Comprehension Check due May 27, 2021 22:59 +03

Question 1

1/1 point (graded)

In the video, we use the function not_inches to identify heights that were incorrectly entered

```
      not_inches <- function(x, smallest = 50, tallest = 84) {
  inches <- suppressWarnings(as.numeric(x))
  ind <- is.na(inches) | inches < smallest | inches > tallest
  ind
}
```

In this function, what TWO types of values are identified as not being correctly formatted in inches?

- [ ] Values that specifically contain apostrophes ('), periods (.) or quotations (").
- [✓] Values that result in NA's when converted to numeric
- [✓] Values less than 50 inches or greater than 84 inches
- [ ] Values that are stored as a character class, because most are already classed as numeric.

✓

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem

Question 2

1/1 point (graded)

Which of the following arguments, when passed to the function `not_inches()`, would return the vector `c(FALSE)`?

- ○ `c(175)`
- ○ `c(“5’8\””)`
- ● `c(70)`
- ○ `c(85)` (the height of Shaquille O'Neal in inches)

✓

Answer
Correct:
The entry 70 can be converted to a numeric entry by as.numeric and is within the range that we set. Therefore, the result of this function would be FALSE (i.e., our entry is correctly formatted in inches).

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem

Question 3

1/1 point (graded)
Our function `not_inches()` returns the object `ind`. Which answer correctly describes ind?

- ● `ind` is a logical vector of TRUE and FALSE, equal in length to the vector `x` (in the arguments list). TRUE indicates that a height entry is incorrectly formatted.
- ○ `ind` is a logical vector of TRUE and FALSE, equal in length to the vector `x` (in the arguments list). TRUE indicates that a height entry is correctly formatted.

○ `ind` is a data frame like our `reported_heights` table but with an extra column of TRUE or FALSE. TRUE indicates that a height entry is incorrectly formatted.

○ `ind` is a numeric vector equal to `reported_heights$heights` but with incorrectly formatted heights replaced with NAs.

✓

Answer
Correct:
Our function returns a logical vector, with TRUE indicating that a height entry is incorrectly formatted and FALSE indicating that a height entry is formatted properly in inches. We then use this logical vector to filter our raw_heights data to only show incorrectly formatted entries.

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem

Question 4

1/1 point (graded)

Given the following code

```
      > s
[1] "70"        "5 ft"      "4'11"      ""          "."         "Six feet"
```

What `pattern` vector yields the following result?

```
      str_view_all(s, pattern)
70
5  ft
4'11

.
Six feet
```

● `pattern <- "\\d|ft"`

○ `pattern <- "\d|ft"`

○ `pattern <- "\\d\\d|ft"`

○ `pattern <- "\\d|feet"`

✓

Answer
Correct: This regex identifies any numeric characters or the text “ft”.

Submit   You have used 1 of 2 attempts

Answers are displayed within the problem

Question 5

1/1 point (graded)
You enter the following set of commands into your R console. What is your printed result?

```
      animals <- c("cat", "puppy", "Moose", "MONKEY")
pattern <- "[a-z]"
str_detect(animals, pattern)
```

○ TRUE

○ TRUE TRUE TRUE TRUE

[✓ TRUE TRUE TRUE FALSE]

○ TRUE TRUE FALSE FALSE

✓

Answer

Correct:
While your first three strings have at least one lowercase letter [a-z], the string MONKEY does not have any lowercase letters and will return a FALSE.

Submit You have used 1 of 2 attempts

ℹ Answers are displayed within the problem

Question 6

1/1 point (graded)
You enter the following set of commands into your R console. What is your printed result?

```
      animals <- c("cat", "puppy", "Moose", "MONKEY")
pattern <- "[A-Z]$"
str_detect(animals, pattern)
```

- ○ FALSE FALSE FALSE FALSE
- ○ FALSE FALSE TRUE TRUE
- [✓ FALSE FALSE FALSE TRUE]
- ○ TRUE TRUE TRUE FALSE

✓

Answer
Correct:
Your regex pattern tells str_detect to look for an uppercase ([A-Z]) letter at the end of the string ($): this is only true for the string “MONKEY”.

Submit You have used 1 of 2 attempts

ℹ Answers are displayed within the problem

Question 7

1/1 point (graded)

You enter the following set of commands into your R console. What is your printed result?

```
      animals <- c("cat", "puppy", "Moose", "MONKEY")
pattern <- "[a-z]{4,5}"
str_detect(animals, pattern)
```

● FALSE TRUE TRUE FALSE

○ TRUE TRUE FALSE FALSE

○ FALSE FALSE FALSE TRUE

○ TRUE TRUE TRUE FALSE

✓

Answer

Correct:

Your regex command tells R to look for either 4 or 5 lowercase letters in a row anywhere in the string. This is true for the animals “puppy” and “Moose”.

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem

Question 8

1/1 point (graded)

Given the following code:

```
      animals <- c("moose", "monkey", "meerkat", "mountain lion")
```

Which TWO “pattern” vectors would yield the following result?

```
      > str_detect(animals, pattern)
[1] TRUE TRUE TRUE TRUE
```

- [✓]
```
        pattern <- “mo*”
```
- [✓]
```
        pattern <- “mo?”
```
- [ ]
```
        pattern <- “mo+”
```
- [ ]
```
        pattern <- “moo*”
```

✓

Answer
Correct:
This regex pattern looks for an “m” followed by zero or more “o” characters. This is true for all strings in the animal vector.
This regex pattern looks for an “m” followed by zero or one “o” characters. This is true for all strings in the animal vector. Even though “moose” has two “o”s after the “m”, it still matches the pattern.

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem

Question 9

1/1 point (graded)

You are working on some data from different universities. You have the following vector:

```
      > schools
[1] "U. Kentucky"                "Univ New Hampshire"          "Univ. of M
[5] "U California"               "California State University"
```

You want to clean this data to match the full names of each university:

```
      > final
[1] "University of Kentucky"      "University of New Hampshire" "University
[5] "University of California"    "California State University"
```

What of the following commands could accomplish this?

- ◯
```
       schools %>%
str_replace("Univ\\.?|U\\.?", "University ") %>%
str_replace("^University of |^University ", "University of ")
```

- [✓]
```
       schools %>%
str_replace("^Univ\\.?\\s|^U\\.?\\s", "University ") %>%
str_replace("^University of |^University ", "University of ")
```

- ◯
```
       schools %>%
str_replace("^Univ\\.\\s|^U\\.\\s", "University") %>%
str_replace("^University of |^University ", "University of ")
```

○
```
        schools %>%
    str_replace("^Univ\\.?\\s|^U\\.?\\s", "University") %>%
    str_replace("University ", "University of ")
```

✓

Answer
Correct:
This code properly replaces all versions of “U”, “U.”, “Univ” and “Univ.” with “University” and then adds the word “of”.

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem

Question 10

1/1 point (graded)

Rather than using the `pattern_with_groups` vector from the video, you accidentally write in the following code:

```
      problems <- c("5.3", "5,5", "6 1", "5 .11", "5, 12")
pattern_with_groups <- "^([4-7])[,\\.](\\d*)$"
str_replace(problems, pattern_with_groups, "\\1'\\2")
```

What is your result?

[✓ ● `[1] "5'3" "5'5" "6 1" "5 .11" "5, 12"`]

○ `[1] "5.3" "5,5" "6 1" "5 .11" "5, 12"`

○ `[1] "5'3" "5'5" "6'1" "5 .11" "5, 12"`

○ `[1] "5'3" "5'5" "6'1" "5’11" "5’12"`

✓

Answer
Correct:
Yes. You forgot to check for any spaces in your regex pattern. While the first two entries of “problems” have commas and periods correctly replaced, the last three entries are not identified as part of the pattern and are not replaced.

Submit You have used 1 of 2 attempts

Answers are displayed within the problem

Question 11

1/1 point (graded)
You notice your mistake and correct your pattern regex to the following

```
      problems <- c("5.3", "5,5", "6 1", "5 .11", "5, 12")
pattern_with_groups <- "^([4-7])[,\\.\\s](\\d*)$"
str_replace(problems, pattern_with_groups, "\\1'\\2")
```

What is your result?

- ○ `[1] "5'3" "5'5" "6 1" "5 .11" "5, 12"`
- ○ `[1] "5.3" "5,5" "6 1" "5 .11" "5, 12"`
- [✓ `[1] "5'3" "5'5" "6'1" "5 .11" "5, 12"`]
- ○ `[1] "5'3" "5'5" "6'1" "5’11" "5’12"`

✓

Answer
Correct:
The new regex pattern now checks for one character, either a comma, period or space, between the first digit and the last one or two digits, and replaces it with an apostrophe (‘). However, because your last two problem strings have additional space between the digits, they are not corrected.

Submit You have used 1 of 2 attempts

Answers are displayed within the problem

Question 12

1/1 point (graded)

In our example, we use the following code to detect height entries that do not match our pattern of x'y":

```
      converted <- problems %>%
  str_replace("feet|foot|ft", "'") %>%
  str_replace("inches|in|''|\"", "") %>%
  str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")

pattern <- "^[4-7]\\s*'\\s*\\d{1,2}$"
index <- str_detect(converted, pattern)
converted[!index]
```

Which answer best describes the differences between the regex string we use as an argument in `str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")` and the regex string in `pattern <- "^[4-7]\\s*'\\s*\\d{1,2}$"` ?

- ○ The regex used in `str_replace()` looks for either a comma, period or space between the feet and inches digits, while the pattern regex just looks for an apostrophe; the regex in str_replace allows for one or more digits to be entered as inches, while the pattern regex only allows for one or two digits.
- ○ The regex used in `str_replace()` allows for additional spaces between the feet and inches digits, but the pattern regex does not.
- ○ The regex used in `str_replace()` looks for either a comma, period or space between the feet and inches digits, while the pattern regex just looks for an apostrophe; the regex in str_replace allows none or more digits to be entered as inches, while the pattern regex only allows for the number 1 or 2 to be used.

● The regex used in `str_replace()` looks for either a comma, period or space between the feet and inches digits, while the pattern regex just looks for an apostrophe; the regex in str_replace allows for none or more digits to be entered as inches, while the pattern regex only allows for one or two digits.

✓

Answer
Correct:
This answer describes two important differences in the `str_replace()` regex and the pattern regex.

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem

Question 13

1/1 point (graded)
You notice a few entries that are not being properly converted using your `str_replace()` and `str_detect()` code:

```
      yes <- c("5 feet 7inches", “5 7”)
no <- c("5ft 9 inches", "5 ft 9 inches")
s <- c(yes, no)

converted <- s %>%
  str_replace("feet|foot|ft", "'") %>%
  str_replace("inches|in|''|\"", "") %>%
  str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")

pattern <- "^[4-7]\\s*'\\s*\\d{1,2}$"
str_detect(converted, pattern)
[1]  TRUE TRUE FALSE FALSE
```

It seems like the problem may be due to spaces around the words feet|foot|ft and inches|in. What is another way you could fix this problem?

[✓]

```
        converted <- s %>%
str_replace("\\s*(feet|foot|ft)\\s*", "'") %>%
str_replace("\\s*(inches|in|''|\")\\s*", "") %>%
str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")
```

```
        converted <- s %>%
str_replace("\\s+feet|foot|ft\\s+”, "'") %>%
str_replace("\\s+inches|in|''|\"\\s+", "") %>%
str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")
```

```
        converted <- s %>%
str_replace("\\s*|feet|foot|ft", "'") %>%
str_replace("\\s*|inches|in|''|\"", "") %>%
str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")
```

```
        converted <- s %>%
str_replace_all(“\\s”, “”) %>%
str_replace("\\s|feet|foot|ft", "'") %>%
str_replace("\\s|inches|in|''|\"", "") %>%
str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")
```

✓

Answer
Correct:
You can add a none or more space character (\\s*) before and after each word to properly replace the word and any additional spaces with an apostrophe.

Submit

You have used 1 of 2 attempts

Answers are displayed within the problem