CD2-vocabularies-survey

Dorien Huijser

7 April 2021

About this document

In this R markdown document, we analyze input from the CID community (Consortium on Individual Development) on the vocabularies that we will use in the data portal as developed during the CD2 project (Connecting Data in Child Development).

We perform the following steps:

1. Set the parameters and load packages
2. Read in the data
3. Create mappings: lists of numbers and instrument names
4. Put the data from the Keyword question in a flat, usable format
5. Put the data from the Category priority question in a flat, usable format

Required files to run this script:
- Data file (located in `data/raw` as a .csv file) - For each cohort, a list of numbers, names and descriptions of the measures as used in the Qualtrics loop and merge blocks (located in `docs/instrumentnrs`)

0. Set the parameters

First, we will set the parameters.

```
# Files to be read in
datafile <- "../data/raw/CD2_vocabularies_20220407.csv"

# Cohorts: same order as instrument numbers files
cohorts <- c("TRAILS", "GenR", "RADAR", "LCID", "YOUth", "NTR")

categorynames <- c("Parenting",
                   "Physiology",
                   "Physical health",
                   "Mental health",
                   "Demographics",
                   "Personality",
                   "Cognition",
                   "Lifestyle",
                   "Life history",
                   "Social cognition and behavior",
                   "Social and emotional development")
```

Download and load the necessary packages:

```
#install.packages("data.table")
library(data.table)
```

```r
# install.package("tidyverse")
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x lubridate::second()  masks data.table::second()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
# install.packages("wordcloud2")
library(wordcloud2)

# install.packages("webshot")
library(webshot)
```

1. Read in the data file and select relevant data

To read in the data, we're using the `fread()` function from the `data.table` package, which is significantly faster than `read.csv()`, especially with a large file like the one in this survey. Additionally, we're filtering out irrelevant responses (previews, non-consented, no cohort provided) and automatic variables (e.g., "StartDate", "ResponseId"), and turning character variables into numeric where necessary.

```r
cd2survey <- fread(datafile,
                   na.strings = "")

# Skip the first 2 rows as they contain the question text
cd2survey <- cd2survey[-c(1, 2), ]

# Skip the lines that were just for testing
cd2survey <- cd2survey[-grep("preview",
                             cd2survey$DistributionChannel), ]

# Select only the responses where consent was provided
cd2survey <- cd2survey[cd2survey$Consent == "Yes",]
```

```r
# Select only the responses where Cohort is not NA
cd2survey <- cd2survey[!is.na(cd2survey$Cohort),]

# Make variables numeric to allow calculations
cd2survey$Duration <- as.numeric(cd2survey$`Duration (in seconds)`)
cd2survey$Progress <- as.numeric(cd2survey$Progress)

# Remove irrelevant variables
cd2survey <- cd2survey[,-c("StartDate","EndDate","Status",
                           "Duration (in seconds)", "Finished", "RecordedDate",
                           "ResponseId", "DistributionChannel", "UserLanguage")]
```

Structure of the data file

The data file consists of roughly 3 types of variables that are repeated for all cohorts and all measures in that cohort, using Qualtrics's Loop and Merge functionality:

1. Keywords question: which keywords would the respondent apply to the measure in question? Format: `[instrument_number]_[cohort]_Keywords`: Keywords string response separated by commas.
2. Category rankings: how would the respondent rank the categories provided for the measure in question? Format: `[instrument_number]_[cohort]_Cat_[category-number]`: the response is a number delineanating the priority given to the category.
3. Category custom categories: if the respondent thinks of another suitable category for the measure in question, they can provide up to 3 custom categories that are in turn ranked. Format: `[instrument_number]_[cohort]_Cat_1[2/3/4]_TEXT`: string indicating the custom category.

2. Data exploration

There are 64 respondents for this survey. It took them a median of 23.1 minutes to complete the survey (ranging from 0.5 to 4.17685×10^4 minutes). This large range is possibly due to the fact that not all respondents finished the survey: the completion rate is on average 66.4% (SD = 33.7).

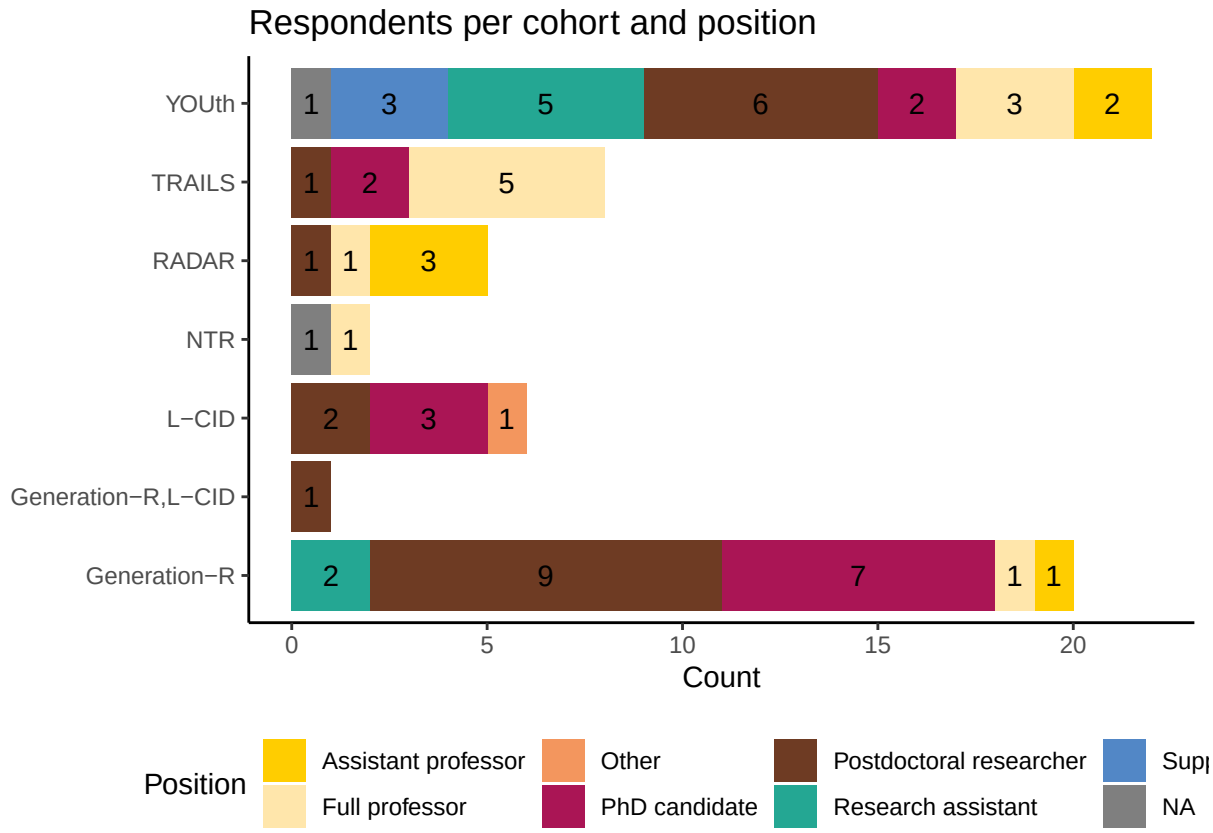
Below you can find a visualization of the amount of respondents for each cohort and their positions:

```r
# R color brewers palettes: http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/
# UU colors: https://www.uu.nl/organisatie/huisstijl/huisstijlelementen/kleur

UU_pallette <- c("#FFCD00",
                 "#FFE6AB",
                 "#F3965E",
                 "#AA1555",
                 "#6E3B23",
                 "#24A793",
                 "#5287C6",
                 "#001240",
                 "#5B2182")

ggplot(cd2survey, aes(x = Cohort, fill = Position)) +
  geom_bar(stat = "count") +
  stat_count(geom = "text",
             aes(label = after_stat(count)),
             position = position_stack(vjust=0.5)) +
  coord_flip() +
  theme_classic() +
  theme(legend.position = "bottom") +
```

```
  scale_fill_manual(values = UU_pallette) +
  labs(x = "", y = "Count", title = "Respondents per cohort and position")
```



Researchers with different expertises participated in the survey. Below you can find a word cloud of the expertises in the sample:

```
# URL: https://r-graph-gallery.com/196-the-wordcloud2-library.html

# Prereqs: separate all words in a list from the Expertise column, count each
# occurrence. Result should be: dataframe with columns word and freq
expertise <-
  cd2survey$Expertise[!is.na(cd2survey$Expertise)] %>%
  str_split(",") %>%
  unlist() %>%
  str_trim() %>%
  str_to_lower()

expertisecounts <- as.data.frame(sort(table(expertise), decreasing=TRUE))
wordcloudexpertise <- wordcloud2(data=expertisecounts, size=1.6, color=rep_len(UU_pallette, nrow(experti
            minRotation = 0, maxRotation = 0, rotateRatio = 1)

# Save the wordcloud as png
# Code copied from https://r-graph-gallery.com/196-the-wordcloud2-library.html#export
webshot::install_phantomjs(force = TRUE)
```

```
## Warning in utils::download.file(url, method = method, ...): the 'wininet'
## method is deprecated for http:// and https:// URLs
```

```
## phantomjs has been installed to C:\Users\4174259\AppData\Roaming\PhantomJS
```

```
library("htmlwidgets")
saveWidget(wordcloudexpertise,"tmp.html",selfcontained = T)
webshot("tmp.html","../assets/wordcloud_expertise.png", delay =5, vwidth = 480, vheight=480)
```


```
file.remove('tmp.html')
```

Export the email addresses for the prize competition:

```
emails <- cd2survey$Email[!is.na(cd2survey$Email)]
write.csv(emails, "../../cd2survey_emailaddresses.csv", row.names = FALSE)
```

3. Create reference lists

In order to make sense of the variable names and data, we need to make a mapping of the instrument numbers and names as they were used in Qualtrics.

```
# Instrument names and numbers
# Put all instrument number files in a list and name them correctly
instrumentnrsfiles <- c("../assets/instrumentnrs/TRAILS_instrumentnrs.csv",
                        "../assets/instrumentnrs/GenR_instrumentnrs.csv",
                        "../assets/instrumentnrs/RADAR_instrumentnrs.csv",
                        "../assets/instrumentnrs/LCID_instrumentnrs.csv",
                        "../assets/instrumentnrs/YOUth_instrumentnrs.csv",
                        "../assets/instrumentnrs/NTR_instrumentnrs.csv")

instrumentnrs <- lapply(instrumentnrsfiles, read.csv, sep= ";", header=TRUE)
names(instrumentnrs) <- paste0("instrumentnrs_", cohorts)
```

```
for (cohort in 1:length(instrumentnrs)) {
  # Turn into character (are automatically read in as factor)
  instrumentnrs[[cohort]]$Instrument_name <-
    as.character(instrumentnrs[[cohort]]$Instrument_name)
  instrumentnrs[[cohort]]$Instrument_description <- as.character(instrumentnrs[[cohort]]$Instrument_des

  # Add rows for which there is no instrument number, to be complete
  # This happens in the Qualtrics Loop & Merge if a row is accidentally removed.
  addrowlist <- list()
  counter <- 0

  for (i in 1:tail(instrumentnrs[[cohort]]$Instrument_nr, 1)) {
    # If there is no instrument numer corresponding to the loop index
    if (!i %in% instrumentnrs[[cohort]]$Instrument_nr) {
      counter <- counter + 1
      vect <- c(i, NA, NA)
      # Add the missing index number and NAs to the list that will be added to the instrument numbers l
      addrowlist[[counter]] <- vect
    }
  }

  addrows <- do.call("rbind", addrowlist) # bind the list elements together in a matrix
  addrowsdf <- data.frame(Instrument_nr = addrows[, 1], # Turn matrix into dataframe
                          Instrument_name = addrows[, 2],
                          Instrument_description=addrows[, 3])

  instrumentnrs[[cohort]] <- rbind(addrowsdf, instrumentnrs[[cohort]])
  instrumentnrs[[cohort]] <- instrumentnrs[[cohort]][order(instrumentnrs[[cohort]]$Instrument_nr), ]
}
```

4. The keywords question

This code is meant to summarize the raw data from the Keywords question, so that we can quickly see which keywords were used for each measure and how often, and how many people provided keywords for the measure. Concretely, the code below does the following:

- In part 1, we select the keywords variables for each cohort and count the occurrences of each keyword given by all participants for each instrument and each cohort separately. The keyword counts belonging to each cohort are saved in a variable called `keywords_[cohortname]_count`.
- In part 2, we save the list to a flat tabular format, with 1 column for the instrument name, 1 column for the keywords, 1 column for the frequencies of those keywords and 1 column with the number of respondents for that specific instrument. This is saved both in a separate variable for each cohort: `keywordcounts_[cohort]_flat` and in a combined list `keywordcounts_CD2`.

```
keywordcounts_CD2 <- vector(mode = "list", length = length(cohorts))
names(keywordcounts_CD2) <- cohorts

for (cohort in 1:length(cohorts)) {

  # PART 1: COUNT THE OCCURENCES OF EACH KEYWORD FOR EACH INSTRUMENT
  # Select all keyword variables of this cohort
  rawkeywords <- cd2survey[,grep(paste0(cohorts[cohort], "_Keywords"),
                                 names(cd2survey),
                                 value = TRUE),
```

```
                          with=FALSE]

# Create an empty list to put keyword counts for the current cohort (each list item = an instrument)
keywords_count_tempname <- vector(mode = "list",
                                  length = length(instrumentnrs[[cohort]][, 1]))

# For each instrument in the cohort, make a list item containing the keywords and their counts
for (instrument in 1:length(keywords_count_tempname)) {
  # Initiate empty vector variable
  keyword <- character(0)

  # Put the contents of each instrument in 1 combined vector with spaces removed (trimmed)
  if (!is.null(rawkeywords[[instrument]])) {
    keyword <- str_trim(unlist(str_split(rawkeywords[[instrument]], ","),
                               use.names = FALSE))
  }

  # Remove the NAs and -99s, lower case all
  keyword <- keyword[keyword != "-99" & !is.na(keyword)]
  keyword <- tolower(keyword)

  # For each instrument, count the number of occurrences of each unique keyword
  y <- as.data.frame(sort(table(keyword), decreasing=TRUE))

  # Put the keywords and counts in the count list
  keywords_count_tempname[[instrument]] <- y

  # Assign instrument names (not numbers) to the list items
  tempname <- instrumentnrs[[cohort]][instrumentnrs[[cohort]]$Instrument_nr == instrument,
                                      "Instrument_name"]
  names(keywords_count_tempname)[instrument] <- paste0(instrument,"_", tempname)
}

# Save the keyword counts for each cohort in a cohort-count-list
assign(paste0("keywords_", cohorts[cohort],"_count"), keywords_count_tempname)

# PART 2: SAVE TO USABLE FORMAT
# Initialize columns to put together in a dataframe
instrumentnames <- names(keywords_count_tempname)
instrumentkeywords <- character(0)
instrumentcounts <- character(0)
numberofrespondents <- numeric(0)

# Fill vectors with values for the flat dataframe
for (instrument in 1:length(keywords_count_tempname)) {
  instrumentkeywords[instrument] <- paste0(keywords_count_tempname[[instrument]]$keyword,
                                           collapse = ",", sep = "")
  instrumentcounts[instrument] <- paste0(keywords_count_tempname[[instrument]]$Freq,
                                         collapse = ",", sep = "")

  # Count the number of respondents for each instrument, ignoring NAs
  if (length(rawkeywords[[instrument]] != 0)) {
    instr <- rawkeywords[[instrument]]
```

```
      numberofrespondents[instrument] <- length(instr[!is.na(instr)])
    } else {
      numberofrespondents[instrument] <- NA
    }
  }

  keywordcounts_temp <- data.frame(Instrument_name=instrumentnames,
                                   Keywords = as.character(instrumentkeywords),
                                   Keyword_count = as.character(instrumentcounts),
                                   Keyword_respondents = numberofrespondents)

  # Remove the rows for which the Instrument_name is [#]_NA: these are nonexistent instruments
  if(sum(str_detect(keywordcounts_temp$Instrument_name,"_NA")) > 0){
    keywordcounts_temp <- keywordcounts_temp[-grep("[0-9]_NA",
                                                   keywordcounts_temp$Instrument_name), ]
  }

  # Save the counts for each cohort in a separate variable
  assign(paste0("keywordcounts_", cohorts[cohort], "_flat"), keywordcounts_temp)

  # Alternative: save each flat keywordcount dataframe in a place in the CD2-wide list
  keywordcounts_CD2[[cohort]] <- keywordcounts_temp
}
```

5. The Categories question

The Categories question is the most problematic question, because it results in 17 separate variables for each measure in each cohort. The aim of the code below is therefore to process all these variables into a few, more informative, variables for each measure, namely:

- which categories were selected for this measure (in order of priority score)?
- how often was each of the selected categories mentioned?
- how many respondents scored this measure?

Reasoning

The major issue for this question is how to fairly calculate an average priority score for each instrument. We probably need to apply weights to the options and then calculate the average priority score. Because respondents can choose how many categories they prioritise, this needs to happen for each respondent separately (besides also for each cohort and instrument separately). But also: each respondent's priorities should be considered equally valid: one priority should not weigh more than the other.

An example:

- Let's say participant 1 has given the instrument "CBCL" 2 priorities: nr 1 = Mental health, nr 2 = Social behavior. If we calculate their weights in absolute manner, Mental health gets 2 points and Social behavior gets 1.
- Let's say participant 2 has given the instrument "CBCL" 3 priorities: nr 1 = Mental health, nr 2 = Personality, nr 3 = Social behavior. In this case, Mental health gets 3 points, Personality gets 2 and Social behavior gets 1.

In this (absolute) example, the average score for Mental health is 3+2/2 = 2.5, for Social behavior: 1+1/2 = 1 and for Personality: 0+2/2 = 1. This is strange as Personality was only chosen by one of the participants, but it still receives the same score as Social behavior.

The example becomes different if we calculate **relative** priority scores:

- Participant 1: Mental health gets 2/(1+2) = 0.66 and Social behavior gets (1/1+2) = 0.33
- Participant 2: Mental health gets 3/(1+2+3) = 0.5, Personality gets 2/(1+2+3) = 0.33 and Social behavior gets 1/(1+2+3) = 0.167.
- Now let's say Participant 3 chooses both Mental health and Social behavior as 1, and Personality as 2. After reverse scoring, Mental health and Social behavior get 2 and Personality gets 1. Relatively, Mental health gets 2/(2+2+1) = 0.4, Social behavior gets 2/(2+2+1) = 0.4 and Personality gets 1/(2+2+1) = 0.2.

Now, the average score for Mental health is: (0.66+0.5+0.4)/3 = 0.52, for Social behavior: (0.33+0.167+0.4)/3 = 0.299 and for Personality: (0+0.33+0.2)/3 = 0.177. In the priorities score, it is reflected that the category Personality was only chosen by a subset of the participants and therefore gets a lower score.

The conclusion from participant 3 - someone giving the same priority to multiple categories - is that the **reverse** scores need to be summed, not the raw scores.

The code

The code below does the following for each cohort:

- In part 1, we put all priorities (text) questions belonging to 1 instrument in a list variable called `categoriesprior_raw`.
- In part 2, we calculate the relative priorities for each participant and instrument separately into a variable called `categoriesprior_rel`.
- In part 3, we calculate the average priorities for each instrument over all participants:
  - We first calculate the average priorities of the non-custom (already provided) categories
  - Then, we do the same for the custom (own-provided) categories
  - Finally, we combine the custom and non-custom average priority scores and save them to a usable, flat format similar to that of the keyword question.

Warning! This code is highly inefficient. It does work, but keep in mind that it will take some time and computational resources to run.

```
# Supress warnings
defaultW <- getOption("warn")
options(warn = -1)

catvar <- "Cat_"

for(cohort in 1:length(cohorts)){

  ### PART 1: PREP THE DATA IN A USABLE FORMAT (SEPARATED BY COHORT AND INSTRUMENT) ###

  # Create empty variables which will be filled in the for-loops
  categoriesprior_raw <- vector(mode = "list",
                                length = length(instrumentnrs[[cohort]][, 1]))
  customcategories <- categoriesprior_raw
  noncustomaverage <- categoriesprior_raw
  customcols <- categoriesprior_raw
  customaverage <- categoriesprior_raw
  averageprioritieslist <- categoriesprior_raw

  measurenames <- character(0)
  instrumentcategories <- character(0)
  priorityscores <- numeric(0)
  nrrespondents <- numeric(0)

  # For each instrument number in the cohort, make a list item
```

```r
for(listitem in 1:length(categoriesprior_raw)){

  # Save name of instrument in a vector for easier subsetting
  measurenames[listitem] <- instrumentnrs[[cohort]][instrumentnrs[[cohort]]$Instrument_nr == listitem
                                                    "Instrument_name"]

  print(paste0("Working on cohort: ", cohorts[cohort], ". Instrument: ", measurenames[listitem]))

  # Select the category variables from the raw dataset for each cohort and list item
  x <- cd2survey[, grep(paste0("^",
                               listitem, "_",
                               cohorts[cohort],
                               "_",
                               catvar),
                        names(cd2survey),
                        value = TRUE),
                 with=FALSE]

  # Put the selected variables in the ith place in the categoriesprior_raw list
  categoriesprior_raw[[listitem]] <- x

  # Assign a name to the dataframe in the list
  names(categoriesprior_raw)[listitem] <- paste0(listitem, "_",
                                                 measurenames[listitem])

  # Assign variable names to the dataframe in each list item that are easier to understand
  # For each column in the df in the list
  for(column in 1:dim(categoriesprior_raw[[listitem]])[2]){
    # If there are values in the column at all
    if(length(categoriesprior_raw[[listitem]][[column]]) != 0){
      # If the column is no TEXT column
      if(!grepl("TEXT",
                names(categoriesprior_raw[[listitem]])[column],
                fixed = TRUE)){
        # Turn character data into numeric to be able to calculate relative priorities later on
        categoriesprior_raw[[listitem]][[column]] <- as.numeric(categoriesprior_raw[[listitem]][[colu
      } # End if TEXT columns

      # Rename column in each list item's dataframe with category names
      # Cols 1-9 are named with the category name
      if(column <= length(categorynames)){
        names(categoriesprior_raw[[listitem]])[column] <- as.character(categorynames[column])
        # Cols 10 and over are named other_#
      } else if(column > length(categorynames)){
        names(categoriesprior_raw[[listitem]])[column] <- paste0("other_",
                                                                 sub(paste0(".*",
                                                                            cohorts[cohort],
                                                                            "_",
                                                                            catvar),
                                                                     "",
                                                                     names(categoriesprior_raw[[listi
      } # End column renaming if-else
    } # End if nr of cols
```

```
  } # End for each column

  # Put the TEXT variables (custom category names) in a separate list.
  textvars <- cd2survey[, grep(paste0("^",
                                      listitem,
                                      "_",
                                      cohorts[cohort],
                                      "_",
                                      catvar,
                                      "1[0-9]_TEXT"),
                               names(cd2survey),
                               value = TRUE),
                           with=FALSE]
  customcategories[[listitem]] <- textvars
  names(customcategories)[listitem] <- paste0(listitem, "_",
                                              measurenames[listitem])

  # Remove the TEXT variables from categoriesprior_raw for easier calculation of relative priorities
  categoriesprior_raw[[listitem]] <- categoriesprior_raw[[listitem]][,
                                                                     !grepl("TEXT", names(categoriespr
                                                                     with = FALSE]

} # End for each listitem in the cohort


### PART 2: CALCULATE RELATIVE PRIORITIES FOR EACH PARTICIPANT AND INSTRUMENT SEPARATELY ###
# Calculate the relative priorities and put them, for each instrument, in a dataframe in categoriespr
# Copy raw list to replace raw values with relative values later on
categoriesprior_rel <- categoriesprior_raw

# Empty vector for nr of participants for each measure
nrparticipants <- numeric(0)

# For each instrument
for(instrument in 1:length(categoriesprior_rel)){

  participantcount <- 0

  for(row in 1:dim(categoriesprior_rel[[instrument]])[1]){ # For each participant

    # Set -99 to NA, example: df[1][df[1]==2] <- NA
    subset <- categoriesprior_rel[[instrument]][row, ]
    subset[subset == -99] <- NA # Doesn't seem to work
    categoriesprior_rel[[instrument]][row, ][categoriesprior_rel[[instrument]][row, ] == -99] <- NA

    # If there are values in the column
    if(length(categoriesprior_rel[[instrument]] != 0)){
      # Determine how many categories have been scored
      cats_scored <- max(subset, na.rm = TRUE)

      # Calculate relative priorities only if not all values are NA
      if(cats_scored != -Inf){
        # Reverse the scoring and calculate the relative priority score
```

```
      reverse_scored <- ((cats_scored + 1) - subset)
      categoriesprior_rel[[instrument]][row, ] <- (reverse_scored)/sum(reverse_scored,
                                                                      na.rm = TRUE)
    } # End if not all values are NA

  } # End if there are values in the column

  # Count how many participants had to assess the instrument and put it
  # nrparticipants (for group average)
  if(length(na.omit(as.numeric(unlist(categoriesprior_rel[[instrument]][row, ])))) > 0){
    participantcount <- participantcount + 1
  }

} # End for each row in the instrument


### PART 3: CALCULATE AVERAGE PRIORITIES OVER ALL PARTICIPANTS ###
## PART 3A: CALCULATE AVERAGE PRIORITIES OF NON-CUSTOM CATEGORIES (I.E., THE GIVEN CATEGORIES)

#Initialize empty vectors for each instrument
averagepriority <- numeric(0)
columnnames <- character(0)
timesmentioned <- numeric(0)
nrparticipants[instrument] <- participantcount

# For each non-custom column in the instrument  (i.e., the first 11 variables)
for(column in 1:length(categoriesprior_rel[[instrument]])){
  # If there are any columns at all
  if(length(categoriesprior_rel[[instrument]] != 0)){
    # If colname does not contain "other"
    if(!grepl("other",names(categoriesprior_rel[[instrument]])[column])){

      # Calculate the average priority score for each column
      sumpriority <- sum(categoriesprior_rel[[instrument]][[column]],
                         na.rm=TRUE)

      # Divide the sumpriority scores by the total nr of people
      # who assessed the instrument, not the total nr of participants
      averagepriority[column] <- sumpriority/nrparticipants[instrument]

      columnnames[column] <- names(categoriesprior_rel[[instrument]])[column]
      timesmentioned[column] <- sum(!is.na(categoriesprior_rel[[instrument]][[column]]))
    } # end if colname does not contain other
  } # end if there are any columns at all
} # end for each column in the instrument

# Populate noncustomaverage list with dataframes containing priority scores
df <- data.frame(categoryname = columnnames,
                 averagepriority = averagepriority,
                 timesmentioned = timesmentioned)

noncustomaverage[[instrument]] <- df
# Assign usable names
```

```
  names(noncustomaverage)[instrument] <- paste0(instrument, "_",
                                                measurenames[instrument])


  ## PART 3B: CALCULATE AVERAGE PRIORITIES OF CUSTOM CATEGORIES
  ## (I.E., PROVIDED IN FREE TEXT, " OTHER" VARIABLES)

  # Add the 'other' columns from relative priorities list (scores)
  # to the customcategories (text) list
  # Find columns containing `other` and put them in a dataframe
  otherscores <- categoriesprior_rel[[instrument]][, grepl("other",
                                                           names(categoriesprior_rel[[instrument]])),
                                                    with = FALSE]

  # Combine the customcategories (text) and average priorities (numeric)
  # dataframes in the customcols list
  customcols[[instrument]] <- cbind(customcategories[[instrument]],
                                    otherscores)

  # Look for same category names in the dataframe and average them
  # If there are any columns at all
  if(length(customcols[[instrument]] != 0)){
    # For each column
    for(column in 1:length(customcols[[instrument]])){
      # Set -99 to NA, example: df[1][df[1]==2] <- NA
      subset <- customcols[[instrument]][[column]]
      subset[subset == "-99"] <- NA
      subset[subset == -99] <- NA
    } # end for each column

    # Select the text and number columns separately
    textcolumns <- customcols[[instrument]][, grepl("TEXT",
                                                    names(customcols[[instrument]])),
                                            with=FALSE]
    numbercolumns <- customcols[[instrument]][, grepl("[0-9]$",
                                                      names(customcols[[instrument]])),
                                              with=FALSE]

    # Unlist both the values and the text into a vector
    categories <- unlist(textcolumns)
    categories <- tolower(categories) # lower case strings

    priority <- unlist(numbercolumns)

    # Put the names and priorities into a dataframe
    noncustomdf <- data.frame(categories, priority)

    # For each unique categoryname, calculate the average priority and assign it to customaverage
    # Prepare variables and dataframe
    categoryname <- unique(noncustomdf$categories)
    averagepriority <- numeric(length(categoryname))
    timesmentioned <- numeric(length(categoryname))
    uniquesaverages <- data.frame(categoryname, averagepriority, timesmentioned)
```

```
    # Calculate the average priority for each unique category and put it into customaverage
    for(uniqueitem in 1:length(categoryname)){
      # Find all indices of the unique category
      indices <- which(noncustomdf$categories %in% categoryname[uniqueitem])

      # Calculate the average priority and the nr. of times each category is mentioned
      sumcustompriors <- sum(noncustomdf$priority[indices], na.rm=TRUE)

      # Divide by nr of people who assessed the instrument, not the total nr of participants
      uniquesaverages$averagepriority[uniqueitem] <- sumcustompriors/nrparticipants[instrument]
      uniquesaverages$timesmentioned[uniqueitem] <- length(indices)
    } # end for each unique item

    # Assign the averages dataframe to customaverage and
    customaverage[[instrument]] <- uniquesaverages
    # Assign usable list item names
    names(customaverage)[instrument] <- paste0(instrument, "_", measurenames[instrument])

  } # End if there are columns at all


  ## PART 3C: COMBINE THE CUSTOM AND NONCUSTOM PRIORITY SCORES IN 1 LIST ITEM AND SAVE TO USABLE FORM

  averageprioritieslist[[instrument]] <- rbind(noncustomaverage[[instrument]], customaverage[[instrume
  # Assign usable names
  names(averageprioritieslist)[instrument] <- paste0(instrument, "_", measurenames[instrument])

  # Sort each dataframe on averagepriority and round average priority to 2 decimals
  averageprioritieslist[[instrument]] <- averageprioritieslist[[instrument]][order(averageprioritieslis
                                                                                   decreasing=TRUE),]
  averageprioritieslist[[instrument]]$averagepriority <- round(averageprioritieslist[[instrument]]$ave

  # Remove rows in the dataframe when a category is mentioned 0 times:
  # Find indices of rows where timementioned == 0 and remove them
  zeroindices <- which(averageprioritieslist[[instrument]]$timesmentioned %in% 0)
  averageprioritieslist[[instrument]] <- averageprioritieslist[[instrument]][-zeroindices, ]

  # Fill vectors with values for the flat dataframe
  instrumentcategories[instrument] <- paste0(averageprioritieslist[[instrument]]$categoryname,
                                             collapse=",", sep="")
  priorityscores[instrument] <- paste0(averageprioritieslist[[instrument]]$averagepriority,
                                       collapse=",", sep="")
  nrrespondents[instrument] <- paste0(averageprioritieslist[[instrument]]$timesmentioned,
                                      collapse=",", sep="")

} # End for each instrument

categories <- data.frame(Instrument_name=names(averageprioritieslist),
                         Categories = as.character(instrumentcategories),
                         Priority_scores = as.character(priorityscores),
                         Times_mentioned = nrrespondents,
                         Respondents = nrparticipants)
```

```
  # Remove the rows for which the Instrument_name is [#]_NA: these are nonexistent instruments
  if(sum(str_detect(categories$Instrument_name, "_NA")) > 0){
    categories <- categories[-grep("[0-9]_NA", categories$Instrument_name), ]
  }

  # Save the relative priorities and non-custom categories for each cohort separately
  assign(paste0("categories_", cohorts[cohort]), categories)

} # End for all cohorts
print("Done!")

# Turn on warnings again
options(warn = defaultW)
```

Combine the keyword and category output into 1 sheet

For easier exporting and studying the results, here we combine the output from the keyword and category question for each cohort into 1 variable that we then export as a .csv file.

```
# Combine the variables into 1 dataframe for each cohort
results_TRAILS <- merge(x = keywordcounts_TRAILS_flat,
                        y = categories_TRAILS,
                        by = "Instrument_name",
                        all = TRUE)
results_TRAILS$Cohort <- rep("TRAILS", dim(results_TRAILS)[1])


results_GenR <- merge(x = keywordcounts_GenR_flat,
                      y = categories_GenR,
                      by = "Instrument_name",
                      all = TRUE)
results_GenR$Cohort <- rep("GenR", dim(results_GenR)[1])


results_RADAR <- merge(x = keywordcounts_RADAR_flat,
                       y = categories_RADAR,
                       by = "Instrument_name",
                       all = TRUE)
results_RADAR$Cohort <- rep("RADAR", dim(results_RADAR)[1])

results_LCID <- merge(x = keywordcounts_LCID_flat,
                      y = categories_LCID,
                      by = "Instrument_name",
                      all = TRUE)
results_LCID$Cohort <- rep("LCID", dim(results_LCID)[1])


results_YOUth <- merge(x = keywordcounts_YOUth_flat,
                       y = categories_YOUth,
                       by = "Instrument_name",
                       all = TRUE)
results_YOUth$Cohort <- rep("YOUth", dim(results_YOUth)[1])

results_NTR <- merge(x = keywordcounts_NTR_flat,
```

```
                      y = categories_NTR,
                      by = "Instrument_name",
                      all = TRUE)
results_NTR$Cohort <- rep("NTR", dim(results_NTR)[1])


todaysdate <- as.character(Sys.Date())
results_folder <- file.path(paste0("../data/processed/",
                                   todaysdate,
                                   "_results/"))
dir.create(results_folder)
```

```
## Warning in dir.create(results_folder): '..\data\processed\2023-08-22_results'
## already exists
```

```
# Write each dataframe to a .csv file
write.csv(results_TRAILS, file.path(paste0(results_folder, "/", todaysdate,
                                           "_results_TRAILS.csv")), row.names = TRUE)
write.csv(results_GenR, file.path(paste0(results_folder, "/", todaysdate,
                                         "_results_GenR.csv")), row.names = TRUE)
write.csv(results_RADAR, file.path(paste0(results_folder, "/", todaysdate,
                                          "_results_RADAR.csv")), row.names = TRUE)
write.csv(results_LCID,file.path(paste0(results_folder, "/", todaysdate,
                                        "_results_LCID.csv")), row.names = TRUE)
write.csv(results_YOUth,file.path(paste0(results_folder, "/", todaysdate,
                                         "_results_YOUth.csv")), row.names = TRUE)
write.csv(results_NTR,file.path(paste0(results_folder, "/", todaysdate,
                                       "_results_NTR.csv")), row.names = TRUE)

# Combine all cohorts in 1 file (delete instrument nrs for easier analysis)
results_all <- rbind(results_TRAILS,results_GenR, results_RADAR,
                     results_LCID, results_YOUth, results_NTR)

results_all$Instrument_name <- str_replace_all(as.character(results_all$Instrument_name),
                                               "^[0-9]+_", "")

write.csv(results_all, file.path(paste0(results_folder, "/", todaysdate,
                                         "_results_all.csv")),row.names = TRUE)
```