Code-along 04

FirstName LastName

Setup

Packages

Load the standard packages.

```
library(here)
library(tidyverse)
library(haven) # not core tidyverse
library(gssr)
library(gssrdoc)
library(summarytools)
```

GSS Panel Data: Download

https://gss.norc.org/get-the-data/stata

Download GSS 2016-2020 Panel (Release 1a, April 2022)

Save the **unzipped** file in your class `data` folder.

GSS Panel Data: Load

```
# Use here() to construct the file path
gss_panel.dta <- here("data", "GSS_2020_panel_stata_1a/gss2020panel_r1a.dta")

#load the data using `haven::read_dta()`
data <- read_dta(gss_panel.dta)
```

```
# Or, do both at the same time!
# data <- read_dta(here("data", "GSS_2020_panel_stata_1a/gss2020panel_r1a.dta"))
```

GSS 2016-2020 Panel Dataset

```
set.seed(815)  # Ensures you get the same sample every time

data |>
  select(yearid, starts_with("year_"), starts_with("age_")) |>
  slice_sample(n = 10)
```

```
# A tibble: 10 x 7
     yearid year_1a year_1b year_2 age_1a    age_1b    age_2
      <dbl>   <dbl>   <dbl>  <dbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
 1 20182183      NA    2018     NA NA(i)        52     NA(i)
 2 20180711      NA    2018     NA NA(i)        19     NA(i)
 3 20182189      NA    2018   2020 NA(i)        37        39
 4 20160354    2016      NA     NA    56     NA(i)     NA(i)
 5 20180452      NA    2018   2020 NA(i)        29        31
 6 20181503      NA    2018   2020 NA(i)        58        60
 7 20162744    2016      NA   2020    71     NA(i)        75
 8 20160315    2016      NA   2020    69     NA(i)        73
 9 20160170    2016      NA     NA    75     NA(i)     NA(i)
10 20161888    2016      NA     NA    71     NA(i)     NA(i)
```

Manipulating Dataframes

Selection helpers

Match variables according to a given pattern.

- `starts_with()`: Starts with an exact prefix.
- `ends_with()`: Ends with an exact suffix.
- `contains()`: Contains a literal string.
- ...

```r
my_data <- data |>
  select(yearid, wtssnr_2,
         starts_with("age_"),
         starts_with("family16_"),
         starts_with("socfrend_"),
         starts_with("childs_"))
```

```r
# You can supply multiple prefixes or suffixes.
my_data <- data |>
  select(yearid, wtssnr_2,
         starts_with(c("age_", "family16_", "socfrend", "childs"))
         )

my_data <- as_factor(my_data) # Apply labels to data
```

`head()` & `tail()`

Look at the first few column names and **first** few rows with `head()`.

```r
head(my_data, n = 5)
```

```
# A tibble: 5 x 14
    yearid wtssnr_2 age_1a age_1b age_2 family16_1a       family16_1b family16_2
     <dbl>    <dbl> <fct>  <fct>  <fct> <fct>             <fct>       <fct>
1 20160001    1.44  47     <NA>   51    both own mother ~ <NA>        not avail~
2 20160002    0.722 61     <NA>   65    both own mother ~ <NA>        not avail~
3 20160003   NA     72     <NA>   <NA>  both own mother ~ <NA>        iap
4 20160004    2.89  43     <NA>   47    mother only       <NA>        not avail~
5 20160005   NA     55     <NA>   <NA>  both own mother ~ <NA>        iap
# i 6 more variables: socfrend_1a <fct>, socfrend_1b <fct>, socfrend_2 <fct>,
#   childs_1a <fct>, childs_1b <fct>, childs_2 <fct>
```

Look at the first few column names and **last** few rows with `tail()`.

```r
# YOUR CODE HERE
```

Tidy data

pivot_longer()

```
my_data <- my_data |>
  pivot_longer(
    cols = 3:14,
    names_to = c(".value", "panel"),
    names_sep = "_")

head(my_data, n = 5)
```

```
# A tibble: 5 x 7
    yearid wtssnr_2 panel age   family16                    socfrend      childs
     <dbl>    <dbl> <chr> <fct> <fct>                       <fct>         <fct>
1 20160001    1.44  1a    47    both own mother and father  several time~ 3
2 20160001    1.44  1b    <NA>  <NA>                        <NA>          <NA>
3 20160001    1.44  2     51    not available for this year several time~ 3
4 20160002    0.722 1a    61    both own mother and father  several time~ 0
5 20160002    0.722 1b    <NA>  <NA>                        <NA>          <NA>
```

pivot_wider()

```
my_data |> # not overwriting my_data
  pivot_wider(                                                    ①
    names_from = panel,                                           ②
    values_from = c(-yearid, -wtssnr_2))                          ③

head(my_data, n = 5)
```

```
# A tibble: 5,215 x 17
     yearid wtssnr_2 panel_1a panel_1b panel_2 age_1a age_1b age_2 family16_1a
      <dbl>    <dbl> <chr>    <chr>    <chr>   <fct>  <fct>  <fct> <fct>
 1 20160001    1.44  1a       1b       2       47     <NA>   51    both own mot~
 2 20160002    0.722 1a       1b       2       61     <NA>   65    both own mot~
 3 20160003   NA     1a       1b       2       72     <NA>   <NA>  both own mot~
 4 20160004    2.89  1a       1b       2       43     <NA>   47    mother only
 5 20160005   NA     1a       1b       2       55     <NA>   <NA>  both own mot~
 6 20160006   NA     1a       1b       2       53     <NA>   <NA>  other
```

```
 7 20160007   NA     1a       1b       2       50     <NA>   <NA>  both own mot~
 8 20160008    1.44  1a       1b       2       23     <NA>   27    both own mot~
 9 20160009    1.44  1a       1b       2       45     <NA>   49    both own mot~
10 20160010   NA     1a       1b       2       71     <NA>   <NA>  both own mot~
# i 5,205 more rows
# i 8 more variables: family16_1b <fct>, family16_2 <fct>, socfrend_1a <fct>,
#   socfrend_1b <fct>, socfrend_2 <fct>, childs_1a <fct>, childs_1b <fct>,
#   childs_2 <fct>
# A tibble: 5 x 7
    yearid wtssnr_2 panel age   family16                    socfrend      childs
     <dbl>    <dbl> <chr> <fct> <fct>                       <fct>         <fct>
1 20160001    1.44  1a    47    both own mother and father  several time~ 3
2 20160001    1.44  1b    <NA>  <NA>                        <NA>          <NA>
3 20160001    1.44  2     51    not available for this year several time~ 3
4 20160002    0.722 1a    61    both own mother and father  several time~ 0
5 20160002    0.722 1b    <NA>  <NA>                        <NA>          <NA>
```

Recode the reshaped variable

Use `mutate()` and `case_when()` to recode the `panel` variable so 1a = 2016, 1b = 2018, and 2 = 2020.

Use `TRUE ~ NA_integer_` for missing values.

Then, use `head()` to check your results.

```
# YOUR CODE GOES HERE

head(my_data, n = 3)
```

```
# A tibble: 3 x 7
    yearid wtssnr_2 panel age   family16                    socfrend      childs
     <dbl>    <dbl> <chr> <fct> <fct>                       <fct>         <fct>
1 20160001     1.44 1a    47    both own mother and father  several time~ 3
2 20160001     1.44 1b    <NA>  <NA>                        <NA>          <NA>
3 20160001     1.44 2     51    not available for this year several time~ 3
```

`relocate()`

```r
my_data <- my_data |>
  relocate(panel)

head(my_data, n = 2)
```

```
# A tibble: 2 x 7
  panel   yearid wtssnr_2 age   family16                    socfrend       childs
  <chr>    <dbl>    <dbl> <fct> <fct>                       <fct>          <fct>
1 1a    20160001     1.44 47    both own mother and father several times~ 3
2 1b    20160001     1.44 <NA>  <NA>                        <NA>           <NA>
```

```r
my_data <- my_data |>
  relocate(panel, .after = yearid)

head(my_data, n = 2)
```

```
# A tibble: 2 x 7
    yearid panel wtssnr_2 age   family16                    socfrend       childs
     <dbl> <chr>    <dbl> <fct> <fct>                       <fct>          <fct>
1 20160001 1a        1.44 47    both own mother and father several times~ 3
2 20160001 1b        1.44 <NA>  <NA>                        <NA>           <NA>
```

arrange()

```r
my_data |>
  arrange(panel) |>
  select(yearid, panel, age) |>
  head(n = 5)
```

```
# A tibble: 5 x 3
    yearid panel age
     <dbl> <chr> <fct>
1 20160001 1a    47
2 20160002 1a    61
3 20160003 1a    72
4 20160004 1a    43
5 20160005 1a    55
```

```r
my_data |>
  arrange(desc(panel)) |>
  select(yearid, panel, age) |>
  head(n = 5)
```

```
# A tibble: 5 x 3
    yearid panel age
     <dbl> <chr> <fct>
1 20160001 2     51
2 20160002 2     65
3 20160003 2     <NA>
4 20160004 2     47
5 20160005 2     <NA>
```

Joining Dataframes

Example datasets

dataframe 1

```
  coupleid  name age
1        2  John  42
2        1 Megan  36
3        3   Bin  38
```

dataframe 2

```
  coupleid    name age
1        1     Sue  40
2        3  Ye-jin  39
3        2 Chrissy  35
```

dataframe 3

```
  coupleid marstat numchild country
1        3       1        1 S.Korea
2        1       0        0      US
3        2       1        4      US
```

append data with `bind_rows()`

```
df_all <- bind_rows(df_partner1, df_partner2)

tibble(df_all)
```

```
# A tibble: 6 x 3
  coupleid name      age
     <dbl> <chr>   <dbl>
1        2 John       42
2        1 Megan      36
3        3 Bin        38
4        1 Sue        40
5        3 Ye-jin     39
6        2 Chrissy    35
```

merge data with `left_join()`

```
df_couples <- left_join(df_partner1, df_family, by = "coupleid")

tibble(df_couples)
```

```
# A tibble: 3 x 6
  coupleid name    age marstat numchild country
     <dbl> <chr> <dbl>   <dbl>    <dbl> <chr>
1        2 John     42       1        4 US
2        1 Megan    36       0        0 US
3        3 Bin      38       1        1 S.Korea
```

Think Like a Statistician

Are married people above or below average in internet use or income? Does it vary by survey year?

Answering this research question takes a few steps. But, the first step is to create a dataframe with all the necessary information.

Step 1. Create a new, reshaped dataframe

- select the variables `yearid`, all `marital` variables, all `wwwhr` variables, and all `realrinc` variables
- pivot the `marital`, `wwwhr`, and `realrinc` variables longer (while keeping `yearid`)
- recode your `panel` variable so 1a = 2016, 1b = 2018, and 2 = 2020

Step 2. Create a summary dataframe

- remove rows with missing `wwwhr` or `realrinc` values
- group the data by your `panel` variable
- create a summary dataframe that contains the averages for `wwwhr` and `realrinc`

Step 3. Put the two dataframes together

- Use full_join to put your two new dataframes together

```
# TIP: It's often easier to play with your code in an R script first.
# Then, copy and past your working R code into this code-chunk
# and delete the `eval` statement when your code is fully working

# Reshape data
think_data <- data |>
    select() |>
    pivot_longer()

# recode variables
think_data <- think_data |>
  mutate()

## summarize by panel
think_summary <- think_data |>
  drop_na()  |>
  group_by() |>
  summarise()

think_full <- full_join()

think_full
```

Your Data Take

What's your conclusion to our initial research question?

NOTE: after you've created a `think_full` *dataframe with the appropriate variables, delete the* `echo` *and* `eval` *statements in each of the code blocks below to produce the necessary tables to answer the research question.*