Blink Detetcion

Code

First load gazeR and some sample data. I am using data from my gazeR package.

Hide

```
library(gazer)

library(cowplot)

library(data.table)

library(patchwork)

library(tidyverse)
```

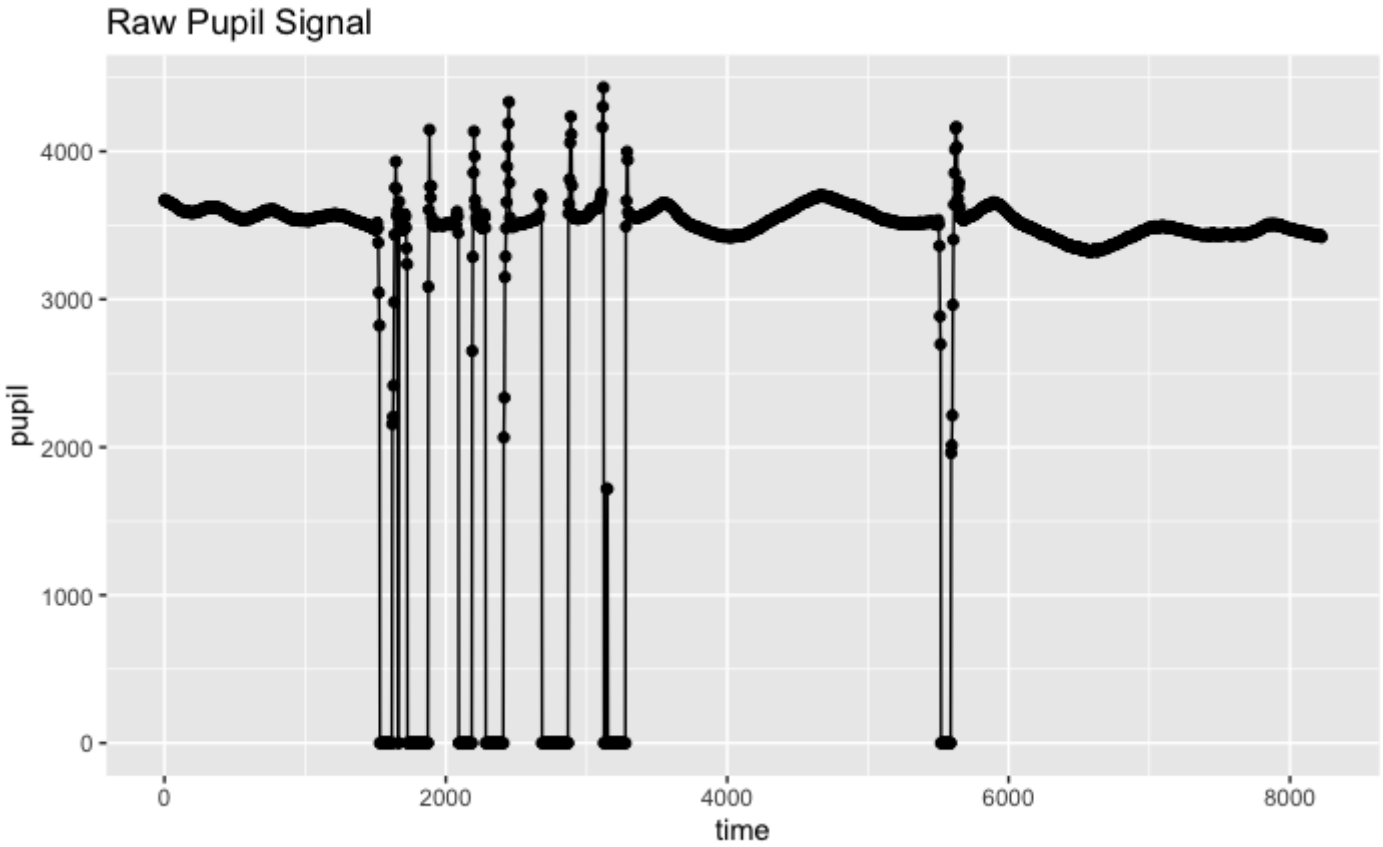
Hide

```
pupil_path <- system.file("extdata", "pupil_sample_files_edf.xls", package = "gazer")

pupil_files1<-fread(pupil_path)

pupil_files1 <- as_tibble(pupil_files1)

summary(pupil_files1)
```

Let's use one trial as an example. It is clear that there are multiple blinks. How well does the blink algo detect them?

Hide

```
interp_graph <- pupil_files1  %>%

  dplyr::mutate(pupil=ifelse(is.na(pupil), 0, pupil)) %>% #turn pupil values from NA to
 0

  dplyr::filter(subject=="11c.edf", trial=="27")



pup_g<- ggplot(interp_graph, aes(x= time, y= pupil)) + geom_point()+ geom_line(colour="b
lack") + ggtitle("Raw Pupil Signal")



  print(pup_g)
```



This is the R script for Ronen's algo.

Hide

```
#load algo

based_noise_blinks_detection <- function(pupil_data, sampling_rate_in_hz){



  library(forecast)

  library(ggplot2)

  library(pracma)



  sampling_interval     <- round(1000/sampling_rate_in_hz) #compute the sampling time in
terval in milliseconds.

  gap_interval          <- 100 # set the interval between two sets that appear consecuti
vely for concatenation.



   blinks_data <- pupil_data==0

  blinks <- c(-1*which(diff(blinks_data) %in% 1), which(diff(blinks_data) %in% -1)+1)



  # Case 1: there are no blinks



  if (length(blinks)==0)

    return(blinks);



  # Sort the blinks by absolute value. in this way we are getting an array of blinks whe
n the offset appears after the onset

  blinks <- blinks[order(abs(blinks))]



  # Edge cases

  # Case 2: the data starts with a blink. In this case, blink onset will be defined as t
he first missing value.
```

```
  if (length(blinks)>0 && blinks[1]>0 && pupil_data[1]==0)

    blinks = c(-1, blinks);



  # Case 3: the data ends with a blink. In this case, blink offset will be defined as th
e last missing sample

  if(length(blinks)>0 && tail(blinks, 1)<0 && tail(pupil_data, 1)==0)

    blinks = c(blinks, nrow(pupil_data))



  # Smoothing the data in order to increase the difference between the measurement noise
and the eyelid signal.

  ms_4_smooting  <- 10                                      # using a gap of 10 ms for t
he smoothing

  samples2smooth <- ceiling(ms_4_smooting/sampling_interval) # amount of samples to smoo
th

  smooth_data    <- ma(pupil_data, samples2smooth)

  smooth_data[1, 1] <- pupil_data[1, 1]

  smooth_data[2, 1] <- pupil_data[2, 1]



  smooth_data[smooth_data==0] <- NaN;  # replace zeros with NaN values

  diff_smooth_data            = diff(smooth_data);



  # Finding the blinks' onset and offset

  blink                <- 1;                          # initialize blink index for it
eration

  blinks_data_positions <- matrix(0, length(blinks), 1)  # initialize the array of blink
s

  prev_offset          <- -1                          # initialize the previous blink
offset (in order to detect consecutive sets)



  while (blink < length(blinks)){
```

```
    onset_candidate <- blinks[blink]

    blink           <- blink + 1  # increase the value for the offset



    # set the offset candidate

    offset_candidate <- blinks[blink]

    blink            <- blink + 1  # increase the value for the next blink



    # find blink onset

    data_before <- diff_smooth_data[2:abs(onset_candidate)] # returns all the data befor
e the candidate

    blink_onset <- tail(which(data_before>0), 1)            # returns the last 2 samples
before the decline



    # Case 2 (the data starts with a blink. In this case, blink onset will be defined as
the first missing value.)

    if (isempty(blink_onset==TRUE))

      ifelse(onset_candidate == blinks[1], blink_onset <- 0, blink_onset <- -abs(onset_c
andidate))



    # correct the onset if we are not in case 2

    if (onset_candidate>0 || pupil_data[onset_candidate+2, 1]>0)

      blink_onset      = blink_onset+2



    # find blink offset

    data_after   <- diff_smooth_data[abs(offset_candidate):length(diff_smooth_data)] # r
eturns all data after the candidate

    blink_offset  <- abs(offset_candidate)+head(which(data_after<0), 1)
# returns the last sample before the pupil increase
```

```
    # Case 3 (the data ends with a blink. In this case, blink offset will be defined as
 the last missing sample.)

    if (length(blink_offset)==0)

      blink_offset <- nrow(pupil_data)+1



    # Set the onset to be equal to the previous offset in case where several sets of mis
sing values are presented consecutively

    if (sampling_interval*blink_onset > gap_interval && sampling_interval*blink_onset-sa
mpling_interval*prev_offset<=gap_interval)

      blink_onset <- prev_offset



    prev_offset <- blink_offset-1

    # insert the onset into the result array

    blinks_data_positions[blink-2] <- -sampling_interval*blink_onset

    # insert the offset into the result array

    blinks_data_positions[blink-1] <- sampling_interval*(blink_offset-1)

  }



  duplicated_values     <- blinks_data_positions[duplicated(blinks_data_positions)]



  #blinks_data_positions <- blinks_data_positions[!blinks_data_positions %in% duplicated
_values];



  res <- blinks_data_positions

  id = 1;

  while(id<length(res)-2)

  {

    if(res[id]>0 && res[id]==-res[id+1]){
```

```
      toremove <- matrix(TRUE, length(res), 1);

      toremove[id] <- FALSE

      toremove[id+1] <- FALSE

      res <- res[toremove]

    }else{

      id = id+1

    }

  }

  return(abs(res))

}
```

The algo seems to do a good job at detecting them, but there seems to be some spurious values still kept in the data.

Hide

```
# It will take much longer when passed as one large dataset. Alternatively, recode the a
lgorithm to work with vectors.

df_blinks<-interp_graph %>%

  # pass into a list

  summarise(Blink_Index = list(based_noise_blinks_detection(as.matrix(interp_graph$pupi
l), 250))) %>%

  unnest(c(Blink_Index))



# Label indices onset/offset - should always end in 'offset'

df_blinks$Label <- rep(c("Onset", "Offset"), length.out=nrow(df_blinks))



df_blinks
```

Blink_Index	Label
<dbl>	<chr>

Blink_Index	Label
<dbl>	<chr>
1516	Onset
1892	Offset
2084	Onset
2448	Offset
2676	Onset
2892	Offset
3120	Onset
3292	Offset
5500	Onset
5632	Offset

1-10 of 10 rows

Hide

```
#list of onsets/offsets for each trial
```

Hide

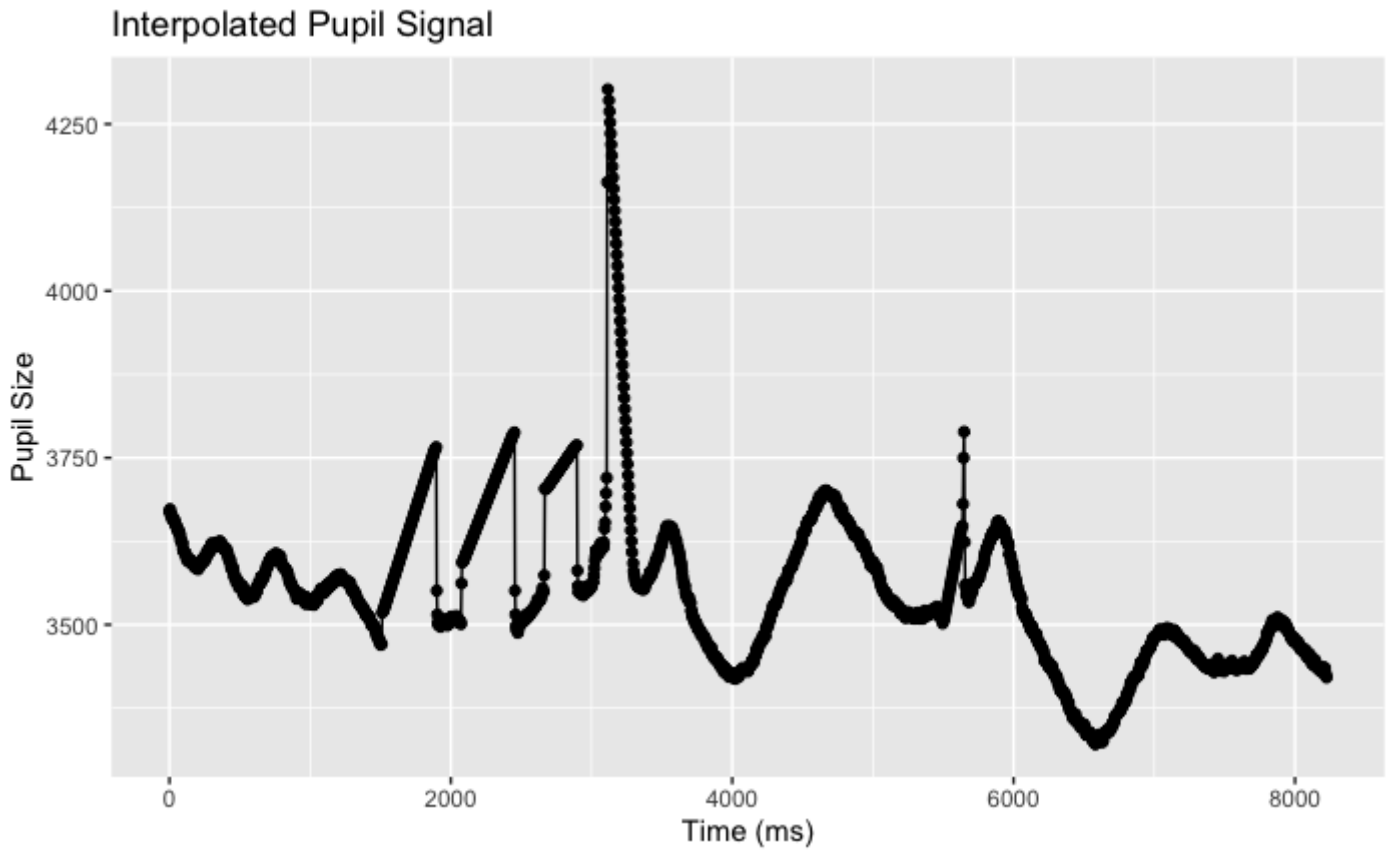
```
# Code to map blink onset/offset index to original dataframe

# Solution found here: https://stackoverflow.com/a/53105626/2653210

i1 <- match(interp_graph$time, df_blinks$Blink_Index)

interp_graph[names(df_blinks)[1:2]] <- lapply(df_blinks[1:2], `[`, i1)



# once merged onsets must be duplicated across until offset in order to mark as blinks

blinks <- interp_graph %>%

        dplyr::group_by(grp = cumsum(!is.na(Label))) %>%

        dplyr::mutate(Label = replace(Label, first(Label) == 'Onset', 'Onset')) %>%

        dplyr::mutate(Blinks_New=ifelse(Label=="Onset" | Label=="Offset", 1, 0)) %>% #tu
rn Onset and Offset 1

        mutate(Blinks_New=ifelse(is.na(Blinks_New), 0, Blinks_New)) %>% # turn rest of v
alues 0

        dplyr::ungroup()



Blink_Pupil<- blinks %>%

        group_by(subject, trial) %>%

  mutate(blinks_pupil=ifelse(Blinks_New==1, pupil==NA, pupil)) %>%

  mutate(blinks_pupil=ifelse(blinks_pupil==0, NA, blinks_pupil))%>%

  #mutate(extendpupil=extend_blinks(blinks_pupil, fillback=10, fillforward=10, hz=250))
 %>%

  dplyr::mutate(interp = zoo::na.approx(blinks_pupil, na.rm = FALSE, rule=2))
```

This is what it looks like after NAs are interpolated. It seems to do a poor job at capturing the offset.

Any Thoughts?

Hide

```
pup_g3 <- ggplot(Blink_Pupil, aes(x= time, y= interp)) +

  geom_point()+ geom_line(colour="black") +

  xlab("Time (ms)") +

  ylab("Pupil Size") +

  ggtitle("Interpolated Pupil Signal")

print(pup_g3)
```



Hide

```
NA

NA
```