Preparation 10

Beatrice Kemboi

28 February, 2022

## {stringr}

The starwars dataset is loaded with the {tidyverse} and contains information from the Star Wars API on many Star Wars characters. For this Preparation, we will be focusing on the hair\_color, skin\_color, and eye\_color variables.

starwars

## # A tibble: 87 x 14  
## name height mass hair\_color skin\_color eye\_color birth\_year sex gender  
## <chr> <int> <dbl> <chr> <chr> <chr> <dbl> <chr> <chr>   
## 1 Luke S… 172 77 blond fair blue 19 male mascu…  
## 2 C-3PO 167 75 <NA> gold yellow 112 none mascu…  
## 3 R2-D2 96 32 <NA> white, bl… red 33 none mascu…  
## 4 Darth … 202 136 none white yellow 41.9 male mascu…  
## 5 Leia O… 150 49 brown light brown 19 fema… femin…  
## 6 Owen L… 178 120 brown, grey light blue 52 male mascu…  
## 7 Beru W… 165 75 brown light blue 47 fema… femin…  
## 8 R5-D4 97 32 <NA> white, red red NA none mascu…  
## 9 Biggs … 183 84 black light brown 24 male mascu…  
## 10 Obi-Wa… 182 77 auburn, wh… fair blue-gray 57 male mascu…  
## # … with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,  
## # films <list>, vehicles <list>, starships <list>

### Hair Color Frequencies

Looking at the frequencies of hair color for the Star Wars characters, we can see that some characters have multiple colors:

starwars %>%   
 drop\_na(hair\_color) %>%   
 group\_by(hair\_color) %>%   
 summarise(n = n()) %>%   
 arrange(n) %>%   
 print(n = length(hair\_color))

## Error in get\_n\_print(n, nrow(x)): object 'hair\_color' not found

Say that we wish to treat each color as a unique instance so if a character has two shades present in their hair, we would count each shade separately. What we need to do is separate the hair\_color column into individual columns, then restructure so that we have this information presented in a long format so that it will work within our data pipeline.

Notice that the most hair colors that a character has is two.

From the restructuring datasets materials earlier this semester, we would do the following (note that we are only *select*ing the column of interest to help focus the output):

restructure\_method <- starwars %>%   
 select(hair\_color) %>%   
 separate(hair\_color,   
 into = c("hair\_first", "hair\_second")) %>%   
 pivot\_longer(cols = everything(),   
 names\_to = "hair\_order",   
 values\_to = "hair\_color",   
 values\_drop\_na = TRUE) %>%   
 select(hair\_color)

## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 79 rows [1, 4, 5,  
## 7, 9, 11, 13, 14, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, ...].

restructure\_method

## # A tibble: 85 x 1  
## hair\_color  
## <chr>   
## 1 blond   
## 2 none   
## 3 brown   
## 4 brown   
## 5 grey   
## 6 brown   
## 7 black   
## 8 auburn   
## 9 white   
## 10 blond   
## # … with 75 more rows

## Alternatively... so clean!  
#  
# starwars %>%   
# select(hair\_color) %>%   
# separate\_rows(hair\_color)

In the code chunk below, you are provided with a character vector of the original hair\_color column.

hair\_vector <- starwars %>%   
 pull(hair\_color)

Use {stringr} functions to *split* the multiple colors in the pipeline below (i.e., replace the two \_\_\_ with the appropriate code).

str\_hair <- hair\_vector %>%   
 str\_split(pattern = " ,") %>%   
 unlist() # converts the list to a vector  
  
str\_hair

## [1] "blond" NA NA "none"   
## [5] "brown" "brown, grey" "brown" NA   
## [9] "black" "auburn, white" "blond" "auburn, grey"   
## [13] "brown" "brown" NA NA   
## [17] "brown" "brown" "white" "grey"   
## [21] "black" "none" "none" "black"   
## [25] "none" "none" "auburn" "brown"   
## [29] "brown" "none" "brown" "none"   
## [33] "blond" "none" "none" "none"   
## [37] "brown" "black" "none" "black"   
## [41] "black" "none" "none" "none"   
## [45] "none" "none" "none" "none"   
## [49] "white" "none" "black" "none"   
## [53] "none" "none" "none" "none"   
## [57] "black" "brown" "brown" "none"   
## [61] "black" "black" "brown" "white"   
## [65] "black" "black" "blonde" "none"   
## [69] "none" "none" "white" "none"   
## [73] "none" "none" "none" "none"   
## [77] "none" "brown" "brown" "none"   
## [81] "none" "black" "brown" "brown"   
## [85] "none" "unknown" "brown"

The code chunk below is to compare your solution to the “old method”. You do not need to add any code; however, verify that you understand what the code is doing.

restructure\_table <- restructure\_method %>%   
 drop\_na(hair\_color) %>%   
 group\_by(hair\_color) %>%   
 summarise(n = n())  
  
str\_table <- tibble(hair\_color = str\_hair) %>%   
 drop\_na(hair\_color) %>%   
 group\_by(hair\_color) %>%   
 summarise(n = n())  
  
waldo::compare(restructure\_table, str\_table)

## `attr(old, 'row.names')[7:9]`: 7 8 9   
## `attr(new, 'row.names')[7:12]`: 7 8 9 10 11 12  
##   
## old$hair\_color | new$hair\_color   
## [1] "auburn" | "auburn" [1]   
## - "auburn, grey" [2]   
## - "auburn, white" [3]   
## [2] "black" | "black" [4]   
## [3] "blond" | "blond" [5]   
## [4] "blonde" | "blonde" [6]   
## [5] "brown" | "brown" [7]   
## - "brown, grey" [8]   
## [6] "grey" | "grey" [9]   
## [7] "none" | "none" [10]   
## ... ... ... and 1 more ...  
##   
## `old$n`: 3 13 3 1 19 3 37 and 2 more...  
## `new$n`: 1 1 1 13 3 1 18 1 1 37 ...

Did you get No differences?

#### Challenge: Skin Color

Can you get the frequencies of the separate values for the variable skin\_color?

str\_skin <- starwars %>%   
 pull(skin\_color) %>%   
 str\_split(pattern = ", ") %>%   
 unlist()  
tibble(skin\_color = str\_skin) %>%   
 drop\_na(skin\_color) %>%   
 group\_by(skin\_color) %>%   
 summarise(n = n())

## # A tibble: 21 x 2  
## skin\_color n  
## <chr> <int>  
## 1 blue 8  
## 2 brown 6  
## 3 brown mottle 1  
## 4 dark 6  
## 5 fair 18  
## 6 gold 1  
## 7 green 9  
## 8 green-tan 1  
## 9 grey 12  
## 10 light 11  
## # … with 11 more rows

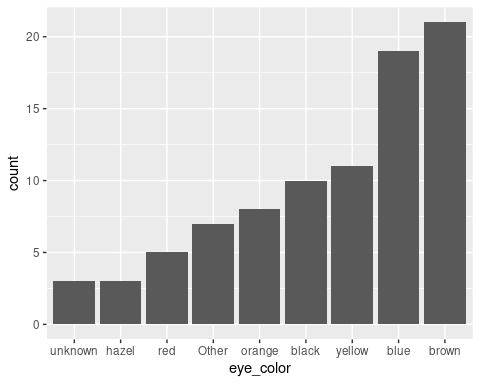
## {forcats}

This will continue using the starwars dataset.

## Plot Frequencies of Eye Color

Plot a bar chart of the variable eye\_color. However, eye colors that have less than 2 observations should be *lump*ed into an “Other” color. You do not need to separate the values of eye\_color, but you must use {forcats} functions to create this plot.

starwars %>%   
 mutate(eye\_color = fct\_lump(eye\_color, prop = 0.03), # total of 87, 2 out of 87 < 3%  
 eye\_color = fct\_infreq(eye\_color), # put levels in dec. order  
 eye\_color = fct\_rev(eye\_color)) %>% # put levels in inc. order  
 ggplot(mapping = aes(x = eye\_color)) +  
 geom\_bar()



Update your plot, using additional {forcats} functions, so that the bars are ordered by the heights (experiement with increasing and decreasing bar heights).

## {lubridate}

You will not be working with any particular data set for these items.

### Calculate age

Update the my\_birthday object below to contain your birthdate in the specified format, then calculate your age in the *number of days*.

my\_birthday <- "1993-11-01"  
as\_datetime(today()) - as\_datetime(my\_birthday)

## Time difference of 10346 days

Calculate your instructor’s birthday in *number of weeks*

bradford\_birthday <- "December 17, 1985"  
(today() - mdy(bradford\_birthday)) %/% dweeks(1)

## [1] 1888

(today() - mdy(bradford\_birthday)) %/% ddays(7)

## [1] 1888

### Vector of days

Create a vector called first\_days\_2021 that contains the first day of every month in 2020. That is, this vector should contain "2021-01-01" through "2021-12-01". Do this using {lubridate} arithmetic. DO NOT hand type each date.

first\_days\_2021 <- as\_datetime("2021-01-01") + months(0:11)  
first\_days\_2021

## [1] "2021-01-01 UTC" "2021-02-01 UTC" "2021-03-01 UTC" "2021-04-01 UTC"  
## [5] "2021-05-01 UTC" "2021-06-01 UTC" "2021-07-01 UTC" "2021-08-01 UTC"  
## [9] "2021-09-01 UTC" "2021-10-01 UTC" "2021-11-01 UTC" "2021-12-01 UTC"

Using your first\_days\_2021 and {lubridate} functions, identify which day of the week these first days were (e.g., Monday). That is, "2021-01-01" occurred on a Friday so your output should say "Friday".

first\_days\_wday <- lubridate::wday(first\_days\_2021, label = TRUE)  
first\_days\_wday

## [1] Fri Mon Mon Thu Sat Tue Thu Sun Wed Fri Mon Wed  
## Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat

## Knitting

To download this file and then upload to Blackboard, do the following:

1. Knit your report and verify that it looks as you want it to (that is, graphs and other output are displaying as you intended),
2. On line 5, change the output from “html\_document” to “word\_document”,
3. Knit your report again to create your preparation10.docx file,
4. Download this document by checking the box next to this docx file in the **Files** pane (lower-right-hand pane), then click More > Export…