Eye Tracking Preprocessing - From .asc to standardized file structure

Free viewing faces

Benjamin Gagl & Klara Gregorova

11.09.2019

```r
# set path for input and output folders
# note, you need a folder freeviesfaces_raw_data where there are the raw .asc files
# saved and a folder freeviewfaces_raw_data_structure which is empty at the beginning

input_folder = "./freeviewfaces_raw_data/"
output_folder = "./freeviewfaces_raw_data_structure/"

# list all .asc files in the input folder with raw data
input_file_names = list.files(input_folder, pattern= "*.asc$")

# function for creation of output folders in case they do not exist.
cb_cd = function(path){
  if(dir.exists(path)){
    print(paste("Folder existis: ",path,sep=""))
  }else{
    dir.create(path, recursive = T)
    print(paste("Create: ",path,sep=""))
  }
}

# loop over all asc files in raw data
for (i in 1:length(input_file_names)){

  # extract the name of the looped [i] .asc file, the subject code (vp) and session (aq)
  id = strsplit(input_file_names[i],split=".asc")[[1]] [1]
  vp = strsplit(id,split="_")[[1]] [1]
  aq = strsplit(id,split="_")[[1]] [2]

  # paste the path for subfolder /eyetrack and create directory with cb_cd function
  indi_folder_path = paste(output_folder,"sub-",vp,"/eyetrack",sep="")
  cb_cd(indi_folder_path)

  # copy the looped [i] .asc file into the file structure.
  # paste the file names containing the subject code (vp) and the session (acq)
  file.copy(paste(input_folder,input_file_names[i],sep="")
          , paste(indi_folder_path,"/sub-",vp,"_acq-",aq,
                  "_task-freeviewfaces_eyetrack.asc",sep="")
          )
  # read the .asc file from the output folder
  tmp_file = readLines(paste(indi_folder_path,"/sub-",vp,"_acq-",aq,
                             "_task-freeviewfaces_eyetrack.asc",sep=""))

  # get cal info -
# grap all rows with message !CAL VALIDATION, excluded aborted validation
# element separator set to split = ""
# save the list
tmp_file_cal = strsplit(tmp_file[grepl("!CAL VALIDATION",tmp_file)&
                                   !grepl("ABORTED", tmp_file)],split = " ")

# loop over all calibration messages in tmp_file_cal from the [i] .asc file
for (cal_nr in 1:length(tmp_file_cal)){
  # for the first calibration: extract information about:
  if (cal_nr ==1){
    calibration = tmp_file_cal[[1]][4] # calibration type
    eye = tmp_file_cal[[cal_nr]][6] # right / left eye
    error_max = tmp_file_cal[[cal_nr]][12] # maximal calibration error
    error_avg = tmp_file_cal[[cal_nr]][10] # avaraged calibration error
    time_cal = as.numeric(strsplit(tmp_file_cal[[cal_nr]], split="\t")[[1]][2],
                          split=" ") # calibration time
  }
  # for calibration on the right eye except of the first one
  # add extracted information about calibration type, eye, max. and averaged error to vectors
  else if(tmp_file_cal[[cal_nr]][6]=="RIGHT"){
    calibration[cal_nr] = tmp_file_cal[[cal_nr]][4]
    eye[cal_nr] = tmp_file_cal[[cal_nr]][6]
    error_max[cal_nr] = tmp_file_cal[[cal_nr]][11]
    error_avg[cal_nr] = tmp_file_cal[[cal_nr]][9]
    time_cal[cal_nr] = as.numeric(strsplit(tmp_file_cal[[cal_nr]], split="\t")[[1]][2], split=" ")
  }

  # for calibrations on the left eye
  # add extracted information about calibration type, eye, max. and averaged error to vectors
  else{
    calibration[cal_nr] = tmp_file_cal[[cal_nr]][4]
    eye[cal_nr] = tmp_file_cal[[cal_nr]][6]
    error_max[cal_nr] = tmp_file_cal[[cal_nr]][12]
    error_avg[cal_nr] = tmp_file_cal[[cal_nr]][10]
    time_cal[cal_nr] = as.numeric(strsplit(tmp_file_cal[[cal_nr]], split="\t")[[1]][2], split=" ")
}
}

# create a dataframe out of the vectors with informations about the calibration
cal_df = data.frame(calibration, eye, error_max, error_avg, time_cal)

# cell type to numeric for max. and avg. calibration error
cal_df$error_max=as.numeric(as.character(cal_df$error_max))
cal_df$error_avg=as.numeric(as.character(cal_df$error_avg))

# extract the information about sampling frequency
sampl_freq = round(as.numeric(strsplit(tmp_file[grepl(
  "SAMPLES\tGAZE\tLEFT\tRIGHT\tRATE",tmp_file)],split = "\t")[[1]][6],0))

# get stim info for events file
# note, messages for trials were set only in the experimental blocks, not in the practicing part

# two subsets: (i) with start & and end of each trial (ii) with the start only
```

```r
tmp_file_sub_all = tmp_file[grepl("FACESTART|ENDPRESENTATION",tmp_file)]
tmp_file_sub = tmp_file_sub_all[grepl("FACESTART",tmp_file_sub_all)]

# loop over trials (messages for trial start)
for (ii in 1:length(tmp_file_sub)){

  # extract information from the starting message
  time = as.numeric(strsplit(strsplit(tmp_file_sub[ii], split="\t")[[1]][2],
                          split=" ")[[1]][1]) # time
  info_trial = as.numeric(strsplit(strsplit(tmp_file_sub[ii], split="\t")[[1]][4],
                          split=" ")[[1]][2]) # counter for the number of the trial
  # conter for the number of the trial within the block
  info_blocktrial = as.numeric(strsplit(
    strsplit(tmp_file_sub[ii], split="\t")[[1]][3], split=" ")[[1]][2])
  # info about the faces shown in the matrix
  info_faces = strsplit(tmp_file_sub[ii], split="\t")[[1]][6]
  # row index for on which position we can find the starting message
  # in the bigger subset including also the ending messages)
  index = match(tmp_file_sub[ii],tmp_file_sub_all)
  # extract the end time of the trial
  t_end = as.numeric(strsplit(
    strsplit(tmp_file_sub_all[index+1], split="\t")[[1]][2], split=" ")[[1]][1])


  # for the first trial, create the first element of a vector containing
  # information about the events:
  if (ii == 1){
    # time point of the start of the first trial (1st event message)
    exp_start_time = time
    start_time = time-exp_start_time # start time relative to the experimental start time
    time_raw = time # raw start time
    duration = t_end-time # fixation duration as a difference between the end and the start time
    trial = info_trial # number of the trial
    blocktrial = info_blocktrial # number of the trial within the block
    faces = info_faces # info about the presented faces

  }else{
    start_time[ii] = time-exp_start_time # start time relative to the experimental start time
    time_raw[ii] = time
    duration[ii] = t_end-time
    trial[ii] = info_trial
    blocktrial[ii] = info_blocktrial
    faces[ii] = info_faces
  }
}

#create a dataframe out of the event information
event_df = data.frame(start_time,duration,time_raw,trial, blocktrial, faces)
event_df$start_time = event_df$start_time/1000 # transform from ms to s
event_df$duration = event_df$duration/1000 # transform from ms to s

stim_info=read.csv("./stimuli_matrices.csv", header=TRUE)
      # import matrix with full information about the presented faces
# put the information about the faces into the event dataframe
event_df$faces_stim = stim_info$index
# delete the column with the incomplete information about the faces which
# were replaced by the information from the stim_info dataframe
event_df$faces=NULL

# write the event dataframe
write.table(event_df,row.names = F, file = paste(indi_folder_path,"/sub-",vp,"_acq-",aq,
                                                 "_task-freeviewfaces_events.tsv",sep=""))

# add exp time (relative to the starting point of the first trial) to the cal_df and export
cal_df$time_cal = (cal_df$time-exp_start_time)/1000
# write the dataframe with the calibration data
write.table(cal_df,row.names = F, file = paste(
  indi_folder_path,"/sub-",vp,"_acq-",aq,"_task-freeviewfaces_cal.tsv",sep=""))

# export .json file with general characteristics

# detect the calibration before the first experimental block (2 rows before the first
# calibration with positive experimental time - after the start of the experiment)
cal_exp = which(cal_df$time_cal > 0)[1] - 2

# vector with all calibration types within the experiment
for(nr in 1:nlevels(factor(cal_df$calibration[cal_exp:nrow(cal_df)]))){
  if(nr==1){
    cal_tmp=levels(factor(cal_df$calibration[cal_exp:nrow(cal_df)]))[nr]
    calibration=cal_tmp}
  else{cal_tmp=levels(factor(cal_df$calibration[cal_exp:nrow(cal_df)]))[nr]
  calibration=paste(calibration,cal_tmp, sep=", ")}}

# vector with tracked eye
for(nr in 1:nlevels(factor(cal_df$eye[cal_exp:nrow(cal_df)]))){
  if(nr==1){
    eye_tmp=levels(factor(cal_df$eye[cal_exp:nrow(cal_df)]))[nr]
    eye=eye_tmp}
  else{eye_tmp=levels(factor(cal_df$eye[cal_exp:nrow(cal_df)]))[nr]
  eye=paste(eye,eye_tmp, sep=", ")}}
# write .json file with all important general information
write(paste('{\n\t"SamplingFrequency": ',sampl_freq,',\n\t'
            ,'"StartMessage": "FACESTART",\n\t"EndMessage": "ENDPRESENTATION",\n\t'
            ,'"EventIdentifier": "EFIX",\n\t'
            ,'"Manifacturer": "SR-Research",\n\t"ManufacturersModelName": '
            ,'"EYELINK II CL v4.56 Aug 18 2010",\n\t'
            ,'"SoftwareVersions": "SREB2.2.61 WIN32 LID:5F0D424B '
            ,'Mod:2019.07.10 14:33 MESZ",\n\t'
            ,'"TaskDescription": "Free viewing of a 4x4 matrices including faces with'
            ,' positive, negative and neutral emotional expression",\n\t'
            ,'"Instructions":"Natural viewing of matrices, no special task",\n\t'
            ,'"InstitutionName": "Goethe-University of Frankfurt; '
            ,'Department of Psychology",\n\t'
            ,'"InstitutionAddress": "Theodor-W.-Adorno-Platz 6 60323 '
            ,'Frankfurt am Main; Germany",\n\t'
            ,'"Calibration type": "',calibration,'",\n\t'
```

```
               ,'"Recorded eye": "',eye,'",\n\t'
               ,'"Maximal calibration error (accross all calibrations excluding training)":'
               ,' ',max(cal_df$error_max[cal_exp:nrow(cal_df)]),',\n\t'
               ,'"Average calibration error '
               ,'(mean accross all calibrations excluding training)":'
               ,' ', mean(cal_df$error_avg[cal_exp:nrow(cal_df)]),"\n}",sep = "")
               , file = paste(indi_folder_path,"/sub-",vp,"_acq-",aq,
                              "_task-freeviewfaces_eyetrack.json",sep=""))

}
```

```
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-001/eyetrack"
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-001/eyetrack"
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-002/eyetrack"
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-002/eyetrack"
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-003/eyetrack"
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-003/eyetrack"
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-004/eyetrack"
## [1] "Folder existis: ./freeviewfaces_raw_data_structure/sub-004/eyetrack"
```