Supplemental Information: Analysis Code

This document provides RCode in support of Seybold et al., 2023. The code can be broken into three main components: (i) code environment setup, (ii) download and analysis function, and (iii) function execution. To copmlete this analysis, we use the GAGESII database (click here for link to data). Furthing, this script relies on the tidyverse, parallel, and dataRetrieval R packages. Finally, here is a tutorial click here for more information on functional programming. Please contact Erin Seybold, Nate Jones, or Sam Zipper if you have any questions.

(i) Setup Coding Environment

```
#Clear memory
remove(list=ls())

#Load packages of interest
library(tidyverse) #join the cult
library(patchwork) #combine multiple plot objects
library(dataRetrieval) #download USGS data
library(parallel) #parallel processing

#Define dir of interest
output_dir <- "docs/"
data_dir   <- "data/"

#Load data
gages <- read_csv(paste0(data_dir, "gagesII.csv"))
```

(ii) Download and Analysis Function

```
#Function
fun <- function(gage_id, start_date = "1979-01-01", end_date = "2018-12-30"){
  # gage_id = USGS site number in "########"
  # start_date = first date in "YYYY-MM-DD" format
  # end_date = last date in "YYYY-MM-DD" format

  # read discharge data
  pCodes = c("00060") # discharge = 00060, stage = 00065
  daily_raw <-
    dataRetrieval::readNWISdv(siteNumbers = gage_id,
                              parameterCd = pCodes,
                              startDate = start_date,
                              endDate = end_date,
                              statCd = "00003")

  # get rating curve
  sw_meas <-
    dataRetrieval::readNWISmeas(siteNumbers = gage_id) |>
    subset(!is.na(measured_rating_diff)) |>
    subset(discharge_va > 0) |>
    subset(measured_rating_diff %in% c("Good", "Excellent"))
```

```
  # find minimum good/excellent measurement
  min_good_q <- sw_meas$discharge_va[which.min(sw_meas$discharge_va)]

  # get percent of days with flow less than min_good_q
  prc_days_good <- sum(daily_raw$X_00060_00003 < min_good_q)/length(daily_raw$X_00060_00003)

  # make data frame to output
  df_out <- data.frame(
    gage_id,
    minGoodQ_cfs = min_good_q,
    daysSubGood_prc = prc_days_good)

  return(df_out)

}
```

(iii) Execute Function

```
#Create wrapper function for error handling
error_fun<-function(n){
  tryCatch(
    expr = fun(gages$STAID[n]),
    error = function(e)
      tibble(
        gage_id = gages$STAID[n],
        minGoodQ_cfs = NA,
        daysSubGood_prc = NA)
    )
}

#Start timer
t0<-Sys.time()

#Determine number of processing cores available on your machine
n.cores<-detectCores()-1

#Create clusters
cl<-makeCluster(n.cores)

#Send libraries to cluster
clusterEvalQ(cl, {
  library(dataRetrieval)
  library(tidyverse)
  library(lubridate)
})

#Export data to cluter environments
clusterExport(cl, c("fun", "gages"))

#Now run function
output<-parLapply(
```

```
  cl=cl,
  seq(1, nrow(gages)),
  error_fun)

#Now, bind rows from list output
output<-output %>% bind_rows()

#Stop the clusters
stopCluster(cl)

#End Time
tf<-Sys.time()
tf-t0

#interrogate results
df<-output %>% drop_na()

#Export results
write.csv(df, "data/SubGoodQ.csv")
```