Creating a Harmonized Cultural Access & Participation Dataset for Music

Daniel Antal

```
library(retroharmonize)
library(dplyr, quietly = TRUE)
cap_files <- c('ZA4529_v3-0-1.sav', 'ZA5688_v6-0-0.sav')
```

We are going to create a harmonized dataset from two Cultural Access & Participation datasets, which each contain harmonized surveys from a 2007 and 2013. We will further harmonize them into a single, longitudional file.

- European Commission (2012). Eurobarometer 67.1 (Feb-Mar 2007). GESIS Data Archive, Cologne. ZA4529 Data file Version 3.0.1, https://doi.org/10.4232/1.10983.
- European Commission, Brussels (2016). Eurobarometer 79.2 (2013). GESIS Data Archive, Cologne. ZA5688 Data file Version 6.0.0, https://doi.org/10.4232/1.12577.

To replicate this example, you must acquire these files from GESIS after accepting their terms of use. The `retroharmonize` project is not affiliated with GESIS.

This vignette article shows you how to replicate our dataset. The results can be found and referenced as: * Daniel Antal. (2022). Harmonized Cultural Access & Participation Dataset for Music (Version 20220129) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.5917742

- Daniel Antal. (2022). Creating a Harmonized Cultural Access & Participation Dataset for Music (Version 20220129) [Data set]. Zenodo. 10.5281/zenodo.5917714

```
## This is a dummy path. You should use the path where you saved these files.
gesis_dir <- file.path("C:", "your_data", "gesis_files")
```

```
source(retroharmonize::here("not_included", "daniel_env.R"))
```

```
## Create full paths to the three selected files
cap_files <- retroharmonize::here(gesis_dir, c('ZA4529_v3-0-1.sav', 'ZA5688_v6-0-0.sav'))
```

Import and Inventory

It is likely that the surveys will fit into your computer's memory, so you can omit the first step. If you work with many files, and you want to keep working sequentially with survey files, it is a good idea to convert them to R objects.

```
cap_surveys <- read_surveys(cap_files)
```

```
survey_doc <- document_surveys(survey_list = cap_surveys)
survey_doc
#> # A tibble: 2 x 5
#>   id           filename           ncol  nrow object_size
#>   <chr>        <chr>             <int> <int>       <dbl>
#> 1 ZA4529_v3-0-1 ZA4529_v3-0-1.sav   842 27746   190821336
#> 2 ZA5688_v6-0-0 ZA5688_v6-0-0.sav   719 27563   162098912
```

The files are rather large. We have `sum(document_cap_files$nrow)` of `sum(document_cap_files$ncol)` variables. We will only harmonize a few of them.

Concepts to harmonize

```
cap_metadata <- metadata_create(survey_list = cap_surveys)
```

Harmonization of Variables

Weights

From the metadata description, we select the post-stratification weight variables and projection weights.

```
weight_variables <- cap_metadata %>%
  filter ( .data$var_name_orig %in% c("isocntry", "wex", "wextra", "v47", "v7", "w1") |
             .data$var_label_orig %in% c("w_1_weight_result_from_target",
                                         "w_3_weight_special_germany",
                                         "weight_result_from_traget_united_germany",
                                         "w_4_weight_special_united_kingdom",
                                         "weight_result_from_traget_united_kingdom"))
```

A *schema crosswalk* is a table that shows equivalent elements (or "fields") in more than one structured data source. With `crosswalk_table_create()` we create an empty schema crosswalk, then we fill up its values. Researchers who feel more comfortable working in a spreadsheet application can create a similar crosswalk table in Excel, Numbers, or OpenOffice, and import the data from a `csv` or any tabular file.

```
## See https://github.com/rOpenGov/retroharmonize/issues/21
weigthing_crosswalk_table <- crosswalk_table_create(
  ## Create an empty schema crosswalk for the weight variables
  weight_variables
  ) %>%
  mutate (
    # Define the new, harmonized variable names
    var_name_target = case_when (
      # grepl("weight_result_from_target", .data$val_label_target)  ~ "w1",  [this is the issue]
      .data$var_name_orig  %in% c("wex", "wextra", "v47")        ~ 'wex',
      .data$var_name_orig  %in% c("w1", "v8")                    ~ "w1",
      .data$var_name_orig %in% c("w3a", "v12")                   ~ "w_de",
      .data$var_name_orig %in% c("w4a", "v10")                   ~ "w_uk",
      .data$var_name_orig == "rowid"  ~ 'rowid',  # do not forget to keep the unique row IDs
      TRUE ~ "geo"),
    # Define the target R class for working with these variables.
    class_target = ifelse(.data$var_name_target %in% c("geo", "v47"), "factor", "numeric")
    ) %>%
  select (
    -all_of(c("val_numeric_orig", "val_numeric_target", "val_label_orig", "val_label_target"))
    )
```

The crosswalk table contains the original (source) variable names that must be converted to the target variable names, i.e. `v7` and `isocntry` to `geo` and `v47`, `wextra` to `wex`. The weights should remain numeric variables, and the `geo` variable should be a categorical factor variable.

```
weigthing_crosswalk_table
#> # A tibble: 10 x 6
#>    id            filename  var_name_orig var_name_target class_orig class_target
```

```
#>    <chr>          <chr>     <chr>          <chr>           <chr>      <chr>
#>  1 ZA4529_v3-0-1 ZA4529_v~ v7             geo             character  factor
#>  2 ZA4529_v3-0-1 ZA4529_v~ v8             w1              numeric    numeric
#>  3 ZA4529_v3-0-1 ZA4529_v~ v10            w_uk            numeric    numeric
#>  4 ZA4529_v3-0-1 ZA4529_v~ v12            w_de            numeric    numeric
#>  5 ZA4529_v3-0-1 ZA4529_v~ v47            wex             numeric    numeric
#>  6 ZA5688_v6-0-0 ZA5688_v~ isocntry       geo             character  factor
#>  7 ZA5688_v6-0-0 ZA5688_v~ w1             w1              numeric    numeric
#>  8 ZA5688_v6-0-0 ZA5688_v~ wextra         wex             numeric    numeric
#>  9 ZA5688_v6-0-0 ZA5688_v~ w3a            w_de            numeric    numeric
#> 10 ZA5688_v6-0-0 ZA5688_v~ w4a            w_uk            numeric    numeric
```

The Eurobarometer surveys contains separate samples for the former West Germany (`DE-W`), East Germany (`DE-E`), Northern Ireland (`GB-NIR`), and Great Britain, i.e. England, Scotland and Wales (`GB-GBN`). For the variable `geo` the appropriate post-stratification weight is the `w1` variable. For the two German subsamples, it is `w_de`, and for the two United Kingdom subsamples it is the `w_uk`. We create two new variables, the `country_code` and `w`. When the two UK and the two German samples are joined, the appropriate post-stratification weight is `w`.

```
weight_vars <- crosswalk(survey_list = cap_surveys,
                         crosswalk_table =  weigthing_crosswalk_table,
                         na_values = NULL)
```

```
set.seed(2022)
weight_vars %>% sample_n(6)
#> # A tibble: 6 x 7
#>   id            rowid               geo       w1  w_uk  w_de    wex
#>   <chr>         <chr>               <fct>  <dbl> <dbl> <dbl>  <dbl>
#> 1 ZA5688_v6-0-0 ZA5688_v6-0-0_14673 GB-GBN 0.827  1.04 0.827 41413.
#> 2 ZA5688_v6-0-0 ZA5688_v6-0-0_7893  IT     1.23   1.23 1.23  62878.
#> 3 ZA5688_v6-0-0 ZA5688_v6-0-0_13922 DE-E   1.27   1.27 0.749 32132.
#> 4 ZA4529_v3-0-1 ZA4529_v3-0-1_2751  GR     0.758  0    0      6593.
#> 5 ZA5688_v6-0-0 ZA5688_v6-0-0_21882 LT     1.08   1.08 1.08   2963.
#> 6 ZA4529_v3-0-1 ZA4529_v3-0-1_10473 AT     1.16   0    0      7892.
```

You must use the `w1` weights when you work with the `geo` variable, which has separate values for `GB-GBN`, `GB-NIR`, `DE-E` and `DE-W`. Using the special country weights for the United Kindom and (united) Germany, we create the appropriate weight variable `w` to be used with ISO country codes, i.e. `GB` and `DE`.

```
weight_vars <- weight_vars %>%
  mutate ( country_code = substr(geo, 1,2),
           w = case_when (
             country_code == "DE" ~  w_de,  # Unified Germany post-stratification weight
             country_code == "GB" ~  w_uk,  # UK = Great Britain + Northern Ireland
             TRUE ~ w1 )) %>%
  mutate ( year_survey = case_when(
    .data$id == "ZA4529_v3-0-1" ~ '2007',
    .data$id == "ZA5688_v6-0-0" ~ '2013'
  )) %>%
  mutate ( year_survey = as.factor(.data$year_survey))
```

Beware that the country codes for the United Kingdom and for Greece follow the ISO standard, i.e. GB and GR, not the Eurostat country codes UK and EL.

```
weight_vars <- weight_vars %>%
  select (all_of(c("rowid", "country_code", "geo", "w", "w1", "wex", "id")))
```

```
set.seed(2022)
weight_vars %>% sample_n(6)
#> # A tibble: 6 x 7
#>   rowid               country_code geo        w    w1    wex id
#>   <chr>               <chr>        <fct>  <dbl> <dbl>  <dbl> <chr>
#> 1 ZA5688_v6-0-0_14673 GB           GB-GBN 1.04  0.827 41413. ZA5688_v6-0-0
#> 2 ZA5688_v6-0-0_7893  IT           IT     1.23  1.23  62878. ZA5688_v6-0-0
#> 3 ZA5688_v6-0-0_13922 DE           DE-E   0.749 1.27  32132. ZA5688_v6-0-0
#> 4 ZA4529_v3-0-1_2751  GR           GR     0.758 0.758  6593. ZA4529_v3-0-1
#> 5 ZA5688_v6-0-0_21882 LT           LT     1.08  1.08   2963. ZA5688_v6-0-0
#> 6 ZA4529_v3-0-1_10473 AT           AT     1.16  1.16   7892. ZA4529_v3-0-1
```

Demography

We will use two important demography variables: the age of the respondent, and the age of school leaving of the respondent. This latter, `age_education` variable is an important, *ex ante* harmonized variable in Eurobarometer. Because each European country has different education systems, furthermore, because education systems are not the same across different demographic groups (for example, people schooled before the World War II often went to very different schools), a more precise education level would require a very complicated mapping with education-specific knowledge.

The age_education variable has three special values: 1. the value for declined answers, which should be coded in a numeric representation to the special `NA` value or R;

2. An integer outside the valid range of responses, which means that the person is still tudying. We will recode the still studying answers to 0, and later we will further recode them to the current age of the person. We will also mark this special group as students—in their case, the `age_education` has a different meaning. A 17 year-old respondent in Europe is likely to study further; a middle aged person who replied with 17 has a low education level according to the current norms. In this case, a student's response of 17 is not the same as the middle aged person's;

3. And at last, the special value `No formal education` means that the person did not finish the primary school. Different countries define differently the minimum mandatory age when a person can leave the school system. We code these answers to 14, which is lower than the lowest age in our sample (15). This is also a special group, so we will mark them with a dummy variable, too.

```
demography_crosstable <- cap_metadata %>%
  filter ( .data$var_name_orig == "rowid" |
           .data$var_label_orig %in% c("age_exact", "age_education")) %>%
  crosswalk_table_create() %>%
  mutate ( var_name_target = case_when(
    # Do not leave out the unique row identifiers!
    .data$var_name_orig == "rowid" ~ "rowid",
    TRUE ~ .data$var_label_orig
  )) %>%
  mutate ( na_label_target = case_when(
    .data$na_label_orig %in% c("DK", "Refusal") ~ "Declined",
    TRUE ~ .data$na_label_target
  )) %>%
  mutate ( val_numeric_target = case_when(
    .data$val_label_orig == "No full-time education" ~ 14,
    .data$val_label_orig == "Still studying"         ~ 0,
    TRUE ~ .data$val_numeric_target
  )) %>%
  mutate ( na_numeric_target = case_when (
```

```
    .data$na_label_target == "Declined" ~ 99999,
    TRUE ~ NA_real_
  )) %>%
  mutate ( class_target = case_when (
    .data$var_name_target == "rowid" ~ 'character',
    TRUE ~ "numeric"
  ))
```

```
set.seed(123456)
demography_crosstable %>% sample_n(5)
#> # A tibble: 5 x 16
#>   id    filename var_name_orig var_name_target val_numeric_orig val_numeric_tar~
#>   <chr> <chr>    <chr>         <chr>                      <dbl>            <dbl>
#> 1 ZA56~ ZA5688_~ d8            age_education                  0                0
#> 2 ZA56~ ZA5688_~ d11           age_exact                     98               98
#> 3 ZA45~ ZA4529_~ rowid         rowid                         NA               NA
#> 4 ZA56~ ZA5688_~ d8            age_education                 98                0
#> 5 ZA45~ ZA4529_~ v727          age_exact                     15               15
#> # ... with 10 more variables: val_label_orig <chr>, val_label_target <chr>,
#> #   class_orig <chr>, class_target <chr>, na_label_orig <chr>,
#> #   na_label_target <chr>, na_numeric_orig <dbl>, na_numeric_target <dbl>,
#> #   var_label_orig <chr>, var_label_target <chr>
```

```
demography_vars <- crosswalk(survey_list     = cap_surveys,
                             crosswalk_table =  demography_crosstable,
                             na_values = NULL) %>%
  mutate ( is_student = ifelse ( .data$age_education == 0, 1, 0),
           age_education = ifelse (.data$age_education ==0,
                                   .data$age_exact,
                                   .data$age_education))
```

Now we have three variables, with a student dummy. You can test yourself if people with `age_educaton`=20 are a homogeneous group, or they should be treated as a still studying group who have already been educated up to 20 years, and a group who had already left eductation at the age of 20. If the two groups are homogeneous, then you will no longer need to use the `is_student` variable.

```
set.seed(1235)
demography_vars %>% sample_n(6)
#> # A tibble: 6 x 5
#>   id            rowid               age_education age_exact is_student
#>   <chr>         <chr>                       <dbl>     <dbl>      <dbl>
#> 1 ZA5688_v6-0-0 ZA5688_v6-0-0_6029             16        48          0
#> 2 ZA5688_v6-0-0 ZA5688_v6-0-0_13               22        53          0
#> 3 ZA5688_v6-0-0 ZA5688_v6-0-0_21100            18        18          1
#> 4 ZA4529_v3-0-1 ZA4529_v3-0-1_13916            16        33          0
#> 5 ZA5688_v6-0-0 ZA5688_v6-0-0_7834             21        37          0
#> 6 ZA5688_v6-0-0 ZA5688_v6-0-0_13482            18        32          0
```

Cultural Access & Participation Variables

Our variables of interest are visiting frequencies to concerts and public libraries. In the Eurobarometer CAP surveys, the answers have

1. `Never in the last twelve months` or `None` will be coded as 0 visits, and their factor label will be harmonized to `never`. In this case, the questionnaire reveals that whilst this was an *ex ante* harmonized

question, the answers were not consistently labelled in the three files. We can safely bring these responses to the same value
2. `1-2 times` will be coded to the programatically easier to use `1_2_times`, and it will get a numeric value of 1.5. Later, we will try to establish a better numeric value, now it is important that all responses labelled as `1-2 times` should have the same numeric code.
3. `3-5 times` will be coded to the numeric value of 3.5 and 3_5_times.
4. `More than 5 times` will be `more_than_5` and have the numeric value of 6.
5. The special case of `DK` will be coded as `declined` (to anwer) and the *Inap...* lables to `inap`. These are special codes in Eurobarometer meaning that the item is inappropriate, i.e. due to some filtering, this particular question was not asked from the respondant. While respondents who consciously decline an answer usually form a rather homogenous category of their own, the ianppropriate answers are truly missing answers. Their numeric representation will be both `NA` (f.e. they should be omitted from a numeric average.)

For code readability, we show the following data pipeline in sections, but you could of course create this code in a single expression—or create the resulting table in a spreadsheet application, if you are not aiming for full reproducibility in your research.

1. Let's create a search term for the metadata inventory:

```
search_term_labels <- c("rowid|cultural_activities_public_library|artistic_activities_sung|cultural_acti
")
```

2. Create an empty schema crosswalk table:

```
cap_vars_crosstable <- cap_metadata %>%
  filter ( .data$var_name_orig == "rowid" |
             grepl(search_term_labels, .data$var_label_orig) ) %>%
  crosswalk_table_create ()
```

3. Set the target variable names for the harmonization of the variables (columns) of the final, tidy, unified dataset.

```
cap_vars_crosstable <- cap_vars_crosstable %>%
  mutate ( var_name_target = case_when(
    .data$var_name_orig == "rowid" ~ "rowid",
    grepl("concert", .data$var_label_orig)       ~ "visit_concert",
    grepl("sung", .data$var_label_orig)          ~ "activity_sung",
    grepl("played_music", .data$var_label_orig)  ~ "activity_played_music",
    TRUE ~ .data$var_label_orig
  ))
```

4. Define the special (missing value) labels. It is important to convert any data with such labels as the R special `NA_real_` value when you give a numeric representation to your data:

```
cap_vars_crosstable <- cap_vars_crosstable %>%
  mutate ( na_label_target = case_when(
    .data$na_label_orig %in% c("DK", "Refusal") ~ "declined",
    grepl("^Inap", .data$na_label_orig)         ~ "inap",
    .data$val_label_orig == "DK"                ~ "declined",
    TRUE ~ .data$na_label_target
  )) %>%
  mutate ( na_numeric_target = case_when (
    .data$na_label_target == "declined" ~ 99999,
    .data$na_label_target == "inap"     ~ 99998,
    TRUE ~ NA_real_
  ))
```

5. Define the labels in the normal value range. These will be the factor levels in a categorical representation of your data:

```
cap_vars_crosstable <- cap_vars_crosstable %>%
  mutate ( val_label_target = case_when (
    tolower(.data$val_label_orig) == "mentioned"             ~ "mentioned",
    tolower(.data$val_label_orig) == "not mentioned"         ~ "not_mentioned",
    tolower(.data$val_label_orig) == "1-2 times"             ~ "1_2_times",
    tolower(.data$val_label_orig) == "3-5 times"             ~ "3_5_times",
    grepl("^more", tolower(.data$val_label_orig))            ~ "more_than_5",
    grepl("never|^not\\s\\in|none", tolower(.data$val_label_orig))  ~ "never",
    !is.na(.data$na_label_target) ~  .data$na_label_target,
    # do not forget the rowid, it should not be labelled
    .data$var_name_target == "rowid" ~ NA_character_,
    TRUE ~ "coding_error"   # If this appears in your scheme crosswalk table, you have a problem.
  ))
```

6. Define the numeric representation of your labels. These will be the factor levels in a numeric representation of your data:

```
cap_vars_crosstable <- cap_vars_crosstable %>%
  mutate ( val_numeric_target = case_when (
    .data$val_label_target == "never"         ~ 0,
    .data$val_label_target == "1_2_times"     ~ 1.5,
    .data$val_label_target == "3_5_times"     ~ 3.5,
    .data$val_label_target == "more_than_5"   ~ 6,
    .data$val_label_target == "mentioned"     ~ 1,
    .data$val_label_target == "not_mentioned" ~ 0,
    TRUE ~ .data$na_numeric_target
  ))
```

We create two versions of the crosswalk table. One will rename the `visit_concert` variable to `fct_visit`, and give them the categorical representation. The `crosswalk()` function will use the retroharmonize `as_factor` method instead of base R's `as.factor` to properly treat missing values.

```
cap_vars_crosstable_fct <- cap_vars_crosstable %>%
  mutate ( var_name_target = ifelse(
           test = grepl("^visit|^artistic", .data$var_name_target),
           yes  = paste0("fct_", .data$var_name_target),
           no   = .data$var_name_target),
           class_target = ifelse(
             test  = .data$var_name_target == 'rowid',
             yes = "character", # the rowid remains a character
             no  = "factor" )
           )

cap_vars_fct <- crosswalk(
  crosswalk_table = cap_vars_crosstable_fct,
  survey_list = cap_surveys)
```

The other will give these variable a numeric representation. The `crosswalk()` function will use the retroharmonize `as_numeric` method instead of base R's `as.numeric` to avoid unexpected coerction in the case of labelled missing cases.

```
cap_vars_crosstable_num <- cap_vars_crosstable %>%
   mutate( class_target = ifelse(
      test  = .data$var_name_target == 'rowid',
```

```
      yes = "character",  # the rowid remains a character
      no = "numeric" )
  )

cap_vars_num <- crosswalk(
  crosswalk_table = cap_vars_crosstable_num,
  survey_list = cap_surveys)
```

Join the harmonized CAP dataset

Using dplyr::left_join we join together by the survey `id` and the observation `rowid` the categorical and numeric representations of the CAP surveys, then add the demography variables, then add the weights.

```
harmonized_cap_file <- cap_vars_fct %>%
  left_join ( cap_vars_num,    by = c("id", "rowid") ) %>%
  left_join ( demography_vars, by = c("id", "rowid") ) %>%
  left_join ( weight_vars,     by = c("id", "rowid") )
```

Unit testing and corrections

The following simple logical tests should give an indication if our results are as expected.

Weights

The average post-stratification weight must be 1 for `w` and `w1`. The `wex` variable has no naturally defined mean value.

```
weight_vars %>%
  group_by ( .data$id ) %>%
  mutate ( across(starts_with("w"), function(x) ifelse(x==0, NA, x))) %>%
  dplyr::summarise( across(starts_with('w'), mean, na.rm=TRUE))
```

The post-stratification weights have a very small deviation from zero. Use these values when you calculate weighted averages. The projection weight should be used when you calculate sums.

Missing labels

All concert visits labelled as `declined` and `inap` must be coded to a numeric `NA` value. Furthermore, there must not be other `NA` values than cases in the `ZA6925_v1-0-0` survey, where this question was not asked. For consistency, these observations should be also coded as `inap` in the factor representation. We use dplyr for the final data wrangling, but of course, you can do this in DT or base R if you prefer.

```
harmonized_cap_file %>%
  select ( starts_with(c("id", "fct", "visit")) ) %>%
  filter ( is.na(.data$visit_concert) ) %>%
  distinct_all()
#> # A tibble: 3 x 3
#>   id            fct_visit_concert visit_concert
#>   <chr>         <fct>                     <dbl>
#> 1 ZA4529_v3-0-1 declined                     NA
#> 2 ZA4529_v3-0-1 inap                         NA
#> 3 ZA5688_v6-0-0 declined                     NA
```

Create a visiting binary variable

We create `is_visit_concert`, which takes the value of 1 if the person visited at least one concert in the previous 12 months, and 0 otherwise, retaining the missing values from declined answers.

```
harmonized_cap_file <- harmonized_cap_file %>%
  mutate ( is_visit_concert = ifelse(.data$visit_concert > 1, 1, .data$visit_concert))
```

And checking if it is indeed smaller than the original (estimated) visiting frequency value.

```
harmonized_cap_file %>%
  group_by ( .data$id) %>%
  dplyr::summarise ( across(starts_with("visit"),     mean, na.rm=TRUE),
                     across(starts_with("is_visit"),  mean, na.rm=TRUE),
                     .groups = "keep")
#> # A tibble: 2 x 3
#> # Groups:   id [2]
#>   id            visit_concert is_visit_concert
#>   <chr>                 <dbl>            <dbl>
#> 1 ZA4529_v3-0-1         1.02             0.402
#> 2 ZA5688_v6-0-0         0.937            0.383
```

Exporting

You can find this harmonized dataset on Zenodo in the Digital Music Observatory and the Cultural Creative Sectors Industries Data Observatory repositories. It was exported with the following code:

```
saveRDS(harmonized_cap_file,  file.path(tempdir(), "harmonized_cap_dataset.rds"))
write.csv(harmonized_cap_file,
          file.path(tempdir(), "harmonized_cap_file_20220129_doi_10_5281_zenodo_5917742.csv"),
          row.names = FALSE)
```

Daniel Antal. (2022). Harmonized Cultural Access & Participation Dataset for Music (Version 20220129) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.5917742

Daniel Antal. (2022). Creating a Harmonized Cultural Access & Participation Dataset for Music (Version 20220129) [Data set]. Zenodo. 10.5281/zenodo.5917714