>45)|(Hydro$Long[i]<(-90)|Hydro$Long[i]>(-70))){  
 Hydro$Lat[i]<-Hydro$MPLat[i]  
 Hydro$Long[i]<-Hydro$MPLong[i]  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
}  
#Same process as above, but on a facilitiy level rather than an MP one. If no valid facility ID can be found,  
#MP data may be used in its place. If no passable geometry can be fournd, return original data  
HydroFacs<-as.data.frame(Hydro %>% group\_by(fac\_hydroid) %>% summarize(Name=first(fac\_name),Sum=plus(tsvalue)))  
HydroFacs$Lat<-numeric(length(HydroFacs$fac\_hydroid))  
HydroFacs$Long<-numeric(length(HydroFacs$fac\_hydroid))  
  
for (i in 1:length(HydroFacs$fac\_hydroid)){  
 HydroFacs$Lat[i]<-Hydro$FacLat[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Long[i]<-Hydro$FacLong[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 HydroFacs$Lat[i]<-Hydro$FacLong[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Long[i]<-Hydro$FacLat[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 HydroFacs$Lat[i]<-Hydro$FacLat[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Long[i]<-(-1)\*Hydro$FacLong[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 HydroFacs$Lat[i]<-Hydro$FacLong[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Long[i]<-(-1)\*Hydro$FacLat[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 }  
 }  
 }  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 j<-1  
 MP<-Hydro[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i],]  
 while(j<(length(MP$fac\_name)+1)){  
 HydroFacs$Lat[i]<-MP$MPLat[j]  
 HydroFacs$Long[i]<-MP$MPLong[j]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 HydroFacs$Lat[i]<-MP$MPLong[j]  
 HydroFacs$Long[i]<-MP$MPLat[j]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 HydroFacs$Lat[i]<-MP$MPLat[j]  
 HydroFacs$Long[i]<-(-1)\*MP$MPLong[j]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 HydroFacs$Lat[i]<-MP$MPLong[j]  
 HydroFacs$Long[i]<-(-1)\*MP$MPLat[j]  
 HydroFacs$Lat[i]<-NAReplace(HydroFacs$Lat[i])  
 HydroFacs$Long[i]<-NAReplace(HydroFacs$Long[i])  
 }  
 }  
 j<-j+1  
 }else{  
 j<-length(MP$fac\_name)+1  
 }  
 }  
 if((HydroFacs$Lat[i]<35|HydroFacs$Lat[i]>45)|(HydroFacs$Long[i]<(-90)|HydroFacs$Long[i]>(-70))){  
 HydroFacs$Lat[i]<-Hydro$FacLat[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 HydroFacs$Long[i]<-Hydro$FacLong[Hydro$fac\_hydroid==HydroFacs$fac\_hydroid[i]][1]  
 }  
 }  
}