```
# This code will examine to hydrograph dataset, select matching days
# and times and conduct a regression that can be used to fill in missing data
# Author: Nicholas A. Sutfin
# Date: Oct. 18th 2017, last modified May 8th, 2020

library("plyr")
#library("smwrBase", lib.loc="~/R/win-library/3.2")
library("lattice") #, lib.loc="C:/Program Files/R/R-3.3.0/library")
library("lubridate")
library("hydroGOF")

# Set user space
loadpath = '/Users/NicholasSutfin/Documents/EastRiver/ER_Rcode/'
savepath = '/Users/NicholasSutfin/Documents/EastRiver/ER_Rcode/Baseflow_1.91_BestFit/' #
Calculating slope as line between 1st and last points (2p)
setwd(loadpath)

All_DailyQ_1935_2020 = read.csv("All_DailyQ_1935_2020.csv", stringsAsFactors = F)
#"All_DailyQ_1910_2020.csv", stringsAsFactors = F)

# Load ALmont data for 2015-2017 as csv file, convert to SI units, code the date as a date, and
define the year
Alm_Q <- read.csv("ER_AlmQ_2015-2019.csv", header=TRUE)
Alm_Q$Q_cfs = as.numeric(as.character(Alm_Q$Q_cfs))
Alm_Q$Alm_Q_cms = Alm_Q$Q_cfs*0.0283168
Alm_DailyQ = as.data.frame(Alm_Q)
Alm_DailyQ = ddply(Alm_DailyQ, ~date, summarise, Alm_Q_cms = mean(Alm_Q_cms))
Alm_Qdaily <- Alm_DailyQ[order(as.Date(Alm_DailyQ$date, format="%m/%d/%y")),]
Alm_Qdaily$Date = as.Date(Alm_Qdaily$date, "%m/%d/%y")
Alm_Qdaily$year = year(Alm_Qdaily$Date)
Alm_Qdaily$month = month(Alm_Qdaily$Date)
Alm_Qdaily$Calday = day(Alm_Qdaily$Date)
Alm_Qdaily$day = yday(Alm_Qdaily$Date)

# Load Pump house data for 2015-2017 as csv file, convert to SI units, code the date as a date,
and define the year
#PH_Qdaily <- read.csv("ER_PH_2015-17Q.csv", header=TRUE )
PH_Data <- read.csv("ER_PHQ_2014-2018.csv", header=TRUE)
PH_DailyQ = ddply(PH_Data, ~date, summarise, PHQ_cms = mean(PHQ_cms))
PH_Qdaily <- PH_DailyQ[order(as.Date(PH_DailyQ$date, format="%m/%d/%y")),]
PH_Qdaily$Date = as.Date(PH_Qdaily$date, "%m/%d/%y")
PH_Qdaily$year = year(PH_Qdaily$Date)
PH_Qdaily$month = month(PH_Qdaily$Date)
PH_Qdaily$Calday = day(PH_Qdaily$Date)
```

```
PH_Qdaily$day = yday(PH_Qdaily$Date)
names(PH_Qdaily)[2]<-paste("PH_Q_cms")

#_____
____
# Find matching dates and create new dataset
DailyQ_diff <- setdiff(PH_Qdaily$Date, Alm_Qdaily$Date)
DailyQ_int <- intersect(PH_Qdaily$Date, Alm_Qdaily$Date)

# Find PH Q data for dates overlapping the with Almont gage
PH_DailyQ_match <- PH_Qdaily[PH_Qdaily$Date %in% DailyQ_int, ]
# Find Almont gauge data that overlaps with pump house study site data
Alm_DailyQ_match <- Alm_Qdaily[Alm_Qdaily$Date %in% DailyQ_int, ]
# Merge the two overlapping datasets side my side by matching dates
All_DailyQ_15_18 <- cbind(Alm_DailyQ_match, PH_DailyQ_match)

rows = length(All_DailyQ_15_18$PH_Q_cms) #[All_DailyQ_15_18$day > 105 &
All_DailyQ_15_18$day < 319])
Qmat <- matrix(0, rows, 3)
Q = as.data.frame(Qmat)
names(Q)[1]=paste("PH")
names(Q)[2]=paste("AL")
names(Q)[3]=paste("day")

# April 15th = 105 Nov 15th = 319, so 104-320 is good
Q$PHDate = All_DailyQ_15_18$Date[which(is.na(All_DailyQ_15_18$PH_Q_cms) == FALSE)]
#[All_DailyQ_15_18$day > 105 & All_DailyQ_15_18$day < 319]
Q$PH = All_DailyQ_15_18$PH_Q_cms[which(is.na(All_DailyQ_15_18$PH_Q_cms) == FALSE)]
#[All_DailyQ_15_18$day > 105 & All_DailyQ_15_18$day < 319]
Q$ALDate = All_DailyQ_15_18$Date[which(is.na(All_DailyQ_15_18$PH_Q_cms) == FALSE)]
#[All_DailyQ_15_18$day > 105 & All_DailyQ_15_18$day < 319]
Q$AL = All_DailyQ_15_18$Alm_Q_cms[which(is.na(All_DailyQ_15_18$Alm_Q_cms) == FALSE)]
#[All_DailyQ_15_18$day > 105 & All_DailyQ_15_18$day < 319]
Q$day = All_DailyQ_15_18$day[which(is.na(All_DailyQ_15_18$Alm_Q_cms) == FALSE)]
#[All_DailyQ_15_18$day > 105 & All_DailyQ_15_18$day < 319]

Qreg <- lm(Q$PH ~ Q$AL, data = Q)
summary(Qreg)
Qreg # adjusted R squared = 0.97

# For all days: PHQ = -0.081804 + 0.211284(Alm)
# Excluding frozen days, regression output: PHQ = 0.010948 + 0.211611(Alm)

par(mfrow=c(1,1), mar=c(4,5,2,2), cex = 1.5, lwd = 1)
```

```
plot(All_DailyQ_15_18$Alm_Q_cms, All_DailyQ_15_18$PH_Q_cms, col = "blue",
    xlab = expression(paste("Discharge at Almont (m"^"3", "s"^"-1",")")),
    ylab = expression(paste("Discharge (m"^"3", "s"^"-1",")")))
lines(All_DailyQ_15_18$Alm_Q_cms, Qreg$coefficients[1] +
Qreg$coefficients[2]*All_DailyQ_15_18$Alm_Q_cms,
     col = "black")
par(cex = 1)
#points(Q$AL, Q$PH, pch = 19, col = "red")
text(10, 15, expression("r"^{2} ~"= 0.97"), cex = 1.5)

# Load Almont discharge data from 1910 to 2020, cut data to timeframe of interest (1955-2015)
# and convert to cms
#_____
____
Alm_Qdaily_1910_2020 <- read.csv("Alm_Q_cfs_1910_2020.csv", header=TRUE)
Alm_Qdaily_1910_2020$Alm_Q_cms = Alm_Qdaily_1910_2020$Alm_Q_cfs*0.0283168
Alm_Qdaily_1910_2020$Date = as.Date(Alm_Qdaily_1910_2020$Date, "%m/%d/%Y")

All_DailyQ_1910_2020 = Alm_Qdaily_1910_2020
All_DailyQ_1910_2020$year = format(All_DailyQ_1910_2020$Date, "%Y")
All_DailyQ_1910_2020$month = format(All_DailyQ_1910_2020$Date, "%m")
All_DailyQ_1910_2020$day = format(All_DailyQ_1910_2020$Date, "%d")
All_DailyQ_1910_2020$yday = yday(All_DailyQ_1910_2020$Date)
All_DailyQ_1910_2020$Mod_PH_Q_cms = Qreg$coefficients[1] +
Qreg$coefficients[2]*All_DailyQ_1910_2020$Alm_Q_cms

# Use regression to extend daily Q for PH based on Almont flow
#_____
____
# regression output: PHQ = x + y(Alm)
par(mfrow=c(1,1), mar=c(4,5,3,2), cex = 1.5)
All_DailyQ_2014_2020 = All_DailyQ_1910_2020[37987:length(Alm_Qdaily_1910_2020$Date), ]

#_____
____
# plot observed vs. modeled data for East River and calculate Nash-Sutcille and RMSE
par(mfrow=c(1,1), mar=c(4,4,2,2), cex = 1.1)

Date = All_DailyQ_2014_2020$Date
Modeled_PHQ = subset(All_DailyQ_2014_2020, Date > "2014-9-30")
#min(WaterYear15):max(WaterYear15)))

# Select only uniqe values
Observed_PHQ = All_DailyQ_15_18[,c(3,9)]
```

```
PH_Q_int <- intersect(Observed_PHQ$Date[order(Observed_PHQ$Date)],
Modeled_PHQ$Date[order(Modeled_PHQ$Date)])
Modeled_Q_match <- Modeled_PHQ[Modeled_PHQ$Date %in% PH_Q_int, ]
Observed_Q_match <- Observed_PHQ[Observed_PHQ$Date %in% PH_Q_int, ]
PHQ_15_18 = cbind(Modeled_Q_match, Observed_Q_match)

Qreg2 <- lm(PHQ_15_18$PH_Q_cms ~ PHQ_15_18$Alm_Q_cms, data = All_DailyQ_15_18)
summary(Qreg2)
Qreg2

par(mfrow=c(1,1), mar=c(4,5,2,2), cex = 1.5, lwd = 1)
# Plot Almont flow data
plot(All_DailyQ_15_18$Date, All_DailyQ_15_18$Alm_Q_cms, lwd = 2, type = "l",
    col = "black", xlab = "Year", ylab = expression(paste("Discharge (m"^"3", "s"^"-1",")")), lty =
5, cex = 1.5)
# Plot observed ER study site flow data
lines(PHQ_15_18$Date[order(PHQ_15_18$Date)],
PHQ_15_18$PH_Q_cms[order(PHQ_15_18$Date)], lty = 1, col = "blue", lwd = 2, type = "l",
    xlab = expression(paste("Discharge (m"^"3", "s"^"-1",")")), ylab = "Time (years)")
#polygon(PHQ_15_17$date, PHQ_15_17[,5], col = "blue")

# Plot modeled ER study site flow data
lines(PHQ_15_18$Date[order(PHQ_15_18$Date)],
PHQ_15_18$Mod_PH_Q_cms[order(PHQ_15_18$Date)], col = 'red', lwd = 2, lty = 2)
legend("topright", col = c("black", "blue", "red"), lty = c(5,1,2),
      lwd = 2, legend = c('Almont', 'Observed', 'Modeled'))

NSE(PHQ_15_18[,10],PHQ_15_18[,8])
text(10, 15, expression("NSE = 0.97"), cex = 1.5)
# Nash-Sutcliffe coeeficient = 0.97

#_____
____
# Format data for hydrograph analysis
write.csv(All_DailyQ_2014_2020,"All_DailyQ_2014_2020.csv")
write.csv(All_DailyQ_1910_2020,"All_DailyQ_1910_2020.csv")

ER_Q_35_20 <- All_DailyQ_1910_2020[All_DailyQ_1910_2020$year > 1934, ]
write.csv(ER_Q_35_20, "All_DailyQ_1935_2020.csv")

####################################
# Create plots of Almont and East RIver
```

```
#_____
____
par(mfrow=c(1,1), mar=c(4,5,1,1), cex = 1)
All_Q_1910_2020 = All_DailyQ_1910_2020
ER_Q_55_20 <- All_Q_1910_2020[All_Q_1910_2020$year > 1954, ]

#_____
____
# Create a stacked plot of hydrographs for the period of record
#_____
____

par(mfrow=c(1,1), mar=c(4,5,2,2), cex = 1.5)

# Create an initial plot to add hydrographs from all years
plot(ER_Q_55_20$yday[ER_Q_55_20$year == 1955],
ER_Q_55_20$Mod_PH_Q_cms[ER_Q_55_20$year == 1955], type = "l",
   ylim = c(0,25), xlab = "Day of Year",
   ylab = expression(paste("Modeled discharge (m"^"3", "s"^"-1",")")), lwd = 1,
   main = "East River 1955-2015")

# Create a smaller zoomed in plot to add hydrographs from all years
#plot(ER_Q_55_20$day[ER_Q_55_20$year == 1955], ER_Q_55_20[ER_Q_55_20$year == 1955,
3], type = "l",
 #    ylim = c(0,11), xlim = c(160,220), xaxt = "n", xlab = "Day of Year", ylab = "Discharge (cms)",
lwd = 1, main = "East River 1955-2017")

# Create a list of unique years for the period of interest
years = unique(ER_Q_55_20$year)

# A for loop to plot hydrographs for all years on top of one another
for (i in 1:65) {
  years2plot = years[i]
  print(years2plot)
  dat.yr = subset(ER_Q_55_20, year == years2plot)
  print(dat.yr)
  lines(dat.yr$yday, dat.yr$Mod_PH_Q_cms, col = "royalblue1", lwd = 1)
}

# Calculate the mean and 95% confidence level for all hydrographs in the period of interest
AllFlow = ddply(ER_Q_55_20, ~yday, summarise,
            MeanFlow = mean(Mod_PH_Q_cms),
            LCI = quantile(Mod_PH_Q_cms, 0.025, na.rm = TRUE),
            UCI = quantile(Mod_PH_Q_cms, 0.975, na.rm = TRUE))
```

```
# Plot a transparent band representing the 95% confidence level
polygon(c(AllFlow$yday,rev(AllFlow$yday)),c(AllFlow$LCI,rev(AllFlow$UCI)),border=NA,
      col = rgb(red = 0.0, green = 0.0, blue = 0.5, alpha = 0.25))


#-----------------------------------------
# Plot mean hydrographs for 6 time intervals

Q_55_73 = ER_Q_55_20[ER_Q_55_20$year < 1974, ]
Q_74_83 = ER_Q_55_20[ER_Q_55_20$year > 1973 & ER_Q_55_20$year < 1984, ]
Q_84_90 = ER_Q_55_20[ER_Q_55_20$year > 1983 & ER_Q_55_20$year < 1991, ]
Q_91_01 = ER_Q_55_20[ER_Q_55_20$year > 1990  & ER_Q_55_20$year < 2002, ]
Q_02_11 = ER_Q_55_20[ER_Q_55_20$year > 2001  & ER_Q_55_20$year < 2012, ]
Q_12_17 = ER_Q_55_20[ER_Q_55_20$year > 2011, ]
Q_12_15 = ER_Q_55_20[ER_Q_55_20$year > 2011  & ER_Q_55_20$year < 2016, ]


par(mfrow=c(1,1), mar=c(4,4,2,2), cex = 1.5)

# Calculate the mean and 95% confidence level for all hydrographs in the period of interest
Flow73 = ddply(Q_55_73, ~yday, summarise,
           MeanFlow = mean(Mod_PH_Q_cms),
           LCI = quantile(Mod_PH_Q_cms, 0.025, na.rm = TRUE),
           UCI = quantile(Mod_PH_Q_cms, 0.975, na.rm = TRUE))
lines(Flow73$yday, type = "line", #ylim = c(0, 11),
    Flow73$MeanFlow, col = "red", lwd = 2.5,
   xlab = "Day of the year", ylab = "Discharge (cms)") # Plot the mean hydrograph value

# Calculate the mean and 95% confidence level for all hydrographs in the period of interest
Flow83 = ddply(Q_74_83, ~yday, summarise,
           MeanFlow = mean(Mod_PH_Q_cms),
           LCI = quantile(Mod_PH_Q_cms, 0.025, na.rm = TRUE),
           UCI = quantile(Mod_PH_Q_cms, 0.975, na.rm = TRUE))
lines(Flow83$yday,
    Flow83$MeanFlow, col = "orange", lwd = 2.5) # Plot the mean hydrograph value

# Calculate the mean and 95% confidence level for all hydrographs in the period of interest
Flow90 = ddply(Q_84_90, ~yday, summarise,
           MeanFlow = mean(Mod_PH_Q_cms),
           LCI = quantile(Mod_PH_Q_cms, 0.025, na.rm = TRUE),
           UCI = quantile(Mod_PH_Q_cms, 0.975, na.rm = TRUE))
lines(Flow90$yday,
    Flow90$MeanFlow, col = "yellow", lwd = 2.5) # Plot the mean hydrograph value
```

```
# Calculate the mean and 95% confidence level for all hydrographs in the period of interest
Flow01 = ddply(Q_91_01, ~yday, summarise,
          MeanFlow = mean(Mod_PH_Q_cms),
          LCI = quantile(Mod_PH_Q_cms, 0.025, na.rm = TRUE),
          UCI = quantile(Mod_PH_Q_cms, 0.975, na.rm = TRUE))
lines(Flow01$yday,
    Flow01$MeanFlow, col = "green", lwd = 2.5) # Plot the mean hydrograph value

# Calculate the mean and 95% confidence level for all hydrographs in the period of interest
Flow11 = ddply(Q_02_11, ~yday, summarise,
          MeanFlow = mean(Mod_PH_Q_cms),
          LCI = quantile(Mod_PH_Q_cms, 0.025, na.rm = TRUE),
          UCI = quantile(Mod_PH_Q_cms, 0.975, na.rm = TRUE))
lines(Flow11$yday,
    Flow11$MeanFlow, col = "darkblue", lwd = 3.5) # Plot the mean hydrograph value

# Calculate the mean and 95% confidence level for all hydrographs in the period of interest
Flow17 = ddply(Q_12_15, ~yday, summarise,
          MeanFlow = mean(Mod_PH_Q_cms),
          LCI = quantile(Mod_PH_Q_cms, 0.025, na.rm = TRUE),
          UCI = quantile(Mod_PH_Q_cms, 0.975, na.rm = TRUE))
lines(Flow17$yday,
    Flow17$MeanFlow, col = "black", lwd = 2.5) # Plot the mean hydrograph value

par(mfrow=c(1,1), mar=c(4,4,2,2), cex = 1.2)
legend(280, 25, legend = c("1955-1973", "1974-1983", "1984-1990", "1991-2001", "2002-2011",
"2012-2015"),
     col = c("red", "orange", "yellow", "green", "darkblue", "black"),
     lty = 1.2, lwd = 2.5, bg = "gray85")

#*********************************************************

###Stream Flow Frequency Analysis and Recession Limb Quantification

###################################
# From time lapse photos and the stage data, bankfull stage appears to occur at about 4 cms
###################################

#setwd(loadpath)
#All_DailyQ_1935_2020 = read.csv("All_DailyQ_1935_2020.csv", stringsAsFactors = F)
#"All_DailyQ_1910_2020.csv", stringsAsFactors = F)
data = All_DailyQ_1935_2020 #"All_DailyQ_1910_2020.csv", stringsAsFactors = F)
dat.er = data[ ,c(2,3,5:9)]
```

```
dat.er$flow.er = dat.er$Mod_PH_Q_cms

# estimate lowflow conditions and a reference basflow by which to measure the recession limb
Lowflow = mean(na.omit(dat.er$flow.er[dat.er$month %in% list("10","11","12","1","2","3")]))
Baseflow =  1.91 #Lowflow #mean(na.omit(dat.er$flow.er[dat.er$month %in% list("9")]))
BFQ = 8 # define a threshold approximation for bankfull discharge
# Estimated bankkfull at 8 cms

# Initialize storage variables
years = unique(dat.er$year) # Unique years for indexing (using water years (10/01-9/30))
years = years[years > 1934]

# Aggregate Yearly (or monthly) data by mean, median, max, and min (or anything else)
x = subset(dat.er, year %in% c(1935:2019))
statistics = as.data.frame(as.list(aggregate(flow.er ~ year ,data = x, FUN=function(x) c(mean
=mean(x), median=median(x), max = max(x),min = min(x)))))

maxflow = as.data.frame(matrix(ncol=10,nrow =85))#length(years)))
# define the list of column names for the dataframe
names(maxflow) = c("year","peakdate","flow.er","BFflow", "BF_EndDay", "enddate",
"TotalSlope","BFslope","BF_StartDay","PeakSlope")

for (k in 2:85){
 # Skip years where insufficient data was collected using a # of days in year as threshold. bad
 if (length(dat.er$Date[dat.er$year == years[k]]) < 250) {
 }
 else {
  # find peak flows greater than 500cfs and corresponding year and Date
  dat.sub = subset(dat.er, year == years[k]) # Subset larger data set
  dat.sub$Date = as.Date(dat.sub$Date, format="%Y-%m-%d")
  medianflow = mean(dat.sub$flow.er[dat.sub$month %in% list("10","11","12")])
#median(na.omit(dat.sub$flow)) # find median flow (used as a threshold, need better method)
  maxflow[k,3] = max(na.omit(dat.sub$flow.er)) # find and store peak flows
  maxflow[k,1] = years[k] # store year
  index = tail(which(dat.sub$flow.er == maxflow[k,3]), n=1) # find index of peak flow to
detrmine the exact Date
  maxflow[k,2] = as.character(dat.sub$Date[index]) # Date of peak flow
  #as.Date(index, origin = dat.sub$Date[1]) #

  # Bankfull flow
  if (max(dat.sub$flow.er >= 8)) {
   indX1 = min(which(dat.sub$flow.er >= 8)) # index the date flow rises above BF
   indX = max(which(dat.sub$flow.er >= 8)) # index the date flow drops below BF
   BF_start = as.character(dat.sub$Date[indX1]) # Assign first date flow exceeds BF
```

```
    maxflow[k,9] = BF_start # Assign first date flow exceeds BF
    BF_end = as.character(dat.sub$Date[indX]) # Assign last date flow drops below BF
    maxflow[k,5] = BF_end # Assign last date flow drops below BF
    maxflow[k,4] = dat.sub$flow.er[indX]
   }
   else {
    maxflow[k,5] = NA
    maxflow[k,4] = NA
    maxflow[k,9] = NA
    indX = NA
    BF_start = NA
    BF_end = NA
    print(years[k])
    }

   ## Extracting Recession limb
   # This section finds the Dates corresponding to the peakflow (already found above) and a
later
   # Date corresponding to "normal" flow conditions. I am currently using the median but it's a
bad
   # metric.
   # Starting at the index of the peak flow Date, step forward one day (increasing the index by 1)
and
   # check if the flow that day is a certain percentage from the median value.
   PeakDate = as.character(dat.sub$Date[index]) # used for extracting recession limb
   maxdepth = maxflow[k,3] # used for extracting recession limb
   repeat{
    index = index+1
    maxdepth = dat.sub$flow.er[index] # flow one day later
    if (is.na(maxdepth)){ # check if no flow was recorded
    } else if (Baseflow > (maxdepth)){ # Check if flow is within X% of median value
      break # was preiously ((medianflow) + Qmin) > maxdepth))
      # The "index" term now identifies the obs where Q reaches a baseflow condition ~0.8cms
    } else if (index == length(dat.sub$flow.er)) {
      print(paste(dat.sub$year[1])) # identify the year
      break
      # This forces the loop to break if Q never falls below baseflow
    }
   }
  #*************************************
   # Indexing for bankfull slope calculation
   BFDate = maxflow[k,5]

   if (is.na(maxflow[k,5]) == FALSE) {
```

```
      repeat{
     indX = indX+1 #increment one more day after last BF flow
     BFQ = dat.sub$flow.er[indX] # flow one day later
     if (is.na(BFQ)){ # check if no flow was recorded and do nothing
     } else if (Baseflow > (BFQ)){ # Check if flow is within threshold of median value was
previously ((medianflow) + Qmin > (BFQ))
      break # Exist loop if Q drops below baseflow and saved that Q value as BFQ
     } else if (indX == length(dat.sub$flow.er)) {
      print(paste(dat.sub$year[1]))
      break # Exit loop if flow does not drop below baseflow
     }
    }
    }

    BaseDate = as.character(dat.sub$Date[index])
    maxflow[k,6] = as.character(dat.sub$Date[index])
    #FirstDate = dat.sub$Date[1] #Set the first date of the year

    # Convert Dates to yday for duration calculations
    BaseDay=yday(BaseDate)
    PeakDay=yday(PeakDate)
    BF_endDay=yday(BF_end)
    BF_startDay=yday(BF_start)
    Last_index=length(dat.sub$Date)
    LastDay = yday(dat.sub$Date[Last_index])
    BaseFlow_Date = as.Date(BaseDay, origin = dat.sub$Date[1])

    ##############################################################################
    # Calculate and plot slopes of recession limb at various stages
    #_____________________________________________________________________________

    # Calculate recession slope based on best fit regression line between all points
    TotSlopeQ = dat.sub$Mod_PH_Q_cms[dat.sub$yday %in% c(PeakDay:BaseDay)]
    TotSlopeDate = dat.sub$Date[dat.sub$yday %in% c(PeakDay:BaseDay)]
    TotSlopeReg = lm(TotSlopeQ ~ TotSlopeDate)
    summary(TotSlopeReg)

    maxflow[k,7] = -1*TotSlopeReg$coefficients[2] #((maxflow[k,3])-Baseflow)/(BaseDay-
PeakDay) # Slope of line from start to end of recession limb
    plot(dat.sub$Date, dat.sub$Mod_PH_Q_cms, type = "line", main = paste(years[k]),
        ylab = "Discharge (cms)", xlab = NA)
    points(TotSlopeDate, TotSlopeQ, pch = 19, col = "violet")
    lines(TotSlopeDate, predict(TotSlopeReg), col = "purple", lwd = 2)
```

```
  # Calculate slope as line between two points
  #maxflow[k,7] = (maxflow[k,3]-Baseflow)/(BaseDay-PeakDay)
  #plot(dat.sub$Date, dat.sub$Mod_PH_Q_cms, type = "line", main = paste(years[k]),
   #    ylab = "Discharge (cms)", xlab = NA)
  #points(TotSlopeDate, TotSlopeQ, pch = 19, col = "violet")
  #QPoints = c(maxflow[k,3],Baseflow)
  #TotDayPts =c(PeakDate, BaseDate)
  #DayPoints = as.Date(TotDayPts, "%Y-%m-%d")
  #lines(DayPoints, QPoints, col = "purple", lwd = 2)


  # Calculate the recession slope from the peak to bankfull flow as the best fit line
  if (is.na(maxflow[k,4])) {
    maxflow[k,10] = NA #Calculate slope of highest peak lower than bankfull to baseflow
  }
  else {

  # Calculate recession slope based on best fit regression line between all points
  PeakSlopeQ = dat.sub$Mod_PH_Q_cms[dat.sub$yday %in% c(PeakDay:BF_endDay)]
  PeakSlopeDate = dat.sub$Date[dat.sub$yday %in% c(PeakDay:BF_endDay)]
  PeakSlopeReg = lm(PeakSlopeQ ~ PeakSlopeDate)
  summary(PeakSlopeReg)
  points(PeakSlopeDate, PeakSlopeQ, pch = 20, col = "pink")
  lines(PeakSlopeDate, predict(PeakSlopeReg), col = "red", lwd = 2)
  maxflow[k,10] = -1*PeakSlopeReg$coefficients[2] #((maxflow[k,3])-
(maxflow[k,4]))/(BF_endDay-PeakDay) #SLope from peak to bankfull

  # Calculate slope as line between two points
  #maxflow[k,10] = (maxflow[k,3]-maxflow[k,4])/(BF_endDay-PeakDay)
  #points(PeakSlopeDate, PeakSlopeQ, pch = 20, col = "pink")
  #QPoints = c(maxflow[k,3],maxflow[k,4])
  #PeakDayPts =c(PeakDate, BF_end)
  #DayPoints = as.Date(PeakDayPts, "%Y-%m-%d")
  #lines(DayPoints, QPoints, col = "red", lwd = 2)
  }

  # Calculate the bankfull slope from bankfull to base flow
  if (is.na(maxflow[k,4])) {
    maxflow[k,8] = NA #Calculate slope of highest peak lower than bankfull to baseflow
  }
  else {
    # Calculate recession slope based on best fit regression line between all points
    BFSlopeQ = dat.sub$Mod_PH_Q_cms[dat.sub$yday %in% c(BF_endDay:BaseDay)]
    BFSlopeDate = dat.sub$Date[dat.sub$yday %in% c(BF_endDay:BaseDay)]
```

```
  BFSlopeReg = lm(BFSlopeQ ~ BFSlopeDate)
  summary(BFSlopeReg)
  points(BFSlopeDate, BFSlopeQ, pch = 20, col = "lightblue")
  lines(BFSlopeDate, predict(BFSlopeReg), col = "blue", lwd = 2)
  maxflow[k,8] = -1*BFSlopeReg$coefficients[2]

  # Calculate slope as line between two points
  #maxflow[k,8] = (maxflow[k,4]-Baseflow)/(BaseDay-BF_endDay)
  #points(BFSlopeDate, BFSlopeQ, pch = 20, col = "lightblue")
  #QPoints = c(maxflow[k,4],Baseflow)
  #BFDayPts =c(BF_end,BaseDate)
  #DayPoints = as.Date(BFDayPts, "%Y-%m-%d")
  #lines(DayPoints, QPoints, col = "blue", lwd = 2)

 }

 # Save year-days for duration calculations
 maxflow[k,11] = BF_startDay
 maxflow[k,12] = PeakDay
 maxflow[k,13] = BF_endDay
 maxflow[k,14] = BaseDay
 maxflow[k,15] = BF_endDay - BF_startDay # Duration Of recession Limb
 maxflow[k,16] = BaseDay - PeakDay # Duration Of recession Limb
 maxflow[k,17] = BaseFlow_Date
 maxflow[k,18] = LastDay # Last recorded day of the year

 # Cumulative days before and after bankfull
 if (is.na(BF_endDay)==FALSE) { # If there was a bankfull flow (i.e., BF_endDay is not NA)
  maxflow[k,19] = LastDay - BF_endDay  # Calculate the days since BF ended
 }
 else { # if there was no bankfull flow that year...
  maxflow[k,19] = LastDay + maxflow[k-1,19] # add the total number of days in the year to the
days since BF in the previous year
 }

 if (is.na(BF_endDay)==FALSE) { # If there was a bankfull flow (i.e., BF_endDay is not NA)
  maxflow[k,20] = BF_startDay + maxflow[k-1,19] # Days since bankfull
 }
 else {
  maxflow[k,20] = LastDay + maxflow[k-1,19]
 }
 BaseStart = min(which(dat.sub$flow.er >= Baseflow))
 maxflow[k,21] = dat.sub$yday[BaseStart]
```

```
  }
}

names(maxflow) = c("year","peakdate","flow.er","BFflow", "BF_EndDate", "enddate",
                   "TotalSlope","BFslope","BF_StartDate","PeakSlope","BF_startDay",
                   "PeakDay","BF_endDay","Base_endDay","BankfullDuration","RecDuration",
                   "BaseFlow_Date","LastDay", "CummDaysAfterBF", "CummDaysBeforeBF",
                   "Base_startDay")

#maxflow = na.omit(maxflow) # Remove missing flow
#if (is.na(maxflow[,2]) == FALSE) {}
#maxflow$peakdate = as.Date(maxflow$peakdate)
#maxflow$enddate = as.Date(maxflow$enddate)
maxflow$duration = yday(maxflow$enddate)-yday(maxflow$peakdate) # Duration Of recession
Limb

# Generate ranks (note that R ranks opposite of what is desired)
maxflow$rank = (length(maxflow$year)+1)-rank(maxflow$flow.er)
maxflow$RI = (length(maxflow$year)+1)/maxflow$rank
# Calculate excedence probablity
maxflow$exceedence = 1/maxflow$RI
#maxflow$NonBFdays = maxflow$LastDay - (maxflow$BF_endDay - maxflow$BF_startDay)
#THis does not account for days before first and last BF day that do not have BF flow
maxflow$BaseDuration = maxflow$Base_endDay - maxflow$Base_startDay #THis does not
account for days before first and last BF day that do not have BF flow

maxflow1 = maxflow[2:85,]
maxflow = maxflow[,c(1,9,2,5,6,3,4,7,10,8,20,21,22,23,26,11:19,24,25)]

setwd(savepath)
write.csv(maxflow1, file = "Maxflow1_6.29.20_Base_1.91_BestFit.csv")
write.csv(maxflow, file = "Maxflow_6.29.20_Base_1.91_BestFit.csv")

#*****************************************************
# Create plots
maxflow1$enddate = as.Date(maxflow1$enddate, format="%Y-%m-%d")
maxflow1$peakdate = as.Date(maxflow1$peakdate, format="%Y-%m-%d")

plot(flow.er ~ maxflow1$RI, maxflow1, log = 'x',
    xlab = "Recurrence Interval (years)",
    ylab = "Annual Maximum discharge (cfs)",
    main = "Flood Frequency Curve of Estimated Peak Flows")

rm(list=setdiff(ls(), c("maxflow","dat","dat.almont","dat.bc","dat.er",
```

```
                "hydrobounds","statistics","yearstats","years","colfunc",
                "loadpath","savepath","mod2","best.span", "Baseflow")))


##########
#
#   Recession Limb Characteristics
#
##########


hydrobounds = as.data.frame(matrix(ncol = 2, nrow = 85)) # create data frame for flow regime
characteristics
names(hydrobounds) = c("start","end") # create colums for end and start dates for bankfull
flow
#hydrobounds$start = maxflow$BF_StartDay
#hydrobounds$end = maxflow$BFdata
hydrobounds$EndDay = maxflow$BaseDay # assign the ending date
#maxflow$BF_StartDate = as.Date(maxflow$BF_StartDay)

for (k in 1:85){
  #print(k)
  years2plot = years[k] # create a list of each of the 83 years of record
  dat.sub = subset(dat.er, year%in%years2plot) # create a subset of data for the current year
  FirstDate = dat.sub$Date[1] #Set the first date of the year

  #_____________________________________________
  # Calculate cummulative annual volume of water discharged by East River
  #dat.sub$yearVol[1] = dat.sub$flow.er[1]*86400   # set initial flow volume for 1st day
  dat.sub$AnnualVol[1] = dat.sub$flow.er[1]*86400   # set initial flow volume for 1st day

  for (n in 2:length(dat.sub$Date)){ # create for loop to add consecutive Q resulting in
cumulative annual Q
    dat.sub$AnnualVol[n] =  dat.sub$AnnualVol[n-1] + dat.sub$flow.er[n]*86400 # sum each
consecutive flow volume for cummulative volume
  }

  #print(n)
  maxflow$AnnualVol[k] = dat.sub$AnnualVol[n] # assign the total ANnual volume of discharge
for each year
  dat.sub$BFVol = NA #create column for bankfull flow volume and fill with NA

  #_____________________________________________
  # Calculate cummulative volume of overbank flow discharged by the East River
```

```
  for (m in 1:length(dat.sub$Date)) {

    if (is.na(maxflow$BF_StartDate[k]) == FALSE) {
      # Set initial volume for first day above Bankful flow
    dat.sub$BFVol[which(maxflow$BF_StartDate[k]==dat.sub$Date)] =
dat.sub$flow.er[which(maxflow$BF_StartDate[k]==dat.sub$Date)]*86400   # set initial flow
volume for 1st day
    #Create indices for the start and end of bankfull flow
    BF_StartIndex = which(maxflow$BF_StartDate[k]==dat.sub$Date) # Index the row for the first
day of bankful flow begins
    BF_EndIndex = which(maxflow$BF_EndDate[k]==dat.sub$Date) #index the row for the last
day of bankful flow ends

    #Creat a loop to add cumulative volume of bankfull discharge
    for (p in BF_StartIndex+1:(BF_EndIndex-BF_StartIndex)) { # create for loop to add consecutive
Q resulting in cumulative annual Q
      #print(p)
      # Old calculations that estimates max BF volume for all days between 1st and last day of
bankfull flow. THis is an iver estimate
      dat.sub$BFVol[p] = dat.sub$BFVol[p-1] + dat.sub$flow.er[p]*86400 # sum each consecutive
flow volume for cummulative volume
      #print(dat.sub$Date[p])
      }
    maxflow$BFVol[k] = dat.sub$BFVol[p] # Assign yearly volume of flow above bankful to the
annual summary
  }
  else {
   dat.sub$BFVol[m] = NA #Assign days without bankful flow as NA values
   maxflow$BFVol[k] = NA #Assign years without bankful flow as NA values
   p=NA

  }
  }

  hydrobounds$cvol.er[k] = dat.sub$AnnualVol[length(dat.sub$AnnualVol)]
  hydrobounds$BFVol[k] = dat.sub$BFVol[max(which(is.na(dat.sub$BFVol) == FALSE))]

  ### Model peaks and valleys

  baseflowinitial = mean(dat.sub$flow.er[dat.sub$month %in% list("1","2")]) # Set initial
baseflow conditions as the mean of flow in Jan and Feb
  baseflowend = mean(dat.sub$flow.er[dat.sub$month %in% list("12")]) # Set ending baseflow
conditions as the mean flow in Dec
```

```
 #create column index for the peaks defined by a rise in flow followed by a decline in flow
ocurring in three consecutive days
 peaks = which(diff(sign(diff(dat.sub$flow.er)))==-2)+1
 #create column index for the valleys defined by a decrease in flow followed by an increase in
flow ocurring in three consecutive days
 valleys = which(diff(sign(diff(dat.sub$flow.er)))==2)+1

 peakbase = dat.sub$flow.er[peaks]-baseflowinitial
 #print(peakbase)
 valleybase = dat.sub$flow.er[valleys] - baseflowinitial
 hydrographstart = 1 # Define HYDRGRAPHSTART

 for (n in 1:length(peakbase)){
  if (length(valleys) < 1){
   hydrographstart = peaks[n]
   peaks[n]
   break
  }

  if(peakbase[n] > 40){ # Check if threshold was met
   if (peaks[n] < valleys[1])  { # Check if first peak is greater than threshold
    hydrographstart = peaks[n]
    break
    }
   else {
    firstvalley = max(valleys[valleys<peaks[n]])
    }

    hydrographstart = firstvalley
    break
   }
 }

 bankfullflow = dat.sub$flow.er[dat.sub$flow.er > 8]
 maxflow$bankfullvol[k] = sum((bankfullflow)*86400) # sum the volume of water exceeding
bankfull flow
 maxflow$bankfulldays[k] = length(bankfullflow)
 hydrobounds[k,1] = hydrographstart
 BaseDays = dat.sub$flow.er[dat.sub$flow.er > Baseflow]
 maxflow$BaseflowDays[k] = length(BaseDays)
 maxflow$NonBFdays[k] = maxflow$LastDay[k] - maxflow$bankfulldays[k]

 if (k%%10 == 0){
```

```
  }
  hydrobounds$startdate[k] = as.character(dat.sub$Date[hydrobounds$start[k]])

}

# Write csv file of the temporary dat.sub datasheets for each year
#setwd(savepath)
write.csv(maxflow, "AnnualStats_6.29.20_Base_1.91_BestFit.csv", row.names = TRUE)

rm(list=setdiff(ls(), c("maxflow","dat","dat.almont","dat.bc","dat.er",
                        "hydrobounds","statistics","yearstats","years","colfunc",
                        "loadpath","savepath","mod2", "best.span")))

#### Extract Local Peaks above a specific flow rate above "bankfull"
#library("signal", lib.loc="~/R/win-library/3.2")
library("signal")

# Estimated bankkfull at 8 cms

for (k in 1:85){
  years2plot = years[k]
  dat.sub = subset(dat.er,year == years2plot)
  x1 = dat.sub$flow.er
  x1
  y1 = dat.sub$day

  #myfilter = butter(1, .2, type = 'low', plane='z')
  myfilter2 = filter(filt = sgolay(p = 12, n = 23), x = x1) # PEak FIlter started at 11
  #myfilter3 = fftfilt(rep(1, 10)/10, x1, n = 365)
  myfilter4 = filter(filt = sgolay(p = 7, n = 15), x = x1) # p = 5, n = 17 # 10 & 15 Oct 2017 # VALLEY
filter good as it gets

  #yfiltered = as.matrix(filter(myfilter, x1)) # apply filter
  yfiltered = myfilter2
  zfiltered = myfilter4
  ##print("**********************************************")
  ##print(years2plot)
  plot(dat.sub$flow.er,type = "n", main = paste(years2plot))
  lines(yfiltered,col = "red")
  lines(dat.sub$flow.er)
  points(dat.sub$flow.er)

  #points(yfiltered[peaks]~dat.sub$day[peaks], pch = 19)
```

```
 # PEaks
 peaks = which(diff(sign(diff(yfiltered)))==-2)+1 #identify the peaks by setting a threshold
where the next point decresaes by 2
 ##print(peaks)
 points(yfiltered[peaks]~dat.sub$yday[peaks], pch = 20, col = "orange")
 peaks2keep = (peaks[yfiltered[peaks] > 8])
 ##print("peaks 2 keep")
 ##print(length(peaks2keep))
 #SortPeaks <- peaks2keep[order(dat.sub$flow.er)]
 ###print(SortPeaks)
 ##print(peaks2keep)
 points(yfiltered[peaks2keep]~dat.sub$yday[peaks2keep], pch = 19, col = "red")

 # Valleys
 valleys = which(diff(sign(diff(zfiltered)))==2)+1 #identify the trophs by setting a threshold
where the next point incresaes by 2
 print("valleys")
 print(valleys)
 points(zfiltered[valleys]~dat.sub$yday[valleys], pch = 20, col = "green")
 valleys2keep = (valleys[zfiltered[valleys] < 100])
 print("valleys2keep")
 print(valleys2keep)
 points(zfiltered[valleys2keep]~dat.sub$yday[valleys2keep], pch = 19, col = "blue")

 #PeakFlows = yfiltered(dat.sub$flow.er[peaks2keep])

 truepeak = c()
 truepeak[1] = tail(which(dat.sub$flow.er == maxflow$flow.er[k]), n=1) # FInd the date of the
max flow and assign to peak flow
 ###print(truepeak)

 RealPeaks = c()
 leftthresh = c()
 rightthresh = c()
 PeakCount = 1
 #NotPeak = 0
 p = 0
 Rp = 0
 IsPeak = c()

 for (n in 1:length(peaks2keep)) {

  if (length(peaks2keep) == 0){ # If no peaks exceed bankfull...
```

```
    #truepeak = yday(maxflow$peakdate[k]) #Determine julian day of max peakflow if below
bankfull
    ###print(peaks2keep)
    PeakCount = 0
    ##print(PeakCount)
    break
  }


  IsPeak[n] = "N"
  leftthresh[n] = max(valleys2keep[valleys2keep < peaks2keep[n]]) # identify the valley
immediately before each peak above bankfull
  rightthresh[n] = min(valleys2keep[valleys2keep > peaks2keep[n]]) # identify the valley
immediately after each peack aboe bankfull
  p=p+1

  ##print(valleys2keep)
  ##print(leftthresh[n])
  ##print(peaks2keep[n])
  ##print(rightthresh[n])
  ##print(years[k])
  ##print(leftthresh[n])
  ##print(dat.sub$flow.er[leftthresh[n]])
  ##print(peaks2keep[n])
  ##print(dat.sub$flow.er[peaks2keep[n]])
  ##print(rightthresh[n])
  ##print(dat.sub$flow.er[rightthresh[n]])
  #if (abs(yfiltered[peaks2keep[n]]-yfiltered[leftthresh[n]]) < 5 |  # was <50 eliminates
  #   abs(yfiltered[peaks2keep[n]]-yfiltered[rightthresh[n]]) < 4){   # was <50
  #q = 0
  if (
     ((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[leftthresh[n]]) > 2)
       &  # peaks that are  >2 cms from valey to left
     (dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[rightthresh[n]]) > 2 & # peaks that are
>2 cms from valey to right
     ((dat.sub$flow.er[rightthresh[n]]) < 10 | (dat.sub$flow.er[leftthresh[n]]) < 10) &
     #(n < length(peaks2keep) & peaks2keep[n+1] < rightthresh[n]) |
     if (n > 1) {
       TRUE
       if (peaks2keep[n-1] < leftthresh[n]) {
         TRUE
         }
         else {
           FALSE
```

```
          #IsPeak[n] = "N"
          }
        } else {TRUE} #JUst changed this from FALSE to TRUE
      )
  {
        truepeak[n] = leftthresh[n]-1+tail(which(dat.sub$flow.er[leftthresh[n]:rightthresh[n]] ==
max(dat.sub$flow.er[leftthresh[n]:rightthresh[n]])),n=1)
        Rp = Rp + 1
        RealPeaks[Rp] = peaks2keep[n]
        IsPeak[n] = "Y"
        #print("1st check _____________________________________________")
        #print(peaks2keep[n])
        #print(IsPeak[n])
        ##print(p)
        ##print("1st Peaks to keep")
        ##print(peaks2keep[n])
        ##print(dat.sub$flow.er[peaks2keep[n]])
        ##print(rightthresh[n])
        ##print(dat.sub$flow.er[rightthresh[n]])
        ##print("Real peaks")
        ##print(length(RealPeaks))
        ##print(RealPeaks)
        ##print(RealPeaks[p])
        ##print(peaks2keep[n-1])
        ##print(RealPeaks[p-1])
        }

   else {
    ##print("Length of peaks 2 keep")
    ##print(length(peaks2keep))
    ##print("RealPeaks")
    ##print(length(RealPeaks))
    IsPeak[n] = "N"

   if (length(peaks2keep) == 2 & n == 1) { #length(RealPeaks == 0)) {
    #Rp = Rp + 1
    RealPeaks[1] = peaks2keep[n]
    IsPeak[n] = "Y"
    Rp = Rp + 1
    RealPeaks[Rp] = peaks2keep[n]
    ##print(length(RealPeaks))
    ##print("conditional met")
    ##print(length(RealPeaks))
    #print("3rd check_____________________________________________")
```

```
  #print(peaks2keep[n])
  #print(IsPeak[n])
 } else {

 #Check all but the last and first point for issues
 if ((n > 1) & (n < length(peaks2keep))) { # NEED TO CORRECT THIS LINE
  ##print("checking small cluster peaks")
  #print("4th check_____________________________________________")
  #print(peaks2keep[n])
  #print(IsPeak[n])
  IsPeak[n] = "N"
  #TRUE

 if(
  (((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[rightthresh[n]]) > 2) &
   (((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[leftthresh[n]]) < 2))# |
    #(dat.sub$flow.er[leftthresh[n]] > 10))
    &
   ((IsPeak[n-1] == "N") &
    (dat.sub$flow.er[leftthresh[n]] < 10 | dat.sub$flow.er[leftthresh[n-1]] < 10 )))  |

  (((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[rightthresh[n]]) < 2) &
   (((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[leftthresh[n]]) > 2)) &
   (dat.sub$flow.er[leftthresh[n]] < 10) &
  (leftthresh[n] > peaks2keep[n-1] | IsPeak[n-1] == "N") &
  rightthresh[n] < peaks2keep[n+1])
   #& (IsPeak[n-1] == "N")
   # THis creates an error because there is no value when there is no peak detected
  )
  {
  #TRUE
  truepeak[n] = leftthresh[n]-1+tail(which(dat.sub$flow.er[leftthresh[n]:rightthresh[n]] ==
max(dat.sub$flow.er[leftthresh[n]:rightthresh[n]])), n=1)
  Rp = Rp + 1
  RealPeaks[Rp] = peaks2keep[n]
  IsPeak[n] = "Y"
  #print("5th check _____________________________________________")
  #print(peaks2keep[n])
  #print(IsPeak[n])
  ##print(Rp)
  ##print("2nd Peaks to keep")
  ##print(peaks2keep)
  ##print(peaks2keep[n])
  ##print(peaks2keep[n-1])
```

```
  ##print(dat.sub$flow.er[peaks2keep[n]])
  ##print(rightthresh[n])
  ##print(dat.sub$flow.er[leftthresh[n]])
  ##print(dat.sub$flow.er[peaks2keep[n]])
  ##print("Real peaks")
  ##print(length(RealPeaks))
  ##print(RealPeaks) # Results in NA with no detected peak
  ##print(RealPeaks[Rp])
  ##print(RealPeaks[Rp-1])

 }

 } else {
 IsPeak[n] = "N"
 #print("6th check_____________________________________________")
 #print(peaks2keep[n])
 #print(IsPeak[n])

 }

 #Check last point and first point for discrepencies
  if (n == length(peaks2keep)) {
   #print("8th check_____________________________________________")
   #print(peaks2keep[n])
   IsPeak[n] = "N"
   #print(IsPeak[n])
   TRUE

   if(  ((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[leftthresh[n]]) > 2 &
      (dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[rightthresh[n]]) > 1 & # peaks that
are >2 cms from valey to right
      (dat.sub$flow.er[leftthresh[n]]) < 10 &
      leftthresh[n] > peaks2keep[n-1]) |

     (((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[rightthresh[n]]) > 2) &
      (((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[leftthresh[n]]) < 2)) &
      #(IsPeak[n-1] == "N"|
       (leftthresh[n] != rightthresh[n-1]))  #|

     #(((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[rightthresh[n]]) < 2) &
     # (((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[leftthresh[n]]) > 2)) &
     # (dat.sub$flow.er[leftthresh[n]] < 10))# &
     # leftthresh[n] > peaks2keep[n-1] &
      #rightthresh[n] < peaks2keep[n+1])
```

```
      )
      {
       TRUE
       truepeak[n] = leftthresh[n]-1+tail(which(dat.sub$flow.er[leftthresh[n]:rightthresh[n]] ==
max(dat.sub$flow.er[leftthresh[n]:rightthresh[n]])), n=1)
       Rp = Rp + 1
       RealPeaks[Rp] = peaks2keep[n]
       IsPeak[n] = "Y"
       #print("9th check _____________________________________________")
       #print(peaks2keep[n])
       #print(IsPeak[n])
      }

    } else {
      FALSE
      if (n == 1) {
       #print("10th check _____________________________________________")
       #print(peaks2keep[n])
       #print(IsPeak[n])
       ##print(dat.sub$flow.er[peaks2keep[n]])
       ##print(dat.sub$flow.er[rightthresh[n]])
       TRUE


       if ((dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[rightthresh[n]]) > 2 &
          (dat.sub$flow.er[peaks2keep[n]] - dat.sub$flow.er[leftthresh[n]]) > 2 &
          dat.sub$flow.er[leftthresh[n]] < 10 &
          dat.sub$flow.er[rightthresh[n]] < 10 &
          rightthresh[n] < peaks2keep[n+1]) {
         TRUE
         IsPeak[n] = "Y"
         Rp = Rp + 1
         RealPeaks[Rp] = peaks2keep[n]
         #print("11th check_____________________________________________")
         #print(peaks2keep[n])
         #print(IsPeak[n])
       }
      }
     }

   }
   }
   if (length(RealPeaks) == 0 & length(peaks2keep) != 0) {
    #TRUE
```

```
    RealPeaks[1] = 1
   }
   PeakCount = length(RealPeaks) #PeakCount + p
   ##print("PeakCount")
   ##print(PeakCount)
  }

  truepeak = na.omit(truepeak)
  ##print(truepeak)
  ##print(peaks2keep)
  #points(dat.sub$flow.er[truepeak]~dat.sub$day[truepeak], pch = 19)
  #points(yfiltered[valleys]~dat.sub$day[valleys], pch = 19, col = "blue")

  #hydrobounds$peak[k] = length(truepeak)
  hydrobounds$peak[k] = PeakCount
  bankfullflow = dat.sub$flow.er[dat.sub$flow.er > 8] # define bankfull flow threshold
  hydrobounds$bankfullvol[k] = sum((bankfullflow)*86400) # sum the volume of water
exceeding bankfull flow
  hydrobounds$bankfulldays[k] = length(bankfullflow)

}


yearstats = cbind(maxflow[,-c(4,5)],hydrobounds[,-c(1,2)],statistics[,-1])
# You will have to rename the headers in excel unless I get some time to go back and clean
things up a bit

#setwd(savepath)
write.csv(yearstats,"YearlyStatistics_6.29.20_Base_1.91_BestFit.csv")


rm(list=setdiff(ls(), c("maxflow","dat","dat.almont","dat.bc","dat.er",
                        "hydrobounds","statistics","yearstats","years","colfunc",
                        "loadpath","savepath","mod2", "best.span")))


# This code will average variables for periods between imagery along the East River

# Author: Nicholas A. Sutfin
# Date: April 2020

library("plyr")
#library("smwrBase", lib.loc="~/R/win-library/3.2")
library("lattice") #, lib.loc="C:/Program Files/R/R-3.3.0/library")
```

```
library("lubridate")
library("hydroGOF")

# User space same as save path from steps 1-4
savepath = '/Users/NicholasSutfin/Documents/EastRiver/ER_Rcode/Baseflow_1.91_BestFit/' #
Calculating slope as line between 1st and last points (2p)
setwd(savepath)
# Load ALmont data for 2015-2017 as csv file, convert to SI units, code the date as a date, and
define the year
#Alm_Q <- read.csv("ER_AlmQ_2015-2017.csv", header=TRUE)
AnnualStats <- read.csv("YearlyStatistics_6.29.20_Base_1.91_BestFit.csv", header=TRUE)
AnnualStats$period = NA

for (i in 2:length(AnnualStats$year)) {
 #AnnualStats$TimeSinceBF[i] = AnnualStats$BF_startDay[i] + AnnualStats$DaysSinceBF[i-1]
 if (AnnualStats$year[i] < 1955){
  AnnualStats$period[i] = "before1955"
 }
 if (AnnualStats$year[i] > 1954 & AnnualStats$year[i] < 1974){
  AnnualStats$period[i] = "1955to1973"
 }
 if (AnnualStats$year[i] > 1973 & AnnualStats$year[i] < 1984){
  AnnualStats$period[i] = "1974to1983"
 }
 if (AnnualStats$year[i] > 1983 & AnnualStats$year[i] < 1991){
  AnnualStats$period[i] = "1984to1990"
 }
 if (AnnualStats$year[i] > 1990 & AnnualStats$year[i] < 2002){
  AnnualStats$period[i] = "1991to2001"
 }
 if (AnnualStats$year[i] > 2001 & AnnualStats$year[i] < 2012){
  AnnualStats$period[i] = "2002to2011"
 }
 if (AnnualStats$year[i] > 2011 & AnnualStats$year[i] < 2016){
  AnnualStats$period[i] = "2012to2015"
 }
 if (AnnualStats$year[i] > 2015){
  AnnualStats$period[i] = "after2015"
 }
}

#na.rm(AnnualStats)

DecadalStats = ddply(AnnualStats, ~period, summarise,
```

```
            MeanPeakDay = mean(PeakDay),
            MeanPeakQ = mean(flow.er), MaxPeakQ = max(flow.er),
            MeanBFDuration = mean(BankfullDuration, na.rm=TRUE), MaxBFDuration =
max(BankfullDuration, na.rm=TRUE),
            MeanBFDays = mean(bankfulldays, na.rm=TRUE), MaxBFDays = max(bankfulldays,
na.rm=TRUE),
            MeanBaseDuration = mean(BaseDuration, na.rm=TRUE), MaxBaseDuration =
max(BaseDuration, na.rm=TRUE),
            MeanBaseDays = mean(BaseflowDays, na.rm=TRUE), MaxBaseDays =
max(BaseflowDays, na.rm=TRUE),
            MeanDaysAfterBF = mean(CummDaysAfterBF, na.rm=TRUE), MaxDaysAfterBF =
max(CummDaysAfterBF),
            MeanDaysB4_BF = mean(CummDaysBeforeBF, na.rm=TRUE), MaxDaysB4_BF =
max(CummDaysBeforeBF, na.rm=TRUE),
            MeanNonBFdays = mean(NonBFdays, na.rm=TRUE), MaxNonBFdays =
max(NonBFdays, na.rm=TRUE),
            MeanBaseDay = mean(Base_endDay, na.rm=TRUE), MeanBF_EndDay =
mean(BF_endDay, na.rm=TRUE),
            MeanPeaks = mean(peak, na.rm=TRUE), MaxPeaks = max(peak, na.rm=TRUE),
            MeanTotSlope = mean(TotalSlope, na.rm=TRUE), MaxTotSlope = max(TotalSlope,
na.rm=TRUE),
            MeanBFSlope = mean(BFslope, na.rm=TRUE), MaxBFSlope = max(BFslope,
na.rm=TRUE),
            MeanPeakSlope = mean(PeakSlope, na.rm=TRUE), MaxPeakSlope = max(PeakSlope,
na.rm=TRUE),
            MeanAnnualVol = mean(AnnualVol), MaxAnnualVol = max(AnnualVol),
TotAnnualVol = sum(AnnualVol),
            # ALtered 6.26.2020 to include volume for days above BF rather than all days
between first and last BF days
            MeanBFVol = mean(bankfullvol,na.rm=TRUE), MaxBFVol =
max(bankfullvol,na.rm=TRUE),
            TotBFDuration = sum(BankfullDuration, na.rm=TRUE), TotBaseDuration =
sum(BaseDuration, na.rm=TRUE),
            TotNonBFdays = sum(NonBFdays, na.rm=TRUE), TotBF_EndDay = sum(BF_endDay,
na.rm=TRUE),
            TotDaysB4_BF = sum(CummDaysBeforeBF, na.rm=TRUE), TotDaysAfterBF =
sum(CummDaysAfterBF),
            TotBFVol = sum(BFVol, na.rm=TRUE))

#setwd(savepath)
write.csv(DecadalStats, "TimePeriodStats_6.29.20_1.91_BestFit.csv", row.names = TRUE)

# This code will examine 15 min hydrograph datasets from the ALmont gage and East RIver
study site
```

```
# to quantify fluctuations above and below bankfull along the recession limb

# Author: Nicholas A. Sutfin
# Date: Oct. 18th 2017

# This code will examine to hydrograph dataset, select matching days
# and times and conduct a regression that can be used to fill in missing data
# Author: Nicholas A. Sutfin
# Date: Oct. 18th 2017

library(plyr)
library(chron)
library(tidyr)
#library(smwrBase, lib.loc=~/R/win-library/3.2)
library(lattice) #, lib.loc=C:/Program Files/R/R-3.3.0/library)
library(lubridate)
library(hydroGOF)
library(OHLCMerge)
library(corrplot)
library(lmtest)
library(car)
library(MASS)
library(Hmisc)

# Set user space on LANL PC
loadpath = '/Users/NicholasSutfin/Documents/EastRiver/ER_Rcode'
savepath = '/Users/NicholasSutfin/Documents/EastRiver/ER_Rcode'
setwd(loadpath)
#setwd("/Users/306722/Documents/EastRiver/ER_Rcode")

# Load ALmont data for 2015-2017 as csv file, convert to SI units, code the date as a date, and
define the year
Alm_15Q <- read.csv("Almont_30minQ_1987_2020.csv", header=TRUE) #load USGS discharge
data
Alm_15Q$Discharge_cfs =
as.numeric(levels(Alm_15Q$Discharge_cfs))[Alm_15Q$Discharge_cfs] # convert Q factors to
numeric values
which(is.na(Alm_15Q$Discharge_cfs) == TRUE) #Check for NA values
Alm_15Q$AlmQ_cms = Alm_15Q$Discharge_cfs*0.0283168 # Calulate Q conversion from cfs to
cms
which(is.na(Alm_15Q$Discharge_cfs) == TRUE) # check for NA values after numeric conversion

Alm_15Q$date = as.Date(Alm_15Q$date, format="%m/%d/%y") # convert Q factors to numeric
values
```

```
Alm_15Q$DaTime = paste(Alm_15Q$date, Alm_15Q$time)
Alm_15Q$DateTime = as.POSIXct(Alm_15Q$DaTime, format = "%Y-%m-%d %H:%M")
Alm_15Q$year = year(Alm_15Q$Date)
Alm_15Q$month = month(Alm_15Q$Date)
Alm_15Q$Calday = day(Alm_15Q$Date)
Alm_15Q$Yday = yday(Alm_15Q$Date)
#Alm_15Q$Yday = yday(Alm_15Q$Date)
Alm_15Q = as.data.frame(Alm_15Q)
#_____________________________________________________________________
# Load Pump house data for 2015-2017 as csv file, convert to SI units, code the date as a date,
and define the year
PH_10Q <- read.csv("PHQ_2014_2018.csv", header=TRUE)
#PH_10Q <- read.csv("PH_10Q.csv", header=TRUE) #load East RIver pump house discharge data
PH_10Q$DateTime = as.POSIXct(PH_10Q$date, format = "%m/%d/%y %H:%M")
PH_10Q$year = year(PH_10Q$DateTime)
PH_10Q$month = month(PH_10Q$DateTime)
PH_10Q$Calday = day(PH_10Q$DateTime)
PH_10Q$Time = format(as.POSIXct(strptime(PH_10Q$DateTime, "%Y-%m-%d %H:%M",tz=""))
,format = "%H:%M")
PH_10Q$Yday = yday(PH_10Q$DateTime)
PH_10Q = as.data.frame(PH_10Q)
#plot(PH_10Q$DateTime, PH_10Q$PHQ_cms, type = "l", col = "blue")

#_____________________________________________________________________
# Find matching date-time combinations and create new dataset
#PH_Q_match =
Alm_15Qnew1  = Alm_15Q[,c(4,6,7,8,9,2,10)][!duplicated(Alm_15Q$DateTime),]
Alm_15Qnew = Alm_15Qnew1[which(is.na(Alm_15Qnew1$DateTime) == FALSE),]
PH_10Qnew  = PH_10Q[,c(2:8)]

Q_int <- intersect.POSIXct(PH_10Qnew$DateTime, Alm_15Qnew$DateTime)
Alm_Q_match <- Alm_15Qnew[Alm_15Qnew$DateTime %in% Q_int, ] #Alm_15Q[Q_int, ] #
PH_Q_match <- PH_10Qnew[PH_10Qnew$DateTime %in% Q_int, ] #PH_10Q[Q_int, ] #
Q_diff <- setdiff(PH_Q_match$DateTime, Alm_Q_match$DateTime)
#which(PH_Q_match$DateTime == NA)
#which(Alm_Q_match$DateTime == NA)
All_Qmatch <- cbind(Alm_Q_match, PH_Q_match)

# Create a smaller zoomed in plot to view Q around Bankfull Q (8 cms)
plot(All_Qmatch$DateTime, All_Qmatch$PHQ_cms, type = "l",
   ylim = c(5,10), xlab = "Day of Year", ylab = "Discharge (cms)", lwd = 1, main = "East River 2015
recession")

# Plot discharge data
```

```
plot(All_Qmatch$DateTime, All_Qmatch$AlmQ_cms, col = "blue", type = "l")
lines(All_Qmatch$DateTime, All_Qmatch$PHQ_cms, col = "royalblue", type = "l")
#_____________________________________________________________________
# Linear regression between the Almont and PH gauges 2014-2016

Qreg <- lm(All_Qmatch$PHQ_cms ~ All_Qmatch$AlmQ_cms, data = All_Qmatch)
summary(Qreg)
Qreg # adjusted R squared = 0.95
# For all days: PHQ = -0.081804 + 0.211284(Alm)
# Excluding frozen days, regression output: PHQ = 0.010948 + 0.211611(Alm)

par(mfrow=c(1,1), mar=c(4,4,2,2), cex = 1, lwd = 1)
plot(All_Qmatch$AlmQ_cms, All_Qmatch$PHQ_cms, col = "blue",
   xlab = "Discharge at Almont (cms)", ylab = "Discharge at Study Site (cms)")
lines(All_Qmatch$AlmQ_cms, Qreg$coefficients[1] +
Qreg$coefficients[2]*All_Qmatch$AlmQ_cms,
    col = "black")
par(cex = 0.6)
#points(All_Qmatch$AlmQ_cms, All_Qmatch$PHQ_cms, pch = 19, col = "red")
text(10, 15, expression("r"^{2} ~"= 0.94"), cex = 1.5)

# Use regression to extend daily Q for PH based on Almont flow
#_____________________________________________________________________________
____
# regression output: PHQ = -0.081804 + 0.211284(Alm)

# Reduce Almont Data size
Alm_15Q_sel = Alm_15Qnew[((Alm_15Qnew$time == "0:00") | (Alm_15Qnew$time == "1:00")
| (Alm_15Qnew$time == "2:00") |
                  (Alm_15Qnew$time == "3:00") |(Alm_15Qnew$time == "4:00") |
(Alm_15Qnew$time == "5:00") |
                  (Alm_15Qnew$time == "6:00") |(Alm_15Qnew$time == "7:00") |
(Alm_15Qnew$time == "8:00") |
                  (Alm_15Qnew$time == "9:00") |(Alm_15Qnew$time == "10:00") |
(Alm_15Qnew$time == "11:00") |
                  (Alm_15Qnew$time == "12:00") |(Alm_15Qnew$time == "13:00") |
(Alm_15Qnew$time == "14:00") |
                  (Alm_15Qnew$time == "15:00") | (Alm_15Qnew$time == "16:00") |
(Alm_15Qnew$time == "17:00") |
                  (Alm_15Qnew$time == "18:00") | (Alm_15Qnew$time == "19:00") |
(Alm_15Qnew$time == "20:00") |
                  (Alm_15Qnew$time == "21:00") | (Alm_15Qnew$time == "22:00") |
(Alm_15Qnew$time == "23:00") |
                  (Alm_15Qnew$time == "24:00")), ]
```

```
All_Q_1987_2020 = Alm_15Q_sel[which(is.na(Alm_15Q_sel$AlmQ_cms) == FALSE), ] #[
,c(6,1,7:9,2,10,4)]
All_Q_1987_2020$Mod_PHQ_cms = Qreg$coefficients[1] +
Qreg$coefficients[2]*All_Q_1987_2020$AlmQ_cms

# Plot a zoomed in window of the recession limb for 2017
Flow2017 = All_Q_1987_2020[All_Q_1987_2020$year == 2017,]
Recession2017 = Flow2017[Flow2017$month == 6,]
Recession2017 = Recession2017[Recession2017$Calday > 6,]
DailyQ = ddply(Recession2017, ~Yday, summarise,
          MeanQ = median(Mod_PHQ_cms),
          DateTime = min(DateTime))

Rmax = max(Recession2017$DateTime)
Rmin = min(Recession2017$DateTime)
window1 <- data.frame(xmin=Rmin, xmax=Rmax, ymin=8, ymax=11)
window2 <- data.frame(xmin=Rmin, xmax=Rmax, ymin=5, ymax=12)


ggplot(data=Recession2017, aes(x=DateTime, y=Mod_PHQ_cms)) +
  geom_path() +
  geom_line(data = DailyQ, aes(x = DateTime , y = MeanQ, colour = 003399)) +
  geom_line(data=Recession2017, aes(x=DateTime, y=Mod_PHQ_cms)) +
  labs(y = expression(paste("Discharge (m"^"3", "s"^"-1",")")), x = "") +
  theme(axis.title.x = element_blank()) +
  theme(text = element_text(size=13)) +
  scale_y_continuous(minor_breaks = seq(6,16,1), breaks = seq(6,16,2)) +
  geom_rect(data=window2, aes(xmin=Rmin, xmax=Rmax, ymin=5, ymax=10), fill="blue",
alpha=0.20, inherit.aes = FALSE) +
  geom_rect(data=window1, aes(xmin=Rmin, xmax=Rmax, ymin=7.95, ymax=8.05), fill="red",
alpha=0.5, inherit.aes = FALSE)


  #geom_rect(x=x, aes(xmin=Rmin, xmax=Rmax, ymin=8, ymax=11, alpha=.5))
  #geom_density(aes(, alpha=.5))


#########################################################################

#########################################################################
#   Recession Limb Characteristics
#########################################################################

#########################################################################
```

```
years = c("1988","1989","1990","1991","1992","1993","1994","1995","1996",
        "1997","1998","1999","2000","2001","2002","2003","2004","2005",
        "2006","2007","2008","2009","2010","2011","2012","2013","2014",
        "2015","2016","2017","2018","2019")

DielYears = data.frame("Years" = years)
DielYears$PeakDate = as.POSIXlt(All_Q_1987_2020$DateTime[1], format = "%Y-%m-%d
%H:%M:%S")
par(cex = 1, mar = c(4,4,2,1))
BFmin = 5
BFmax = 10
DielFluctuation = 2

for (p in 1:length(years)) {
  DataYear = years[p]
  DielData =  subset(All_Q_1987_2020, year%in%DataYear)
  DielRec = 0
  AllDiel = 0
  DielYears$PeakFlow[p] = max(DielData$Mod_PHQ_cms[which(is.na(DielData$Mod_PHQ_cms)
== FALSE)]) #max(DielData$Mod_PHQ_cms)
  DielYears$PeakDate[p] = as.POSIXlt(DielData$DateTime[max(which(DielData$Mod_PHQ_cms
== DielYears$PeakFlow[p]))], format = "%Y-%m-%d %H:%M:%S")
  DielYears$PeakDay[p] = yday(DielYears$PeakDate[p])
  DielYears$PostPeakDays[p] = max(DielData$Yday) - DielYears$PeakDay[p]
  PeakIndex = which(DielData$DateTime == DielYears$PeakDate[p])
  DielPeaks = c()
  DielTotal = 0
  maxDiel = 0
  minDiel = 0
  #print("_____")
  #print(years[p])
  #print(DielPeaks)
  #print(minDiel)
  #print(maxDiel)
  #print(AllDiel)
  #print(DielRec)


  #Find unique days for the year on record
  UniqDays = unique(DielData$Yday)
  PostPeakUniq = UniqDays[UniqDays > DielYears$PeakDay[p]]
```

```
 if (DielYears$PeakFlow[p] > 6) {

  for (r in 2:length(UniqDays)) {
   # Assign daily max and min discharge values
   DailyFlow = subset(DielData, DielData$Yday == UniqDays[r])
   Dmax = max(DailyFlow$Mod_PHQ_cms)
   #DmaxIndex = which(DailyFlow$Mod_PHQ_cms == Dmax)
   Dmin = min(DailyFlow$Mod_PHQ_cms)

   if (((Dmax < BFmax) | (Dmin > BFmin)) & ((Dmax - Dmin) > DielFluctuation)) {
    AllDiel = AllDiel + 1
   }
   DielYears$AllDiel[p] = AllDiel # Record number of times Q crosses BF during the entire year
  }
  #print("----------------------")
  #print(years[p])
  #print("YES")
  for (q in 1:length(PostPeakUniq)) {
   # Assign daily max and min discharge values
   DailyFlow = subset(DielData, DielData$Yday == PostPeakUniq[q])
   Dmax = max(DailyFlow$Mod_PHQ_cms)
   #DmaxIndex = which(DailyFlow$Mod_PHQ_cms == Dmax)
   Dmin = min(DailyFlow$Mod_PHQ_cms)

   if (((Dmax < BFmax) | (Dmin > BFmin)) & ((Dmax - Dmin) > DielFluctuation)) {

    DielRec = DielRec + 1
    DielPeaks[DielRec] = DailyFlow$Yday # Index the day of year for each Q that crosses BF
after peak flow
    #print(length(DielPeaks))
    #print(DielPeaks)
    maxDiel = max(DielPeaks)
    minDiel = min(DielPeaks)
    DielRange = Dmax - Dmin
    DielTotal = DielTotal + DielRange
    DielYears$minDiel[p] = minDiel
    DielYears$maxDiel[p] = maxDiel

    # Plot portion of recession limb within bankfull window
    days = c(minDiel, maxDiel)
    Qlow = c(BFmin, BFmin)
    Qhigh = c(BFmax, BFmax)
    #plot(DielData$day, DielData$Mod_PHQ_cms, type = "l",  main = paste(years[p]),
```

```
        #ylim = c(6,10), xlim = c(DielYears$minDiel[p]-1,DielYears$maxDiel[p]+1),
        #xlab = "Day of Year", ylab = "Discharge (cms)", lwd = 1)
    #lines(c(0,250), c(8,8), col="blue")

    # plot a transparent band around the bankfull window
    #polygon(c(days, rev(days)), c(Qlow, Qhigh), border = NA,
         #col = rgb(red = 0.0, green = 0.0, blue = 0.5, alpha = 0.4))
   }
   AveDielRange = DielTotal/DielRec
   DielYears$TotalDielRange[p] = DielTotal
   DielYears$AveDielRange[p] = AveDielRange
   DielYears$DielRec[p] = DielRec # Record number of times Q crosses BF during recession limb

  }
  #plot(DielData$day, DielData$Mod_PHQ_cms, type = "l",  main = paste(years[p]),
      #xlab = "Day of Year", ylab = "Discharge (cms)", lwd = 1)
 }

 else {
  #print("----------------------")
  #print(years[p])
  #print("NO")
  DielYears$TotalDielRange[p] = NA
  DielYears$AveDielRange[p] = NA
  DielYears$DielRec[p] = NA
  DielYears$minDiel[p] = 0
  DielYears$maxDiel[p] = 0
  }
 }

DielYears

# THis data was combined with the average statistics form the hydrologic and
# imagery analysis to produce the datasheet used below

################################################################################
####################################
# Conduct Multiple Regression to examine role of diel fluctuations on erosion
################################################################################
####################################

# Load data on Mac with slope analysis from primary 60 year analysis derived from daily mean
data
# Set user space
```

```
savepath =
'/Users/NicholasSutfin/Documents/EastRiver/ER_Rcode/Baseflow_0.49_2p_corrected/' #
Calculating slope as line between 1st and last points (2p)
setwd(savepath)
write.csv(DielYears,"DielRecessionDate_6.30.20_2cms_>6_5_10.csv")

# Load other hydrologic variables from baoder analysis and 6 year hydro record
YearlyHydroStats <- read.csv("DielRecessionRegData_6.29.20.csv", header=TRUE)

# cbind annual hydrologic data with diel data
DielRegData = cbind(DielYears, YearlyHydroStats)

DielRegData = DielRegData[(which(is.na(DielRegData$DielRec) == FALSE)), ]

for (i in 1:length(DielRegData$Years)) {
  if (DielRegData$DielRec[i] == 0) {
    DielRegData$AveDielRange[i] = 0
  }
}

#=============================================================
#Assign variables
#RespVar = DielRegData$AveDielRange
Preds = subset(DielRegData, select = c(6:9,16:18)) #c(3:6,9:52))
Preds[, c(1:7)] <- sapply(Preds[, c(1:7)], as.numeric)

# examine subset correlations
par(mfrow=c(1,1), mar=c(3,3,3,2), cex = 1.3)
DataCorr = cor(Preds, method = "pearson")
corrplot(DataCorr)

CorrT = rcorr(as.matrix(Preds), type = "pearson")
CorrRtable = data.frame(CorrT$r)
CorrPtable = data.frame(CorrT$P)
CorrT

write.csv(CorrRtable, file = "DielData_RCorrs_6.30.20_2cms_>6_5_10.csv") # with new data
from new stats calculated June 2020
write.csv(CorrPtable, file = "DielData_PCorrs_6.30.20_2cms_>6_5_10.csv")

################################################
# Number of Diel Fluctuations
#_____________________________________________
```

```
cor.test(Preds$TotalSlope, Preds$DielRec)
DielRecReg = lm(Preds$TotalSlope ~ Preds$DielRec, data=Preds)
summary(DielRecReg)

ggplot(Preds, aes(x=TotalSlope, y=DielRec)) +
  geom_point(color='#D55E00', size = 3) +
  geom_smooth(method=lm, color='#2C3E50', linetype="dashed") +
  theme(text = element_text(size=13)) +
  labs(title = "2cms fluctuations >6cms from 5-10cms window",
       y=expression(paste("Number of diel fluctuations > 2 m"^"3", "s"^"-1")),
       x = expression(paste("Slope of recession limb (m"^"3", "s"^"-1", "day"^"-1",")")))

################################################
# Total sum magnitude of diel fluctuation
#_____________________________________________

cor.test(Preds$TotalSlope, Preds$TotalDielRange)

ggplot(Preds, aes(x=TotalSlope, y=TotalDielRange)) +
  geom_point(color='#D55E00', size = 3) +
  geom_smooth(method=lm, color='#2C3E50', linetype="dashed") +
  theme(text = element_text(size=13)) +
  labs(title = "2cms fkuctuations >6cms from 5-10cms window",
       y=expression(paste("Summed magnitude of diel fluctuation")),
       x = expression(paste("Slope of recession limb (m"^"3", "s"^"-1", "day"^"-1",")")))

################################################
# Average magnitude of diel fluctuation
#_____________________________________________

cor.test(Preds$TotalSlope, Preds$AveDielRange)

ggplot(Preds, aes(x=TotalSlope, y=AveDielRange)) +
  geom_point(color='#D55E00', size = 3) +
  geom_smooth(method=lm, color='#2C3E50', linetype="dashed") +
  theme(text = element_text(size=13)) +
  labs(title = "2cms fkuctuations >6cms from 5-10cms window",
       y=expression(paste("Average magnitude of diel fluctuation (m"^"3","s"^"-1", ")")),
       x = expression(paste("Slope of recession limb (m"^"3", "s"^"-1", "day"^"-1",")")))
```