Supplementary Appendix 1

####R code to fix the baseline of a TDR######

Length of the depth data minus 30 data points. Dives were never over 120 seconds, so a conservative #cutoff of 180 seconds (6*30) was used to assess #baseline deviations length of the depth data minus 30 #data points

```
steps<-1:(length(data$depth)-30)

#A function to calculate mode

mode <- function(j) {

  unis <- unique(j)

  unis[which.max(tabulate(match(j, unis)))]

}

#code to iterate through depth recordings and return them to the baseline.

for(i in (steps)){

#Find data where at least two thirds of the 30 recordings are below 0 and add the most frequently #occurring number below zero (the current baseline) to all subsequent values. On all occasions of #testing this lead to a correct baseline change.

  if (captureperiod$data[i] < 0 && length(which(captureperiod$data[i:(i+30)] < 0)) > 20){

    captureperiod$depth[i:(length(steps)+30)]<-captureperiod$depth[i:(length(steps)+30)] -

      mode((captureperiod$depth[i:(i+30)])[captureperiod$depth[(i:(i+30))]< 0])

  }

#detect deviations above 0 and add the minimum recorded value, which is the shifted #downwards baseline.

  else if (captureperiod$depth[i] > 0 && min(captureperiod$depth[i:(i+30)]) > 0){

    captureperiod$depth[i:(length(steps)+30)]<-captureperiod$depth[i:(length(steps)+30)]
-min(captureperiod$depth[i:(i+30)])

  }
```

```
  else {NULL}

}

#Effectiveness of the code to fix the depth baseline may vary by device and species. For cases
of more #frequent and sporadic baseline changes this code may be inappropriate. Always test
the results against #the raw data to ensure validity of the method.


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~#


#####function to summarise dives#####

#dataday must be in POsiXt format; datadepth should be numeric and represent depth at a
given time; #datastatus must be either "Dive" or "Surface"; rate should be the time in second
between records; #temp is the recorded temperature at each dive and can be removed if
unavailable by removing it from #divedata and the function arguments


divefunction <- function(dataday, datadepth, datastatus, rate, temp){


  #sets up a data frame to receive dive information

  divedata <- data.frame(matrix(ncol = 7, nrow = 0))

  colnames(divedata)<- c("divelength", "divemax", "divemaxtime", "enddivetime", "temp","desc",
"asc")

  divedata$divelength <- as.numeric(divedata$divelength)

  divedata$divemax <- as.numeric(divedata$divemax)

  divedata$divemaxtime <- as.numeric(divedata$divemaxtime)

  divedata$asc<-as.numeric(divedata$acs)

  divedata$desc<-as.numeric(divedata$desc)


#numbers and objects to be used in the loop
```

```
  numb <- 1
  divetemp <- c()

#loop to calculate dive characteristics
  for (i in 1:length(datadepth)){
    if (datastatus[i] == "Dive" && substr(dataday[i],0,10) == substr(dataday[i+1],0,10)){
      divetemp <- c(divetemp, as.numeric(datadepth[i]))

      divedata[numb,] <- c((as.numeric(length(divetemp))*rate),
                           as.numeric(max(divetemp)),
                           as.numeric(length(divetemp[divetemp>max(divetemp)*0.75])*rate),
                           dataday[i+1], temp[i],
                           divetemp[min(which(divetemp>max(divetemp)*0.75))]/
                             (min(which(divetemp>max(divetemp)*0.75))*6),
                           divetemp[max(which(divetemp>max(divetemp)*0.75))]/
                             ((length(divetemp)-max(which(divetemp>max(divetemp)*0.75)))*6))
    }
    else{
      divetemp <- c()
      numb <- ifelse(is.na(divedata[numb,1]),
                     numb, numb+1)
    }
  }
  return(divedata)
}
```