Code

STAT 209: Lab 6

Aggregating Data by Group

Goal

To practice summarizing subsets of data separately using `group_by()` together with `summarize()`

Resources

For convenience, the link to the `dplyr` reference sheet is here (http://colindawson.net/stat209/resources/dplyr-quick-reference.pdf)

The Data

We'll continue some more with the `babynames` dataset. We'll start out exploring the question I asked you to "think about" but not actually answer from the last lab:

Exercise 1 For a chosen name, find the year that that name was the most equally split between male and female babies: that is, the year when the sex distribution was closest to 50/50.

Preliminaries (loading packages and data):

Code

Uncle Jess(i)e vs… Great Aunt Jessie?

80s Heart-throb Uncle Jesse, Born During Jessie's Most Male Era (Source: Bustle (https://www.bustle.com/articles/78132-7-times-full-houses-uncle-jesse-was-your-biggest-crush-because-this-rock-star-stole-the))

A Random Redditor's Great Aunt Jessie, Apparently; Born during Peak Jessie (Source: Reddit (https://www.reddit.com/r/OldSchoolCool/comments/49249f/my_great_aunt_jessie_1940s/))

For the examples below I'll look at the name "Jessie", which bounced around in its gender connotation during the period of this data, but has consistently been at least reasonably popular for both males and females.

One way to answer the question of interest would be to extract the counts for "Jessie" and manually scan the data to see when they are closest to equal.

Code:

Code

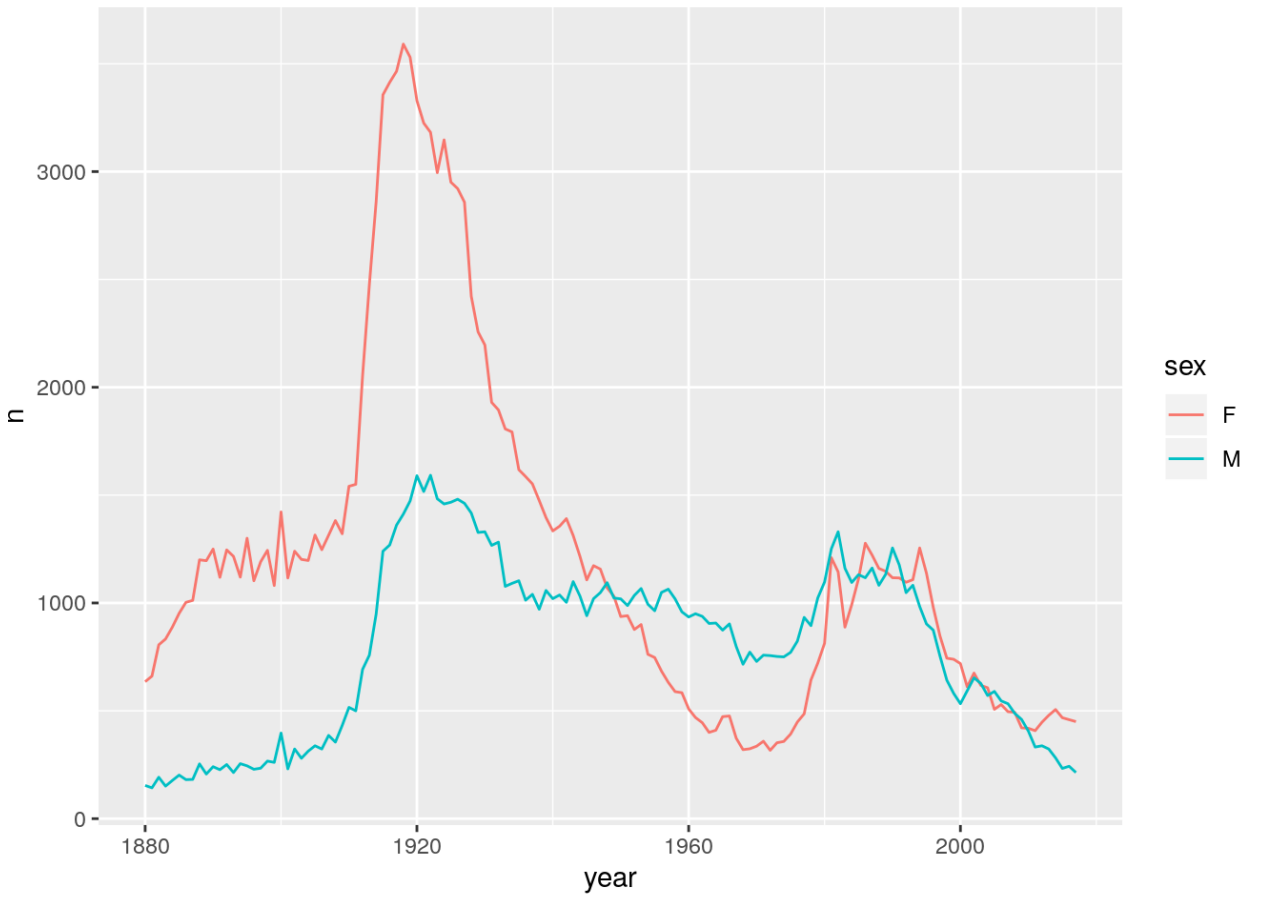
```
## # A tibble: 276 x 5
##     year sex   name       n    prop
##    <dbl> <chr> <chr>  <int>   <dbl>
##  1  1880 F     Jessie   635 0.00651
##  2  1880 M     Jessie   154 0.00130
##  3  1881 F     Jessie   661 0.00669
##  4  1881 M     Jessie   143 0.00132
##  5  1882 F     Jessie   806 0.00697
##  6  1882 M     Jessie   192 0.00157
##  7  1883 F     Jessie   833 0.00694
##  8  1883 M     Jessie   151 0.00134
##  9  1884 F     Jessie   888 0.00645
## 10  1884 M     Jessie   177 0.00144
## # … with 266 more rows
```

But this is time consuming and requires mental arithmetic.

We can create a quick plot to estimate the solution visually:

Code:

Code



It looks like the lines cross somewhere around 1950, and then cross a few more times between 1980 and 2010 or so. But this still isn't an exact solution.

What we'd like is to create a variable that actually tells us the relative number of males and females associated with the name "Jessie" in each year.

Exercise 2 Often when we want to create a new variable, we can use `mutate()`. But that won't work in this case. Why not?

Explanation:

Code

Using `summarize()` to do conditional counting

The quantity we are interested in requires us to aggregate data from multiple cases. This is what `summarize()` is good for: take a dataset and apply a function that takes a set of values as input and returns a single value as output. These functions, such as `sum()`, `mean()`, `max()`, and `n()`, are called **aggregate** functions.

For example, I can compute the number of male Jessies born in a particular year (let's say 1982) by using `filter()` to extract the data from 1982, and then using `summarize()` together with `sum()`, combined with a conditional function like `ifelse()` to replace the values I'm not interested in with zeroes. Like so:

Code:

Code

```
## # A tibble: 1 x 2
##   total num_males
##   <int>     <dbl>
## 1  2474      1330
```

The `ifelse()` command operates on each entry in a variable, returning a list of values the same length as the input. In this case, `ifelse(sex == "M", n, 0)` says to look at the `sex` column, and for each entry if it is equal to `"M"`, place the value of `n` in that same position, otherwise put zero. Then, the `sum()` function adds up the results.

Exercise 3 Take the command above and use `mutate()` to get the proportion of babies named Jessies in 1982 that were male.

Code:

Code

```
## # A tibble: 1 x 3
##   total num_males prop_males
##   <int>     <dbl>      <dbl>
## 1  2474      1330      0.538
```

So `filter()` together with `summarize()` gives us a proportion for a specific year. We could in principle repeat this for each year in the data and see which one comes out closest to 0.5. But this would be tedious, not to mention error-prone.

Combining `group_by()` and `summarize()` to summarize by group

Instead, we can use `group_by()` to "slice" the data by year, and `summarize()` each slice:

Code:

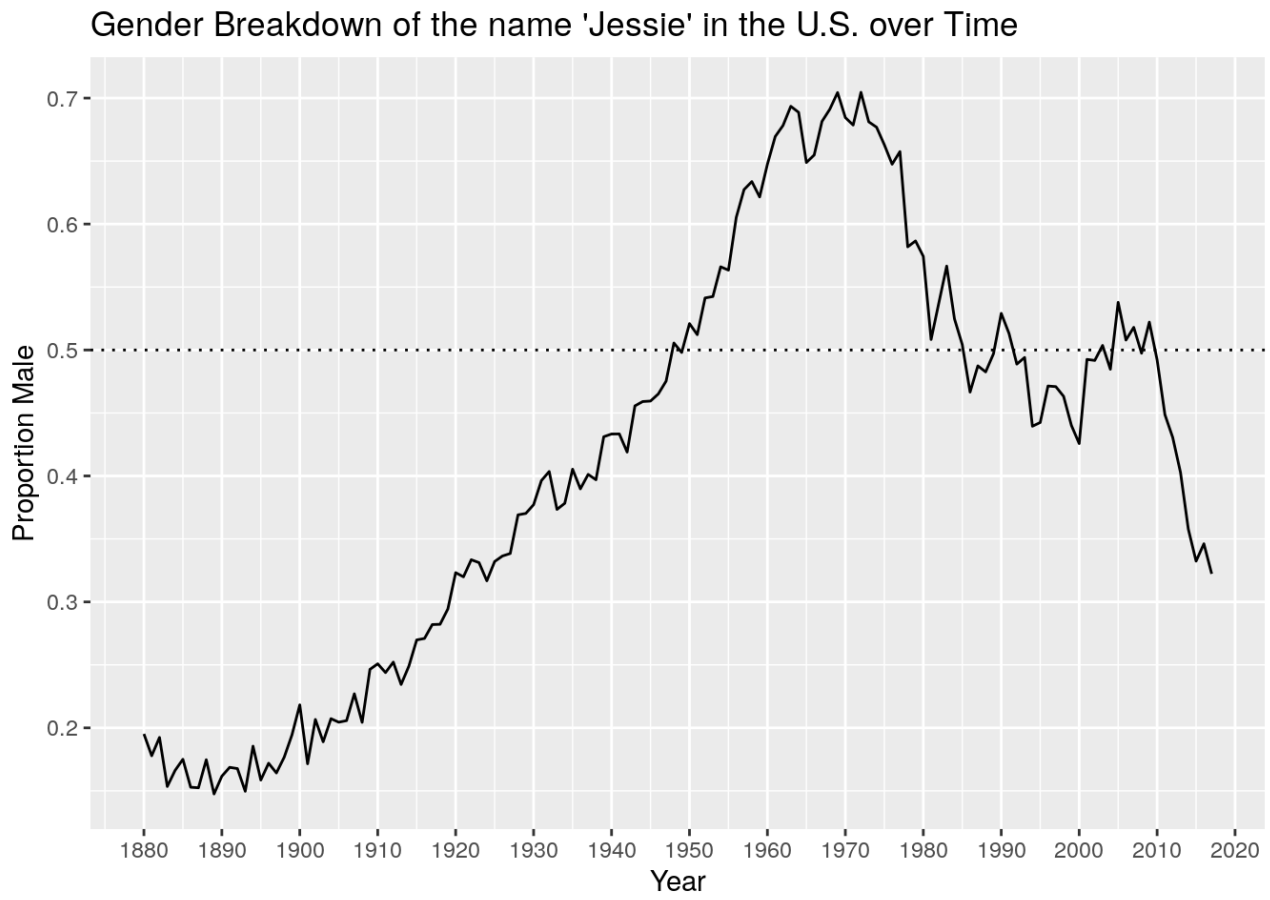
Code

```
## # A tibble: 138 x 6
##     year num_rows total_births num_males num_females prop_males
##    <dbl>    <int>        <int>     <dbl>       <dbl>      <dbl>
##  1  1880        2          789       154         635      0.195
##  2  1881        2          804       143         661      0.178
##  3  1882        2          998       192         806      0.192
##  4  1883        2          984       151         833      0.153
##  5  1884        2         1065       177         888      0.166
##  6  1885        2         1154       202         952      0.175
##  7  1886        2         1184       181        1003      0.153
##  8  1887        2         1194       182        1012      0.152
##  9  1888        2         1454       254        1200      0.175
## 10  1889        2         1403       207        1196      0.148
## # … with 128 more rows
```

Let's plot the proportion by year, just for fun:

Code:

Code



We can now see with one line what we could see before by seeing when our two lines crossed.

But we'd still like an exact answer to the question "In what year was the proportion closest to 50/50?"

`summarize()`ing a summary

Before that, though, here are some other questions you can answer with the `group_by()` and `summarize()` one-two punch:

Exercise 4 In what year was the largest *total* number of Jessies born (combining sexes)? How many Jessies were born that year? (Hint: treat the output of `summarize()` as just another data set, and use `summarize()` a second time to find the year with the largest total)

Sample solution:

Code

More Practice with `group_by()` and `summary()`

Exercise 5 Create a summary table that shows, for each name, the first year it appears in the data and the last year it appears in the data.

Sample solution:

Code

Exercise 6 There are 16 names that were assigned to babies of both sexes in *every* year from 1880 to 2017. List them. Hint: if a name has zero births for a particular sex in a particular year, there is no entry in the data for that year/sex/name combination. Names that appear for both sexes (and only those names) will have $2 \cdot (2017 - 1880 + 1)$ entries in the table.

Sample solution

Code

Interestingly, some of these (like John and William) have a strong connotation with a particular sex today, but they are such common names overall that a few instances of the opposite sex appear.

Exercise 7 Write a pipeline to return the 10 most common names (combining sexes) of the 1990s, arranged in descending order of popularity.

Sample solution:

Code

```
## # A tibble: 10 x 2
##    name        num_births
##    <chr>            <int>
##  1 Michael         464249
##  2 Christopher     361251
##  3 Matthew         352341
##  4 Joshua          330046
##  5 Jessica         303854
##  6 Ashley          303125
##  7 Jacob           298926
##  8 Nicholas        275906
##  9 Andrew          273515
## 10 Daniel          273347
```

Exercise 8 This one is really challenging, but provides a strong test of how well you understand these verbs and how to combine them: Find the names that were “popular” in your birth year (defined as being assigned to 1% or more of all births, irrespective of sex), and determine which one made its first appearance latest. That is, what’s the “youngest” popular name for your birth year? (This (I think) requires more than one “pipeline”: one to get a list of popular names in your birth year and another to extract the earliest appearance of each name and summarize. You will probably need to use `pull(DataSet, VariableName)` to extract a column as a free-standing object, and the `%in%` operator to check whether an entry is in a set)!

Sample solution:

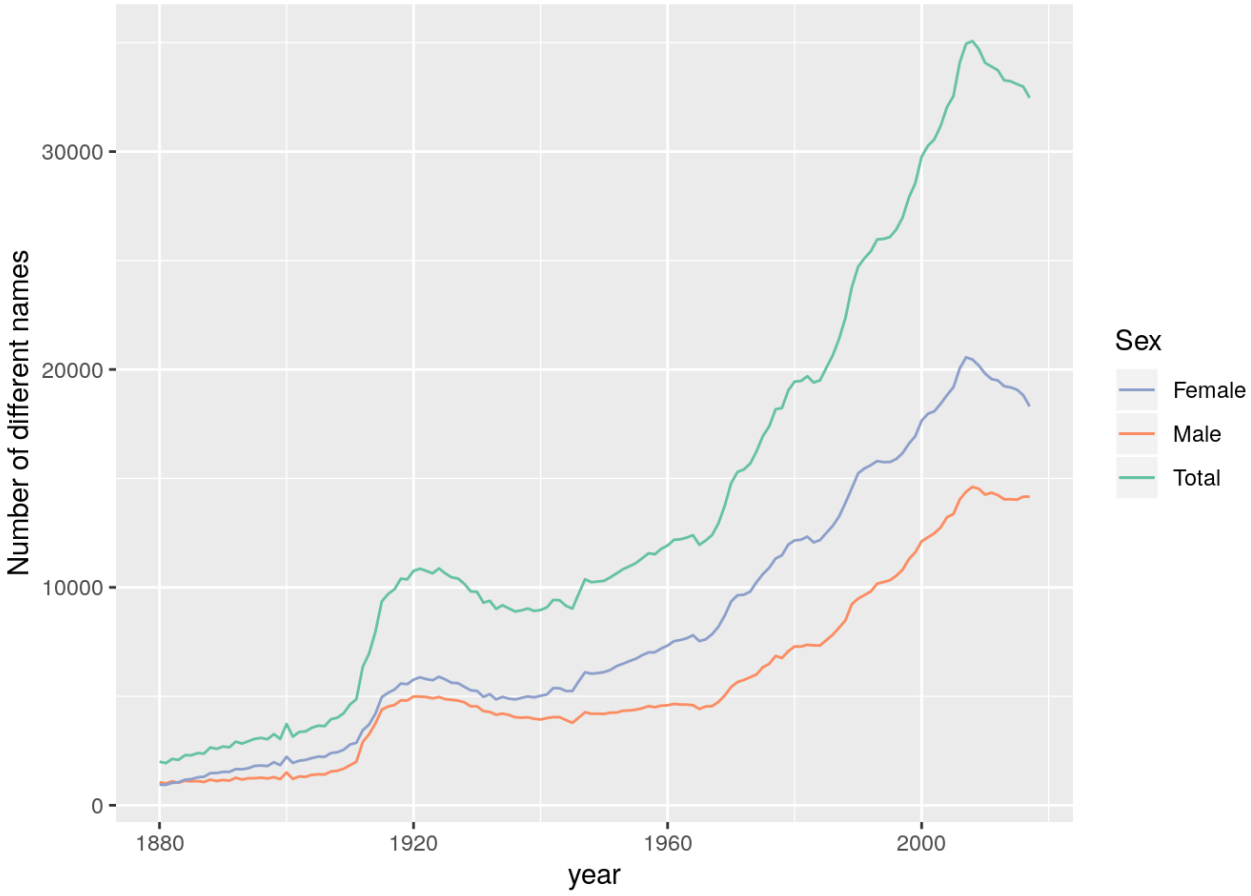
Code

In 1982, Jennifer was a popular name, and there were (theoretically) no U.S. born Jennifers who at the time were older than 66.

Exercise 9 Create and plot a variable that represents the "name diversity" in each year, defined as the number of distinct names that appear in the dataset. If you are so inclined, create additional variables that compute this for only males and for only females.

Sample solution

Code



Exercise 10 Use `group_by()`, `summarize()`, and `ggplot()` together (along with perhaps some other wrangling verbs) to create an interesting and informative visualization on a topic of your choosing (other than baby names). You might want to look back at Lab 2 for links to data sources. Share a code snippet and graphic to the `#lab6` channel on Slack.

Exercise 11 What did you find particularly interesting/challenging about this lab? Post your response alongside your plot on the `#lab6` channel.