Census data wrangling

This document will first go over combining all of the census datasets. Then it will go over removing test answers from the venue surveys.

Census data

To begin let's read all the files.

```
# Reading all the census data file names from the data directory
file_names <- list.files("../../input/data_raw/census_data/")
file_names <- paste0("../../input/data_raw/census_data/", file_names)
kable(file_names)
```

x
../../input/data_raw/census_data/Audience+Survey+-+DE_October+25,+2024_12.15.sav
../../input/data_raw/census_data/Audience+Survey+-+FN_October+25,+2024_12.14.sav
../../input/data_raw/census_data/Audience+Survey+-+LT_October+25,+2024_12.14.sav
../../input/data_raw/census_data/Audience+Survey+-+UA_October+25,+2024_12.14.sav

```
# Reading all the census files using read_surveys function from retroharmonize
census_data <- read_surveys(file_names, .f = "read_spss")
```

```
# The German data has data from two different cities so first the data
# will be split into two
census_data <- append(census_data,
                      census_data[[1]] %>%
                        filter(Q24 == 1) %>%
                        list())
census_data <- append(census_data,
```

```
                      census_data[[1]] %>%
                        filter(Q24 == 2) %>%
                        list())
# Removing the combined German data
census_data <- census_data[-1]

# Adding city name as a column to every dataframe
city_names <- c("Helsinki", "Vilnius", "Lviv", "Mannheim", "Heidelberg")
census_data <- lapply(seq_along(census_data), function(i) {
    census_data[[i]] %>%
        add_column(City = city_names[i])
})

# Naming the list components
names(census_data) <- c("fi", "lt", "ua", "man", "heide")

# Let's now make sure that the columns of all the datasets
# match as much as possible
# Removing Heidelberg related questions from Mannheim data
census_data <- census_data %>%
  purrr::imap(~ if (.y == "man") .x %>%
                select(-c(Q23, Q24, Q25, Q26, starts_with("Q27"), Q28)) else .x)

# Removing Mannheim related questions from Heidelberg data
census_data <- census_data %>%
    purrr::imap(~ if (.y == "heide") .x %>%
                    select(-c(Q24, Q4, Q5, Q11, starts_with("Q17"), Q18)) else .x)


# Reading excel file that has harmonization instructions
heide_harmo <- readxl::read_excel("../../input/other/codebook.xlsx", sheet = 1)

# Harmonizing Heidelberg column names
census_data <- census_data %>%
  purrr::imap(~ if (.y == "heide") .x %>%
                renamefrom(cw_file = heide_harmo, raw = heide_raw,
                           clean = clean, keep_label = TRUE, drop_extra = FALSE)
              else .x)

# Changing Heidelberg and Mannheim 3/4 coding into 1/2 coding
# to match the other datasets
variable_harmo <- readxl::read_excel("../../input/other/codebook.xlsx", sheet = 2)
```

```r
# Creating helper function for harmonization
harmonize_variable <- function(harmo_data, variable_name, y, cities) {
  # Selecting the right rows from the harmonization data
  harmonization_data <- harmo_data %>%
    filter(variable == variable_name)

  # Extracting the values for harmonization
  from_values <- harmonization_data$from
  to_values <- harmonization_data$to
  values <- harmonization_data$value

  variable <- rlang::sym(variable_name)

  # Doing the harmonization
  y <- y %>%
      purrr::imap(~ if(.y %in% cities) .x %>%
                      mutate(!!variable :=
                          harmonize_values(.data[[variable]], harmonize_labels = list(
                          from = from_values,
                          to = to_values,
                          numeric_values = values
                      ), na_values = FALSE) %>%
                        labelled::labelled())
                  else .x)

  return(y)
}

# Changing the variable coding for Q4 and Q5 for Mannheim and Heidelberg
census_data <- harmonize_variable(variable_harmo, "Q4",
                                  census_data, c("man", "heide"))
census_data <- harmonize_variable(variable_harmo, "Q5",
                                  census_data, c("man", "heide"))

# Binding all the datasets together into one
census_data_all <- bind_rows(census_data) %>%
  select(-rowid)

# Changing variable labeling
var_labels <- readxl::read_excel("../../input/other/codebook.xlsx", sheet = 3)
census_data_all <- census_data_all %>%
    renamefrom(cw_file = var_labels, raw = variable, clean = variable, label = label,
```

```
                keep_label = TRUE, drop_extra = FALSE)

# Changing factor level labels
level_labels <- readxl::read_excel("../../input/other/codebook.xlsx", sheet = 4)
for (i in 1:nrow(level_labels)) {
  variable <- rlang::sym(level_labels$variable[i])
  census_data_all <- census_data_all %>%
    mutate(!!variable := harmonize_values(.data[[variable]],
                                                    harmonize_labels = list(
                            from = level_labels$from[i],
                            to = level_labels$to[i],
                            numeric_values = level_labels$value[i]
                          ), na_values = FALSE)
    )
}
```

```
# Removing test responses and responses with a lot of missing values)
# Identify test responses
census_data_all <- census_data_all %>%
  mutate(test = ifelse(as.Date(EndDate) < as.Date("2024-10-11"), 1, 0))

# Calculate age
census_data_all <- census_data_all %>%
  mutate(Q3 = as.numeric(Q3),  # Convert Q3 from labelled to numeric if needed
         Age = 2024 - Q3)

# Add age brakcets into the data
census_data_all <- census_data_all %>%
  mutate(
    Age_group = case_when(
          Age < 18 ~ "0-17",
          Age > 17 & Age < 30 ~ "18-29",
          Age > 29 & Age < 40 ~ "30-39",
          Age > 39 & Age < 50 ~ "40-49",
          Age > 49 & Age < 60 ~ "50-59",
          Age > 59 & Age < 70 ~ "60-69",
          Age > 69 ~ "70+"
    ),
    Age_group = factor(Age_group, levels = c(
      "0-17", "18-29", "30-39", "40-49", "50-59", "60-69", "70+"))
  )
```

```r
# Create variables to flag missing values
census_data_all <- census_data_all %>%
  # Create 'Q6_missing'
  mutate(Q6_missing = ifelse(rowSums(is.na(select(., Q6_1:Q6_5))) == 5, 1, 0)) %>%

  # Create 'Q8_missing'
  mutate(Q8_missing = ifelse(rowSums(is.na(select(., Q8_1:Q8_10))) == 10, 1, 0)) %>%

  # Create 'Q12_missing'
  mutate(Q12_missing = ifelse(rowSums(is.na(select(., Q12_1:Q12_12))) == 12, 1, 0)) %>%

  # Create 'Q15_missing'
  mutate(Q15_missing = ifelse(rowSums(is.na(select(., Q15_1:Q15_12))) == 12, 1, 0)) %>%

  # Create 'Q17_missing'
  mutate(Q17_missing = ifelse(rowSums(is.na(select(., Q17_1:Q17_11))) == 11, 1, 0))

# Create 'missing_all' variable = cases with missing values in ALL of the above questions
census_data_all <- census_data_all %>%
  mutate(missing_all = ifelse(Q6_missing == 1 & Q8_missing == 1 & Q12_missing == 1 &
                              Q15_missing == 1 & Q17_missing == 1, 1, 0))

# Identify bad responses = age over 99 missing values
census_data_all <- census_data_all %>%
  # Create 'bad' variable: 1 if age > 99 or missing_all == 1, otherwise 0
  mutate(bad = ifelse(Age > 99 | missing_all == 1, 1, 0))

# Filter out test and bad responses
census_data_all <- census_data_all %>%
  # Create 'exclude' variable: 1 if test == 1 or bad == 1, otherwise 0
  mutate(exclude = ifelse(test == 1 | bad == 1, 1, 0),
         # Create 'screenout' variable: 1 if exclude == 0, otherwise 0
         screenout = ifelse(exclude == 0, 1, 0))
# Dropping all the extra columns if needed
census_data_all <- census_data_all %>%
  select(-c(test, contains("missing"), bad))

# Getting a dataset without the bad observations
census_filtered <- census_data_all %>%
  filter(exclude == 0) %>%
  select(-c(exclude, screenout))
```

```r
# Remove useless columns
census_data_all <- census_data_all %>%
  select(-c(StartDate, EndDate, Status, Progress, Duration__in_seconds_, Finished, RecordedDa
# Saving the combined dataset as a .sav file
write_sav(census_data_all, "../../input/data_processed/combined_census_data.sav")
# write_sav(census_filtered,
#           paste0(getwd(), "/filtered_census_data.sac"))
```