# Code-along 02

FirstName LastName

## 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()`.

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

### Load your data & codebook

```
# 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:**

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

```
a
```

```
[1] 7
```

```
b
```

```
[1] 3
```

```
addition
```

```
[1] 10
```

```
subtraction
```

```
[1] 4
```

```
multiplication
```

```
[1] 21
```

```
division
```

```
[1] 2.333333
```

```
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
```

```r
more_than
```

```
[1] TRUE
```

```r
less_than_or_equal_to
```

```
[1] FALSE
```

```r
more_than_or_equal_to
```

```
[1] TRUE
```

### Logical operators

```r
x <- TRUE
y <- FALSE

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

```r
and_operator
```

```
[1] FALSE
```

```r
or_operator
```

```
[1] TRUE
```

```r
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
# REPLACE THE BLANK LINES WITH THE NAME OF THE VARIABLE

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

### Convert labels to factors

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

```r
# REPLACE THE BLANK LINES WITH THE NAME OF THE VARIABLE

gss24$premarsx <- zap_missing(gss24$________)

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

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

```
# REPLACE THE BLANK LINES WITH THE NAME OF THE VARIABLE

gss24$________ <- as_factor(gss24$________) # replaces the values with labels
table(gss24$________, useNA = "ifany") # no longer need to wrap the variable in as_factor
```

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

```
# REPLACE THE BLANK LINES WITH THE NAME OF THE VARIABLE

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

Now, use `zap_missing()`, `as_factor()`, and `droplevels()` to do the same for the `sex` variable.

Then, use `table()` to see your results.

```
# ADD YOUR CODE HERE
```

## Look at variables

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

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

```
Tagged NA values were detected and will be reported as regular NA; use haven::as_factor() to
```

```
Frequencies
gss24$sex
Label: Respondents sex
Type: Numeric (labelled)

                   Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------------- ------ --------- -------------- --------- --------------
         male [1]   1467     44.59          44.59     44.33          44.33
       female [2]   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
```

```
# ADD YOUR CODE HERE
```

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)
```

```
Tagged NA values were detected and will be reported as regular NA; use haven::as_factor() to
```

```
                    Freq        %    % Cum.
----------------- ------ --------- ---------
         male [1]   1467     44.59     44.59
       female [2]   1823     55.41    100.00
            Total   3290    100.00    100.00
```

```
# ADD YOUR CODE HERE
```

### 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)
```

```
      1   2
  1 146 209
  2  44  77
  3 127 130
  4 616 758
```

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
```

```
Tagged NA values were detected and will be reported as regular NA; use haven::as_factor() to
Tagged NA values were detected and will be reported as regular NA; use haven::as_factor() to


Frequencies
gss24$premarsx
Label: Sex before marriage
Type: Numeric (labelled)
Group: sex = 1

                                 Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
------------------------------ ------ --------- -------------- --------- --------------
              always wrong [1]    146     15.65          15.65      9.95           9.95
       almost always wrong [2]     44      4.72          20.36      3.00          12.95
      wrong only sometimes [3]    127     13.61          33.98      8.66          21.61
          not wrong at all [4]    616     66.02         100.00     41.99          63.60
                     other [5]      0      0.00         100.00      0.00          63.60
                          <NA>    534                              36.40         100.00
                         Total   1467    100.00         100.00    100.00         100.00

Group: sex = 2

                                 Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
------------------------------ ------ --------- -------------- --------- --------------
              always wrong [1]    209     17.80          17.80     11.46          11.46
       almost always wrong [2]     77      6.56          24.36      4.22          15.69
      wrong only sometimes [3]    130     11.07          35.43      7.13          22.82
          not wrong at all [4]    758     64.57         100.00     41.58          64.40
                     other [5]      0      0.00         100.00      0.00          64.40
                          <NA>    649                              35.60         100.00
                         Total   1823    100.00         100.00    100.00         100.00
```

Use summarytools::ctable instead!

```r
ctable(gss24$premarsx, gss24$sex,
       prop = "c",
       format = "p",
       useNA = "no")
```

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

```
---------- ----- -------------- --------------- ---------------
             sex              1               2           Total
  premarsx
         1          146 ( 15.6%)    209 ( 17.8%)    355 ( 16.8%)
         2           44 (  4.7%)     77 (  6.6%)    121 (  5.7%)
         3          127 ( 13.6%)    130 ( 11.1%)    257 ( 12.2%)
         4          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’.