```
# I can't upload a tar archive or even a text file to pp
# Hence just copy and paste code from this pdf into a text file:
# each following section specifies a file name
# so look further down for the shell and the R-script

# First install R from https://www.r-project.org/, there are
# also windows- and mac versions

# How to create the expected directory structure:
# root dir can be anywhere, /tmp is just an example.

mkdir -p /tmp/audio/bodycams
cd /tmp/audio/bodycams
ln -s ~/some-data-dir/1382_202407131806_Unit5-0.mp4 .

# extract shots from mp4:
cd ..
sh extract-shots.sh

# run shot1corr.R
cd ..
# put the file shot1corr.R in the current directory (= /tmp)

#run R
R
# at command prompt (>)
source('shot1corr.R')

# but expect some trivial yet acidic problems on windows ...
# and perhaps also the appearance of some strange characters
# when pasting into a text file. The pdf-format sometimes makes
# use of special character codes just for making fun of the unexperienced.
--------------------------------
```

```sh
#!/bin/sh
# File: extract-shots.sh
# use this shell script to extract audio streams from police cruiser dashcams
# use aac format because original format is aac in video source
# adapted from original by @https://tribe.peakprosperity.com/u/brian60221
# don't change ".stream-[123]." if you're using these data with shot1corr.R
# or change the R-script accordingly

src='1382_202407131806_Unit5-0.mp4'

ffmpeg -i "bodycams/${src}" -vn -map 0:a:0 -ss 00:04:55 -t 00:00:10 "${src}.stream-1.aac"
ffmpeg -i "bodycams/${src}" -vn -map 0:a:1 -ss 00:04:55 -t 00:00:10 "${src}.stream-2.aac"
ffmpeg -i "bodycams/${src}" -vn -map 0:a:2 -ss 00:04:55 -t 00:00:10 "${src}.stream-3.aac"

--------------------------------
```

```r
# File: shot1corr.R
# Author https://tribe.peakprosperity.com/u/pk2019

if (! require(av)) {
      install.packages('av')
      library(av)
}
if (! require(tibble) ) {
      install.packages('tibble')
      library(tibble)
}
```

```r
if (! require(Cairo) ) {
      install.packages('Cairo')
      library(Cairo)
}


if (! require(lattice) ) {
      install.packages('lattice')
      library(lattice)
}

sq<-function(s) paste0("'",s,"'")


# read audio data from fn and take a sound sample at t0..t0+dt
streamCreate <- function(fn,t0,dt) {
  base::message(paste('reading',sq(fn),'...'))
  dta  <- av::read_audio_bin(fn)
  ddf  <- tibble::tibble(t     = seq_along(dta)/attr(dta,'sample_rate'),
                         type  = 'data',  # = data
                         data  =  dta)

  idx  <- with(ddf, t >= t0 & t < t0+dt)
  sdf  <- ddf[idx,]
  sdf  <- within(sdf, {type <- 'sample'}) # s = sample


  self <- list(ddf = ddf, sdf = sdf, cdf = NULL)

  attr(self,'file')   <-  fn
  class(self)         <- 'stream' # turn the list 'self' into an object of class 'stream'
  self
}

# compute sample correlation using a shifting data window
# this is very slow, could at least make use the R package parallely
streamCorr <- function(self) {
  stopifnot(inherits(self,'stream'))
  fn <- attr(self,'file')
  base::message(paste('correlating',sq(fn),'...'))

  # correlate data column of df with sample s at offset i..i+len(s)-1
  # and return the Pearson's r or NA if i is out of range
  sampleCorr <- function(df,s,i) {
     idx <- seq(i,i+NROW(s)-1)
     if ( tail(idx,1) > NROW(df) ) {
          return(NA)
     }
     stats::cor(s$data,df$data[idx])
  }


  idx <- seq_along(self$ddf$data)
  res <- numeric(max(idx))
  for ( i in  idx ) {
       res[i] <- with(self, sampleCorr(ddf,sdf,i))
  }
  cdf      <-  self$ddf
  cdf$type <- 'correlation' # c == correlation
  cdf$data <-  res
  self$cdf <-  cdf
  self
}



streamPlot <- function(self) {
  stopifnot(inherits(self,'stream'))
  fn   <- attr(self,'file')
  stem <- base::basename(fn)
  fpdf <- paste0(stem,'.corr-shot-1.pdf')

  grDevices::graphics.off()
  Cairo::CairoPDF(file=fpdf,author='pk2019',title=fpdf)

  base::message(paste('plotting to file',sq(fpdf),'...'))


  # construct sample diagram
  xat <- round(seq(min(self$sdf$t),max(self$sdf$t),0.05),2)
  p   <- lattice::xyplot( data ~ t|type,
                          data = self$sdf,
                        subset = type == 'sample',
                        xlab   = 't [s]',
                        ylab   = NULL,
                        sub    = list(paste('Shot 1 sample from\n', paste(fn,sep="'")),cex=0.5),
                        type   = 'p',
                        cex    = 0.001,
                        strip  = FALSE,
                    strip.left = TRUE,
                    layout     = c(1,1),
                        scales = list (y = list(rot=0), x=list(at=xat)),
                           xat = xat, # needed to effectively pass down xat to the panel function below
                         panel = function(x,y,...,xat) {
                                    panel.abline(v = xat, col='lightgray',alpha=0.4)
                                    panel.abline(h = 0,   col='lightgray',alpha=0.4)
                                    panel.xyplot(x,y,...,alpha=0.9)
                    })
  base::plot(p)

  # construct data and correlation diagram
  xat <- as.integer(seq(min(self$ddf$t),max(self$ddf$t),1))
  p   <- lattice::xyplot( data ~ t | type,
                          data = rbind(self$ddf,self$cdf),
                          sub  = list(paste('Data & Shot1 Correlation from\n',paste(fn,sep="'")),cex=0.5),
                          xlab ='t [s]',
                          ylab = NULL,
                          type = c('p'),
                           cex = 0.001,
                         strip = FALSE,
                    strip.left = TRUE,
                        layout = c(1,2),
                        scales = list (y = list( relation = 'free', rot=0),
                                       x = list( at = xat)),
                           xat = xat, # needed to pass xat down to panel-function
                         panel = function(x,y,...,xat) {
                                    panel.abline(v=xat,col='lightgray',alpha=0.4)
                                    panel.abline(h=0,col='lightgray',alpha=0.4)
                                    if ( panel.number() == 1) {
                                         hpos <- c(0.25,0.5,0.75,1)
                                         panel.abline(h=c(-hpos,hpos),col='lightgray',alpha=0.4)
                                    }
                                    panel.xyplot(x,y,...,alpha=0.9)
                    })
 base::plot(p)
 grDevices::graphics.off()
 NULL
}


# processing starts here

base::warning('@offtheback: the parameters ps, t0 and dt are worth to be checked thrice; perhaps individual star
ps   <- 0.0045 # 9 ms phase shift between stream 1 and 3
t0   <- 2.024  # first shot begins at 2.024 s within stream 2 of 1..3 (= track 1 of 0..2)
dt   <- 0.15   # first shot response length in seconds (no snick)

# you may want to change stem to match .aac file locations
stem <- 'audio/1382_202407131806_Unit5-0.mp4'

s1   <- streamCorr(streamCreate(paste0(stem,'.stream-1.aac'), -ps + t0, dt))
s2   <- streamCorr(streamCreate(paste0(stem,'.stream-2.aac'),       t0, dt))
s3   <- streamCorr(streamCreate(paste0(stem,'.stream-3.aac'),  ps + t0, dt))



#produce pdfs seperately (even a single stream results in very large output)
streamPlot(s1)
streamPlot(s2)
streamPlot(s3)
```
