Trees and pipelines

QUESTIONS

40 marks

Recall, that in an earlier assignment a number of questions were asked about student guesses of the tallest coast redwood tree. The entire context can be found in the again in the `TreesContext` file.

The results of the online quiz were summarized in the file `trees.csv` for analysis, but this is not how the data were provided by the online quiz service.
Instead, they provided as two different ".csv" files, one for each course, in quite different structure and containing lots of extraneous information.

Anonymized versions of these data are given in the files:

- `DataViz_trees_anon.csv` containing the results from the students in `Data Visualization`, and
- `EDA_trees_anon.csv` containing the results from the students in `Exploratory Data Analysis`

The purpose of the present question is to exactly reproduce the data of `trees.csv` by reading in and reshaping the data in the above two files. But first, there will be a little exploration of the data as it comes in.

- **All programming** to accomplish this reshaping is to be use `readr` functions for inputting data and `magrittr` **pipelines** to reshape the data.
- In answering all questions, **show your code** as part of your answer **unless** you are specifically asked not to.
- Whenever asked to print the result, just use the default printing (which will only print a subset of the result). We don't need to see the whole thing, just whatever `R` prints.

a. **(2 marks)** Using `read_csv()`, read in the data

- from `EDA_trees_anon.csv`, assigning the result to `EDA_raw`, and
- from `DataViz_trees_anon.csv` assigning the result to `DV_raw`.

```
(EDA_raw <- read_csv(file.path(dataDirectory, "EDA_trees_anon.csv")))
(DV_raw <- read_csv(file.path(dataDirectory, "DataViz_trees_anon.csv")))
```

b. **(12 marks)** Some practice with pipes.

i. *(3 marks)* The variable `"Org Defined ID"` contains the (anonymized) student id number.

Construct a pipeline beginning with `EDA_raw`. It should

- change the name of the variable `"Org Defined ID"` to simply `ID`,
- determine the number of students answering the quiz, and
- print the result

```
EDA_raw %>%
rename(ID = `Org Defined ID`) %>%
select(ID) %>%
n_distinct()
```

```
## [1] 36
```

i. *(3 marks)* Construct a pipeline beginning with `EDA_raw`. It should

- determine the number of students who started answering the quiz in the `AM`,
- as well as the number in the `PM`, and then
- print the result showing both numbers

```
EDA_raw %>%
distinct(`Org Defined ID`, .keep_all = TRUE) %>%
filter(grepl("AM", `Attempt Start`)) %>%
select(`Attempt #`) %>%
summarise(countAM = n(), countPM = 36-countAM)
```

```
## # A tibble: 1 x 2
##   countAM countPM
##     <int>   <dbl>
## 1      13      23
```

i. *(3 marks)* The function `str_extract(x, "[0-9]+")` will extract the first set of contiguous digits from x.

Construct a pipeline beginning with `EDA_raw`. It should

- change the name of `"Answer Match"` to `answer`,
- find all the distinct answers,
- change these answers to be the first set of contiguous digits, and
- print the result

```
EDA_raw %>%
rename(answer = `Answer Match`) %>%
distinct(answer) %>%
str_extract("[0-9]+")
```

```
## [1] "75"
```

i. *(3 marks)* The function `substring(x, first, last)` will extract from `x` the contiguous characters from the position identified by `first` to that defined by `last` (default value of `last` is `1000000L`). Another function `toupper(x)` will convert the characters in `x` to be upper case (only affects letters).

Construct a pipeline beginning with `DV_raw` (from the Data Visualization class).

It should

- change the name of `"Answer Match"` to `answer`,
- find all the distinct answers,
- select the first 5 characters of `answer`
- apply `toupper` to `answer`
- print the result

```
DV_raw %>%
rename(answer = `Answer Match`) %>%
```

```
distinct(answer) %>%
mutate(answer = substring(answer, first = 1, last = 5)) %>%
toupper()
```

```
## [1] "C(NA, \"A\", \"110M\", \"A. LE\", \"100 M\", \"38\", \"B.\", \"100M\", \"B\", \"70\", \"160\",
```

```
#mutate(answer = toupper(substring(answer, first = 1, last = 5))) -> this returns col of dataframe
```

c. **(26 marks)** Now you will construct pipelines to produce the data sets `trees_EDA`, and `trees_DV`. These will then be combined to produce `trees_both` containing results from both classes. In the end, your data set `trees_both` should exactly match the `trees` data set from a previous assignment.

Each of these data sets will have exactly the following four variables (in this order):

- `class` – a character vector identifying the class as either `"DataViz"` or `"EDA"`
- `anchor` – a numeric `double` vector containing the value of the anchor used in the question
- `greater` – a logical vector indicating whether the student answered that the height was greater than (`TRUE`) or less than (`FALSE`) the value given by the anchor
- `height` – the value guessed by the student

The following information may be helpful in constructing your pipeline

- elements of `"Q Title"` identify which context (either `"Trees 1"` or `"Trees 2"`), that is set of questions, is being asked.
- the text of the questions (including the anchor value) is contained in `"Q Text"`
- which question (first or second) is being answered in `"Answer"`
- the answers are contained in `"Answer Match"`
- if, when answering the first question, anyone writing `"less than"` should have answered `"A"`, and anyone writing `"more than"`, or `"greater than"`, should have answered `"B"`.
- if, when answering the first question, they wrote the value of the anchor, then that should be interpreted as answering `"A"`
- for the height guessed, the first numerical answer they give is to be used (e.g. take just the first value of a range)

i. *(20 marks)* Beginning with `EDA_raw`, construct a single pipeline to produce `trees_EDA`.

Show the result `trees_EDA`

```
EDA_raw %>%
filter(Difficulty == 1) %>%
mutate(anchor = factor(if_else(`Q Title` == "Trees 1", 50, 150))) %>%
pivot_wider(names_from = Answer, values_from = 'Answer Match') %>%
transmute(class = "EDA",
          anchor = anchor,
          greater = factor(if_else(`Answer for blank # 1` == "A", "FALSE", "TRUE")),
          height = (if_else(`Answer for blank # 2` == "100-110m", 100,extract_numeric(`Answer for blan
) %>%
mutate(height = factor(if_else(`height` == 115.92, 115, height))) ->
trees_EDA
trees_EDA
```

```
## # A tibble: 36 x 4
##    class anchor greater height
##    <chr> <fct>  <fct>   <fct>
##  1 EDA   150    FALSE   75
##  2 EDA   50     TRUE    70
##  3 EDA   150    FALSE   100
##  4 EDA   50     TRUE    100
##  5 EDA   50     TRUE    65
##  6 EDA   150    FALSE   85
##  7 EDA   50     TRUE    120
##  8 EDA   50     TRUE    200
##  9 EDA   50     TRUE    100
## 10 EDA   50     TRUE    100
## # ... with 26 more rows
```

i. *(4 marks)* Reuse your code to construct `trees_DV` from `DV_raw`

- identify where your previous code is changed with a comment like:

```
# <----  change
```

at the end of the line where the change occurred.

Show the result `trees_DV`

```
DV_raw %>%
filter(Difficulty == 1) %>%
mutate(anchor = factor(if_else(`Q Title` == "Trees 1", 50, 150))) %>%
pivot_wider(names_from = Answer, values_from = 'Answer Match') %>%
mutate(blank1 = toupper(substr(`Answer for blank # 1`, 1, 1))) %>%            # <----  change
transmute(class = "DataViz",
          anchor = anchor,
          greater = factor(if_else(blank1 == "B", "TRUE", "FALSE")),      # <----  change
          height = extract_numeric(`Answer for blank # 2`)           # <----  change
) ->
trees_DV
trees_DV
```

```
## # A tibble: 93 x 4
##    class   anchor greater height
##    <chr>   <fct>  <fct>    <dbl>
##  1 DataViz 150    FALSE      110
##  2 DataViz 150    FALSE      100
##  3 DataViz 50     FALSE       38
##  4 DataViz 50     TRUE       100
##  5 DataViz 50     TRUE        70
##  6 DataViz 150    TRUE       160
##  7 DataViz 50     TRUE       150
##  8 DataViz 150    TRUE       300
##  9 DataViz 50     TRUE        70
## 10 DataViz 50     FALSE       20
## # ... with 83 more rows
```

i. *(2 marks)* Combine the two data sets by appending to the bottom of `trees_DV` the rows of `trees_EDA` to create `trees_both`.

Read in the values of `trees.csv` (with `read_csv()`) and assign the result to `trees`.

Check that these are identical by evaluating

```
sum(trees_both != trees)
```

This should be 0.

```
trees_both <- rbind(trees_DV, trees_EDA)
trees <- read_csv(file.path(dataDirectory, "trees.csv"))
```

```
sum(trees_both != trees)
```

```
## [1] 0
```