If Statements

Contents

Warm-up: Gaining Comfort with Booleans

Predict the output of the following statements before running the code:

```
5 != 10
```

```
## [1] TRUE
```

Answer: TRUE

```
!is.na(NA)
```

```
## [1] FALSE
```

Answer: FALSE

```
4 %in% c(1, 2, 3)
```

```
## [1] FALSE
```

Answer: FALSE

```
TRUE | FALSE
```

```
## [1] TRUE
```

Answer: TRUE

```
4 > 3 & is.character("four")
```

```
## [1] TRUE
```

Answer: TRUE

```r
(1 == 1 | FALSE) & (-10 <= 0)
```

```
## [1] TRUE
```

Answer: TRUE

If it's not clear, try out each part in parentheses separately first.

Extension: Write a complicated boolean expression that returns `TRUE` in which you use "and", "or" and "not".

```r
# Example
(2+2==4)&(2>1)&(FALSE==FALSE)!=(FALSE==TRUE)|(FALSE)
```

```
## [1] TRUE
```

Adjust your previous expression slightly to return `FALSE`.

Vectorized booleans

We have seen that we can make vectorized comparisons. For example:

```r
c(1, 2, 3, 4, -5) > c(0, 0, 0, 0, 0)
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE
```

It returns the vector of `c(TRUE, TRUE, TRUE, TRUE, FALSE)`, because 1 through 4 are greater than 0 and -5 is not. It's equivalent to writing:

```r
# run this
c(1, 2, 3, 4, -5) > 0
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE
```

(This is the underlying idea that makes filter() work!)

What will the following code return? Predict and then test out:

```r
c(1, 2, 3, 4) == 4
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```r
c(1, 2, 3, 4) == c(2, 3, 4, 5)
```

```
## [1] FALSE FALSE FALSE FALSE
```

```r
c(TRUE, FALSE) | c(FALSE, TRUE)
```

```
## [1] TRUE TRUE
```

1. Fill in the blank to provide the desired result

```r
example_vec <- c(10, 3, 2, -1, -12)

# provide a number where the result is 5 TRUEs
example_vec <= c(12, 15, 19, 1, 10)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```r
# provide a vector where the result is 5 TRUEs
example_vec  == c(10, 3, 2, -1, -12)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```r
# provide a vector where the result alternates
# c(TRUE, FALSE, TRUE, FALSE, TRUE)
example_vec != c(10, 0, 2, 1, -12)
```

```
## [1] FALSE  TRUE FALSE  TRUE FALSE
```

```r
example_vec %in% c(10,0,2,0,-12)
```

```
## [1]  TRUE FALSE  TRUE FALSE  TRUE
```

Extension: Write an expression that returns the vector `c(FALSE, FALSE, FALSE, FALSE, FALSE)` using `truth <- c(NA, 2, -4, 4, 0.5)`.

```r
truth <- c(NA, 2, -4, 4, 0.5)
!is.na(truth) & truth > 5
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

Hint: You'll need to use `is.na()` along with an `&`

boolean expressions in `ifelse()`

`ifelse()` takes a vector of logicals (e.g. `my_vec %in% c(2, 3, 5, 7)`) in it's first position.

Guess the output before running the code

```r
my_vec <- 1:10

ifelse(my_vec < 7, "less than 7", "greater than or equal to 7")
```

```
##  [1] "less than 7"                "less than 7"               
##  [3] "less than 7"                "less than 7"               
##  [5] "less than 7"                "less than 7"               
##  [7] "greater than or equal to 7" "greater than or equal to 7"
##  [9] "greater than or equal to 7" "greater than or equal to 7"
```

```
ifelse(my_vec %in% c(2, 3, 5, 7), "prime", "not prime")
```

```
##  [1] "not prime" "prime"     "prime"     "not prime" "prime"     "not prime"
##  [7] "prime"     "not prime" "not prime" "not prime"
```

```
ifelse(my_vec %in% c(2, 3, 5, 7), my_vec, NA)
```

```
##  [1] NA  2  3 NA  5 NA  7 NA NA NA
```

You have a tibble as below and you want to add a column to your data called `is_prime` which is `TRUE` for primes and `FALSE` otherwise. Use `mutate()` with `ifelse()` to add that column.

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

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

```
my_vec <- 1:10

my_tibble <- tibble::tibble(numbers = my_vec)

my_tibble <- my_tibble %>%
  mutate(is_prime = ifelse(my_vec %in% c(2, 3, 5, 7), "prime", "not prime"))
```

Applying Booleans to the Survey of Household Economics Dynamics

We are going to explore the Survey of Household Economics Dynamics (SHED) by the Federal Reserve. The data file is somewhat large, so we are only providing the columns of interest for this workshop. The data for the workshop is available at `data/shed_data_abridged.dta`. The original data including a code book can be found here.

Note: You'll notice that this file uses a different file format than you might be used to—it's a .dta. These is the file format that Stata uses. While there has been a large shift towards languages like R and Python, Stata is still the most common languages used for applied economics work. The nice thing about R is that we can read any of these files easily. In particular, the `haven` package will do the trick!

Our work will focus on the following question: **Do financially stable respondents have higher levels of social trust than those facing instability?**

First, we read the data. Make sure we've loaded `haven` using the `library()` function (and throw in `tidyverse` since we'll use it too), then use the `read_dta()` function from it to read our file:

```r
library(tidyverse)
library(haven)
shed_data <- read_dta("data/shed_data_abridged.dta")
```

Start simple. What are the dimensions of the data? How many rows and columns?

```r
dim(shed_data)
```

```
## [1] 11316    14
```

Hint: Use `dim()`.

We cannot ask people directly if they are "financially stable". Rather, one survey question asks how people would pay for a unexpected $400 expense (e.g. a hospital bill). There are 9 options, captured in different columns as indicator variables (1 if answer is yes, 0 if answer is no).

Note: Indicator variables might also be called dummy variables.

The options are:

- EF3_a: Put it on my credit card and pay it off in full at the next statement
- EF3_b: Put it on my credit card and pay it off over time
- EF3_c: With the money currently in my checking/savings account or with cash
- EF3_d: Using money from a bank loan or line of credit
- EF3_e: By borrowing from a friend or family member
- EF3_f: Using a payday loan, deposit advance, or overdraft
- EF3_g: By selling something
- EF3_h: I wouldn't be able to pay for the expense right now
- EF3_i: Other
- EF3_Refused: Refused to answer

We could argue that people who can pay for an expense shock with cash or credit which they pay off in full are financially stable.

1. Use `mutate` and `ifelse` to add an indicator variable called `financially_stable` that is `1` if the respondent can pay in cash (`EF3_c`) or pay with a credit card which they'll pay off at the end of the month (`EF3_a`) and `0` otherwise. Name the resulting tibble `shed_with_financial_stability`.

```r
shed_with_financial_stability <- shed_data %>%
  mutate(financially_stable = ifelse(EF3_c == 1|EF3_a == 1, 1, 0))
```

Hint: If you need to check how `ifelse()` works, remember you can call up any function's documentation with `?`, i.e., `?ifelse`.

2. Use `group_by` and `summarize` to determine the number of financially stable people in the data. Alternatively, use `table()` or `count()`. Try each approach out!

```r
shed_with_financial_stability %>%
  group_by(financially_stable) %>%
  summarize(financially_stable_count = n())
```

```
## # A tibble: 2 x 2
##   financially_stable financially_stable_count
##                <dbl>                    <int>
## 1                  0                     3426
## 2                  1                     7890
```

How many people are financially stable in the data?

Answer: 7890.

```
table(shed_with_financial_stability$financially_stable)
```

```
## 
##    0    1 
## 3426 7890
```

```
shed_with_financial_stability %>% count(financially_stable)
```

```
## # A tibble: 2 x 2
##   financially_stable     n
##                <dbl> <int>
## 1                  0  3426
## 2                  1  7890
```

Our code assigns `0` to all people who refused to answer the question. It's preferable to label them as `NA`. Adjust the `financially_stable` variable to reflect this non-availablility of the data.

```
shed_with_financial_stability <- shed_data %>%
    mutate(financially_stable = ifelse(EF3_c == 1|EF3_a == 1, 1, 0),
           financially_stable = ifelse(EF3_Refused == 1, NA, financially_stable))
```

Hint: first create `financially_stable` as before, and then with a second line in the `mutate()` use an `ifelse(<BOOLEAN>, NA, financially_stable)`. The first line will create a vector of `1`'s and `0`'s, the second will keep the same `1`'s and `0`'s unless they refused to answer. If they refused to answer, their `0` will be replaced with`NA`

Note: Again, going back to this whole "many ways to do it" thing, you could have chained the `ifelse` calls like so:

```
shed_with_financial_stability <- shed_data %>%
    mutate(financially_stable = ifelse((EF3_c == 1|EF3_a == 1), 1,
                                       ifelse(EF3_Refused == 1, NA, 0)))
```

Adding Trust

We have quantified financial stability in our population in `shed_with_financial_stability`. Now we are ready to incorporate a measure of trust to address our research question.

Variable B11 captures response to the question: "And now a general question about trust. On a scale from zero to ten, where zero is not at all and ten is completely, in general how much do you trust most people?"

Let's start by finding the distribution of trust for the full population. Using any of the approaches mentioned when you counted how many people are financially stable, find the values of B11 and their distribution.

```r
shed_with_financial_stability %>%
  group_by(B11) %>%
  summarize(n = n())
```

```
## # A tibble: 12 x 2
##                    B11     n
##              <dbl+lbl> <int>
##  1 -1 [Refused]          20
##  2  0 [0 Not at all]    435
##  3  1                   331
##  4  2                   609
##  5  3                  1144
##  6  4                  1120
##  7  5                  2632
##  8  6                  1645
##  9  7                  2088
## 10  8                  1068
## 11  9                   176
## 12 10 [10 Completely]    48
```

```r
# Or
table(shed_with_financial_stability$B11)
```

```
##
##   -1    0    1    2    3    4    5    6    7    8    9   10
##   20  435  331  609 1144 1120 2632 1645 2088 1068  176   48
```

```r
# Or
shed_with_financial_stability %>% count(B11)
```

```
## # A tibble: 12 x 2
##                    B11     n
##              <dbl+lbl> <int>
##  1 -1 [Refused]          20
##  2  0 [0 Not at all]    435
##  3  1                   331
##  4  2                   609
##  5  3                  1144
##  6  4                  1120
##  7  5                  2632
##  8  6                  1645
##  9  7                  2088
## 10  8                  1068
## 11  9                   176
## 12 10 [10 Completely]    48
```

- What is the modal (most common) response?
- What do you think -1 means?

Answers: 5 and refused to answer.

Now, create a new indicator variable called `trusting` which is `1` when a respondent responds with 6 or higher to the question. Deal with refusal as you did before.

```
shed_with_financial_stability <- shed_with_financial_stability %>%
  mutate(trusting = ifelse(B11 >= 6, 1, 0),
         trusting = ifelse(B11 == -1, NA, trusting))
# Or
shed_with_financial_stability <- shed_with_financial_stability %>%
  mutate(trusting = ifelse(B11 >= 6, 1,
                           ifelse(B11 == -1, NA, 0)))
```

2) Use `group_by(financially_stable)` and summarize to find the proportion of trusting people by financial security group.

Hint: Recall that you have missing values and that they're sticky! Refer to the "Dealing with Missing Data" section of our "Reading Files" tutorial if you need a refresher.

```
shed_with_financial_stability %>%
  group_by(financially_stable) %>%
  summarize(proportion_trusting = mean(trusting, na.rm = TRUE))
```

```
## # A tibble: 3 x 2
##   financially_stable proportion_trusting
##                <dbl>               <dbl>
## 1                  0               0.299
## 2                  1               0.508
## 3                 NA               0.301
```

What proportion of "financially-stable" respondents were "trusting"?

Answer: 51%.

Extension

Our definition of financially stable could be more nuanced. Particularly, some of the possible responses indicate higher levels of financial instability such as those who require payday loans, selling things or say "I wouldn't be able to pay for the expense right now". Create an additional level to your financially stable variable and repeat the analysis. Does this affect your previous interpretation?