Code-along 02

Your Name Goes Here

Setup

Packages

Install the summarytools package, available on CRAN. Copy and paste the following code into your Console pane. Then hit enter.

```
install.packages("summarytools")
```

Load the standard packages and our new package `summarytools()`.

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

Load your data & codebook

```r
# Get the data only for the 2024 survey respondents
gss24 <- gss_get_yr(2024)

# Load the codebook
data(gss_dict)
```

Coding Basics

You can use R to do basic math calculations

```r
1 + 2
```

```
[1] 3
```

```r
2 * 5
```

```
[1] 10
```

```r
(1 + 2) / 2
```

```
[1] 1.5
```

You can create new objects with the assignment operator <-

```r
x <- 3 * 4
x
```

```
[1] 12
```

You can (and should) make comments in your code

```r
# R will ignore any text after # for that line

primes <- c(2, 3, 5, 7, 11, 13) # create vector of prime numbers
primes
```

```
[1]  2  3  5  7 11 13
```

Object names must start with a letter and can only contain letters, numbers, _, and .

```
i_use_snake_case
otherPeopleUseCamelCase
some.people.use.periods
And_aFew.People_RENOUNCEconvention
```

Demo:

```r
a <- 7
b <- 3
addition <- a + b
subtraction <- a - b
multiplication <- a * b
division <- a / b
exponentiation <- a^2
```

```r
a
```

```
[1] 7
```

```r
b
```

```
[1] 3
```

```r
addition
```

```
[1] 10
```

```r
subtraction
```

```
[1] 4
```

```r
multiplication
```

```
[1] 21
```

```r
division
```

```
[1] 2.333333
```

```r
exponentiation
```

```
[1] 49
```

Operators in R

Operators in R are symbols directing R to perform various kinds of mathematical, logical, and decision operations.

Comparison operators

```r
x <- 5
y <- 3
equal <- x == y
not_equal <- x != y
less_than <- x < y
more_than <- x > y
less_than_or_equal_to <- x <= y
more_than_or_equal_to <- x >= y
```

```r
x
```

```
[1] 5
```

```r
y
```

```
[1] 3
```

```r
equal
```

```
[1] FALSE
```

```r
not_equal
```

```
[1] TRUE
```

```r
less_than
```

```
[1] FALSE
```

```
more_than
```

```
[1] TRUE
```

```
less_than_or_equal_to
```

```
[1] FALSE
```

```
more_than_or_equal_to
```

```
[1] TRUE
```

Logical operators

```
x <- TRUE
y <- FALSE

and_operator <- x & y
or_operator <- x | y
not_operator <- !x
```

```
and_operator
```

```
[1] FALSE
```

```
or_operator
```

```
[1] TRUE
```

```
not_operator
```

```
[1] FALSE
```

Assignment operators

Make a tiny data frame and save it.

```r
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
```

Variable Types

Data types in R

A property is assigned to objects that determines how generic functions operate with it.

logical - Boolean values `TRUE` and `FALSE`

```r
class(TRUE)
```

```
[1] "logical"
```

character - character strings

```r
class("Sociology")
```

```
[1] "character"
```

Integer - numeric data without decimals
(indicated with an `L`).

```r
class(2L)
```

```
[1] "integer"
```

numeric - default type if values are numbers or if the values contain decimals.

```r
class(2.5)
```

```
[1] "numeric"
```

factors consist of character data with a fixed and known set of possible values

```r
opinion <- factor(c("like", "dislike", "dislike", "hate", "dislike", "hate"))
class(opinion)
```

```
[1] "factor"
```

```r
# By default, the levels are sorted alphabetically.
levels(opinion)
```

```
[1] "dislike" "hate"    "like"
```

```r
# Reorder the levels with the argument `levels` in the `factor()` function
opinion <- factor(opinion, levels = c("hate", "dislike", "like"))
levels(opinion)
```

```
[1] "hate"    "dislike" "like"
```

```r
# If the order has meaning (like rankings), you can make it an ordered factor
opinion <- factor(opinion, levels = c("hate", "dislike", "like"), ordered = TRUE)
levels(opinion)
```

```
[1] "hate"    "dislike" "like"
```

Converting between types

Use a function: `as.logical()`, `as.numeric()`, `as.integer()`, or `as.character()`.

Create a numeric variable.

```r
x <- 1:3
x
```

```
[1] 1 2 3
```

```r
class(x)
```

```
[1] "integer"
```

Change it to a character variable.

```r
y <- as.character(x)
y
```

```
[1] "1" "2" "3"
```

```r
class(y)
```

```
[1] "character"
```

Haven labelled

When you import data into R from software like SPSS, Stata, or SAS, you might notice a special class called `haven_labelled`.

```r
class(gss24$premarsx)
```

```
[1] "haven_labelled" "vctrs_vctr"     "double"
```

```r
table(gss24$premarsx)
```

```


   1    2    3    4 
 357  122  258 1378 
```

It makes data easier to understand without needing a separate codebook.

```r
attr(gss24$premarsx, "label")
```

```
[1] "Sex before marriage"
```

```r
print_labels(gss24$premarsx)
```

```

Labels:
 value                           label
     1                    always wrong
     2             almost always wrong
     3            wrong only sometimes
     4                not wrong at all
     5                           other
 NA(d)                      don't know
 NA(i)                             iap
 NA(j)               I don't have a job
 NA(m)                     dk, na, iap
 NA(n)                       no answer
 NA(p)                  not imputable
 NA(r)                         refused
 NA(s)                  skipped on web
 NA(u)                      uncodeable
 NA(x) not available in this release
 NA(y)    not available in this year
 NA(z)                    see codebook
```

You can use `as_factor` to see the value labels of the variable `premarsx`.

```r
table(as_factor(gss24$premarsx), useNA = "ifany")
```

```

         always wrong          almost always wrong
                  357                          122
 wrong only sometimes             not wrong at all
                  258                         1378
                other                          iap
                    0                         1126
           don't know           I don't have a job
                   50                            0
          dk, na, iap                    no answer
                    0                            6
        not imputable                      refused
                    0                            0
       skipped on web                   uncodeable
```

```
                             12                               0
not available in this release    not available in this year
                              0                               0
                   see codebook
                              0
```

Convert labels to factors

1. Get rid of all the 'missing' (NA) levels using `zap_missing`

```
gss24$premarsx <- zap_missing(gss24$premarsx)
table(as_factor(gss24$premarsx), useNA = "ifany")
```

```

        always wrong  almost always wrong wrong only sometimes
                 357                  122                  258
    not wrong at all                other                 <NA>
                1378                    0                 1194
```

2. Apply the labels instead of numeric values using `as_factor`

```
gss24$premarsx <- as_factor(gss24$premarsx) # replace the values with labels
table(gss24$premarsx, useNA = "ifany") # notice we didn't need to wrap the variable in as_fac
```

```

        always wrong  almost always wrong wrong only sometimes
                 357                  122                  258
    not wrong at all                other                 <NA>
                1378                    0                 1194
```

3. Get rid of the empty levels in `premarsx` using `droplevels`

```
gss24$premarsx <- droplevels(gss24$premarsx)
table(gss24$premarsx)
```

```

        always wrong  almost always wrong wrong only sometimes
                 357                  122                  258
    not wrong at all
                1378
```

Now do the same for the `sex` variable.

```r
gss24$sex <- zap_missing(gss24$sex)
gss24$sex <- as_factor(gss24$sex)
gss24$sex <- droplevels(gss24$sex)

table(gss24$sex)
```

```

  male female 
  1467   1823 
```

Look at variables

Make a frequency table of the variable `sex`. Then, do the same for `premarsx`.

```r
freq(gss24$sex)
```

```
Frequencies  
gss24$sex  
Type: Factor  

               Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
------------ ------ --------- -------------- --------- --------------
        male   1467     44.59          44.59     44.33          44.33
      female   1823     55.41         100.00     55.09          99.43
        <NA>     19                               0.57         100.00
       Total   3309    100.00         100.00    100.00         100.00
```

```r
freq(gss24$premarsx)
```

```
Frequencies  
gss24$premarsx  
Type: Factor  

                              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
--------------------------- ------ --------- -------------- --------- --------------
              always wrong    357     16.88          16.88     10.79          10.79
       almost always wrong    122      5.77          22.65      3.69          14.48
```

```
   wrong only sometimes    258     12.20            34.85      7.80            22.27
       not wrong at all   1378     65.15           100.00     41.64            63.92
                   <NA>   1194                               36.08           100.00
                  Total   3309    100.00           100.00    100.00           100.00
```

Using `report.nas = FALSE` suppresses the missing data.
The `headings = FALSE` parameter suppresses the heading section. Do the same for `premarsx`.

```
freq(gss24$sex, report.nas = FALSE, headings = FALSE)
```

```
               Freq        %   % Cum.
------------ ------ -------- --------
        male   1467    44.59    44.59
      female   1823    55.41   100.00
       Total   3290   100.00   100.00
```

```
freq(gss24$premarsx, report.nas = FALSE, headings = FALSE)
```

```
                             Freq        %   % Cum.
-------------------------- ------ -------- --------
             always wrong     357    16.88    16.88
      almost always wrong     122     5.77    22.65
     wrong only sometimes     258    12.20    34.85
         not wrong at all    1378    65.15   100.00
                    Total    2115   100.00   100.00
```

Cross-tabs

We've been using the `table()` function with one variable at a time, but it also let's you create a frequency table (**crosstab**) with two variables.

```
# 1st variable is the rows, 2nd variable is the columns.
table(gss24$premarsx, gss24$sex)
```

```
                        male female
```

```
  always wrong          146    209
  almost always wrong    44     77
  wrong only sometimes  127    130
  not wrong at all      616    758
```

To run `freq()` by group, pair it with the `stby()` function.

```
stby(gss24$premarsx, gss24$sex, freq)
```

```
NA detected in grouping variable(s); consider using useNA = TRUE

Frequencies
gss24$premarsx
Type: Factor
Group: sex = male

                             Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
-------------------------- ------ --------- -------------- --------- --------------
              always wrong    146     15.65          15.65      9.95           9.95
       almost always wrong     44      4.72          20.36      3.00          12.95
      wrong only sometimes    127     13.61          33.98      8.66          21.61
          not wrong at all    616     66.02         100.00     41.99          63.60
                      <NA>    534                              36.40         100.00
                     Total   1467    100.00         100.00    100.00         100.00

Group: sex = female

                             Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
-------------------------- ------ --------- -------------- --------- --------------
              always wrong    209     17.80          17.80     11.46          11.46
       almost always wrong     77      6.56          24.36      4.22          15.69
      wrong only sometimes    130     11.07          35.43      7.13          22.82
          not wrong at all    758     64.57         100.00     41.58          64.40
                      <NA>    649                              35.60         100.00
                     Total   1823    100.00         100.00    100.00         100.00
```

Use `summarytools::ctable` instead!

```
ctable(gss24$premarsx, gss24$sex,                                          ①
       prop = "c",                                                         ②
       format = "p",                                                       ③
       useNA = "no")                                                       ④
```

```
Cross-Tabulation, Column Proportions
premarsx * sex
Data Frame: gss24

---------------------- ----- -------------- --------------- ---------------
                         sex           male          female           Total
              premarsx
         always wrong          146 ( 15.6%)    209 ( 17.8%)    355 ( 16.8%)
  almost always wrong           44 (  4.7%)     77 (  6.6%)    121 (  5.7%)
 wrong only sometimes          127 ( 13.6%)    130 ( 11.1%)    257 ( 12.2%)
     not wrong at all          616 ( 66.0%)    758 ( 64.6%)   1374 ( 65.2%)
                Total          933 (100.0%)   1174 (100.0%)   2107 (100.0%)
---------------------- ----- -------------- --------------- ---------------
```

Check your knowledge

Based on your table:

- ***[your answer here]*** percentage of respondents believe sex before marriage is 'almost always wrong'?
- A greater percentage of ***[men or women]*** think sex before marriage is 'not wrong at all'.