HW5

Lance Ding

3/2023

Instructions

This HW is worth 10 total points.

1. Change the author and date fields in the header above to your name and the date.
2. Make sure to load any packages you may need right at the start. Do *NOT* include the `learnr` package, ever, unless you are writing an interactive Tutorial (which you won't do in this) - this will cause problems.
3. Ensure that no chunks have the `include = FALSE` or `echo = FALSE` option, as I want to be able to see *all* your code and output.
4. Brief but descriptive headings and document organization (answers under headings, text near relevant code, brief explanatory text as indicated below, etc.) (**3 pts** - this is more than usual due to the short nature of the overall assignment, so spend some time on this instead)
5. The following two questions continue to make use of the `classic_rock_song_list` or `classic_rock_raw_data` data frames from the `fivethirtyeight` package. You may use either one you like as they both contain artist.

   Make the variable `artist` into a factor, then print *just the first 10 levels.*

   Now choose your favorite artist, make them the new first level of the factor, then re-print the first 10 levels. (2 pts)
6. Extract the value "Jimi Hendrix" from the data as a *length-1* character vector. Do this two ways:

   i) Two steps: By first extracting/subsetting a single column vector, then indexing/subsetting that vector WITHOUT USING NUMBERS. If you are confused about how to accomplish the first part of this, my advice is to check the "Getting Single Columns" sub-section under "Subsetting Data Frames and Tibbles" in Tutorial 5.2. There's a couple options there that should help you. (1.5 pts)
   ii) One step: by taking it directly from the data frame, but do NOT use row or column NUMBERS in your code such as `cr[[877,2]]`. HINT: Figure out how you will indicate the correct column and find a proper row that has the value "Jimi Hendrix." (1.5 pts)
7. Consider the following list:

   `list(list("I", "Love", "Rock & Roll", "Put", "Another"), 10, "In", "The", list("Juke", "Box"), "Baaaaaby")`

   From this list, extract/subset me a vector with two elements: 10 and "Juke", in that order. Print it. (1.5 pts)

   What type of vector is this? Why? (0.5 pts)

To submit this assignment:

Ideally, knit straight to PDF by changing `html_document` to `pdf_document` in line 5 above. Otherwise:

1. Knit to HTML. An HTML document should open automatically in another RStudio window.
2. Click "Open in Browser" in that HTML document. It should open as a webpage in your default browser (e.g. Chrome).
3. Click Ctrl+P/Command+P, but instead of printing a hard copy on your printer click "Save as PDF."
4. Save and upload that document to Canvas.

A note on PDF formatting: you may notice that long lines of code "fly off the side of the page" when you knit to PDF. To fix this:

If you're on a Windows machine:

- Install the `formatR` package
- Change your `opts_chunk$set` code line to the following: `knitr::opts_chunk$set(echo = TRUE, tidy.opts=list(width.cutoff=80), tidy=TRUE)`

That should force your code to always wrap rather than fly off the edge of the page of a PDF. Note this does not fix issues of, say, plot titles that are too long getting cut off. But it should fix all the errors with your code not wrapping. Happy PDFing!

If you're on a Mac: I don't have an easy solution for you. Try and keep your lines of code under about 80 characters. Feel free to use more vertical lines of code to accomplish this. But don't waste large amounts of time formatting. I'll ask you for clarification if something critical is missing.

——BEGIN ANSWER BELOW———

Load necessary libraries

```
pacman::p_load(tidyverse)
library(fivethirtyeight)
```

```
## Some larger datasets need to be installed separately, like senators and
## house_district_forecast. To install these, we recommend you install the
## fivethirtyeightdata package by running:
## install.packages('fivethirtyeightdata', repos =
## 'https://fivethirtyeightdata.github.io/drat/', type = 'source')
```

```
data(classic_rock_song_list)
```

Factoring the `artist` Variable from `classic_rock_song_list`

`classic_rock_song_list`, per the data dictionary, is a "data frame with 2230 rows representing unique songs and 7 variables: `song`, `artist`, . . . " In its raw form, the dataframe's `artist` column is stored as a list of character vectors. However, since there could be multiple songs by the same artist, it may be beneficial to instead use a *factor* to store the values from `artist`. Factors are a way of storing categorical data in R, and can be much more space efficient in internal data storage and more time efficient in comparisons because R uses integers under the hood for factors.

We will use `factor()` to turn the `artist` column of `classic_rock_song_list` into a list of factors. We will then use list indicies and `select()` syntax to extract the **first 10 elements** from the list that contains all the levels of `artist` factor. Finally, we will make my favorite artist (*... ? I don't really listen to rock*) Led Zeppelin the first factor by using `fct_relevel()`.

```
crsl <- classic_rock_song_list %>%
  mutate(artist = factor(artist))

# First 10 levels
attributes(crsl$artist)[[1]][1:10]
```

```
##  [1] ".38 Special"       "10cc"              "3 Doors Down"
##  [4] "4 Non Blondes"     "a-ha"              "AC/DC"
##  [7] "Ace"               "Adelitas Way"      "Aerosmith"
## [10] "Alanis Morissette"
```

```
# Make Led Zeppelin first
crsl <- crsl %>%
  mutate(artist = fct_relevel(artist, "Led Zeppelin"))

attributes(crsl$artist)[[1]][1:10]
```

```
##  [1] "Led Zeppelin"  ".38 Special"   "10cc"          "3 Doors Down"
##  [5] "4 Non Blondes" "a-ha"          "AC/DC"         "Ace"
##  [9] "Adelitas Way"  "Aerosmith"
```

Extracting "Jimi Hendrix" from `artists` as an Atomic Vector

Now we want to extract "Jimi Hendrix" as an atomic character vector in 2 ways:

- By extracting a single column vector, then indexing that vector without using numbers
- By taking it directly from the dataframe without using row or column numbers

Here is the first implementation - we extract the `artist` column, then subset it by filtering out the entries that were not by Jimi Hendrix. We then take the first element of the resulting character vector.

```
crsl <- classic_rock_song_list # Load the unmodified data

crsl[["artist"]][crsl$artist == "Jimi Hendrix"][1]
```

```
## [1] "Jimi Hendrix"
```

```
class(crsl[["artist"]][crsl$artist == "Jimi Hendrix"][1])
```

```
## [1] "character"
```

Here is the second implementation - we extract the "Jimi Hendrix" directly from the dataframe by finding the first row that contains "Jimi Hendrix" and the `artist` column, and using indexing and `[[]]` to extract the element as an atomic character vector.

```
crsl[[which(crsl$artist == "Jimi Hendrix")[1], "artist"]]
```

```
## [1] "Jimi Hendrix"
```

```
class(crsl[[which(crsl$artist == "Jimi Hendrix")[1], "artist"]])
```

```
## [1] "character"
```

Extracting a Vector from a Nested List

Lists are complex data structures - they can contain multiple data types, and potentially other lists, making them **recursive**. We will now attempt to extract a vector containing the elements 10 and "Juke" from this recursive, or **nested**, list. The resulting vector should be of type `character`, because `character` is considered more complex and thus takes precedence over `numeric` when data types are coerced in the construction of a vector with elements that are not of the same type.

```
thing <- list(list("I", "Love", "Rock & Roll", "Put", "Another"),
              10,
              "In",
              "The",
              list("Juke", "Box"),
              "Baaaaaby")

subthing <- c(thing[[2]], thing[[5]][[1]])
subthing
```

```
## [1] "10"   "Juke"
```

```
class(subthing)
```

```
## [1] "character"
```