02 CSI online aphasia: Spoken - Preprocessing Preliminary Data for TeaP

Kirsten Stark

16 März, 2022

Load packages

```r
rm(list = ls())

# install.packages("remotes") # uncomment if installation is needed (only once)
# remotes::install_github("rstudio/renv") # uncomment if installation is needed (only once)
# if file is accessed through the R package, all packages should be installed if
# renv::restore()
# is applied. Otherwise use:
# install.packages("tidyr") # uncomment if installation is needed (only once)
# install.packages("dplyr") # uncomment if installation is needed (only once)
# install.packages("here") # uncomment if installation is needed (only once)
# install.packages("knitr")

library(tidyr)
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag

## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
options( "encoding" = "UTF-8" )

# set working directory -- @ Marcus: Am besten, du klickst in dem Ordner, den ich dir als .zip-file ges
# setwd("/Users/kirstenstark/Documents/Research/CSI_online_aphasia")
```

Load and preprocess data

This input file needs to be entered by hand:

```r
files <- list.files(here::here("data", "raw", "preliminary_for_TeaP"),
                    recursive = TRUE)
sosci_files <- files[grep("data.csv", files, fixed=T)]
vot_files <- files[-grep("data.csv", files, fixed=T)]

# input output main data
type <- c("preliminary_teap")
#input <- c("pilot_newdissmtukl_2021-10-13_10-44.csv")
#input <- c("data_dissmtukl_2021-08-12_22-27.csv") # can be a vector of several files
# possibly, the CSV file needs to be opened once and saved as CSV UTF-8 (durch Trennzeichen getrennte D
# then all relevant rows of column A need to be selected (e.g. A1:3). Then click Daten --> Text in Spal
# save again as CSV UTF-8
```

```r
options( "encoding" = "UTF-8" )

# output
output <- c("aphasia_patient_TeaP.csv") # data file for spoken data

# arrays
arrays <- "arrays_umlaut.csv"

# load SOSCISURVEY DATA
datafiles <- list()
for(i in 1:length(sosci_files)) {
   datafiles[[i]] <- read.csv(here::here("data", "raw", "preliminary_for_TeaP",
                                          sosci_files[i]), sep = ";",
                              na = "")
}
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote
## = quote, : unvollständige letzte Zeile von readTableHeader in '/Users/
## kirstenstark/Documents/research/research projects/CSI_online_aphasia/data/raw/
## preliminary_for_TeaP/as1957/as1957data.csv' gefunden

## Warning in read.table(file = file, header = header, sep = sep, quote
## = quote, : unvollständige letzte Zeile von readTableHeader in '/Users/
## kirstenstark/Documents/research/research projects/CSI_online_aphasia/data/raw/
## preliminary_for_TeaP/aw1961/aw1961data.csv' gefunden

## Warning in read.table(file = file, header = header, sep = sep, quote
## = quote, : unvollständige letzte Zeile von readTableHeader in '/Users/
## kirstenstark/Documents/research/research projects/CSI_online_aphasia/data/raw/
## preliminary_for_TeaP/aw1975/aw1975data.csv' gefunden

## Warning in read.table(file = file, header = header, sep = sep, quote
## = quote, : unvollständige letzte Zeile von readTableHeader in '/Users/
## kirstenstark/Documents/research/research projects/CSI_online_aphasia/data/raw/
## preliminary_for_TeaP/CG1970/CG1970data.csv' gefunden

## Warning in read.table(file = file, header = header, sep = sep, quote
## = quote, : unvollständige letzte Zeile von readTableHeader in '/Users/
## kirstenstark/Documents/research/research projects/CSI_online_aphasia/data/raw/
## preliminary_for_TeaP/ek1971/ek1971data.csv' gefunden
## Warning in read.table(file = file, header = header, sep = sep, quote
## = quote, : unvollständige letzte Zeile von readTableHeader in '/Users/
## kirstenstark/Documents/research/research projects/CSI_online_aphasia/data/raw/
## preliminary_for_TeaP/hj1969/hj1969data.csv' gefunden

## Warning in read.table(file = file, header = header, sep = sep, quote
## = quote, : unvollständige letzte Zeile von readTableHeader in '/Users/
## kirstenstark/Documents/research/research projects/CSI_online_aphasia/data/raw/
## preliminary_for_TeaP/rj1967/rj1967data.csv' gefunden
```

```
# if(type == "pilot_patient" || type == "main") {
#   datafiles[[i]]$QUESTNNR <- "online_CSI_spoken"
# }

# save cell instructions
instructions <- datafiles[[1]][1,]

# perform some transformations on each dataframe
for(i in 1:length(sosci_files)) {
  # add type column
  datafiles[[i]]$type <- type[i]
  # delete instruction column
  if(datafiles[[i]][1,1] == "Interview-Nummer (fortlaufend)") {
    datafiles[[i]] <- datafiles[[i]][-c(1),]
  }
  # add subject id: each subject is saved in a separate folder
  datafiles[[i]] <- datafiles[[i]] %>% dplyr::mutate(subject = i)
  # add testing session per subject
  datafiles[[i]] <- datafiles[[i]] %>% dplyr::mutate(session = row_number())
}

# load VOT DATA
datafiles_vot <- list()
for(i in 1:length(vot_files)) {
   datafiles_vot[[i]] <- read.csv(here::here("data", "raw",
                                             "preliminary_for_TeaP",
                                         vot_files[i]), sep = ",",
                              na = "")
   datafiles_vot[[i]]$name <- vot_files[i]
}


# perform some transformations on each dataframe
participant <- 1
#session <- 1
for(i in 1:length(vot_files)) {
  # add type column
  datafiles_vot[[i]]$type <- type[i]
  # add subject personal code
  datafiles_vot[[i]]$OR02_01 <- substr(vot_files[i], 1,6)
  # add subject id: each subject is saved in a separate folder
  ## and add testing session per subject
  if (i == 1) {
    datafiles_vot[[i]] <- datafiles_vot[[i]] %>% dplyr::mutate(subject = participant)
    # datafiles_vot[[i]] <- datafiles_vot[[i]] %>% dplyr::mutate(session = session)
    # session <- session +1
  } else if (datafiles_vot[[i]]$OR02_01[1] == datafiles_vot[[i-1]]$OR02_01[1]){
     datafiles_vot[[i]] <- datafiles_vot[[i]] %>%
       dplyr::mutate(subject = participant)
     # datafiles_vot[[i]] <- datafiles_vot[[i]] %>%
     #   dplyr::mutate(session = session)
     # session <- session + 1
  } else if (datafiles_vot[[i]]$OR02_01[1] != datafiles_vot[[i-1]]$OR02_01[1]){
     participant <- participant +1
     # session <- 1
    datafiles_vot[[i]] <- datafiles_vot[[i]] %>%
      dplyr::mutate(subject = participant)
    # datafiles_vot[[i]] <- datafiles_vot[[i]] %>%
    #   dplyr::mutate(session = session)
    #  session <- session + 1
  } else {
    print("error")
  }

  # add session ID
  datafiles_vot[[i]] <- datafiles_vot[[i]] %>%
    mutate(session=stringr::str_sub(name, start = -16)) %>%
    mutate(session=stringr::str_sub(session, start = 1, end=1)) %>%
    mutate(session=as.numeric(as.character(session)))

  # fix corrupted trial no
  colnames(datafiles_vot[[i]])[1] <- "trial"
}
```

Convert to long format, prepare wide dataframe, and bind long and wide dataframes together

First convert all variables with values for each trial, then bind them together. In a next step bind them to the variables that only have one value per participant.

```
for(i in 1:length(sosci_files)){
  # print(i)

#--------------------------------------------------------------
# Prepare long data frame
    ## if the file also contains spoken data
    if("online_CSI_spoken" %in% datafiles[[i]]$QUESTNNR){
      ### AUDIO FILES
      # In some of the datafiles, two columns seem to be missing and the order of the columns is corrup
      for (j in nrow(datafiles[[i]])) {
        if (!grepl("AR01", datafiles[[i]]$AR01x02[j], fixed=TRUE)) {
          for (k in 407:20){
            datafiles[[i]][j,k] <- datafiles[[i]][j, k-1]
          }
          for (k in 407:38){
            datafiles[[i]][j,k] <- datafiles[[i]][j, k-1]
          }
        }
      }

      # Rename session like the audio files
      datafiles[[i]]$session <-
        stringr::str_sub(datafiles[[i]]$AR01x02, start = -11)

      # Audio files of first 1-80 trials
      df1 <- datafiles[[i]] %>%
              select('subject', starts_with("AR")&contains("x")) %>%
                      pivot_longer(
                        cols = -subject,
                         names_to = c("trial"),
                        values_to = "audio") %>%
                        group_by(subject)
                        # order by AR, but for each repetition separately
      df1$session <- stringr::str_sub(df1$audio, start = -11)
      df1 %>%
        arrange(session, trial) %>%
        mutate(trial = rep(seq(1,80),
                            times = length(unique(df1$session))))-> df1
      # Audio files of last 1-80 trials
      df2 <- datafiles[[i]] %>%
              select('subject', starts_with("AU")&contains("x")) %>%
                      pivot_longer(
                        cols = -subject,
                         names_to = c("trial"),
                        values_to = "audio") %>%
                        group_by(subject)
                        # order by AU, but for each repetition separately
      df2$session <- stringr::str_sub(df2$audio, start = -11)
      df2 %>%
        arrange(session, trial) %>%
        mutate(trial = rep(seq(81,160),
                            times = length(unique(df1$session))))-> df2
      # bind first 80 and last 80 trials together
      df_audio <- bind_rows(df1, df2) %>%
            arrange(subject, session, trial)
      # delete audiofile columns from wide data frame
      datafiles[[i]] <- datafiles[[i]] %>%
          select(!starts_with(c("AR", "AU")))
    }

### Arrange order
    df_main <- df_audio %>% arrange(subject, session, trial)


#--------------------------------------------------------------
# Adapt wide data frame with info that is assessed only once

# for control reasons: calculate time sum by hand:
```

```r
# sum dwell times for each page
datafiles[[i]] <- datafiles[[i]] %>%
  mutate_at(vars(contains("TIME0")), as.numeric)
datafiles[[i]] <- datafiles[[i]] %>% rowwise() %>%
  dplyr::mutate(timetotal = rowSums(across(starts_with("TIME0")), na.rm=TRUE)/60)

# delete columns with info we don't need
datafiles[[i]] <- datafiles[[i]] %>%
  select(-c(SERIAL, REF, MODE, SD22_PRV,SD22_BVS, LASTDATA,
            SD19, SD19_01, SD19_02, SD19_03, MISSING, MISSREL))

# delete columns that contain only NAs
datafiles[[i]]<- datafiles[[i]] %>% select_if(~sum(!is.na(.)) > 0)

# delete practice audio files
if("online_CSI_spoken" %in% datafiles[[i]]$QUESTNNR){
  datafiles[[i]]<- datafiles[[i]] %>% select(!starts_with("PA"))
}

# add comments column if participant left no comment
if(!("IM01_01" %in% colnames(datafiles[[i]]))){
  datafiles[[i]]$IM01_01 <- ""
}

# give columns more recognizable names
  datafiles[[i]] <- datafiles[[i]] %>%
  dplyr::rename(gender = SD01, age = SD02_01,language = SD21,
                     os_system = SD22_OS, browser_automatic = SD22_BNM,
                     system_format  = SD22_FmF,
                     handedness = SD27,
                     comments = IM01_01, time_wo_outlier = TIME_SUM,
                     screen_width = SD22_ScW, screen_height = SD22_ScH,
                     questionnaire_width = SD22_QnW)


#-------------------------------------------------------------
# Bind long and wide data frame together
# Repeat each subjects' rows 160 times (no of trials)
datafiles[[i]] <- datafiles[[i]] %>% slice(rep(seq_len(n()), 160))

# Add trial number to wide data frame
datafiles[[i]]$trial <-
  rep(1:160, times = length(unique(datafiles[[i]]$session)))

# Arrange by session and trial
datafiles[[i]] %>% arrange(subject, session, trial) -> datafiles[[i]]

# bind wide and long info together
datafiles[[i]] <- datafiles[[i]] %>%
  left_join(df_main, by = c("subject", "session", "trial")) %>%
  relocate(subject, session, trial)

# make sure sessions are ordered in the correct order:
```

```
datafiles[[i]] %>% arrange(STARTED) -> datafiles[[i]]

# rename sessions in numerical order
datafiles[[i]] %>% mutate(session=case_when(
  STARTED == unique(datafiles[[i]]$STARTED)[1] ~ 1,
  STARTED == unique(datafiles[[i]]$STARTED)[2] ~ 2,
  STARTED == unique(datafiles[[i]]$STARTED)[3] ~ 3))-> datafiles[[i]]

# prepare for merging
datafiles[[i]] <- datafiles[[i]] %>% select(-CASE)
datafiles[[i]] <- datafiles[[i]] %>%
 #  mutate(CASE = as.character(CASE))%>%
   mutate(OR01_01 = as.numeric(as.character(OR01_01))) %>%
   mutate(gender = as.numeric(as.character(gender))) %>%
   mutate(age = as.numeric(as.character(age))) %>%
  mutate(language = as.numeric(as.character(language))) %>%
  mutate(os_system = as.numeric(as.character(os_system))) %>%
  mutate(system_format = as.numeric(as.character(system_format))) %>%
  mutate(handedness = as.numeric(as.character(handedness))) %>%
  mutate(comments = as.character(comments)) %>%
  mutate(time_wo_outlier = as.numeric(as.character(time_wo_outlier))) %>%
  mutate(screen_width = as.numeric(as.character(screen_width))) %>%
  mutate(screen_height = as.numeric(as.character(screen_height))) %>%
  mutate(questionnaire_width = as.numeric(as.character(questionnaire_width)))%>%
  mutate(browser_automatic = as.numeric(as.character(browser_automatic))) %>%
  mutate(CH01 = as.numeric(as.character(CH01))) %>%
   mutate(CH01_01 = as.numeric(as.character(CH01_01))) %>%
   mutate(CH01_02 = as.numeric(as.character(CH01_02))) %>%
   mutate(CH01_03 = as.numeric(as.character(CH01_03))) %>%
    mutate(CH01_04 = as.numeric(as.character(CH01_04))) %>%
  mutate(CH02 = as.numeric(as.character(CH02))) %>%
   mutate(CH02_01 = as.numeric(as.character(CH02_01))) %>%
   mutate(CH02_02 = as.numeric(as.character(CH02_02))) %>%
   mutate(CH02_03 = as.numeric(as.character(CH02_03))) %>%
    mutate(CH02_04 = as.numeric(as.character(CH02_04))) %>%
  mutate(CH03 = as.numeric(as.character(CH03))) %>%
mutate(MC01 = as.numeric(as.character(MC01))) %>%
   mutate(MC01_01 = as.numeric(as.character(MC01_01))) %>%
   mutate(MC01_02 = as.numeric(as.character(MC01_02))) %>%
   mutate(MC01_03 = as.numeric(as.character(MC01_03))) %>%
    mutate(MC01_04 = as.numeric(as.character(MC01_04))) %>%
  mutate(MC02 = as.numeric(as.character(MC02))) %>%
  mutate(MC03 = as.numeric(as.character(MC03))) %>%
  mutate(FINISHED = as.numeric(as.character(FINISHED))) %>%
  mutate(Q_VIEWER = as.numeric(as.character(Q_VIEWER))) %>%
  mutate(LASTPAGE = as.numeric(as.character(LASTPAGE))) %>%
  mutate(MAXPAGE = as.numeric(as.character(MAXPAGE))) %>%
  mutate(DEG_TIME = as.numeric(as.character(DEG_TIME)))


#------------------------------------------------------------
# Convert numeric variables from string to integer:
#str(df)
```

```
# if(type[i] == "pilot_patient"){
#   datafiles[[i]] <-  datafiles[[i]] %>%
#   mutate_at(vars(!c("type", contains("KBO"))), as.numeric)
# }else if(type[i] == "main") {
#   datafiles[[i]] <-  datafiles[[i]] %>%
#   mutate_at(vars(!c( "browser_other",
#                      "word", "comments", "type",
#                      "array_no", "TIME_RSI", contains("KBO"))), as.numeric)
# } else if(type[i] == "replacement") {
#     datafiles[[i]] <- datafiles[[i]] %>%
#    mutate_at(vars(!c("operator_system_other",
#                       "word", "comments", "type",
#                      "array_no", contains("KBO"))), as.numeric)
# }
}
```

Add array (actual stimuli) for each participant

For the spoken data, this is just to double check that everything went fine upon merging the files and to add the categories.

```
# load arrays
arrays <- read.csv2(here::here("data", "supplementary_info", arrays),
    sep = ";", na = "NA")

for (i in 1:length(sosci_files)) {
  # convert array column in datafiles
  # datafiles[[i]]$array <- as.numeric(
  #   stringr::str_remove(
  #   datafiles[[i]]$array, "Array"))
  datafiles[[i]]$array <- as.numeric(as.character(datafiles[[i]]$OR01_01))
  # bind arrays to data frame
  datafiles[[i]]$item <- NA
    for(j in 1:nlevels(as.factor(datafiles[[i]]$session))){
      array <- arrays[, unique(datafiles[[i]]$array[
                          datafiles[[i]]$session == j])]
      datafiles[[i]]$item[datafiles[[i]]$session == j] <- array
    }

  # add category columns
  datafiles[[i]] <- datafiles[[i]] %>% group_by(subject, session, item) %>%
    dplyr::mutate(category = arrays$categorie[arrays$item == item]) %>%
    dplyr::mutate(supercategory =
                    arrays$supercategorie[arrays$item == item] )

    #   # add position number
    # datafiles[[i]] <- datafiles[[i]] %>% group_by(subject, category) %>%
    #   add_count() %>%
    #   dplyr::mutate(PosOr = seq(1:n)) %>% select(-n)
    # # add ordinal position for fillers
    # table(datafiles[[i]]$PosOr)
    # count <- 1
```

```r
    # for (j in 1:nrow(datafiles[[i]])) {
    #   if(datafiles[[i]]$category [j] == "Filler") {
    #     datafiles[[i]]$PosOr[j] <- count
    #     count <- count+1
    #   }
    #   if(count == 6) {  count <- 1}
    # }
    # table(datafiles[[i]]$PosOr)
}
```

Bind the dataframes together

```r
df <- bind_rows(datafiles)

# Did it the array merging work?
x <- strsplit(df$audio, "[.]")
for (i in 1:nrow(df)){
  if(df$item[i] != tolower(x[[i]][1])){
    print("merging error")
  }
}
```

Combine data frame with audio files

```r
# fix participant ID
for(i in 1:length(datafiles_vot)) {
  if(i <= length(datafiles)) {
     datafiles[[i]] <- datafiles[[i]] %>% mutate(OR02_01=toupper(OR02_01))
  }
  datafiles_vot[[i]] <- datafiles_vot[[i]] %>% mutate(OR02_01=toupper(OR02_01))
  # fix filename
  datafiles_vot[[i]] <- datafiles_vot[[i]] %>% rename(audio=File_name)
}

# bind VOT files of one participant into one data frame
data_vot <- as.data.frame(datafiles_vot[[1]])
for(i in 2:length(vot_files)) {
  data_vot <- rbind(data_vot, datafiles_vot[[i]])
}

# add item to data_vot
x <- strsplit(data_vot$audio, "[.]")
for(i in 1:nrow(data_vot)) {
  data_vot$item[i] <- tolower(x[[i]][1])
}

data_vot$item[data_vot$item == "loewenzahn"] <- "löwenzahn"
data_vot$item[data_vot$item == "wuerfel"] <- "würfel"
```

```r
data_vot$item[data_vot$item == "muelleimer"] <- "mülleimer"
data_vot$item[data_vot$item == "buerste"] <- "bürste"
data_vot$item[data_vot$item == "saege"] <- "säge"
data_vot$item[data_vot$item == "schluessel"] <- "schlüssel"
data_vot$item[data_vot$item == "kopfhoerer"] <- "kopfhörer"
data_vot$item[data_vot$item == "geschirrspueler"] <- "geschirrspüler"
data_vot$item[data_vot$item == "uboot"] <- "u-boot"
data_vot$item[data_vot$item == "kuehlschrank"] <- "kühlschrank"
data_vot$item[data_vot$item == "kaefig"] <- "käfig"
data_vot$item[data_vot$item == "marienkaefer"] <- "marienkäfer"
data_vot$item[data_vot$item == "loewe"] <- "löwe"
data_vot$item[data_vot$item == "loeffel"] <- "löffel"
data_vot$item[data_vot$item == "faecher"] <-  "fächer"
data_vot$item[data_vot$item == "baer"] <- "bär"

# merge data from data_vot with df
df$VOT <- NA
df$correct <- NA
df$AR <- NA
df$error <- NA
df$name <- NA
df$audio_vot <- NA
for(i in 1:nrow(data_vot)) {
  df$VOT[tolower(df$OR02_01) == tolower(data_vot$OR02_01)[i] &
           df$session == data_vot$session[i] &
           df$item == data_vot$item[i]] <- data_vot$VOT[i]
  df$correct[tolower(df$OR02_01) == tolower(data_vot$OR02_01)[i] &
           df$session == data_vot$session[i] &
           df$item == data_vot$item[i]] <- data_vot$correct[i]
  df$AR[tolower(df$OR02_01) == tolower(data_vot$OR02_01)[i] &
           df$session == data_vot$session[i] &
           df$item == data_vot$item[i]] <- data_vot$AR[i]
  df$error[tolower(df$OR02_01) == tolower(data_vot$OR02_01)[i] &
           df$session == data_vot$session[i] &
           df$item == data_vot$item[i]] <- data_vot$error[i]
  df$name[tolower(df$OR02_01) == tolower(data_vot$OR02_01)[i] &
           df$session == data_vot$session[i] &
           df$item == data_vot$item[i]] <- data_vot$name[i]
  df$audio_vot[tolower(df$OR02_01) == tolower(data_vot$OR02_01)[i] &
           df$session == data_vot$session[i] &
           df$item == data_vot$item[i]] <- data_vot$audio[i]
}

# check the differences: just file names once written wit Umlaut, once without
x <- df[stringr::str_sub(df$audio, end = -5) != stringr::str_sub(df$audio_vot, end = -4),]
x <- x %>% select(audio, audio_vot) %>% droplevels()
```

```
## Adding missing grouping variables: `subject`, `session`, `item`
```

```r
sum(is.na(df$VOT)) # 160 audio files missing (session 3 of aw1975)
```

```
## [1] 160
```

Roughly check participants' adherence to the experiment

Did all participants finish the experiment?

```r
# did all participants finish the experiment?
for(i in 1:length(unique(df$subject))) {
  print(paste(i,":"))
  print("Experiment completed?")
  print(table(df$FINISHED[df$subject==i])/160)
  print("What was the last experimental page reached?")
  print(table(df$LASTPAGE[df$subject==i])/160)
}
```

```
## [1] "1 :"
## [1] "Experiment completed?"
##
## 1
## 3
## [1] "What was the last experimental page reached?"
##
## 32
##  3
## [1] "2 :"
## [1] "Experiment completed?"
##
## 1
## 3
## [1] "What was the last experimental page reached?"
##
## 32
##  3
## [1] "3 :"
## [1] "Experiment completed?"
##
## 0 1
## 1 2
## [1] "What was the last experimental page reached?"
##
## 29 32
##  1  2
## [1] "4 :"
## [1] "Experiment completed?"
##
## 1
## 3
## [1] "What was the last experimental page reached?"
##
## 32
##  3
## [1] "5 :"
## [1] "Experiment completed?"
##
## 1
## 3
## [1] "What was the last experimental page reached?"
##
## 32
##  3
## [1] "6 :"
## [1] "Experiment completed?"
##
## 1
## 3
## [1] "What was the last experimental page reached?"
##
## 32
##  3
## [1] "7 :"
## [1] "Experiment completed?"
##
## 0 1
## 1 2
## [1] "What was the last experimental page reached?"
##
## 31 32
##  1  2
```

Participant 7 once only reached the second last page, participant 3 the thrid last.

Exclude participants who did not finish the experiment for approval:

```
# for (i in 1:length(input)) {
#   exclude <- datafiles[[i]]$prolificid[datafiles[[i]]$FINISHED == "0" &
#                               datafiles[[i]]$LASTPAGE != "30"]
#   exclude_id <- datafiles[[i]]$subject[datafiles[[i]]$FINISHED == "0" &
#                               datafiles[[i]]$LASTPAGE != "30"]
#
#   # Exclude from data frame
#   datafiles[[i]] <- datafiles[[i]][!datafiles[[i]]$subject %in% exclude_id,]
#
#   # update subject IDs
#   datafiles[[i]]$subject <-
#     rep(1:nlevels(as.factor(datafiles[[i]]$subject)), each = 160)
#
#   # All the other data look fine, so we can include all others:
#   include <- as.data.frame(
#     datafiles[[i]]$prolificid[datafiles[[i]]$FINISHED == "1" &
#                               datafiles[[i]]$LASTPAGE == "33"])
#
#   # delete prolific ids ton anonymize data frame
#   datafiles[[i]] <- datafiles[[i]] %>%
#    dplyr::select(-prolificid)
# }
#
```

Fix single participants

In the typing experiment, the typed words of participant 1 were not saved (reason unknown). We fix this by putting all the single letter columns together. For some reason, space or enter were also saved as NANA. We will fix that later

```
# df_typing <- df_typing %>% unite(word,starts_with("letter"),sep="", remove=FALSE, na.rm=TRUE)
```

Export prepared data frame

```
#if("online_CSI_spoken" %in% unique(df$QUESTNNR)) {
  write.csv(df, here::here("data", "transient_data_files", output),
            row.names = FALSE)
#}
```