Code-along 03

FirstName LastName

Setup

Packages

Load the standard packages.

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

Load your data & codebook

```r
# Get the data all survey years
data(gss_all)

# Load the codebook
data(gss_dict)
```

Data Management I

The pipe |>

The pipe operator passes what comes before it into the function that comes after it as the first argument in that function.

```r
sum(1, 2)
```

```
[1] 3
```

```r
1 |>
  sum(2)
```

```
[1] 3
```

`dplyr` grammar

What's the advantage of `dplyr` grammar? We can sequence data manipulation!

```r
gss_all |>
  select(year, sex, agekdbrn) |>
  filter(year == 2022) |>
  drop_na(sex, agekdbrn) |>
  group_by(sex) |>
  summarise(avg = mean(agekdbrn))
```

```
# A tibble: 2 x 2
  sex           avg
  <dbl+lbl>   <dbl>
1 1 [male]     26.3
2 2 [female]   23.7
```

`select()`, `filter()`, and `drop_na()`

Use `select()` to pick specific columns from your dataset.
Use `filter()` to keep rows that meet a condition.
Use `drop_na()` to remove rows with missing (NA) values.

```r
gss_all |>
  select(year, sex, agekdbrn) |>
  filter(year == 2022) |>
  drop_na(sex, agekdbrn) |>
```

```
Error in parse(text = input): <text>:6:0: unexpected end of input
4:   drop_na(sex, agekdbrn) |>
5:
  ^
```

group_by() and summarize()

Use group_by() to organize your data into groups based on one or more variables. Use summarize() to compute statistics like total, mean, or median for each group.

```
gss_all |>                                                                    ①
  select(year, sex, agekdbrn) |>                                              ②
  filter(year == 2022) |>                                                     ③
  drop_na(sex, agekdbrn) |>                                                   ④
  group_by(sex) |>                                                            ⑤
  summarise(freq = n())                                                       ⑥
```

① Start with the gss_all data frame:
② Keep only the variables in the dataset that we need.
③ Keep only the respondents from the 2022 survey
④ Remove any observations with missing data for our key variables
⑤ Do the next steps separately for each group in the variable
⑥ Creates a new data frame with one row for each combination of grouping variables

```
# A tibble: 2 x 2
  sex          freq
  <dbl+lbl>   <int>
1 1 [male]     1031
2 2 [female]   1363
```

dplyr() in action

Compare the average and median age at first childbirth for U.S. men and women in 2022.

```
gss_all |>
  select(year, sex, agekdbrn) |>
  filter(year == 2022) |>
  drop_na(sex, agekdbrn) |>
  group_by(sex) |>
  summarise(
    freq = n(),
    avg = mean(agekdbrn),
    med = median(agekdbrn)
    )
```

```
# A tibble: 2 x 4
  sex          freq   avg   med
  <dbl+lbl>   <int> <dbl> <dbl>
1 1 [male]     1031  26.3    25
2 2 [female]   1363  23.7    23
```

mutate() in action

Use mutate() to add new columns or change existing ones.

What proportion of new parents were teenagers (e.g., under 18 years old)?

```
gss_all |>                                                                    ①
  select(year, agekdbrn) |>
  filter(year == 2022) |>
  drop_na(agekdbrn) |>
  mutate(teen_parent = (agekdbrn < 18) * 1) |>
  summarise(proportion = mean(teen_parent))
```

```
# A tibble: 1 x 1
  proportion
       <dbl>
1     0.0773
```

Use case_when() inside mutate() to create values based on conditions.

What proportion of new parents had their first child as teenagers, in their 20s, 30s, or after age 40?

```
gss_all <-  gss_all |>
  mutate(age_groups = case_when(
    agekdbrn < 18 ~ "<18",
    agekdbrn >= 18 & agekdbrn <= 29 ~ "18-29",
    agekdbrn >= 30 & agekdbrn <= 39 ~ "30-39",
    agekdbrn >= 40 ~ "40+",
    TRUE ~ NA_character_)
  )

gss_all |>
  filter(year == 2022) |>
  freq(age_groups, report.nas = FALSE, headings = FALSE)
```

```
              Freq         %   % Cum.
----------- ------ --------- --------
        <18    186      7.73     7.73
      18-29   1704     70.82    78.55
      30-39    463     19.24    97.80
        40+     53      2.20   100.00
      Total   2406    100.00   100.00
```

Assignment operators

Let's make a tiny data frame to use as an example:

```
df <- tibble(x = c(1, 2, 3, 4, 5), y = c("a", "a", "b", "c", "c"))
df
```

```
# A tibble: 5 x 2
      x y
  <dbl> <chr>
1     1 a
2     2 a
3     3 b
4     4 c
5     5 c
```

```
df |>
  mutate(x = x * 2)
```

```
# A tibble: 5 x 2
      x y
  <dbl> <chr>
1     2 a
2     4 a
3     6 b
4     8 c
5    10 c
```

```
df
```

```
# A tibble: 5 x 2
      x y    
  <dbl> <chr>
1     1 a    
2     2 a    
3     3 b    
4     4 c    
5     5 c    
```

```r
 #| label: assignment

df <- df |>
   mutate(x = x * 2)
```

```r
df
```

```
# A tibble: 5 x 2
      x y    
  <dbl> <chr>
1     2 a    
2     4 a    
3     6 b    
4     8 c    
5    10 c    
```

Do something, save result, overwriting original

```r
df <- tibble(
  x = c(1, 2, 3, 4, 5),
  y = c("a", "a", "b", "c", "c")
)

df <- df |>
  mutate(x = x * 2)

df
```

```
# A tibble: 5 x 2
      x y    
  <dbl> <chr>
1     2 a    
```

```
2     4 a
3     6 b
4     8 c
5    10 c
```

Do something, save result, *not* overwriting original

```r
df <- tibble(
  x = c(1, 2, 3, 4, 5),
  y = c("a", "a", "b", "c", "c")
)

df_new <- df |>
  mutate(x = x * 2)

df_new
```

```
# A tibble: 5 x 2
      x y
  <dbl> <chr>
1     2 a
2     4 a
3     6 b
4     8 c
5    10 c
```

Do something, save result, overwriting original when you shouldn't

```r
df <- tibble(
  x = c(1, 2, 3, 4, 5),
  y = c("a", "a", "b", "c", "c")
)

df <- df |>
  group_by(y) |>
  summarize(mean_x = mean(x))

df
```

```
# A tibble: 3 x 2
  y     mean_x
  <chr>  <dbl>
1 a        1.5
2 b        3
3 c        4.5
```

Do something, save result, not overwriting original when you shouldn't

```r
df <- tibble(
  x = c(1, 2, 3, 4, 5),
  y = c("a", "a", "b", "c", "c")
)

df_summary <- df |>
  group_by(y) |>
  summarize(mean_x = mean(x))

df_summary
```

```
# A tibble: 3 x 2
  y     mean_x
  <chr>  <dbl>
1 a        1.5
2 b        3
3 c        4.5
```

Do something, save result, overwriting original data frame

```r
df <- tibble(
  x = c(1, 2, 3, 4, 5),
  y = c("a", "a", "b", "c", "c")
)
df <- df |>
  mutate(z = x + 2)
df
```

```
# A tibble: 5 x 3
      x y         z
  <dbl> <chr> <dbl>
1     1 a         3
2     2 a         4
3     3 b         5
```

```
4     4 c         6
5     5 c         7
```

Do something, save result, overwriting original column

```
df <- tibble(
  x = c(1, 2, 3, 4, 5),
  y = c("a", "a", "b", "c", "c")
)
df <- df |>
  mutate(x = x + 2)
df
```

```
# A tibble: 5 x 2
      x y
  <dbl> <chr>
1     3 a
2     4 a
3     5 b
4     6 c
5     7 c
```

Do something, save result, not overwriting original.

```
gss_all <-  gss_all |>
  mutate(age_groups = case_when(
    agekdbrn < 18 ~ "<18",
    agekdbrn >= 18 & agekdbrn <= 29 ~ "18-29",
    agekdbrn >= 30 & agekdbrn <= 39 ~ "30-39",
    agekdbrn >= 40 ~ "40+",
    TRUE ~ NA_character_)
  )

gss_all |>
  filter(year == 2022) |>
  freq(age_groups, report.nas = FALSE, headings = FALSE)
```

Do something and show me

```
gss_all |>
  select(year, agekdbrn) |>
  filter(year == 2022) |>
  drop_na(agekdbrn)  |>
  mutate(age_groups = case_when(
    agekdbrn < 18 ~ "<18",
    agekdbrn >= 18 & agekdbrn <= 29 ~ "18-29",
    agekdbrn >= 30 & agekdbrn <= 39 ~ "30-39",
    agekdbrn >= 40 ~ "40+",
    TRUE ~ NA_character_)) |>
  group_by(age_groups) |>
  summarise(
    count = n(),
    proportion = round(count / sum(count), 3)
  )
```

Summary Statistics

Median & Mode

Let's use dplyr grammar to find the median and mode for the `childs` variable.

```
gss_all$childs <- zap_missing(gss_all$childs)
gss_all$childs <- as_factor(gss_all$childs)
gss_all$childs <- droplevels(gss_all$childs)

gss_all |>                                        ①
  filter(year == 2024) |>
  freq(childs, report.nas = FALSE) |>             ②
  tb()                                            ③
```

① Use `dplyr` grammar, starting with the name of the df and a pipe
② Use the `freq()` function as usual
③ Add the `tb()` function to turn the table into a `tibble`

```
# A tibble: 9 x 4
  childs     freq   pct pct_cum
  <fct>     <dbl> <dbl>   <dbl>
1 0          1029 31.4     31.4
2 1           484 14.8     46.2
```

```
3 2             851 26.0       72.1
4 3             475 14.5       86.6
5 4             243  7.41      94.0
6 5              96  2.93      96.9
7 6              53  1.62      98.6
8 7              16  0.488     99.1
9 8 or more      31  0.946    100
```

Median & Mean

Let's use dplyr grammar to find the median and mean for the `hrs1` variable.

```
median(gss_all$hrs1, na.rm=TRUE)                                              ①
mean(gss_all$hrs1, na.rm=TRUE)

# show me summary statistics
summary(gss_all$hrs1)
```

① `na.rm` is a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.

```
[1] 40
[1] 41.11279
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00   37.00   40.00   41.11   48.00   89.00   32371
```

`descr()`

Univariate statistics for numerical data

```
gss_all |>
  filter(year == 2024) |>
  drop_na(hrs1)  |>
  descr(hrs1,
        stats = "common") |>                                                  ①
  tb()                                                                        ②
```

① Which stats to produce: "all" (default), "fivenum", "common" (see Details), or a selection. See ?descr

② Makes a tidy dataset out of `freq()` or `descr()` outputs.

```
# A tibble: 1 x 9
  variable  mean    sd   min   med   max n.valid     n pct.valid
  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>     <dbl>
1 year      39.4  13.9     0    40    89    1768  1768       100
```

`summarize()`

```r
gss_all |>
  select(year, hrs1, sex) |>
  filter(year == 2024) |>
  drop_na(hrs1, sex)  |>
  group_by(as_factor(sex)) |>
  summarise(
    count = n(),
    min = min(hrs1),
    median = median(hrs1),
    max = max(hrs1),
    mean = round(mean(hrs1), digits = 2),
    sd = sd(hrs1)
    )
```

```
# A tibble: 2 x 7
  `as_factor(sex)` count min       median max              mean    sd
  <fct>            <int> <dbl+lbl>  <dbl> <dbl+lbl>      <dbl> <dbl>
1 male               869 0             40 89 [89+ hours]   41.7  13.7
2 female             891 0             40 89 [89+ hours]   37.3  13.7
```

Think Like a Statistician

On average, in 2024, did parents with 4 or more kids work fewer hours for pay than other parents?

How do we find out?

- Make `childs` a numeric variable
- `filter()` the data to only 2024 respondents
- `select()` only the variables you need: `year`, `childs`, `hrs`
- use `mutate()` and `case_when()` to create a character variable with 4 categories: 1 child, 2 children, 3 children, 4 or more children (use `TRUE ~ NA_character_` for missing data)
- use `drop_by()` to remove missing data for your new variable and `hrs1`

- use `group_by()` to group the data by your new variable
- use `summarise()` to create count, mean, median, and sd summary statistics

```
# 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.
```

Your Data Take

What's your conclusion to our initial research question?