Accelerated TA Session 7: If statements and conditionals

Harris Coding Camp

Summer 2022

General Guidelines

You may encounter some functions we did not cover in the lectures. This will give you some practice on how to use a new function for the first time. You can try following steps:

1. Start by typing `?new_function` in your Console to open up the help page
2. Read the help page of this `new_function`. The description might be a bit technical for now. That's OK. Pay attention to the Usage and Arguments, especially the argument `x` or `x,y` (when two arguments are required)
3. At the bottom of the help page, there are a few examples. Run the first few lines to see how it works
4. Apply it in your questions

It is highly likely that you will encounter error messages while doing this exercise. Here are a few steps that might help get you through it:

1. Locate which line is causing this error first
2. Check if you have a typo in the code. Sometimes your group members can spot a typo faster than you.
3. If you enter the code without any typo, try googling the error message. Scroll through the top few links see if any of them helps
4. Try working on the next few questions while waiting for help by TAs

I. Warm-up Questions

1. Without running the code, predict what the output will be. Then, see if you were right by running the code in the console.

True or False

a. `TRUE | FALSE`
b. `TRUE | (FALSE & TRUE)`
c. `TRUE | (10 < 4)`
d. `TRUE | (1 & 0)`
e. `TRUE | (4 > pi & 3 < pi & exp(1) >= 3 & 1e6 < 2^30)`
f. `4 > 2 | 2 > 4`
g. `TRUE | NA`
h. What rule do these problems demonstrate?

True and False

a. `TRUE & FALSE`
b. `TRUE & (FALSE & TRUE)`
c. `TRUE & (10 < 4)`
d. `TRUE & (1 & 0)`
e. `TRUE & (4 > pi & 3 < pi & exp(1) >= 3 & 1e6 < 2^30)`
f. `4 > 2 & 2 > 4`
g. `NA & FALSE`
h. What rule do these problems demonstrate?

True and NA

a. There are a few times when `NA` are not contagious! Given your analysis above, predict, test and explain the output of the following code.

```
TRUE & NA
FALSE & NA
TRUE | NA
FALSE | NA
```

Solution: The big idea here is that sometimes missing information will not change the outcome. For `|`, a single `TRUE` implies the expression is `TRUE`. With `&` a single `FALSE` implies the expression is `FALSE`. In these situations, `NA` can be ignored.

2. Without running the code, predict what the output will be. Then, see if you were right by running the code in the console.

```
ifelse(TRUE, "yes", "no")
ifelse(FALSE, "yes", "no")
ifelse(c(TRUE, FALSE, TRUE, FALSE), "yes", "no")
ifelse(c(NA, TRUE, FALSE), "yes", "no")
ifelse(c(NA, NA, TRUE, FALSE), "yes", "no")
ifelse(c(NA, NA, TRUE, FALSE) & TRUE, "yes", "no")
ifelse(c(NA, NA, TRUE, FALSE) | TRUE, "yes", "no")
```

3. What do the following do? Predict the output and then run the code.

```r
# x and y are integer vectors of length 10
x <- c(2:10, NA)
y <- c(NA, 10:2)

# also predict how long will the output be and what data type will we have?
ifelse(x < y, y, x)
ifelse(is.na(x), 0, x)
ifelse(x == 2, paste0(x, "nd"),
       ifelse(x == 3, paste0(x, "rd"),
              paste0(x, "th")))
y  +  ifelse(x > 6, 1, 2)
ifelse(y %% 2 == 0, "even", "odd")
```

4. Write your own `ifelse()` statements using `x` or `y` as input that do the following:

a. Use `&` or `|`.
b. Return a character vector with no `NA`s.
c. Uses nested `ifelse`.
d. All three things at the same time.

There are no correct answers. The idea here is to explore the code to build understanding!

```r
# EXample Solutions
ifelse(x > y & !is.na(y), 1, 0)
ifelse(is.na(y) | is.na(x), "missing", "available")
ifelse(y > 7,
       ifelse(x < 5, TRUE, FALSE),
       NA) # if y is <= 7 we return NA
ifelse(is.na(y))
```

Data and background

For this lab, we will use data from Opportunity Insights: https://opportunityinsights.org/data/

1. Download the Stata file and Readme for "College Level Characteristics from the IPEDS Database and the College Scorecard".
2. Load it. We will refer to the data as `mrc_data`.

You should be able to read the data with no error in a more-or-less tidy format! (Finally a clean data set!)

1. What does each row represent? How many are there?
2. What columns are present? How many are there? Are the names clear? (The README can help clarify information).

II. Common uses of `ifelse`

1. Run the following code and you will see the distinct `tier_names` available in the dataset.

```
mrc_data %>% distinct(tier_name)
```

a. `ifelse` can be used to adjust entries in the `tier_name` column. Change "Two-year (public and private not-for-profit)" to "Two-year (public and private)".[1]

```
# SOLUTION
mrc_data %>%
mutate(tier_name = ifelse(tier_name == "Two-year (public and private not-for-profit)",
                          "Two-year (public and private)",
                          tier_name))
```

b. `ifelse` is often used to collapse tiers. Redefine `tier_name` so that "Nonselective four-year public" and "Nonselective four-year private not-for-profit" are grouped together as "Nonselective four-year (public and private)".[2]

```
# SOLUTION
mrc_data %>%
mutate(tier_name = ifelse( tier_name == "Nonselective four-year private not-for-profit"|
                           tier_name == "Nonselective four-year public",
                          "Nonselective four-year (public and private)",
                          tier_name))
```

2. As you can see below, there are 1466 colleges missing average SAT scores. Sometimes we want to replace `NA`s with a value. For example, linear regressions will drop any row with `NA`s, and we might not want that.

```
mrc_data %>%
  summarise(missing_sat_2013 = sum(is.na(sat_avg_2013)))
```

```
## # A tibble: 1 x 1
##   missing_sat_2013
##              <int>
## 1             1466
```

To avoid dropping rows, sometimes people replace the `NA` with the mean and add a new column that is an indicator of missingness. Here's a small example of what we expect.

```
before <- tibble(fake_data = c(1, 2, NA))
before
```

```
## # A tibble: 3 x 1
##   fake_data
##       <dbl>
## 1         1
## 2         2
## 3        NA
```

[1] Hint: In the first position, put a condition testing if `tier_name` matches the string. If it does, we replace the string with "Two-year (public and private)", otherwise keep the same data.

[2] Hint: The code will be very similar to the previous problem.

```
after
```

```
## # A tibble: 3 x 2
##   fake_data missing_fake_data
##       <dbl>             <dbl>
## 1       1                 0
## 2       2                 0
## 3       1.5               1
```

Fill `NA` in `sat_avg_2013` with the average SAT score of the other colleges and create a column called `missing_sat_avg_2013` that is 1 if `NA` and 0 otherwise.

```
# SOLUTION
mrc_data %>%
mutate(missing_sat_avg_2013 = ifelse(is.na(sat_avg_2013), 1, 0),
       sat_avg_2013 = ifelse(is.na(sat_avg_2013),
                             mean(sat_avg_2013, na.rm = TRUE),
                             sat_avg_2013)) %>%
  select(missing_sat_avg_2013, sat_avg_2013) %>%
  head()
```

```
## # A tibble: 6 x 2
##   missing_sat_avg_2013 sat_avg_2013
##                  <dbl>        <dbl>
## 1                    1        1064.
## 2                    0        1075
## 3                    0         925
## 4                    1        1064.
## 5                    0         984
## 6                    0        1115
```

III. College choice

Imagine the situation: It's 2014 and a group of high school friends want to go to college together. They need to find a college that meets all their preferences. Your job is to find the perfect college.

Name	SAT Score	Preferences
A-plus Abdul	1430	Either ivy plus tier or a flagship school
Snooty Stephen	1450	not a public school
Nourishing Nancy	1590	school in the midwest so she can be near her grandma
Radical Rei	1490	strong social studies (as measured by the percentage of students majoring in social studies > 30 percent)
Cost-conscious Casey	1600	wants a public school in CA or a school where students from homes in the bottom 20th percentile of incomes pay less than 10000 per year

Here are the rules. They want to go to school where their test scores are within 100 points of the school average SAT score. To match their preferences, use the most recent data. You will need a few tools.

1. First, in order to understand what a column contains, you can use the `distinct()` function[3]. For example, say you are trying to figure out how to identify "ivy plus" schools (or what that specifically means). Notice that there is a columns called `tier_name`, then run the code (don't worry if you are not familiar with `%>%` at this moment. We will cover this in Lecture 7):

```
mrc_data %>% distinct(tier_name)
```

```
## # A tibble: 12 x 1
##    tier_name
##    <chr>
##  1 Two-year for-profit
##  2 Selective private
##  3 Nonselective four-year public
##  4 Four-year for-profit
##  5 Selective public
##  6 Two-year (public and private not-for-profit)
##  7 Nonselective four-year private not-for-profit
##  8 Highly selective private
##  9 Less than two-year schools of any type
## 10 Other elite schools (public and private)
## 11 Highly selective public
## 12 Ivy Plus
```

   We see there are 12 tiers and one is "Ivy Plus"! Note the capitalization.

2. Second, we're going to have to find schools that match ranges of SAT scores by using `filter()` function, which filters out all observations that meet the given condition. We can use the `between()` function from `dplyr`. Are the following two approaches generating the same output? If not, what is the difference?

```
mrc_data %>% filter(1330 <= sat_avg_2013, sat_avg_2013 <= 1530)
mrc_data %>% filter(between(sat_avg_2013, 1330, 1530))
```

3. The final thing is a concept. You're probably about to write code that looks like the following pseudo code.[4] Note that `mutate()` is a function that creates new variables based on certain conditions. This function is included in a package called tidyverse (you need to install and load it before running the code), though you don't have to worry about it now – we will cover more details in Lecture 7.

```
# This is pseudo code
mrc_data %>%
  mutate(abdul_choices = ifelse(CONDITIONS, "yes", "no"),
         stephens_choices = ifelse(CONDITIONS, "yes", "no"),
         ...) %>%
  filter(abdul_choices == "yes", stephens_choices == "yes", ...)
```

   We can avoid the extra `== "yes"` by making `abdul_choices` a logical vector. In other words, write code like so:

```
# This is pseudo code
mrc_data %>%
  mutate(abdul_choices = ifelse(CONDITIONS, TRUE, FALSE),
```

[3] from `dplyr`. The codebook is also useful.
[4] pseudo code is a term for fake code that captures the logic of some coding idea without being actual code.

```
         stephens_choices = ifelse(CONDITIONS, TRUE, FALSE),
         ...) %>%
  filter(abdul_choices, stephens_choices, ...)
```

Test out the concept with a simple example. For example, try to figure out Abdul's only condition of being Ivy Plus. If time permits, also try to figure out Stephen's only condition of not being a public school.

4. Now you're ready to find the college for the five friends.

a. What school(s) are acceptable to all five?
b. How many school(s) are available to any of the five? Adjust `filter` statement slightly.[5]

```
# SOLUTION
 bff_super_awesome_college_list <-
    mrc_data %>%
      mutate(abdul_choices = ifelse(between(sat_avg_2013, 1330, 1530) &
                                    (tier_name == "Ivy Plus" | flagship == 1 ),
                                    TRUE, FALSE),
             casey_choices = ifelse(between(sat_avg_2013, 1500, 1600) &
                                    (public == 1 & state == "CA" |
                                    scorecard_netprice_2013 < 10000),
                                    TRUE, FALSE),
             rei_choices = ifelse(between(sat_avg_2013, 1390, 1590) &
                                  (pct_socialscience_2000 > 30),
                                   TRUE, FALSE),
             nancy_choices = ifelse(between(sat_avg_2013, 1490, 1600) &
                                    (region == 2),
                                    TRUE, FALSE),
             sam_choices = ifelse(between(sat_avg_2013, 1350, 1550) &
                                   (public == 0),
                                   TRUE, FALSE),
              )

bff_super_awesome_college_list %>%
  filter(abdul_choices, casey_choices, rei_choices, nancy_choices, sam_choices)
```

1. The five friends have `NA` in their choice sets. Do the the school list change if we replace all the `NA`s with `TRUE`? Without coding, argue why the list will not change if we replace the `NA`s with `FALSE`.

SOLUTION: When we make force missing values to be `TRUE`, they still only agree on UChicago. If we replace the `NA` with `FALSE` the results will not change following the logic from the warm-up. In filter, we're asking a bunch of logical AND (`&`) so if we replace an `NA` with `FALSE` we'll continue to filter away that row!

```
# The straight forward way to solve is to fill all the NAs with TRUE using a
# new call to ifelse() for each person
# ex:
bff_super_awesome_college_list %>%
  mutate(abdul_choices = ifelse(is.na(abdul_choices), TRUE, abdul_choices),
         casey_choices = ifelse(is.na(casey_choices), TRUE, casey_choices),
         rei_choices = ifelse(is.na(rei_choices), TRUE, rei_choices),
```

[5] Hint: Think about the warm-up you did for this lab

```
        nancy_choices = ifelse(is.na(nancy_choices), TRUE, nancy_choices),
        sam_choices = ifelse(is.na(sam_choices), TRUE, sam_choices)) %>%
filter(abdul_choices, casey_choices, rei_choices, nancy_choices, sam_choices)


# you might Google around and figure out something like this; though tbh it
# took a while for me to find it.
bff_super_awesome_college_list %>%
  replace_na(list(abdul_choices = TRUE,
                  casey_choices = TRUE,
                  rei_choices = TRUE,
                  nancy_choices = TRUE,
                  sam_choices = TRUE)) %>%
  filter(abdul_choices, casey_choices, rei_choices, nancy_choices, sam_choices)



# Another approach is to use if_else() which is a dplyr ifelse() that has
# accepts a missing argument.

    mrc_data %>%
      mutate(abdul_choices = if_else(between(sat_avg_2013, 1330, 1530) &
                                     (tier_name == "Ivy Plus" | flagship == 1 ),
                                     TRUE, FALSE, missing = TRUE),
             casey_choices = if_else(between(sat_avg_2013, 1500, 1600) &
                                     (public == 1 & state == "CA" |
                                        scorecard_netprice_2013 < 10000),
                                     TRUE, FALSE, missing = TRUE),
             rei_choices = if_else(between(sat_avg_2013, 1390, 1590) &
                                    (pct_socialscience_2000 > 30),
                                     TRUE, FALSE, missing = TRUE),
             nancy_choices = if_else(between(sat_avg_2013, 1490, 1600) &
                                      (region == 2),
                                     TRUE, FALSE, missing = TRUE),
             sam_choices = if_else(between(sat_avg_2013, 1350, 1550) &
                                           (public == 0),
                                   TRUE, FALSE, missing = TRUE),
              ) %>%
  filter(abdul_choices, casey_choices, rei_choices, nancy_choices, sam_choices)
```

1. **Challenge (optional)** Create a "Five friends college ranking". A college is ranked 1 if it is acceptable to all 5 friends. 2 if it is acceptable to any 4 friends and so on.[6] Colleges that are not acceptable to any friend should be marked "Unranked".

[6] 3 if it is acceptable to 3 friends. 4 if acceptable to 2 friends and 5 if acceptable to 1 friend