Eye Tracking Preprocessing - Fixations

Free viewing faces

Klara Gregorova & Benjamin Gagl

05.11.2019

List pathways to .asc files and event files

```r
# set path to file structure
path = "./freeviewfaces_raw_data_structure"

# list all subject directories
subjects = list.dirs(path, recursive = T)
subjects = subjects[grepl(pattern="eyetrack", x = subjects)]

# get full file names for .asc files and *events.tsv files
input_file_pathways = list.files(subjects, pattern = "*.asc$",full.names = T)
input_event_pathways = list.files(subjects, pattern = "*events.tsv$",full.names = T)
```

EFIX data from all participants to a dataframe, define AOIs

```r
dat_efix_all = data.frame()

# loop over the input pahtways --> over all .asc files which should be preprocessed
for (nr in 1:length(input_file_pathways)){
  tmp_file = readLines(input_file_pathways[nr]) # read the .asc file
  # separate the file name
  file = strsplit(input_file_pathways[nr], split = "eyetrack/")[[1]][2]
  # extract the code for the experimental session
  code = strsplit(file, split = "_task-freeviewfaces_eyetrack")[[1]][1]
  # separate the information about the session from the code of the subject
  sub = strsplit(code, split = "_") [[1]][1] # extract the individual subject code
  session = strsplit(code, split = "_") [[1]][2] # extract the number of the session

  # grep all rows including EFIX (automatic fixation detection by Eyelink)
  # --> for evaluation of the algorithm see Friedman et al. (2018)
  # and unlist them to an individual dataframe
  dat_fix = strsplit(tmp_file[grepl("EFIX",tmp_file)],split = "\t")
  dat_efix = data.frame(matrix(unlist(dat_fix), nrow = length(dat_fix), byrow=T))
  colnames(dat_efix)= c("X1", "endtime", "duration", "x", "y", "pupilsize")
  dat_efix$X1 = as.character(dat_efix$X1)
  # add vectors indicating the subject code and the session
  dat_efix$sub = sub
  dat_efix$session = session

  # extract eye (R/L) and starttime
  for (i in 1:length(dat_efix$sub)){
    # information about the tracked eye
  dat_efix$eye[i] = strsplit(dat_efix$X1[i], split = " ")[[1]][2]
  # start time of the fixation
  dat_efix$starttime[i] = strsplit(dat_efix$X1[i], split = " ")[[1]][5]
  }

  # set the data types
  dat_efix$starttime=as.character(dat_efix$starttime)
  dat_efix$duration=as.character(dat_efix$duration)
  dat_efix$x=as.character(dat_efix$x)
  dat_efix$y=as.character(dat_efix$y)
  dat_efix$starttime=as.numeric(dat_efix$starttime)
  dat_efix$duration=as.numeric(dat_efix$duration)
  dat_efix$x=as.numeric(dat_efix$x)
  dat_efix$y=as.numeric(dat_efix$y)

# read the dataset with individual event characteristics
dat_events=read.table(input_event_pathways[nr], header=TRUE)

# get exact time when each fixation starts and ends:
# substraction of the raw time of the first message in the experimental task
# from the raw time of the message for each fixation
# --> we get time specification relative to the begin of the experimental task
dat_efix$fix_time_start = (dat_efix$starttime - dat_events$time_raw[1])/1000
# add the fixation duration to the fixation start time to get the fixation end time
dat_efix$fix_time_end = dat_efix$fix_time_start + (dat_efix$duration/1000)

# assign fixations reaching to the time window of an experimental trial
# the start time of the fixation should be greater or equal to the start time
# of the trial and lower than the end time of the trial
# OR
# the end time of the fixation should be greater than the start point
# of the trial and lower than the end time of the trial
for (i in c(1:128)){
  dat_efix$trial[
    dat_efix$fix_time_start >= dat_events$start_time[i] &
      dat_efix$fix_time_start < (dat_events$start_time[i]+dat_events$duration[i])|
      dat_efix$fix_time_end >= dat_events$start_time[i] &
      dat_efix$fix_time_end <
      (dat_events$start_time[i]+dat_events$duration[i])]=dat_events$trial[i]}


# assign the start and the end time point of experimental trials -
# for fixations reaching into the time window of the trial
for (i in c(1:128)){
  dat_efix$trial_starttime[dat_efix$trial==i] = dat_events$start_time[i]
  dat_efix$trial_endtime[dat_efix$trial==i] = (dat_events$start_time[i] + dat_events$duration[i])
}

# correction of fixation durations in case the fixation expands out of the
# time window of the experimental trial;
# before: overlap to the time window before the trial
# after: overlap to the time window after the trial
for(i in 1:length(dat_efix$fix_time_start)){
    if (!is.na(dat_efix$trial_starttime[i]) & dat_efix$fix_time_start[i]
        < dat_efix$trial_starttime[i]){
          dat_efix$dur_cor[i] =
            dat_efix$duration[i] -
            ((dat_efix$trial_starttime[i]- dat_efix$fix_time_start[i])*1000)
          dat_efix$type_fixation[i] = "before"
        }
    else if (!is.na(dat_efix$trial_endtime[i]) & dat_efix$fix_time_end[i]
        > dat_efix$trial_endtime[i]){
          dat_efix$dur_cor[i] =
              dat_efix$duration[i] -
              ((dat_efix$fix_time_end[i] - dat_efix$trial_endtime[i])*1000)
          dat_efix$type_fixation[i] = "after"
        }
    else{
      dat_efix$dur_cor[i]=dat_efix$duration[i]
      dat_efix$type_fixation[i]=NA}
  }

  dat_efix$type_fixation[is.na(dat_efix$type_fixation)
                         & !is.na(dat_efix$trial)]="complete"

  # assign which picture position is fixated
  # --> interval of x AND y coordinates has to be met
  dat_efix$stim_pos=NA
  dat_efix$stim_pos[410<=dat_efix$x & dat_efix$x<=610 & 95<=dat_efix$y & dat_efix$y<=295] = 1
  dat_efix$stim_pos[630<=dat_efix$x & dat_efix$x<=830 & 95<=dat_efix$y & dat_efix$y<=295] = 2
  dat_efix$stim_pos[850<=dat_efix$x & dat_efix$x<=1050 & 95<=dat_efix$y & dat_efix$y<=295] = 3
  dat_efix$stim_pos[1070<=dat_efix$x & dat_efix$x<=1270 & 95<=dat_efix$y & dat_efix$y<=295] = 4
  dat_efix$stim_pos[410<=dat_efix$x & dat_efix$x<=610 & 315<=dat_efix$y & dat_efix$y<=515] = 5
  dat_efix$stim_pos[630<=dat_efix$x & dat_efix$x<=830 & 315<=dat_efix$y & dat_efix$y<=515] = 6
  dat_efix$stim_pos[850<=dat_efix$x & dat_efix$x<=1050 & 315<=dat_efix$y & dat_efix$y<=515] = 7
  dat_efix$stim_pos[1070<=dat_efix$x & dat_efix$x<=1270 & 315<=dat_efix$y & dat_efix$y<=515] = 8
  dat_efix$stim_pos[410<=dat_efix$x & dat_efix$x<=610 & 535<=dat_efix$y & dat_efix$y<=735] = 9
  dat_efix$stim_pos[630<=dat_efix$x & dat_efix$x<=830 & 535<=dat_efix$y & dat_efix$y<=735] = 10
  dat_efix$stim_pos[850<=dat_efix$x & dat_efix$x<=1050 & 535<=dat_efix$y & dat_efix$y<=735] = 11
  dat_efix$stim_pos[1070<=dat_efix$x & dat_efix$x<=1270 & 535<=dat_efix$y & dat_efix$y<=735] = 12
  dat_efix$stim_pos[410<=dat_efix$x & dat_efix$x<=610 & 755<=dat_efix$y & dat_efix$y<=955] = 13
  dat_efix$stim_pos[630<=dat_efix$x & dat_efix$x<=830 & 755<=dat_efix$y & dat_efix$y<=955] = 14
  dat_efix$stim_pos[850<=dat_efix$x & dat_efix$x<=1050 & 755<=dat_efix$y & dat_efix$y<=955] = 15
  dat_efix$stim_pos[1070<=dat_efix$x & dat_efix$x<=1270 & 755<=dat_efix$y & dat_efix$y<=955] = 16

  # add column with simulus characteristics out of the event file (for all 16 images)
  dat_events$faces_stim=as.character(dat_events$faces_stim)
  for (i in c(1:128)){
    dat_efix$faces_stim[dat_efix$trial == i] = dat_events$faces_stim[dat_events$trial == i]}

# bind the individual dataset to the dataframe with data from all participants
dat_efix_all = rbind(dat_efix_all, dat_efix)

}
```

Assign characteristics of fixated images

```
# characteristics of fixated positions (pictures)
  # loop over all rows of the full dataset (fixations from all participants)
  # goal: information about the position, model, sex of the model and expressed emotion

dat_efix_all$pic = NA
dat_efix_all$model=NA
dat_efix_all$sex=NA
dat_efix_all$emotion=NA

lines = which(!is.na(dat_efix_all$faces_stim))

for (i in lines){
    # separate the string with the full information about the presented matrix
    dat_efix_all$pic[i] = strsplit(dat_efix_all$faces_stim[i],
                                   split = "__")[[1]][dat_efix_all$stim_pos[i]]
    m = strsplit(dat_efix_all$pic[i], split = "_")[[1]][2]
    dat_efix_all$model[i] = substr(m, start=1, stop=nchar(m)-2)
    dat_efix_all$sex[i] = substr(m, start=nchar(m)-1, stop=nchar(m)-1)
    dat_efix_all$emotion[i] = substr(m, start=nchar(m), stop=nchar(m))
    }
```

Clean the dataframe for usable data only

```
# exclude all fixations which are out of a trial
dat_fix_cleaned = dat_efix_all[!is.na(dat_efix$trial),]

# delete the raw start and end time (we use time relative to the first message)
dat_fix_cleaned$endtime = NULL
dat_fix_cleaned$starttime = NULL
# delete the matrix information which was separated to the single columns
# model, sex and emotion
dat_fix_cleaned$faces_stim = NULL
# delete the first column including information about
# the tracked eye and start time of the fixation
dat_fix_cleaned$X1 = NULL
# as we did not concerned relevant set up for valid pupilsize measures,
# the pupilsize turns not to be usable
dat_fix_cleaned$pupilsize = NULL
# set data types
dat_fix_cleaned$session = as.factor(dat_fix_cleaned$session)
dat_fix_cleaned$eye = as.factor(dat_fix_cleaned$eye)

summary(dat_fix_cleaned)
```

```
##     duration            x                y               sub
##  Min.   :   2.0   Min.   : 333.9   Min.   :  13.2   Length:48768
##  1st Qu.: 155.8   1st Qu.: 715.9   1st Qu.: 411.2   Class :character
##  Median : 203.0   Median : 842.8   Median : 478.0   Mode  :character
##  Mean   : 214.2   Mean   : 838.1   Mean   : 528.3
##  3rd Qu.: 255.0   3rd Qu.: 955.5   3rd Qu.: 660.5
##  Max.   :2184.0   Max.   :1753.2   Max.   :1844.2
##
##   session      eye       fix_time_start      fix_time_end         trial
##  acq-1:24384   L:24408   Min.   :  -0.027   Min.   :   0.231   Min.   :  1.00
##  acq-2:24384   R:24360   1st Qu.: 430.673   1st Qu.: 430.890   1st Qu.: 33.00
##                          Median : 829.473   Median : 829.755   Median : 66.00
##                          Mean   : 776.635   Mean   : 776.849   Mean   : 65.12
##                          3rd Qu.:1198.668   3rd Qu.:1198.947   3rd Qu.: 97.00
##                          Max.   :1516.969   Max.   :1517.023   Max.   :128.00
##
##  trial_starttime  trial_endtime        dur_cor       type_fixation
##  Min.   :   0.0   Min.   :   6.015   Min.   :   1.0   Length:48768
##  1st Qu.: 427.4   1st Qu.: 433.459   1st Qu.: 155.0   Class :character
##  Median : 829.5   Median : 835.515   Median : 200.0   Mode  :character
##  Mean   : 773.7   Mean   : 779.759   Mean   : 212.5
##  3rd Qu.:1193.5   3rd Qu.:1199.553   3rd Qu.: 254.0
##  Max.   :1511.0   Max.   :1517.005   Max.   :2184.0
##
##     stim_pos         pic              model              sex
##  Min.   : 1.000   Length:48768       Length:48768       Length:48768
##  1st Qu.: 5.000   Class :character   Class :character   Class :character
##  Median : 8.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 8.307
##  3rd Qu.:11.000
##  Max.   :16.000
##  NA's   :2664
##    emotion
##  Length:48768
##  Class :character
##  Mode  :character
##
##
##
##
```

Write dataframe with preprocessed data

```r
# define output path
outpath =  "./"

# write csv with cleaned data
write.csv(dat_fix_cleaned, paste0(outpath, "dat_simulated.csv"), row.names = FALSE, quote=F)
```

Literature

Friedman L, Rigas I, Abdulin E, Komogortsev OV (2018). A novel evaluation of two related and two independent algorithms for eye movement classification during reading. *Behav Res Methods 50*(4):1374-1397. Google Scholar