GSS Cleanup

ML

11/3/2021

Set Up

Install the `gssr` package:

```
#remotes::install_github("kjhealy/gssr")
```

You do not have to reinstall the package so put a hashtag in front of the `remotes...` line in the chunk above.

You will have to load the package every time you want to use it. You load it just as you load any other package. We'll also load tidyverse to use its helper functions.

```
library(gssr)
```

```
## Package loaded. To attach the GSS data, type data(gss_all) at the console.
## For the codebook, type data(gss_doc). The gss_all and gss_doc objects will then be available to use.
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

GSS Codebook

Loading the package will give you two helpful messages. To attach the entire GSS dataset as a dataframe, use `data(gss_all)`. To attach just the codebook, use `data(gss_doc)`. We'll start with just the codebook.

```
data(gss_doc)
```

Now open the data spreadsheet view of the `gss_doc` dataframe. The id column lists the variable names. The description column is a simple summary of the question. The text column is the exact wording of the survey question. The properties and marginals columns will come in handy soon.

Use the filter option in the spreadsheet view to find the variable with the description related to subjective class identification. What is the variable name? What is the exact wording of the question?

Once you know the variable id, you can also get the value labels and the wording using the `filter` and `select` functions we have seen before.

```
gss_doc |>
  filter(id == "class") |>
  select(id, description, text)
```

```
## # A tibble: 1 x 3
##   id    description                    text
##   <chr> <chr>                          <chr>
## 1 class Subjective class identification 185a. If you were asked to use one of f~
```

You can pull multiple values at the same time. This could be helpful if you want to create your own codebook with all the variables in your study.

```
gss_doc |>
  filter(id == "class" | id == "degree") |>
  select(id, description, text)
```

```
## # A tibble: 2 x 3
##   id     description                    text
##   <chr>  <chr>                          <chr>
## 1 degree R's highest degree             19. If finished 9th-12th grade: Did yo~
## 2 class  Subjective class identification 185a. If you were asked to use one of ~
```

Let's find the values and the distribution for the class variable across all years.

```
gss_get_marginals("class", gss_doc)
```

```
## # A tibble: 9 x 6
##   variable percent     n value label         id
##   <chr>      <dbl> <int> <chr> <chr>         <chr>
## 1 class        6.3  3872 1     LOWER CLASS   CLASS
## 2 class       45.6 27968 2     WORKING CLASS CLASS
## 3 class       44.9 27519 3     MIDDLE CLASS  CLASS
## 4 class        3.2  1971 4     UPPER CLASS   CLASS
## 5 class        0       1 5     NO CLASS      CLASS
## 6 class       NA    3064 0     IAP           CLASS
## 7 class       NA     185 8     DK            CLASS
## 8 class       NA     234 9     <NA>          CLASS
## 9 class      100   64814 <NA>  Total         CLASS
```

You can see in the table above that values of 0, 8, and 9 are all NA. Values of 5 should probably also be NA, but that kind of decision is up to the researcher. If you just want a quick way to see the values that the GSS identifes with missing responses, access the question's properties.

```
gss_get_props("class", gss_doc)
```

```
## # A tibble: 3 x 4
##   variable property           value   id
##   <chr>    <chr>              <chr>   <chr>
## 1 class    Data type          numeric CLASS
## 2 class    Missing-data codes 0,8,9   CLASS
## 3 class    Record/column      1/929   CLASS
```

GSS Dataset

Now let's look at the data. The package automatically named the dataframe `gss_all` but we have to load it. It might take a little while to load since it's a big file.

```
data(gss_all)
```

Let's see in which years the `class` question was asked.

```
gss_which_years(gss_all, "class")
```

```
## # A tibble: 32 x 2
##     year class
##    <dbl> <lgl>
##  1  1972 TRUE
##  2  1973 TRUE
##  3  1974 TRUE
##  4  1975 TRUE
##  5  1976 TRUE
##  6  1977 TRUE
##  7  1978 TRUE
##  8  1980 TRUE
##  9  1982 TRUE
## 10  1983 TRUE
## # ... with 22 more rows
```

If the question were not asked in a year, we would see FALSE instead of TRUE. We do not need all the observations from every year between 1972 and 2018 for our example. Let's keep the observations from 2008 or 2018 in a new dataframe we will call `gss_subset`.

```
gss_subset <- gss_all |>
  filter(year == 2008 | year == 2018)
```

We'll work with only a few variables to keep things even more simple. Select these variable names: `id`, `year`, `hrs1`, `health`, `class`, and `sex`.

```
gss_subset <- gss_subset |>
  select(id, year, hrs1, health, class, sex)
```

Run a summary of this dataframe to see how the values are coded and if there are any missing values.

```
summary(gss_subset)
```

```
##        id             year           hrs1           health          class
##  Min.   :   1   Min.   :2008   Min.   : 1.00   Min.   :1.000   Min.   :1.000
##  1st Qu.: 547   1st Qu.:2008   1st Qu.:36.00   1st Qu.:2.000   1st Qu.:2.000
##  Median :1093   Median :2018   Median :40.00   Median :2.000   Median :2.000
##  Mean   :1099   Mean   :2013   Mean   :41.62   Mean   :2.086   Mean   :2.414
##  3rd Qu.:1640   3rd Qu.:2018   3rd Qu.:50.00   3rd Qu.:3.000   3rd Qu.:3.000
##  Max.   :2348   Max.   :2018   Max.   :89.00   Max.   :4.000   Max.   :4.000
##                                NA's   :1787    NA's   :1451    NA's   :31
##       sex
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.547
##  3rd Qu.:2.000
##  Max.   :2.000
##
```

It would be a good idea at this point to check the properties for all your variables as well to make sure that the NAs are correctly capturing all the missing values. Here's how to do that will multiple variables.

```
gss_get_props(c("id", "year", "hrs1", "health", "class", "sex"), gss_doc)
```

```
## # A tibble: 16 x 4
##    variable property           value    id
##    <chr>    <chr>              <chr>    <chr>
##  1 year     Data type          numeric  YEAR
##  2 year     Record/columns     1/1-4    YEAR
##  3 id       Data type          numeric  ID
##  4 id       Record/columns     1/5-8    ID
##  5 sex      Data type          numeric  SEX
##  6 sex      Missing-data code  0        SEX
##  7 sex      Record/column      1/297    SEX
##  8 hrs1     Data type          numeric  HRS1
##  9 hrs1     Missing-data codes -1,98,99 HRS1
## 10 hrs1     Record/columns     1/10-11  HRS1
## 11 health   Data type          numeric  HEALTH
## 12 health   Missing-data codes 0,8,9    HEALTH
## 13 health   Record/column      1/787    HEALTH
## 14 class    Data type          numeric  CLASS
## 15 class    Missing-data codes 0,8,9    CLASS
## 16 class    Record/column      1/929    CLASS
```

From the summary, it looks like we need to add variable labels for `health`, `class`, and `sex`. We already saw a way to find the labels associated with the values of class. Let's use that same function to get the labels for `health` and `sex` too.

```
gss_get_marginals(c("class", "health", "sex"), gss_doc)
```

```
## # A tibble: 20 x 6
```

```
##    variable percent     n value label         id
##    <chr>      <dbl> <int> <chr> <chr>         <chr>
##  1 sex         44.1 28614 1     MALE          SEX
##  2 sex         55.9 36200 2     FEMALE        SEX
##  3 sex        100   64814 <NA>  Total         SEX
##  4 health      29.8 14186 1     EXCELLENT     HEALTH
##  5 health      45.3 21559 2     GOOD          HEALTH
##  6 health      19.2  9123 3     FAIR          HEALTH
##  7 health       5.7  2722 4     POOR          HEALTH
##  8 health      NA   17099 0     IAP           HEALTH
##  9 health      NA      35 8     DK            HEALTH
## 10 health      NA      90 9     <NA>          HEALTH
## 11 health     100   64814 <NA>  Total         HEALTH
## 12 class        6.3  3872 1     LOWER CLASS   CLASS
## 13 class       45.6 27968 2     WORKING CLASS CLASS
## 14 class       44.9 27519 3     MIDDLE CLASS  CLASS
## 15 class        3.2  1971 4     UPPER CLASS   CLASS
## 16 class        0       1 5     NO CLASS      CLASS
## 17 class       NA    3064 0     IAP           CLASS
## 18 class       NA     185 8     DK            CLASS
## 19 class       NA     234 9     <NA>          CLASS
## 20 class      100   64814 <NA>  Total         CLASS
```

Use these values and labels to clean up the `sex` variable.

```
gss_subset <- gss_subset |>
  mutate(sex = factor(sex,
                      labels = c("Male", "Female")))
```

You will use the same tools to clean up the `class` and `health` variables in Assignment 6.

Finally, let's save the cleaned up subset to use it again without having to repeat all the above steps.

```
save(gss_subset, file = "../data/gss_cleanup_example.RData")
```

Note that we saved the file with the .Rdata extension rather than the .csv extension. The advantage of .Rdata is that this file type preserves the factor level ordering so you will not have to reassert it every time. To reopen this file, use `load()`.

```
load("../data/gss_cleanup_example.RData")
```