02 CSI online aphasia: Spoken - Preprocessing

Kirsten Stark

17 Oktober, 2021

@ Marcus: Am besten, du klickst in dem Ordner, den ich dir als .zip-file geschickt habe, die Datei "CSI_online_aphasia.Rproj" an. Dann öffnet sich R Studio neu. In dem neuen Fenster unten links unter Files –> Scripts findest du dann dieses Skript hier mit dem Namen "02_CSI_online_aphasia_typing_preprocessing.Rmd"). Doppelklick darauf sollte das Skript öffnen. Jetzt kannst du bei jedem grau hinterlegten Feld nacheinander auf das grüne Dreieck drücken und die einzelnen Abschnitte laufen lassen. Dabei am besten immer auf die Kommentare achten, ob du noch etwas anpassen musst =)

Der Code mit # ist jeweils auskommentiert, alles in grauen Boxen oder # davor wird läuft bei Drücken auf Play durch.

Load packages

```
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)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from '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
# input output main data
type <- c("pilot_patient")
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("pilot_spoken_patient_long.csv") # data file for spoken data

# arrays
arrays <- "arrays_umlaut.csv"

# load data
datafiles <- list()
for(i in 1:length(input)) {
   datafiles[[i]] <- read.csv(here::here("data", "raw", input[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/CSI_online_aphasia/data/raw/
## pilot_newdissmtukl_2021-10-13_10-44.csv' gefunden
```

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

# perform some transformations on each dataframe
for(i in 1:length(input)) {
  # add type column
  datafiles[[i]]$type <- type[i]
  # delete description column and experimenter's tryout data
  # datafiles[[i]] <- datafiles[[i]][-c(1:2),]
  # add subject id
  datafiles[[i]] <- datafiles[[i]] %>% dplyr::mutate(subject = row_number())
}
```

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(input)){

#------------------------------------------------------------
# Prepare long data frame
    ## if the file also contains spoken data
    if("online_CSI_spoken" %in% datafiles[[i]]$QUESTNNR){
      ### AUDIO FILES
      # 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 seperately
        for (k in 1:(nrow(df1)/80)){
          df_k <- df1[((k-1)*80+1):(k*80),]  %>%
            arrange(trial)
            df1[((k-1)*80+1):(k*80),] <- df_k
        }
      df1 <- df1 %>%  dplyr::mutate(trial = seq(1,80, by = 1))
      # 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
              for (k in 1:(nrow(df2)/80)){
                 df_k <- df2[((k-1)*80+1):(k*80),]  %>%
                 arrange(trial)
                df2[((k-1)*80+1):(k*80),] <- df_k
             }
      df2 <- df2 %>%  dplyr::mutate(trial = seq(81,160, by = 1))
      # bind first 80 and last 80 trials together
      df_audio <- bind_rows(df1, df2) %>%
            arrange(subject, 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, trial)
```

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

# for control reasons: calculate time sum by hand:
# 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")))/60)

# delete columns with info we don't need
datafiles[[i]] <- datafiles[[i]] %>%
  select(-c(CASE, MODE, STARTED, SD22_PRV,
                        SD22_BID, 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"))
}

# give columns more recognizable names
if (type[i] == "pilot_patient") {
  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,
                      time_wo_outlier = TIME_SUM,
                      screen_width = SD22_ScW, screen_height = SD22_ScH,
                      questionnaire_width = SD22_QnW, array=OR01_01)
} else if (type[i] == "main") {
  datafiles[[i]] <- datafiles[[i]] %>%
  dplyr::rename(array_no = AY01_01, gender = SD01, age = SD02_01,
                      language.test = SD20, language = SD21,
                      os_system = SD22_OS, browser_automatic = SD22_BNM,
                      system_format  = SD22_FmF, prolificid = SD24_01,
                      handedness = SD27,
                      comments = IM01_01, time_wo_outlier = TIME_SUM,
                      screen_width = SD22_ScW, screen_height = SD22_ScH,
                      questionnaire_width = SD22_QnW)
} else if (type[i] == "replacement") {
  datafiles[[i]] <- datafiles[[i]] %>%
  dplyr::rename(gender = SD01, age = SD02_01,
                      language.test = SD20, language = SD21,
                      os_system = SD22_OS, browser_automatic = SD22_BNM,
                      system_format  = SD22_FmF, prolificid = SD24_01,
```

```
                      handedness = SD27,
                      comments = IM01_01, time_wo_outlier = TIME_SUM,
                      screen_width = SD22_ScW, screen_height = SD22_ScH,
                      questionnaire_width = SD22_QnW,
                     array=OR01_01)
}

#-------------------------------------------------------------
# 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]]$trial <- rep(1:160, times = max(datafiles[[i]]$subject))

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

#-------------------------------------------------------------
# Convert numeric variables from string to integer:
#str(df)
# if(type[i] == "pilot_patient"){
#   datafiles[[i]] <-  datafiles[[i]] %>%
#   mutate_at(vars(!c("type", contains("KB0"))), 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("KB0"))), as.numeric)
# } else if(type[i] == "replacement") {
#     datafiles[[i]] <- datafiles[[i]] %>%
#    mutate_at(vars(!c("operator_system_other",
#                       "word", "comments", "type",
#                      "array_no", contains("KB0"))), as.numeric)
# }
}
```

Roughly check participants' adherence to the experiment

Did all participants finish the experiment?

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

```
## [1] "pilot_patient :"
## [1] "Experiment completed?"
##
## 1
## 3
## [1] "What was the last experimental page reached?"
##
## 32
##  3
```

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)
# }
#
```

Add array (actual stimuli) for each participant

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

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

```
      datafiles[[i]]$item[datafiles[[i]]$subject == j] <- array
    }

  # add category columns
  datafiles[[i]] <- datafiles[[i]] %>% group_by(subject, 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]]$PosOr)
    count <- 1
    for (j in 1:nrow(datafiles[[i]])) {
      if(datafiles[[i]]$category [j] == "Filler") {
        datafiles[[i]]$PosOr[j] <- count
        count <- count+1
      }
      if(count == 6) {  count <- 1}
    }
    table(datafiles[[i]]$PosOr)
}
```

Bind the dataframes together

```
df <- bind_rows(datafiles)
```

Combine data frame with audio files

```
## fix participant ID
df <- df %>% mutate(ID = case_when(is.na(OR02_01) ~ toupper(IM01_01),
                                   !is.na(OR02_01) ~ toupper(OR02_01)))

# fix subject ID and add testing ID
subject <-  1
testing <- 1
df$testing <- NA
for(i in 1:(nrow(df)-1)){
  if(df$ID[i] == df$ID[i+1]) {
    df$subject = subject
  } else {
    df$subject == subject
    subject = subject +1
  }
```

```
  if(i > 1){
    if(df$trial[i] == 1 & df$ID[i] !=df$ID[i-1]) {
      testing = 1
    } else if (df$trial[i] == 1 & df$ID[i] == df$ID[i-1]){
      testing = testing +1
    }
  }
  df$testing[i] <- testing
  if(i == nrow(df)-1) {
    df$testing[i+1] <- testing
  }
}

## read latency file names
filenames <- list.files(here::here("data","raw","latencies"), pattern="*.txt", full.names=FALSE)


temp3 <- NA
for(i in 1:length(filenames)){
  # get file info
  info <- stringr::str_split(filenames[i], c("_"))
  temp <- info[[1]][2]
  temp <- as.numeric(as.character(substr(temp,1,1)))
  info[[1]][2] <- temp

  # read text file
  file <- read.csv(here::here("data","raw","latencies", filenames[i]),
                   sep = "\t", header=FALSE,  encoding = "UTF-8" )
  colnames(file) <- c("audio", "RT")
  # exclude practice files
  file <- file %>% filter(stringr::str_detect(file$audio, ".PA") == FALSE)
  # extract item names
  for(j in 1:length(file$audio)){
    file$item[j] <- tolower(stringr::str_split(file$audio, ".A")[[j]][1])
  }
  # fix names with Umlaute
  file$item[157]<- "würfel" # file$item == "würfel"
  file$item[127]<- "schlüssel" # file$item == "schlüssel"
  file$item[120]<-"säge" # file$item ==  "säge"
  file$item[100]<-"mülleimer" # file$item ==  "mülleimer"
  file$item[95]<-"marienkäfer" # file$item == "marienkäfer"
  file$item[93]<-"löwenzahn" # file$item == "löwenzahn"
  file$item[92]<-"löwe"# file$item == "löwe"
  file$item[91]<-"löffel" # file$item == "löffel"
  file$item[86]<-   "kühlschrank" # file$item == "kühlschrank"
  file$item[81]<- "kopfhörer" # file$item ==  "kopfhörer"
  file$item[71]<-"käfig" # file$item ==  "käfig"
  file$item[48]<-"geschirrspüler" # file$item ==  "geschirrspüler"
  file$item[39]<-"fächer" # file$item == "fächer"
  file$item[31]<-"bürste" # file$item == "bürste"
  file$item[15]<-"bär" # file$item ==  "bär"
```

```
  # select relevant participant from datafile
  temp <- df %>% filter(ID == info[[1]][1])

  # select relevant testing (1,2,3)
  temp2 <- temp %>% filter(testing == as.numeric(info[[1]][2]))

  # combine latencies and further data
  temp2 <- merge(temp2, file,  by="item", all = TRUE)

  # combine all together
  if (is.na(temp3)) {
    temp3 <- temp2
  } else {
    temp3 <- rbind(temp3, temp2)
  }
}
```

```
## Warning in if (is.na(temp3)) {: Bedingung hat Länge > 1 und nur das erste
## Element wird benutzt

## Warning in if (is.na(temp3)) {: Bedingung hat Länge > 1 und nur das erste
## Element wird benutzt
```

```
if (nrow(df) == nrow(temp3)){
  df_all <- temp3
} else {
  print0("Merging error!")
}

# delete audio file names
# df_all <- df_all %>% select(!c("audio.x", "audio.y"))
```

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_all, here::here("data", "transient_data_files", output[1]), row.names = FALSE)
}
```