```
"
Created on 10/27/2017

author: James Raines

Purpose: Thesis data science; organizing, visualizing, and running statistics on forest change and socioeconomics
around natural protected areas in Mexico.
"

#functions
remove.outliers2 <- function(x, na.rm = TRUE) {
  "
  Removes outliers beyond the interquartile distance of the minimum value.

  :param x: 1D vector; values
  :param na.rm: boolean; do not change. Removes NA values from calculations

  "
  qnt <- quantile(x, probs=c(.25, .75), na.rm = na.rm)
  iqr <- IQR(x, na.rm = na.rm)
  multFactor <- abs((min(x, na.rm = na.rm)-qnt[1])/iqr)
  H <- multFactor * iqr

  outs <- x[x > (qnt[2] + H)]
  print(paste('outliers removed: ', length(outs[is.finite(outs)]), sep = ''))
  print(outs[is.finite(outs)])

  y <- x
  y[x > (qnt[2] + H)] <- NA
  y
}

plotSE <- function(changeType, seType, rmOutliers, corTest) {
  "
  Creates a 4x4 plot of socioeconomic factors against forest change type

  :param changeType: string; forest or fragmentation change. Options-'forest', 'frag'
  :param seType: string; population or poverty index. Options-'pop', 'pov'
  :param reOutliers: boolean; removes outliers in changeType.
  :param corTest: boolean; calculates pearson's r and creates title 'sig' if p-value < 0.05

  "
  par(mfrow=c(4,4))
  for (forest in forests) {
    for (buffer in buffers) {
      pop <- df$pop[which(df$forestType == forest & df$bufferzone == buffer)]
      pov <- df$pov[which(df$forestType == forest & df$bufferzone == buffer)]
      frag <- df$fragChange[which(df$forestType == forest & df$bufferzone == buffer)]
      fors <- df$forestChange[which(df$forestType == forest & df$bufferzone == buffer)]
      if (changeType == 'frag') {
        changeT = frag
        ylab = 'frag change (%)'
      } else if (changeType == 'forest') {
        changeT = fors
        ylab = 'forest change (%)'
      }
      if (seType == 'pop') {
        seT = pop
        xlab = 'total population'
      } else if (seType == 'pov') {
        seT = pov
        xlab = 'average poverty index'
      }
      if (rmOutliers == T) {
        changeT <- remove.outliers2(changeT)
      }
      changeT[is.infinite(changeT)] <- NA
      plot(changeT~seT, xlab = xlab, ylab = ylab)
      mod <- lm(changeT~seT)
      abline(mod)
      if (corTest == T) {
        corT <- cor.test(seT, changeT)
        print(corT)
        if (corT$p.value < 0.05) {
          title('sig')
        }
      }
    }
  }
  par(mfrow=c(1,1))
}

plotCB <- function(changeType, rmOutliers) {
  "
  Creates a 2x2 plot of concentric buffer zones against forest change type.
```

```
  Runs either ANOVA or Kruskal-Wallis test depening on if outliers are removed.

  :param changeType: string; forest or fragmentation change. Options-'forest', 'frag'
  :param reOutliers: boolean; removes outliers in changeType.

  "
  par(mfrow=c(2,2))
  xlab = 'buffer zone'
  for (forest in forests) {
    dfFunc <- data.frame()
    for (buffer in buffers) {
      frag <- df$fragChange[which(df$forestType == forest & df$bufferzone == buffer)]
      fors <- df$forestChange[which(df$forestType == forest & df$bufferzone == buffer)]
      dfUpdate <- data.frame(cbind(frag, fors, buffer))
      dfFunc <- rbind(dfFunc, dfUpdate)
    }
    if (changeType == 'frag') {
      changeT = as.numeric(as.character(dfFunc$frag))
      ylab = 'frag change (%)'
    } else if (changeType == 'forest') {
      changeT = as.numeric(as.character(dfFunc$fors))
      ylab = 'forest change (%)'
    }
    changeT[is.infinite(changeT)] <- NA
    if (rmOutliers == T) {
      changeT <- remove.outliers2(changeT)
      plot(changeT~dfFunc$buffer, xlab = xlab, ylab = ylab)
      anov <- anova(lm(changeT~dfFunc$buffer))
      print(anov)
      if((anov$`Pr(>F)` < 0.05)[1]) {
        title('sig')
      }
    } else if (rmOutliers == F) {
      plot(changeT~dfFunc$buffer, xlab = xlab, ylab = ylab)
      krus <- kruskal.test(changeT~dfFunc$buffer)
      print (krus)
      if (krus$p.value < 0.05) {
        title('sig')
      }
    }
  }
  par(mfrow=c(1,1))
}

#sets path to socioeconomic data
path <- scan(what = "character", allowEscapes=F, nlines = 1)
C:\Users\jrain\Documents\Graduate_School\Thesis\Data\
path <- gsub("\\\\","/",path)
setwd(path)

#set variables
df <- read.csv('thesisResults_SeForestFrag.csv')
forests <- unique(df$forestType)
buffers <- unique(df$bufferzone)

### socioeconomics ###
plotSE('forest', 'pop', T, T)
plotSE('forest', 'pov', T, T)
plotSE('forest', 'pop', F, T)
plotSE('forest', 'pov', F, T)

plotSE('frag', 'pop', T, T)
plotSE('frag', 'pov', T, T)
plotSE('frag', 'pop', F, T)
plotSE('frag', 'pov', F, T)


### concentric buffers ###
plotCB('frag', T)
plotCB('frag', F)

plotCB('forest', T)
plotCB('forest', F)
```