```r
##================================
# File: cleaningFuncs.R
# Author: Jianying Li
##================================
peak.quick <- function (x, y){
  return(x[which(diff(sign(diff(y)))==-2)])
}

getFirstPop <- function (dtIN)
{
  get.den <- density(dtIN)
  peaks <- peak.quick (get.den$x, get.den$y)
#str(peaks)

  dt.normed = dtIN - peaks[1]
  dt.first.left <- dt.normed[which(dt.normed < 0)]
  dt.first.right <- -dt.first.left
  dt.first <- c(dt.first.left, dt.first.right)
  str(dt.first)
  return (dt.first)
}

getPopWIndex <- function (dtIN, where2start)
{
  get.den <- density(dtIN)
  peaks <- peak.quick (get.den$x, get.den$y)
  #str(peaks)

  dt.normed = dtIN - peaks[where2start]
  dt.first.left <- dt.normed[which(dt.normed < 0)]
  dt.first.right <- -dt.first.left
  dt.first <- c(dt.first.left, dt.first.right)
  str(dt.first)
  return (dt.first)
}

getSecondPop <- function (dtIN)
{

  get.den <- density(dtIN)
  peaks <- peak.quick (get.den$x, get.den$y)
  #str(peaks)

  dt.normed = dtIN - peaks[2]
  dt.first.left <- dt.normed[which(dt.normed < 0)]
  dt.first.right <- -dt.first.left
  dt.first <- c(dt.first.left, dt.first.right)
  str(dt.first)
  return (dt.first)
}


cleanFirstPop <- function ( firstPeak,
                            dt.first,
                            dt.raw  )

{
  ##  Filter starts here...

  first.den <- density(dt.first + firstPeak)
  tempDen <-  first.den
  tempDen$y <- first.den$y/sum(first.den$y)

  ##  Retain data on the right of the first peak
  ##            SAME AS
  ##    Remove data on the left  of the fist peak

  dt.right.of.peak <- dt.raw[which(dt.raw >=  firstPeak)]

  ##    Retain data on the right of max of the first population

  dt.right.of.peak.max <- dt.right.of.peak [which(dt.right.of.peak >=max(dt.first+firstPeak))]

  ##    Data fall between the right of the first peak and the left of max of the first population
  dt.between.peak.max <- dt.right.of.peak [-which(dt.right.of.peak >=max(dt.first+firstPeak))]

  ##    Now, remove the data according to the "estimated proportion"
  ##    Between two adjacent populations

  adjust = 0;
  dt2filter <-dt.between.peak.max
  #     str(dt2filter)
  for (i in 1:256)
  {
    temp = 0
    l.bound <- i + 255
    h.bound <- i + 256
    num.of.data <- ((tempDen$y[l.bound] + tempDen$y[h.bound])/2)*(length(dt.first))
    candidate <- which(dt2filter > tempDen$x[l.bound] & dt2filter  < tempDen$x[h.bound])
    if (length(candidate) >=1)
    {
      if (length(candidate) > floor(num.of.data))
      {
        temp = num.of.data - floor(num.of.data)
        data2exclude <- sample(candidate, floor(num.of.data))
        if (length(data2exclude) >=1 )
        {
          dt2filter <- dt2filter[-data2exclude]
          adjust = adjust + temp
        }
      }else{
        dt2filter <- dt2filter[-candidate]
      }
    }
  }
  num2salvage <- sample (c(1:length(dt2filter)), ceiling(adjust))       #FIXME: Manully fixting
  dt2filter <- dt2filter[-num2salvage]
  dt.retain <- c(dt.right.of.peak.max, dt2filter)
  return (dt.retain)

}

followUpClean  <- function ( firstPeak,
                            dt.first,
                            dt.raw  )

{
  ##  Filter starts here...
  #returnList <- list()

  first.den <- density(dt.first + firstPeak)
  tempDen <-  first.den
  tempDen$y <- first.den$y/sum(first.den$y)

  ##  Retain data on the right of the first peak
  ##      SAME AS
  ##  Remove data on the left  of the fist peak

  dt.right.of.peak <- dt.raw[which(dt.raw >=  firstPeak)]

  ##    Retain data on the right of max of the first population

  dt.right.of.peak.max <- dt.right.of.peak [which(dt.right.of.peak >=max(dt.first+firstPeak))]

  ##    Data fall between the right of the first peak and the left of max of the first population
  dt.between.peak.max <- dt.right.of.peak [-which(dt.right.of.peak >=max(dt.first+firstPeak))]

  ##    Now, remove the data according to the "estimated proportion"
  ##    Between two adjacent populations

  adjust = 0;
  dt2filter <- dt.between.peak.max
  dt2retain  <- 0
  #     str(dt2filter)
  for (i in 1:256)
  {
    temp = 0
    l.bound <- i + 255
    h.bound <- i + 256
    num.of.data <- ((tempDen$y[l.bound] + tempDen$y[h.bound])/2)*(length(dt.first))
    candidate <- which(dt2filter > tempDen$x[l.bound] & dt2filter  < tempDen$x[h.bound])
    if (length(candidate) >=1)
    {
      if (length(candidate) > floor(num.of.data))
      {
        temp = num.of.data - floor(num.of.data)
        data2exclude <- sample(candidate, floor(num.of.data))
        if (length(data2exclude) >=1 )
        {
          dt2filter     <- dt2filter[-data2exclude]
          dtRetainTemp  <- dt.between.peak.max[data2exclude]
          dt2retain <- c(dt2retain, dtRetainTemp)
          adjust = adjust + temp
        }
      }else{
        dt2filter     <- dt2filter[-candidate]
        dtRetainTemp  <- dt.between.peak.max[candidate]
        dt2retain <- c(dt2retain, dtRetainTemp)
      }
    }
  }



  num2salvage <- sample (c(1:length(dt2filter)), ceiling(adjust))       #FIXME: Manully fixting
  if (length(num2salvage) > 0)
  {
    dt2filter <- dt2filter[-num2salvage]
    dtRetainTemp  <- dt.between.peak.max[num2salvage]
    dt2retain <- c(dt2retain, dtRetainTemp)
  }

  dt.retained <- c(dt.right.of.peak.max, dt2filter)
  returnList <- list(dtFiltered = dt2retain, dtRetained = dt.retained)
  return (returnList)

}


tryDensity <- function(dt) {
    out <- tryCatch(
        {
                den <- density(dt, warn=FALSE)
        },
        error=function(cond) {
             return(NA)
        },
        warning=function(cond) {
               return(NULL)
        },
        finally={
     #       message(paste("Somthing is wrong with the ", dt))
         }
    )
    return(out)
}
```