STAT 33A Workbook 4

CJ HINES (3034590053)

Sep 24, 2020

This workbook is due **Sep 24, 2020** by 11:59pm PT.

The workbook is organized into sections that correspond to the lecture videos for the week. Watch a video, then do the corresponding exercises *before* moving on to the next video.

Workbooks are graded for completeness, so as long as you make a clear effort to solve each problem, you'll get full credit. That said, make sure you understand the concepts here, because they're likely to reappear in homeworks, quizzes, and later lectures.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

In the notebook, you can run the line of code where the cursor is by pressing `Ctrl` + `Enter` on Windows or `Cmd` + `Enter` on Mac OS X. You can run an entire code chunk by clicking on the green arrow in the upper right corner of the code chunk.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.

Three Ways to Subset

Watch the "Three Ways to Subset" lecture video.

Exercise 1

Create a variable `count` that contains the integers from 1 to 100 (inclusive).

The `as.character()` function coerces its argument into a character vector. Coerce `count` into a character vector and assign the result to a variable called `fizzy`. Now you have congruent vectors `count` and `fizzy`.

The modulo operator `%%` returns the remainder after dividing its first argument by its second argument. You can use the modulo operator to test whether a number is divisible by some other number.

For example, suppose you want to test whether the elements in a vector `x` are divisible by 7. Since `x %% 7` returns the remainder after dividing by 7, then then `x %% 7 == 0` returns `TRUE` for elements that are divisible by 7. Note that `%%` comes before `==` in the order of operations.

Use subset assignment (by condition) to replace every number in `fizzy` that's:

- Divisible by 3 with `"Fizz"`

- Divisible by 5 with `"Buzz"`
- Divisible by 15 with `"FizzBuzz"`

Leave all other numbers in `fizzy` as-is.

Print out the final version of `fizzy`. It should begin:

```
 [1] "1"        "2"        "Fizz"     "4"        "Buzz"     "Fizz"
 [7] "7"        "8"        "Fizz"     "Buzz"     "11"       "Fizz"
[13] "13"       "14"       "FizzBuzz" "16"       "17"       "Fizz"
```

Hint 1: Start by using `count` to write conditions that match the three bullet points above.

Hint 2: Use your conditions from Hint 1 to assign words to the corresponding elements of `fizzy`. Remember that `count` and `fizzy` are congruent.

Hint 3: The order you assign words to `fizzy` matters, since numbers that are divisible by 15 are also divisible by 3.

Note: This problem, called the FizzBuzz problem, used to be a common interview question to determine if a candidate has basic programming skills.

YOUR ANSWER GOES HERE:

```r
count = c(seq(1, 100))
fizzy = as.character(count)

i = 1
while (i <= 100) {
  if (count[i] %% 3 == 0) fizzy[i] = "Fizz"
  if (count[i] %% 5 == 0) fizzy[i] = "Buzz"
  if (count[i] %% 15 == 0) fizzy[i] = "FizzBuzz"
  i = i + 1
}
print(fizzy)
```

```
##   [1] "1"        "2"        "Fizz"     "4"        "Buzz"     "Fizz"
##   [7] "7"        "8"        "Fizz"     "Buzz"     "11"       "Fizz"
##  [13] "13"       "14"       "FizzBuzz" "16"       "17"       "Fizz"
##  [19] "19"       "Buzz"     "Fizz"     "22"       "23"       "Fizz"
##  [25] "Buzz"     "26"       "Fizz"     "28"       "29"       "FizzBuzz"
##  [31] "31"       "32"       "Fizz"     "34"       "Buzz"     "Fizz"
##  [37] "37"       "38"       "Fizz"     "Buzz"     "41"       "Fizz"
##  [43] "43"       "44"       "FizzBuzz" "46"       "47"       "Fizz"
##  [49] "49"       "Buzz"     "Fizz"     "52"       "53"       "Fizz"
##  [55] "Buzz"     "56"       "Fizz"     "58"       "59"       "FizzBuzz"
##  [61] "61"       "62"       "Fizz"     "64"       "Buzz"     "Fizz"
##  [67] "67"       "68"       "Fizz"     "Buzz"     "71"       "Fizz"
##  [73] "73"       "74"       "FizzBuzz" "76"       "77"       "Fizz"
##  [79] "79"       "Buzz"     "Fizz"     "82"       "83"       "Fizz"
##  [85] "Buzz"     "86"       "Fizz"     "88"       "89"       "FizzBuzz"
##  [91] "91"       "92"       "Fizz"     "94"       "Buzz"     "Fizz"
##  [97] "97"       "98"       "Fizz"     "Buzz"
```

Logic

Watch the "Logic" lecture video.

Exercise 2

Suppose you conduct a survey and store the results in the following congruent vectors:

```r
# Q: What's your favorite color?
color = c("red", "blue", "blue", "green", "yellow", "green")
color = factor(color)

# Q: Name a dessert you like?
sweet = c("egg tart", "brownie", "ice cream", "ice cream", "fruit", "egg tart")
sweet = factor(sweet)

# Q: Name a desert (not dessert) you like?
dry = c("Kalahari", "Atacama", "Taklamakan", "Sonoran", "Atacama", "Atacama")
dry = factor(dry)

# Q: How old are you?
age = c(23, 15, 92, 21, 28, 45)

# Q: How many UFOs have you seen since 2010?
ufo = c(0, 3, 122, 0, 0, 1)
```

Use the vectors above, comparison operators, and logical operators to compute a logical vector that corresponds to each of the following conditions.

1. People who have seen a UFO.
2. People who have seen a UFO but aren't over 50 years old.
3. People who didn't choose ice cream.
4. People who like both ice cream and the color green.
5. People who like the color red or the color green.

YOUR ANSWER GOES HERE:

```r
# People who have seen a UFO.
seenufo = (ufo > 0)
seenufo
```

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

```r
# People who have seen a UFO but aren't over 50 years old.
below50seenufo = (seenufo) & (age <= 50)
below50seenufo
```

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

```r
# People who didn't choose ice cream.
noticecream = (sweet != "ice cream")
noticecream
```

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

```r
# People who like both ice cream and the color green.
icecreamandgreen = (!noticecream) & (color == "green")
icecreamandgreen
```

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

```r
# People who like the color red or the color green.
redorgreen = (color == "red") | (color == "green")
redorgreen
```

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

Exercise 3

In the expression `(x < 5) == TRUE`, explain why `== TRUE` is redundant.

YOUR ANSWER GOES HERE:

> The `== TRUE` is redundant because `(x < 5)` is a logical class and type. When x is less than 5, the condition returns TRUE. Otherwise, it will return FALSE. So the `== TRUE` is unnecessary.

Logical Summaries

Watch the "Logical Summaries" lecture video.

No exercises for this section. You're halfway finished!

Subset vs. Extract

Watch the "Subset vs. Extract" lecture video.

Exercise 4

A **recursive** list is a list with elements that are also lists.

Here's an example of a recursive list:

```r
mylist = list(list(1i, 2, 3i), list(c("hello", "hi"), 42))
```

Use the recursive list above to answer the following:

1. What's the first element? What's the second element?

2. Use the extraction operator `[[` to get the value 42.
3. Use the extraction operator `[[` twice to get the value `3i`.

YOUR ANSWER GOES HERE:

```r
#1. First Element is list(1i, 2, 3i). Second Element is list(c("hello", "hi"), 42))
mylist[1]
```

```
## [[1]]
## [[1]][[1]]
## [1] 0+1i
##
## [[1]][[2]]
## [1] 2
##
## [[1]][[3]]
## [1] 0+3i
```

```r
mylist[2]
```

```
## [[1]]
## [[1]][[1]]
## [1] "hello" "hi"
##
## [[1]][[2]]
## [1] 42
```

```r
#2.
mylist[[2]][2]
```

```
## [[1]]
## [1] 42
```

```r
#3.
mylist[[1]][3]
```

```
## [[1]]
## [1] 0+3i
```

Exercise 5

For the list `cool_list = list("Hope", "springs", "eternal")`, why is `cool_list[1]` the same as `cool_list[1][1][1]`? Is this property unique to `cool_list`, or is it a property of all lists? Explain your answer.

YOUR ANSWER GOES HERE:

> The first element of cool_list,`cool_list[1]` returns the list with the first element "Hope". In order to reference inside the "Hope" list directly, we'd have to use the extraction operator "[[]]". It is a property of all lists.

Subsets of Data Frames

Watch the "Subsets of Data Frames" lecture video.

Exercise 6

For the dogs data, compute:

1. The mean and median of the longevity column (ignoring missing values).
2. The subset that contains rows 10-20 of the height, weight, and longevity columns.
3. The number of dog breeds whose average weight is greater than 42. *Note: the `weight` column is the average weight of each row's breed.*
4. The subset of large dogs that require daily grooming.

YOUR ANSWER GOES HERE:

```
dogs = readRDS("dogs.rds")

#1
mean(dogs$longevity, na.rm = TRUE)
```

```
## [1] 10.95674
```

```
median(dogs$longevity, na.rm = TRUE)
```

```
## [1] 11.29
```

```
#2
dogs$height[10:20]
```

```
##  [1] 14.50 21.75 10.50 10.25    NA 13.00  5.00 10.50 20.00 19.50 10.50
```

```
dogs$weight[10:20]
```

```
##  [1] 22.0 47.5 15.0   NA 24.0 15.5  5.5   NA   NA 45.0   NA
```

```
dogs$longevity[10:20]
```

```
##  [1] 12.53 12.58 13.92 11.42 12.63 11.81 16.50 11.05 12.87 12.54 12.80
```

```
#3
check = (dogs$weight > 42)
leo = dogs$weight[check]
leo
```

```
##   [1]    NA    NA    NA    NA  47.5    NA    NA    NA  45.0    NA    NA    NA
##  [13]  62.5  59.5  67.5    NA  65.0    NA  60.0    NA    NA  67.5  62.5    NA
##  [25]    NA    NA    NA    NA    NA    NA    NA    NA    NA  70.0    NA  77.5
##  [37]  65.0    NA  55.0 125.0    NA    NA    NA  77.5  82.5    NA  60.0    NA
##  [49]  80.0    NA  85.0    NA 115.0 175.0    NA 155.0  45.0    NA    NA    NA
##  [61]    NA    NA 115.0    NA    NA    NA    NA    NA    NA  62.5    NA  45.0
##  [73]    NA    NA  52.5  62.5    NA    NA    NA    NA    NA    NA    NA    NA
##  [85]    NA    NA    NA    NA    NA    NA    NA  55.0    NA    NA 100.0  92.5
##  [97]    NA    NA    NA    NA 130.0  51.5    NA    NA  97.5    NA  50.0    NA
## [109]  47.5    NA    NA    NA    NA  92.5    NA    NA    NA    NA    NA    NA
## [121]    NA    NA    NA
```

```
leo = leo[!is.na(leo)]
length(leo)
```

```
## [1] 37
```

```
#4
w = which(dogs$size == "large" & dogs$grooming == "daily")
dogs$breed[w]
```

```
## [1] "Briard"           "Giant Schnauzer"  "Afghan Hound"     "Borzoi"
## [5] "Alaskan Malamute" "Saint Bernard"
```

Exercise 7

Workbook 1 mentioned that the advantage of `order()` over `sort()` is that you can use `order()` to sort one vector based on the elements of some other congruent vector.

Use the `order()` function to sort the rows of the dogs data set based on the height column. What are the 3 tallest breeds of dog?

YOUR ANSWER GOES HERE:

```
byheight = order(dogs$height, decreasing = TRUE)
bred = dogs$breed[byheight]
bred[1:4]
```

```
## [1] "Irish Wolfhound" "Mastiff"         "Great Dane"      "Great Pyrenees"
```