Grouping, filtering, and counting

Ben Schmidt

2019-01-29

Week 3 Problem set.

Crews dataset.

1. Using summarize, find the minimum age in the set. (Remember, the function to find a minimum is `min`.)

```
library(tidyverse)
crews = read_csv("../data/CrewlistCleaned.csv")
```

2. Using `filter`: what is the name of that youngest person? When did he or she sail?
3. How many sailors left on Barks between 1850 and 1880? Chain together `filter` and `summarize` with the special `n()` function. Note that this has a number of different conditions in the filter statement. You could build several filters in a row: but you can also include multiple filters by separating them with commas. For instance, `filter(school=="Northeastern",year==2015)` might be a valid filter on some dataset (though not this one.)
4. Question 3 told you how many sailors left on barks in those years. How many distinct voyages left? The variable `Voyage.number` identifies distinct voyages in this set. (This may require reading some documentation: reach out to me or a classmate if you can't figure it out. There are at least two ways: one involves using the `dplyr` function `distinct` before summarizing, and the second involves using the functions `length` and `unique` in your call to `summarize`.)

Grouping

```
crews %>% group_by(Residence)
```

Now when we run a summary operation, it will give the counts. For example, to get the number of people residing in each town by descending order, we could run the following function.

```
crews %>% group_by(Residence) %>% summarize(count=n()) %>% arrange(-count)
```

Filtering, Summarizing, and Grouping Exercises: Population data

Here are some real calisthenics to get used to using dplyr. This is a lot, but will get you into shape.

**Note: these exercises are quite cumulative: you'll probably be building up a single chain over the course of the set. Make use of cut and paste! 1. Read in the data frame `tidied` that you made in the last problem set, and select only the columns for city, state, year, and population.

2. Because the original data was not tidy, you probably have a lot of zeroes in there. Remove them.
3. The `slice` operator is much like `head`, but lets you specify precisely the positions you want. `slice(1:10)` will give you the first ten rows of a frame; `slice(4)` will give you the fourth row of a frame.
   Using `slice`: What were the ten largest cities in the United States in 1820?

4. `slice` works *by group*. Using `slice` and `group_by`, what was the third largest city in the United States for every census year?
5. Define a medium-sized city as any city with over 10,000 population. What **state** had the most medium-sized cities in 1900? (Remember the `n()` function get the count inside a summary.)
6. Tricky: create a list of the **states** with the most medium-sized cities for each year from 1790 to 2010. (Note that `group_by` can use multiple variables at the same time. You can either use `rank` or `arrange` with `slice` after grouping by the appropriate variables. Note that arrange respects existing groups: you may need to use the `ungroup` function even after a `summarize` call.)
7. Boston has been the largest city in its state in every census. What other cities have been the largest in their state every census since 1790?
8. (Tough) What state has had the most different cities as its largest? What are they? (Although you can answer both these questions with a single pipeline, it might be easier to do it with two.)
9. (Tough) For question 8, you found states which had the same largest city since 1790. But most states weren't even in the USA in 1790. Find, instead, the states that have the same largest city for every appearance they make in the census lists.

String functions

To think about working with *text*, I'm going to introduce a new dataset that I think might be interesting: all the books in the Library of Congress catalog that have

The `tidyverse` packages includes some functions for exploring strings

`str_detect` *finds* a regular expression; it returns TRUE or FALSE, so works well with `filter`. `str_replace` *replaces* a regular expression inside a string: it returns a new string, so it works well with mutate. `str_length` returns a number which is the *length* of the summary.

```r
books = read_csv("../data/summaries.csv.gz")

books %>%
  filter(subjects %>% str_detect("France|French")) %>%
  select(year, summary) %>%
  sample_n(10)
```

```r
country_regex = ".*(France|Germany|United Kingdom).*"
books %>%
  filter(subjects %>% str_detect(country_regex)) %>%
  mutate(
  country = subjects %>% str_replace(country_regex, "\\1")
) %>% filter(!is.na(country)) %>%
  group_by(country) %>%
  summarize(count=n())
```