02 CSI online aphasia: Spoken - Preprocessing Final Data Set

Kirsten Stark

13 June, 2023

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)

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

Load and preprocess data

This input file needs to be entered by hand:

```
# input output main data
type <- c("PWA", "control")

## Load files from the control and the PWA group
for(i in 1:length(type)){
  # read in file names
  eval(parse(text=paste0("files_", type[i],
         "<- list.files(here::here('data', 'raw', 'final','",type[i],
         "'), recursive = TRUE)")))
  ## sosci_files
  eval(parse(text=paste0("sosci_files_", type[i],
                         "<- files_", type[i],
                         "[grep('data.csv', files_", type[i],",",
                         fixed=T)]")))
  ## vot files
```

```r
  eval(parse(text=paste0("vot_files_", type[i],
  "<- files_", type[i],
  "[-grep('data.csv', files_", type[i],", fixed=T)]")))
}
```

```r
# output file name
output <- c("aphasia_final.csv")

# arrays
arrays <- "arrays_umlaut.csv" # experimental arrays
```

```r
for(i in 1:length(type)){
  ##### SOSCI FILES
  # reorder sosci files
  eval(parse(text=paste0(
    "sosci_files_", type[i],
    "<- c(sosci_files_", type[i], "[substr(sosci_files_", type[i],
    ", 2,2)=='.'] %>% sort(),",
    "sosci_files_", type[i], "[substr(sosci_files_", type[i],
    ", 2,2)!='.'] %>% sort())")))

  # load sosci file data
  eval(parse(text=paste0("datafiles_", type[i], "<- list()")))
  eval(parse(text=paste0("for(j in 1:length(sosci_files_", type[i],")) {",
        "datafiles_", type[i], "[[j]] <-
        read.csv(here::here('data', 'raw', 'final','", type[i], "',
        sosci_files_", type[i], "[j]), sep = ';', na = '')",
        "}")))

  # save cell instructions
  if(i == 1){
      eval(parse(text=paste0("instructions <- datafiles_",
                             type[i],"[[1]][1,]")))
  }

  # perform some transformations on each data frame
  eval(parse(text=paste0("for(j in 1:length(sosci_files_",type[i],")){",
    # add original name
    "datafiles_", type[i], "[[j]]$name","<- sosci_files_",type[i],"[j];",

    # add type column
    "datafiles_", type[i], "[[j]]$type","<- '",type[i], "';",

    # delete instruction column
    "if(datafiles_", type[i], "[[j]][1,1] ==",
    "'Interview-Nummer (fortlaufend)') {",
          "datafiles_", type[i], "[[j]] <-",
          "datafiles_", type[i], "[[j]][-c(1),]",
    "};",

    # add subject id: each subject is saved in a separate folder
      "datafiles_", type[i], "[[j]]$subject <- i*100+j;",
    # add testing session per subject
    "datafiles_", type[i], "[[j]] <-","datafiles_", type[i], "[[j]] %>% ",
                           "dplyr::mutate(session = row_number())",

    "}")))


    ##### VOT FILES
  # reorder VOT Data
  eval(parse(text=paste0("vot_files_", type[i], "<-
    c(vot_files_", type[i], "[substr(vot_files_", type[i],", 2,2)==
    '.'] %>% sort(),
    vot_files_", type[i], "[substr(vot_files_", type[i], ",2,2)!=
    '.'] %>% sort())")))


  # load VOT DATA
  eval(parse(text=paste0("datafiles_vot_", type[i], "<- list()")))
  eval(parse(text=paste0("for(j in 1:length(vot_files_",type[i], ")) {",
     "datafiles_vot_", type[i], "[[j]] <- ",
     "read.csv(here::here('data', 'raw', 'final', '", type[i], "',
     vot_files_", type[i], "[j]), sep =',', na = '');",
     "datafiles_vot_", type[i], "[[j]]$name <- vot_files_", type[i], "[j];",
  "}")))


  # perform some transformations on each dataframe
  participant <- 1

  eval(parse(text=paste0("for(j in 1:length(vot_files_", type[i], ")) {",

    # # add original name
    # "datafiles_vot_", type[i], "[[j]]$name_vot <-
    # vot_files_", type[i], "[j];",

    # add type column
    "datafiles_vot_", type[i], "[[j]]$type <- '", type[i], "';",
    "}")))

    # add subject personal code
  eval(parse(text=paste0("for(j in 1:length(vot_files_", type[i], ")) {",
    "if(substr(vot_files_", type[i],", 2,2)=='.'){",
      "datafiles_vot_", type[i], "[[j]]$OR02_01 <-",
        "substr(vot_files_", type[i],"[j], 4,9)",
    "} else {",
      "datafiles_vot_", type[i], "[[j]]$OR02_01 <- ",
        "substr(vot_files_", type[i], "[j], 5,10)",
    "};",

    # add subject id: each subject is saved in a separate folder
    "if (j == 1) {",
      "datafiles_vot_", type[i], "[[j]] <- datafiles_vot_", type[i],
        "[[j]] %>% dplyr::mutate(subject = participant)",
    "} else if (datafiles_vot_", type[i], "[[j]]$OR02_01[1] ==",
                "datafiles_vot_", type[i], "[[j-1]]$OR02_01[1]){",
        "datafiles_vot_", type[i], "[[j]] <- datafiles_vot_", type[i],
        "[[j]] %>% dplyr::mutate(subject = participant);",
    "} else if (datafiles_vot_", type[i], "[[j]]$OR02_01[1] !=",
                "datafiles_vot_", type[i], "[[j-1]]$OR02_01[1]){",
        "participant <- participant +1;",
         "datafiles_vot_", type[i], "[[j]] <-",
           "datafiles_vot_", type[i], "[[j]] %>%",
           "dplyr::mutate(subject = participant)",
    "} else {",
       "print('error')",
    "};",

    "datafiles_vot_", type[i], "[[j]] <-
    datafiles_vot_", type[i], "[[j]]", "%>% ",
                           "dplyr::mutate(subject =", i*100, "+
                           datafiles_vot_", type[i], "[[j]]$subject);",


    #   # add session ID
    "datafiles_vot_", type[i], "[[j]] <- datafiles_vot_", type[i],
    "[[j]] %>%
      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_", type[i], "[[j]])[1] <- 'trial';",
  "}")))
  }

# bind sosci-files together
datafiles <- c(datafiles_PWA, datafiles_control)

# bind vot-files together
datafiles_vot <- c(datafiles_vot_PWA, datafiles_vot_control)

## fix single column names
for(i in 1:(length(vot_files_PWA)+length(vot_files_control))){
  if("error......" %in% colnames(datafiles_vot[[i]])){
    datafiles_vot[[i]] <-  datafiles_vot[[i]] %>% rename(error=error......)
  }
}
```

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.

```r
for(i in 1:(length(sosci_files_PWA)+length(sosci_files_control))){
  # print(i)
#-------------------------------------------------------------
# Prepare long data frame
      ### AUDIO FILES
      # In some of the data files, two columns seem to be missing and the order of the columns is corru
      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 =
                 as.numeric(as.character(stringr::str_sub(trial, 3,4))))-> 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 =
                 as.numeric(as.character(
                   stringr::str_sub(trial, 3,4)))+80)-> df2

      # bind first 80 and last 80 trials together
      df_audio <- bind_rows(df1, df2) %>%
            arrange(subject, session, trial)

      if(nrow(df_audio) != 480) {
        print('error - audio files dont have correct length')}

      # delete audio file 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:
  # 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
  if("SD22_BVS" %in% colnames(datafiles[[i]])){
    datafiles[[i]] <- datafiles[[i]] %>%
    select(-c(SERIAL, REF, MODE, SD22_PRV,SD22_BVS,LASTDATA,
              SD19, SD19_01, SD19_02, SD19_03, MISSING, MISSREL))
  } else {
    datafiles[[i]] <- datafiles[[i]] %>%
    select(-c(SERIAL, REF, MODE, SD22_PRV,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
  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,
```

```r
                      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)))
}
```

Check whether there are missing trials in the sosci and VOT files

```
### Check whether there are missing trials
# VOT FILES
print('VOT files:')
```

```
## [1] "VOT files:"
```

```
for(i in 1:(length(vot_files_PWA)+length(vot_files_control))){
  if(i == 1){
    vot <- datafiles_vot[[i]]
  } else {
    vot <- rbind(vot, datafiles_vot[[i]])
  }
}

for(i in 1:length(unique(vot$subject))){
  for(j in 1:length(unique(vot$session))){
    for(k in 1:(nrow(vot[vot$subject == unique(vot$subject)[i] &
                         vot$session == unique(vot$session)[j],])-1)){
      x1 <- vot$trial[vot$subject == unique(vot$subject)[i] &
                        vot$session == unique(vot$session)[j]][k]
      x2 <- vot$trial[vot$subject == unique(vot$subject)[i] &
                        vot$session == unique(vot$session)[j]][k+1]
      id <- vot$OR02_01[vot$subject == unique(vot$subject)[i] &
                          vot$session == unique(vot$session)[j]][k]
      if(length(x1)== 0 | length(x1)==0){
        print(paste0('Trial number not recorded in subject ',
                     unique(vot$subject)[i], ' (',
                     id, '), session ', unique(vot$session[j]), ' row ', k,
                     ' or ', k+1, '.'))
      } else if (is.na(x1) | is.na(x2)){
        print(paste0('Trial number not recorded in subject ',
                     unique(vot$subject)[i], '(',
                     id, '), session ',unique(vot$session[j]), ' row ',
                     k, ' or ', k+1, '.'))
      } else if (x1 == x2) {
        print(paste0('Trial number not recorded in subject ',
                     unique(vot$subject)[i], ' (', id, '), session ',
                     unique(vot$session[j]), ' row ', k, ' and ', k+1,
                     '(trial number ', x1, ' and ', x2, ').'))
      } else if (x1 != x2-1) {
        print(paste0('Trial number missing between row ', k, ' and ', k+1,
                     ' (trial numbers ', x1, ' and ', x2, ') in subject ',
                     unique(vot$subject)[i], ' (', id, '), session ',
                     unique(vot$session[j]), '.'))
      } else if (k == (nrow(vot[vot$subject == unique(vot$subject)[i] &
                                vot$session == unique(vot$session)[j],]))-1 &
                 x2 != 164){
        print(paste0('Last trials are missing in subject ',
                     unique(vot$subject)[i], ' (', id, '), session ',
                     unique(vot$session[j]), '(last trial numbers in row ',
                     k, ' and ', k+1, ' are ', x1, ' and ', x2, ').'))
      }
    }
    }
  }
```

```
## [1] "Trial number missing between row 132 and 133 (trial numbers 132 and 4) in subject 112 ( id196),
## [1] "Trial number missing between row 133 and 134 (trial numbers 4 and 133) in subject 112 ( id196),
## [1] "Last trials are missing in subject 112 ( id196), session 1(last trial numbers in row 161 and 16
## [1] "Last trials are missing in subject 120 ( uw196), session 1(last trial numbers in row 162 and 16
```

```
# SOSCI FILES - Need to be checked after cleaning
print('SoSci files:')
```

```
## [1] "SoSci files:"
```

```
for(i in 1:(length(sosci_files_PWA)+length(sosci_files_control))){
  if(i == 1){
   sosci <- datafiles[[i]] %>% select(session, subject, trial,OR02_01)
  } else {
    sosci <- rbind(sosci, datafiles[[i]] %>%
                     select(session, subject, trial, OR02_01))
  }
}

for(i in 1:length(unique(sosci$subject))){
  for(j in 1:length(unique(sosci$session))){
    for(k in 1:(nrow(sosci[sosci$subject == unique(sosci$subject)[i] &
                           sosci$session == unique(sosci$session)[j],])-1)){
      x1 <- sosci$trial[sosci$subject == unique(sosci$subject)[i] &
                          sosci$session == unique(sosci$session)[j]][k]
      x2 <- sosci$trial[sosci$subject == unique(sosci$subject)[i] &
```

```r
                          sosci$session == unique(sosci$session)[j]][k+1]
      id <- sosci$OR02_01[sosci$subject == unique(sosci$subject)[i] &
                            sosci$session == unique(sosci$session)[j]][k]
      if(length(x1)== 0 | length(x1)==0){
        print(paste0('Trial number not recorded in subject ', unique(sosci$subject)[i], ' (',
                     id, '), session ', unique(sosci$session[j]), ' row ',
                     k, ' or ', k+1, '.'))
      } else if (is.na(x1) | is.na(x2)){
        print(paste0('Trial number not recorded in subject ',
                     unique(sosci$subject)[i], '(',
                     id, '), session ',unique(sosci$session[j]), ' row ',
                     k, ' or ', k+1, '.'))
      } else if (x1 == x2) {
        print(paste0('Trial number not recorded in subject ', unique(sosci$subject)[i], ' (', id, '), s
                     unique(sosci$session[j]), ' row ', k, ' and ',
                     k+1, '(trial number ', x1, ' and ', x2, ').'))
      } else if (x1 != x2-1) {
        print(paste0('Trial number missing between row ', k, ' and ', k+1,
                     ' (trial numbers ', x1, ' and ', x2, ') in subject ',
                     unique(sosci$subject)[i], ' (', id, '), session ',
                     unique(sosci$session[j]), '.'))
      } else if (k == (nrow(sosci[sosci$subject == unique(sosci$subject)[i] &
                                  sosci$session ==
                                    unique(sosci$session)[j],]))-1 &
                 x2 != 160){
        print(paste0('Last trials are missing in subject ',
                     unique(sosci$subject)[i], ' (', id, '), session ',
                     unique(sosci$session[j]), '(last trial numbers in row ',
                     k, ' and ', k+1, ' are ', x1, ' and ', x2, ').'))
      }
    }
    }
  }
```

Add array (actual stimuli in actual order) 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.

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

for (i in 1:(length(sosci_files_PWA)+length(sosci_files_control))) {
  x <- datafiles[[i]]
  ### Array column: OR02_01
  x$array <- as.numeric(as.character(x$OR01_01))

  ### Create a new dataframe with the fitting array
  for(j in 1:nlevels(as.factor(x$session))){
    eval(parse(text=paste0("y",j,"<-
                           arrays[, unique(x$array[x$session == j])]")))
    ## Add stable columns
    eval(parse(text=paste0("y",j,"<- data.frame(
                           subject=rep(x$subject[x$session == j][1],
                           each=nrow(y", j,")),
                           session=rep(x$session[x$session == j][1],
                           each=nrow(y", j,")),
                           trial=rep(1:160),
                           item=y",j,")"
                           )))
    ## Add category and supercategory from the array
    eval(parse(text=paste0("for (k in 1:nrow(y",j,")){
                           y", j,"$category[k] <-
                              arrays$categorie[arrays$item == y",j,"$item[k]];
                           y", j,"$supercategory[k] <-
                              arrays$supercategorie[arrays$item ==
                              y",j,"$item[k]];
                           }")))
    ## Add further stable columns
    for (m in 1:length(colnames(x[x$session == j,]))){
      if(colnames(x[x$session == j,])[m] != "subject" &
         colnames(x[x$session == j,])[m] != "session" &
         colnames(x[x$session == j,])[m] != "trial" &
         colnames(x[x$session == j,])[m] != "audio") {
        if(length(unique(x[x$session == j,][m])) != 1){
          print(paste0('Error! Column ', colnames(x[x$session == j,])[m],
                       ' is not stable'))
        } else{
              eval(parse(text=paste0("y",j,"<- data.frame(y",j,", ",
              colnames(x[x$session == j,])[m],
              "= rep(x[x$session==j,m][[1]][1],
              each=nrow(y", j,")))")))
        }
      }
    }

    ## Add unstable columns if there is info: AUDIO
    o <- 1
    eval(parse(text=paste0("for(n in 1:nrow(y",j,")){
    l <- strsplit(x$audio[x$session==j][o], '.',1);
    l <- tolower(l[[1]][1]);
    if(!is.na(l)){
    if(tolower(y", j,"$item[n]) == l){
      y",j,"$audio[n] <- x$audio[x$session==j][o];
      o <- o+1
    }} else {
      y",j,"$audio[n] <- NA;
      if(is.na(l)){
      o <- o+1
      }
    }}")))
  }
  y <- rbind(y1,y2,y3)
  datafiles[[i]] <- y
```

```
}
```

Check that everything went fine

```
for(i in 1:length(datafiles)){
  x <- as.data.frame(datafiles[[i]])
  for(j in 1:nrow(x)){
      y <- x$audio[j]
        if(tolower(strsplit(y, ".", fixed=TRUE)[[1]][1]) != x$item[j] |
           is.na(y)){
          print(paste0('Audiofiles and Items dont fit in subject ',
                       x$subject[1], ', session ', x$session[j], ',
                       trial ', x$trial[j], '. The audiofilename is ',
                       x$audio[j], ', but the item should be ',
                       x$item[j], '.'))
        }
  }
}
```

```
## [1] "Audiofiles and Items dont fit in subject 112, session 1, \n                       trial 98. The
## [1] "Audiofiles and Items dont fit in subject 112, session 1, \n                       trial 99. The
## [1] "Audiofiles and Items dont fit in subject 112, session 1, \n                       trial 110. Th
## [1] "Audiofiles and Items dont fit in subject 120, session 3, \n                       trial 130. Th
```

There are four trials with missing data -> Technical errors!

Bind the dataframes together

```
df <- bind_rows(datafiles)

# Check whether the expected file length fits the actual file length
nrow(df) == 2*20*3*160
```

```
## [1] TRUE
```

Combine data frame with audio files

```
# Bind VOT files into one
for(i in 1:length(datafiles_vot)) {
  datafiles_vot[[i]]$error <- as.character(datafiles_vot[[i]]$error)
  datafiles_vot[[i]]$correct <- as.character(datafiles_vot[[i]]$correct)
}
vot <- bind_rows(datafiles_vot)

vot %>% group_by(subject, session) %>% count()
```

```
## # A tibble: 120 x 3
## # Groups:   subject, session [120]
##    subject session     n
##      <dbl>   <dbl> <int>
##  1     101       1   164
##  2     101       2   164
##  3     101       3   164
##  4     102       1   164
##  5     102       2   164
##  6     102       3   164
##  7     103       1   164
##  8     103       2   164
##  9     103       3   164
## 10     104       1   164
## # i 110 more rows
```

```
# delete practice trials (4 per subject)
sum(stringr::str_detect(vot$File_name, "PA0"), na.rm=T) == 2*20*3*4
```

```
## [1] TRUE
```

```
vot %>% filter(!stringr::str_detect(vot$File_name, "PA0")) -> vot

# fix participant ID
for(i in 1:nrow(vot)){
  x <- strsplit(vot$name[i], ".", fixed=TRUE)
  x <- strsplit(x[[1]][2], "/", fixed=TRUE)
  x <- stringr::str_sub(x[[1]][2], 1,6)
  vot$OR02_01[i] <- toupper(x)
}
df <- df %>% mutate(OR02_01 = toupper(OR02_01))
# Have both data frames the same participant ID?
unique(df$OR02_01) %in% unique(vot$OR02_01)
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
unique(vot$OR02_01) %in% unique(df$OR02_01)
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
# Fix file name ending in both data frames
df %>% mutate(audio2=case_when(endsWith(audio, ".webm") ~
                                 stringr::str_sub(audio, 1, -18),
                               endsWith(audio, ".ogg") ~
                                 stringr::str_sub(audio, 1, -17))) -> df
vot$File_name2 <- stringr::str_sub(vot$File_name, 1, -17)
```

```r
# Fix Umlaute
vot$File_name2[vot$File_name2 == "Loewenzahn"] <- "Löwenzahn"
vot$File_name2[vot$File_name2 == "L^wenzahn"] <- "Löwenzahn"
vot$File_name2[vot$File_name2 == "Wuerfel"] <- "Würfel"
vot$File_name2[vot$File_name2 == "Muelleimer"] <- "Mülleimer"
vot$File_name2[vot$File_name2 == "Buerste"] <- "Bürste"
vot$File_name2[vot$File_name2 == "Saege"] <- "Säge"
vot$File_name2[vot$File_name2 == "Schluessel"] <- "Schlüssel"
vot$File_name2[vot$File_name2 == "Kopfhoerer"] <- "Kopfhörer"
vot$File_name2[vot$File_name2 == "Geschirrspueler"] <- "Geschirrspüler"
vot$File_name2[vot$File_name2 == "UBoot"] <- "U-Boot"
vot$File_name2[vot$File_name2 == "Kuehlschrank"] <- "Kühlschrank"
vot$File_name2[vot$File_name2 == "Kaefig"] <- "Käfig"
vot$File_name2[vot$File_name2 == "Marienkaefer"] <- "Marienkäfer"
vot$File_name2[vot$File_name2 == "Marienk%fer"] <- "Marienkäfer"
vot$File_name2[vot$File_name2 == "Marienk%fer"] <- "Marienkäfer"
vot$File_name2[vot$File_name2 == "Loewe"] <- "Löwe"
vot$File_name2[vot$File_name2 == "Loeffel"] <- "Löffel"
vot$File_name2[vot$File_name2 == "Faecher"] <-  "Fächer"
vot$File_name2[vot$File_name2 == "F%cher"] <-  "Fächer"
vot$File_name2[vot$File_name2 == "F%cher"] <-  "Fächer"
vot$File_name2[vot$File_name2 == "Baer"] <- "Bär"

# Write VOT data into df
df$File_name <- NA
df$VOT<- NA
df$correct <- NA
df$AR <- NA
df$error <- NA
df$name <- NA
for(i in 1:nrow(vot)){
  df$File_name[df$OR02_01 == vot$OR02_01[i] & df$session == vot$session[i] &
           df$audio2 == vot$File_name2[i]] <- vot$File_name2[i]
  df$VOT[df$OR02_01 == vot$OR02_01[i] & df$session == vot$session[i] &
           df$audio2 == vot$File_name2[i]] <- vot$VOT[i]
  df$correct[df$OR02_01 == vot$OR02_01[i] & df$session == vot$session[i] &
           df$audio2 == vot$File_name2[i]] <- vot$correct[i]
  df$AR[df$OR02_01 == vot$OR02_01[i] & df$session == vot$session[i] &
           df$audio2 == vot$File_name2[i]] <- vot$AR[i]
  df$error[df$OR02_01 == vot$OR02_01[i] & df$session == vot$session[i] &
           df$audio2 == vot$File_name2[i]] <- vot$error[i]
   df$name[df$OR02_01 == vot$OR02_01[i] & df$session == vot$session[i] &
           df$audio2 == vot$File_name2[i]] <- vot$name[i]
}


# Check whether merging worked properly
sum(!is.na(df$File_name)) == nrow(vot)
```

```
## [1] TRUE
```

```r
  # all file names were transmitted to the big df
x <- df %>% filter(is.na(File_name))
```

```
  # four files are missing = technical errors
(x %>% select(subject, session, trial, item, audio, File_name) -> x)
```

```
##   subject session trial    item audio File_name
## 1     112       1    98 kirsche  <NA>      <NA>
## 2     112       1    99   biene  <NA>      <NA>
## 3     112       1   110    bein  <NA>      <NA>
## 4     120       3   130   birne  <NA>      <NA>
```

```
table(df$audio[is.na(df$File_name)])
```

```
## < table of extent 0 >
```

Roughly check participants' adherence to the experiment

Did all participants finish the experiment?

```
# did all participants finish the experiment?
for(i in 1:length(unique(df$subject))) {
  if(all(df$FINISHED[df$subject==unique(df$subject[i])] != 0) |
     all(df$LASTPAGE[df$subject==unique(df$subject[i])] != 32)){
      print(paste(i,":"))
  print("Experiment completed?")
  print(table(df$FINISHED[df$subject==unique(df$subject[i])])/160)
  print("What was the last experimental page reached?")
  print(table(df$LASTPAGE[df$subject==unique(df$subject[i])])/160)
  }
}
```

Make sure patient and control group matching worked fine

```
overview <- data.frame(PWA=unique(df$subject)[unique(df$subject)<200],
           PWA_codes=unique(df$OR02_01[df$subject < 200]),
           control=unique(df$subject)[unique(df$subject)>200],
           control_codes=unique(df$OR02_01[df$subject>200]))
# make sure the extraction worked
for(i in 1:nrow(overview)){
  if(all(df$OR02_01[df$subject==overview$PWA[i]] != overview$PWA_code[i])){
    print('error PWA')
  }
  if(all(df$OR02_01[df$subject==overview$control[i]] != overview$control_code[i])){
    print('error control')
  }
}
# add age and gender
for(i in 1:nrow(overview)){
  overview$PWA_gender[i] <- df$gender[df$subject == overview$PWA[i]][1]
  overview$control_gender[i] <- df$gender[df$subject == overview$control[i]][1]
  overview$PWA_age[i] <- df$age[df$subject == overview$PWA[i]][1]
  overview$control_age[i] <- df$age[df$subject == overview$control[i]][1]
}

# export
write.csv(df, here::here("data", "transient_data_files",
                         "matching_overview.csv"),
            row.names = FALSE)
```

Double-check time taken

```
range(as.numeric(as.character(df$TIME003))) # welcome page
```

```
## [1]    1 5261
```

```
range(as.numeric(as.character(df$TIME004))) # consent
```

```
## [1]    4 2045
```

```
range(as.numeric(as.character(df$TIME005))) # demographics
```

```
## [1]   8 305
```

```
range(as.numeric(as.character(df$TIME006))) # general instructions
```

```
## [1]    1 1937
```

```
range(as.numeric(as.character(df$TIME007)), na.rm=T) # array loading (not visible for participant)
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf

## Warning in max(x): no non-missing arguments to max; returning -Inf

## [1]  Inf -Inf
```

```
range(as.numeric(as.character(df$TIME008))) # familiarization instructions
```

```
## [1]    1 7053
```

```
range(as.numeric(as.character(df$TIME009)), na.rm=T) # familiarization
```

```
## [1]   29 1463
```

```
# Some participants already open the link prior to the experiment. Therefore, well only consider the ti
for(i in 1:nrow(df)){
  df$time_correct[i] <- sum(df[i,] %>%
        select(starts_with("TIME")), na.rm=T)/60
}

# Get an overview
df %>% group_by(subject, session) %>% count(time_correct)
```

```
## # A tibble: 121 x 4
## # Groups:   subject, session [120]
##    subject session time_correct     n
##      <dbl>   <dbl>        <dbl> <int>
##  1     101       1         25.1     1
##  2     101       1         25.6   159
##  3     101       2         29.1   160
##  4     101       3         24.0   160
##  5     102       1         23.2   160
##  6     102       2         27.6   160
##  7     102       3         20.0   160
##  8     103       1         29.0   160
##  9     103       2         20.2   160
## 10     103       3         21.6   160
## # i 111 more rows
```

```
df %>%
  summarise(m = mean(time_correct),
               sd = sd(time_correct),
               min = min(time_correct),
               max = max(time_correct))
```

```
##          m       sd      min      max
## 1 28.59683 21.48414 13.31381 170.7955
```

```
df %>% group_by(type) %>%
  summarise(m = mean(time_correct),
               sd = sd(time_correct),
               min = min(time_correct),
               max = max(time_correct))
```

```
## # A tibble: 2 x 5
##   type        m    sd   min   max
##   <chr>   <dbl> <dbl> <dbl> <dbl>
## 1 PWA      34.5  21.1  15.7  129.
## 2 control  22.7  20.3  13.3  171.
```

```
df %>% group_by(type, session) %>%
  summarise(m = mean(time_correct),
               sd = sd(time_correct),
               min = min(time_correct),
               max = max(time_correct))
```

```
## 'summarise()' has grouped output by 'type'. You can override using the
## '.groups' argument.
```

```
## # A tibble: 6 x 6
## # Groups:   type [2]
##   type    session     m    sd   min   max
##   <chr>     <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 PWA           1  40.4 23.3   21.9 106.
## 2 PWA           2  32.9 23.2   20.1 129.
## 3 PWA           3  30.1 13.8   15.7  76.3
## 4 control       1  29.2 32.9   14.5 171.
## 5 control       2  18.9  3.68  13.3  26.9
## 6 control       3  20.0  8.54  14.5  53.3
```

Calculate mean semantic similarity per group

```
### Semantic similarity
# install.packages("LSAfun")
# install.packages("Rmisc")
# install.packages("tidyverse")
# install.packages("here")
# install.packages("cowplot")
library(LSAfun)
```

```
## Loading required package: lsa

## Loading required package: SnowballC

## Loading required package: rgl
```

```
##  load dewak dataframe
load(here::here("data", "dewak",
  "dewak100k_lsa.rda"))


## Within each category, calculate pairwise semantic similarity and calculate mean and sd
df %>% group_by(category, item) %>% count() %>%
  select(category, item) -> stims
stims$min_cat <- NA
stims$max_cat <- NA
stims$mean_cat <- NA
stims$sd_cat <- NA
vectors <- data.frame(category=NA, stims1=NA, stims2=NA)
for(i in 1:length(unique(df$category))){
  subset <- stims %>% filter(category==unique(df$category)[i])
  subset2 <- rbind(subset[1,], subset[1,], subset[1,], subset[1,],
                   subset[2,], subset[2,], subset[2,],
                   subset[3,], subset[3,],
                   subset[4,])
  subset2$item2 <- c(subset$item[2], subset$item[3], subset$item[4], subset$item[5],
                     subset$item[3], subset$item[4], subset$item[5],
                     subset$item[4], subset$item[5],
                     subset$item[5])
  for(j in 1:nrow(subset2)){
      subset2$cosine[j] <-
    LSAfun::pairwise(subset2$item[j], subset2$item2[j],tvectors = dewak100k_lsa)
  }
  min <- min(subset2$cosine)
```

```r
  max <-
  mean <- mean(subset2$cosine)
  sd <- sd(subset2$cosine)
  stims$min_cat[stims$category==unique(df$category)[i]] <- min(subset2$cosine, na.rm=T)
  stims$max_cat[stims$category==unique(df$category)[i]] <- max(subset2$cosine, na.rm=T)
  stims$mean_cat[stims$category==unique(df$category)[i]] <- mean(subset2$cosine, na.rm=T)
  stims$sd_cat[stims$category==unique(df$category)[i]] <- sd(subset2$cosine, na.rm=T)
}
```

```
## Warning: Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
```

```
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
```

```
## Warning: Unknown or uninitialised column: ‘cosine‘.
```

```
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
```

```
## Warning in min(subset2$cosine, na.rm = T): no non-missing arguments to min;
## returning Inf
## Warning in max(subset2$cosine, na.rm = T): no non-missing arguments to max;
## returning -Inf

## Warning: Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.

## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)

## Warning: Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.
## Unknown or uninitialised column: ‘cosine‘.

## Note: not all elements in y were found in rownames(tvectors)

## Warning: Unknown or uninitialised column: ‘cosine‘.

## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
##
## Note: not all elements in y were found in rownames(tvectors)
##
## Note: not all elements in x were found in rownames(tvectors)
```

```
stims %>% group_by(category,mean_cat, sd_cat, min_cat, max_cat) %>% count()
```

```
## # A tibble: 25 x 6
## # Groups:   category, mean_cat, sd_cat, min_cat, max_cat [25]
##    category     mean_cat sd_cat min_cat max_cat     n
##    <chr>           <dbl>  <dbl>   <dbl>   <dbl> <int>
##  1 Aufbewahrung    0.380 0.186   0.0998   0.601     5
##  2 Bauernhof       0.544 0.180   0.270    0.772     5
##  3 Blumen          0.687 0.116   0.443    0.838     5
##  4 Büro            0.620 0.177   0.389    0.841     5
##  5 Filler          0.167 0.120   0.0723   0.398    40
##  6 Fische          0.604 0.0616  0.515    0.690     5
##  7 Gebäude         0.281 0.157   0.101    0.686     5
##  8 Gemüse          0.803 0.0858  0.682    0.960     5
##  9 Heimwerker      0.590 0.173   0.360    0.833     5
## 10 Huftiere        0.618 0.0971  0.511    0.833     5
## # i 15 more rows
```

```
# Average semantic similarity in experimental categories
paste0('Mean semantic similarity per category: ',
       round(mean(stims$mean_cat[stims$category != "Filler"], na.rm=T),3), " SD: ",
       round(mean(stims$sd_cat[stims$category != "Filler"], na.rm=T), 3),
       " (caution: Some stimuli are not in the dewak database)")
```

```
## [1] "Mean semantic similarity per category: 0.576 SD: 0.152 (caution: Some stimuli are not in the dew
```

```
# Average semantic similarity in fillers
paste0('Mean semantic similarity across fillers: ',
       round(mean(stims$mean_cat[stims$category == "Filler"], na.rm=T),3), " SD: ",
       round(mean(stims$sd_cat[stims$category == "Filler"], na.rm=T), 3),
       " (caution: Some stimuli are not in the dewak database)")
```

```
## [1] "Mean semantic similarity across fillers: 0.167 SD: 0.12 (caution: Some stimuli are not in the d
```

Export prepared data frame

Anonymize data frame

```
df <- df %>% select(-c("QUESTNNR", "STARTED", "OR01_01")) %>% #, "OR02_01")) %>%
  # drop unnecessary columns
  select(-c("os_system", "SD22_BID", "browser_automatic",
            "system_format", "screen_width", "screen_height",
            "questionnaire_width", starts_with("TIME0"),
            "FINISHED", "Q_VIEWER",
            "LASTPAGE", "MAXPAGE", "DEG_TIME"))
```

Reduce data frame to relevant columns

```
# head(df)
df %>% select(type, subject, session, trial, item, category, supercategory,
              VOT, correct, AR, error, gender, age,language, handedness,
              starts_with("CH"), array, comments, timetotal, time_correct,
              OR02_01) -> d
```

```
write.csv(d, here::here("data", "transient_data_files", output),
            row.names = FALSE)
write.csv(df, here::here("data", "transient_data_files",
                         "aphasia_final_complete.csv"),
            row.names = FALSE)
```