Turkey Roost Up/Down Times

Henry Traynor

2024-10-25

```
# March 1-2 2023, ID 11197
require(readr)
```

```
## Loading required package: readr
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
df<- read_csv("../Data/11197_2.csv")
```

```
## Rows: 1440 Columns: 20
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (6): Date, eobs:acceleration-axes, eobs:accelerations-raw, sensor-type,...
## dbl  (9): event-id, Month, Day, Year, data-decoding-software, eobs:accelerat...
## lgl  (2): visible, import-marked-outlier
## time (3): timestamp, Time, eobs:start-timestamp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
#change name of columns
df<-df%>% rename(ID =`individual-local-identifier`, accel=`eobs:accelerations-raw`)

# Function to split 'accel' column into groups of three values (x, y, z axes)
split_accel_data <- function(accel_col) {
  accel_values <- unlist(strsplit(accel_col, " "))
  matrix(accel_values, ncol = 3, byrow = TRUE)
}

# Initialize an empty dataframe to store the expanded data
expanded_df <- data.frame()

# Iterate through each row in the dataframe
for (i in 1:nrow(df)) {
  # Split accel column into x, y, z values
  accel_matrix <- split_accel_data(df$accel[i])

  # Create a temporary dataframe for the current row, repeating the timestamp, date, year, and tag-loca
  temp_df <- data.frame(
    Time = rep(df$Time[i], nrow(accel_matrix)),
    Date = rep(df$Date[i], nrow(accel_matrix)),
    Year = rep(df$Year[i], nrow(accel_matrix)),
    ID = rep(df$ID[i], nrow(accel_matrix)),
    sample_num = 1:nrow(accel_matrix),  # Add a sample number for each sample
    x_axis = accel_matrix[, 1],
    y_axis = accel_matrix[, 2],
    z_axis = accel_matrix[, 3]
  )

  # Append the temporary dataframe to the expanded dataframe
  expanded_df <- rbind(expanded_df, temp_df)
}
expanded_df$x_axis=as.numeric(expanded_df$x_axis)
expanded_df$y_axis=as.numeric(expanded_df$y_axis)
expanded_df$z_axis=as.numeric(expanded_df$z_axis)
```

```r
#Rename expanded_df
df <- expanded_df
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 4.2.3
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(dplyr)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.2.3

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
#Calculate rolling mean of acc for DBA calculation
window_length=40
df$x_mean=append(rollmean(expanded_df$x_axis, window_length, align="left"), replicate(window_length-1, l
df$y_mean=append(rollmean(expanded_df$y_axis, window_length, align="left"), replicate(window_length-1, l
df$z_mean=append(rollmean(expanded_df$z_axis, window_length, align="left"), replicate(window_length-1, l
```

```r
#Calculate dynamic body acceleration (DBA) by row for each x,y,and z measurement
df$ax=df$x_axis-df$x_mean
df$ay=df$y_axis-df$y_mean
df$az=df$z_axis-df$z_mean

#Add up all the absolute values of DBA to get OBDA for each row
df$odba=(abs(df$ax)+abs(df$ay)+abs(df$az))

# Calculate VeDBA by row
df$VeDBA=sqrt(df$ax^2+df$ay^2+df$az^2)

#Calculate log of average VeDBA for each burst
calculate_rolling_average <- function(column, window_size = 40) {
  sapply(seq_along(column), function(i) {
    start <- floor((i - 1) / window_size) * window_size + 1
    end <- min(start + window_size - 1, length(column))
    mean(column[start:end])
  })
}

df=df%>%
  mutate(AVG_VeDBA=calculate_rolling_average(VeDBA))

df$log_avg_VeDBA=log(df$AVG_VeDBA)

#Concatenate Times
df$DateTime = paste(df$Date, df$Time)
df$dt = mdy_hms(df$DateTime, tz="GMT")
df$dt = df$dt - hours(5)

#Calculate rolling median of the log VeDBA for 10 minute window
calculate_rolling_median <- function(column, window_size = 400) {
  sapply(seq_along(column), function(i) {
    start <- floor((i - 1) / window_size) * window_size + 1
    end <- min(start + window_size - 1, length(column))
    median(column[start:end])
  })
}

df=df%>%
  mutate(median_VeDBA=calculate_rolling_median(log_avg_VeDBA))

#plot(df$dt, df$median_VeDBA)

#Calculate rolling median of the avg VeDBA for 6 minute window
df=df%>%
  mutate(median_VeDBA=calculate_rolling_median(AVG_VeDBA))
```

```r
#Pull out raw values and average over each recorded burst
df.raw <- data.frame(
  ID = df$ID,
  x_axis = df$x_axis,
  y_axis = df$y_axis,
```

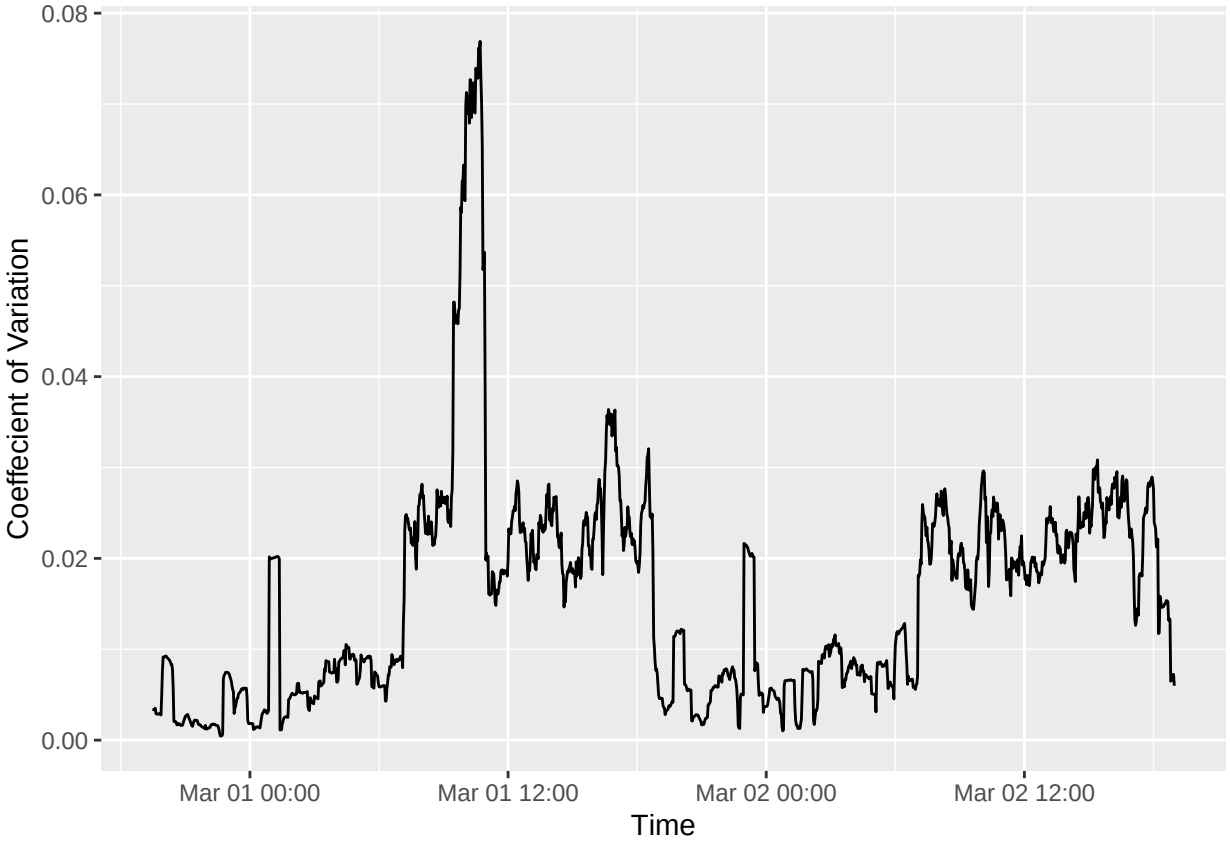
```
  z_axis = df$z_axis,
  dt = df$dt
)

condense_burst <- function(column, window_size = 40) {
  sapply(seq_along(column), function(i) {
    start <- floor((i - 1) / window_size) * window_size + 1
    end <- min(start + window_size - 1, length(column))
    mean(column[start:end])
  })
}

df.raw=df.raw%>%
  mutate(x_avg=condense_burst(x_axis))
df.raw=df.raw%>%
  mutate(y_avg=condense_burst(y_axis))
df.raw=df.raw%>%
  mutate(z_avg=condense_burst(z_axis))

df.raw <- df.raw %>% distinct(dt, .keep_all=TRUE)
df.avg <- df.raw %>%
  select("ID", "dt", "x_avg", "y_avg", "z_avg")
```

```
library(zoo)
library(moments)
intervalAnalysis <- function(df.sample, fun.call, dataCol, windowLength, windowStep) {
  colNum = which(colnames(df.sample)==dataCol)
  #calculation of statistic values
  stat <- rollapply(df.sample[,colNum],
                    FUN=fun.call,
                    width=windowLength,
                    by = windowStep,
                    by.column = TRUE,
                    align = "right")

  timestamp <- seq(from=df.sample$dt[windowLength], to=df.sample$dt[nrow(df.sample)], length.out=length
  #combining stat values with time indices
  df.stat <- data.frame(timestamp,stat)

  return(df.stat)
}

library(goeveg)
```

```
## Warning: package 'goeveg' was built under R version 4.2.3
```

```
## This is GoeVeg 0.7.2 - build: 2024-02-06
```

```
df.stat <- intervalAnalysis(df.avg, "cv", "z_avg", 15, 1)

ggplot(data=df.stat, aes(x=timestamp, y=stat)) + geom_line() + ylab("Coeffecient of Variation") + xlab(
```


0.08
0.06
0.04
0.02
0.00
Coeffecient of Variation
Mar 01 00:00
Mar 01 12:00
Mar 02 00:00
Mar 02 12:00
Time