# Class Activity 6

Your name here

March 19 2024

We will work with the `babynames` dataset again in this class activity. The header of the dataset looks like this:

```
knitr::kable(head(babynames), caption = "A glimpse of the babynames dataset")
```

Table 1: A glimpse of the babynames dataset

| year | sex | name | n | prop |
|---|---|---|---|---|
| 1880 | F | Mary | 7065 | 0.0723836 |
| 1880 | F | Anna | 2604 | 0.0266790 |
| 1880 | F | Emma | 2003 | 0.0205215 |
| 1880 | F | Elizabeth | 1939 | 0.0198658 |
| 1880 | F | Minnie | 1746 | 0.0178884 |
| 1880 | F | Margaret | 1578 | 0.0161672 |

In this tutorial, we will learn about the five main verbs of dplyr and how to use them to manipulate data:

- `select()`: Choose columns from a data frame
- `filter()`: Choose rows based on a condition
- `arrange()`: Sort the rows of a data frame
- `mutate()`: Add new columns based on existing columns
- `summarise()`: Aggregate data and compute summary statistics

## Problem 1: `select()`

Which of these is NOT a way to select the `name` and `n` columns together?

```
select(babynames, -c(year, sex, prop))  #1
select(babynames, name:n)   #2
select(babynames, starts_with("n")) #3
select(babynames, ends_with("n"))  #4
```

*Answer:* 4 is not the way to select the `name` and `n` columns together

## Problem 2: `filter()`

Use `filter()` with the logical operators to extract:

**a. All of the names where prop is greater than or equal to 0.08**

```
filter(babynames, prop >= 0.08)
# A tibble: 3 x 5
   year sex   name        n   prop
  <dbl> <chr> <chr>   <int>  <dbl>
1  1880 M     John     9655 0.0815
2  1880 M     William  9532 0.0805
3  1881 M     John     8769 0.0810
# alternate
babynames %>% filter(prop >= 0.08)
# A tibble: 3 x 5
   year sex   name        n   prop
  <dbl> <chr> <chr>   <int>  <dbl>
1  1880 M     John     9655 0.0815
2  1880 M     William  9532 0.0805
3  1881 M     John     8769 0.0810
```

**b. All of the babies named "Rose"**

```
babynames %>% filter(name == "Rose")
# A tibble: 247 x 5
    year sex   name      n      prop
   <dbl> <chr> <chr> <int>     <dbl>
 1  1880 F     Rose    700 0.00717
 2  1880 M     Rose      7 0.0000591
 3  1881 F     Rose    734 0.00743
 4  1882 F     Rose    886 0.00766
 5  1883 F     Rose    877 0.00730
 6  1883 M     Rose      5 0.0000445
 7  1884 F     Rose   1060 0.00770
 8  1884 M     Rose      5 0.0000407
 9  1885 F     Rose   1164 0.00820
10  1885 M     Rose      9 0.0000776
# i 237 more rows
```

**c. Use `filter()` to choose all rows where name is "John" and sex is "M".**

```
babynames %>% filter(name == "John", sex == "M")
# A tibble: 138 x 5
    year sex   name      n   prop
   <dbl> <chr> <chr> <int>  <dbl>
 1  1880 M     John   9655 0.0815
 2  1881 M     John   8769 0.0810
 3  1882 M     John   9557 0.0783
 4  1883 M     John   8894 0.0791
 5  1884 M     John   9388 0.0765
 6  1885 M     John   8756 0.0755
 7  1886 M     John   9026 0.0758
 8  1887 M     John   8110 0.0742
 9  1888 M     John   9247 0.0712
10  1889 M     John   8548 0.0718
# i 128 more rows
```

## Problem 3: `arrange()`

**a. Use `arrange()` to sort the babynames dataset by the prop column in descending order.**

```
babynames %>% arrange(desc(prop))
# A tibble: 1,924,665 x 5
    year sex   name        n   prop
   <dbl> <chr> <chr>   <int>  <dbl>
 1  1880 M     John     9655 0.0815
 2  1881 M     John     8769 0.0810
 3  1880 M     William  9532 0.0805
 4  1883 M     John     8894 0.0791
 5  1881 M     William  8524 0.0787
 6  1882 M     John     9557 0.0783
 7  1884 M     John     9388 0.0765
 8  1882 M     William  9298 0.0762
 9  1886 M     John     9026 0.0758
10  1885 M     John     8756 0.0755
# i 1,924,655 more rows
```

**b. Use `arrange()` to sort the babynames dataset by year (ascending) and then by prop (descending).**

```
babynames %>% arrange(year, desc(prop))
# A tibble: 1,924,665 x 5
    year sex   name        n   prop
   <dbl> <chr> <chr>   <int>  <dbl>
 1  1880 M     John     9655 0.0815
 2  1880 M     William  9532 0.0805
 3  1880 F     Mary     7065 0.0724
 4  1880 M     James    5927 0.0501
 5  1880 M     Charles  5348 0.0452
 6  1880 M     George   5126 0.0433
 7  1880 M     Frank    3242 0.0274
 8  1880 F     Anna     2604 0.0267
 9  1880 M     Joseph   2632 0.0222
10  1880 M     Thomas   2534 0.0214
# i 1,924,655 more rows
```

## Problem 4: `mutate()`

**a. Use `mutate()` to create a new column called decade which contains the decade the record is in (e.g., 1990 for the years 1990-1999).**

```
babynames %>% mutate(decade = (year %/% 10) * 10)
# A tibble: 1,924,665 x 6
    year sex   name           n   prop decade
   <dbl> <chr> <chr>      <int>  <dbl>  <dbl>
 1  1880 F     Mary        7065 0.0724   1880
 2  1880 F     Anna        2604 0.0267   1880
 3  1880 F     Emma        2003 0.0205   1880
 4  1880 F     Elizabeth   1939 0.0199   1880
 5  1880 F     Minnie      1746 0.0179   1880
 6  1880 F     Margaret    1578 0.0162   1880
 7  1880 F     Ida         1472 0.0151   1880
 8  1880 F     Alice       1414 0.0145   1880
 9  1880 F     Bertha      1320 0.0135   1880
10  1880 F     Sarah       1288 0.0132   1880
# i 1,924,655 more rows
```

## Problem 5: `summarize()` or `summarise()`

Use the codes mentioned so far to compute three statistics:

- the total number of children who ever had your name
- the maximum number of children given your name in a single year
- the mean number of children given your name per year

```
babynames %>%
  filter(name == "John", sex == "M") %>%
  group_by(year) %>%
  summarise(total = sum(n),
            max = max(n),
            mean = mean(n))
# A tibble: 138 x 4
    year total   max  mean
   <dbl> <int> <int> <dbl>
 1  1880  9655  9655  9655
 2  1881  8769  8769  8769
 3  1882  9557  9557  9557
 4  1883  8894  8894  8894
 5  1884  9388  9388  9388
 6  1885  8756  8756  8756
 7  1886  9026  9026  9026
 8  1887  8110  8110  8110
 9  1888  9247  9247  9247
10  1889  8548  8548  8548
# i 128 more rows
```

```
# alternate
summarize(filter(babynames, name =="John", sex == "M"), total = sum(n), max = max(n), mean = mean(n))
# A tibble: 1 x 3
    total   max   mean
    <int> <int>  <dbl>
1 5115466 88318 37069.
```

## Problem 6

**a. Use `min_rank()` and `mutate()` to rank each row in babynames from largest prop to smallest prop.**

```
babynames %>% mutate(rank = min_rank(desc(prop))) %>% arrange(rank)
# A tibble: 1,924,665 x 6
    year sex   name        n   prop  rank
   <dbl> <chr> <chr>   <int>  <dbl> <int>
 1  1880 M     John     9655 0.0815     1
 2  1881 M     John     8769 0.0810     2
 3  1880 M     William  9532 0.0805     3
 4  1883 M     John     8894 0.0791     4
 5  1881 M     William  8524 0.0787     5
 6  1882 M     John     9557 0.0783     6
 7  1884 M     John     9388 0.0765     7
 8  1882 M     William  9298 0.0762     8
 9  1886 M     John     9026 0.0758     9
10  1885 M     John     8756 0.0755    10
# i 1,924,655 more rows
```

**b. Compute each name's rank within its year and sex.**

```
babynames %>% group_by(year, sex) %>% mutate(rank = min_rank(desc(prop)))
# A tibble: 1,924,665 x 6
# Groups:   year, sex [276]
    year sex   name          n   prop  rank
   <dbl> <chr> <chr>     <int>  <dbl> <int>
 1  1880 F     Mary       7065 0.0724     1
 2  1880 F     Anna       2604 0.0267     2
 3  1880 F     Emma       2003 0.0205     3
 4  1880 F     Elizabeth  1939 0.0199     4
 5  1880 F     Minnie     1746 0.0179     5
 6  1880 F     Margaret   1578 0.0162     6
 7  1880 F     Ida        1472 0.0151     7
 8  1880 F     Alice      1414 0.0145     8
 9  1880 F     Bertha     1320 0.0135     9
10  1880 F     Sarah      1288 0.0132    10
# i 1,924,655 more rows
```

**c. Then compute the median rank for each combination of name and sex, and arrange the results from highest median rank to lowest.**

```
babynames %>%
  group_by(year, sex) %>%
  mutate(rank = min_rank(desc(prop))) %>%
  group_by(name, sex) %>%
  summarize(score = median(rank)) %>%
  arrange(score)
# A tibble: 107,973 x 3
# Groups:   name [97,310]
   name      sex   score
   <chr>     <chr> <dbl>
 1 Mary      F       1
 2 James     M       3
 3 John      M       3
 4 William   M       4
 5 Robert    M       6
 6 Michael   M       7.5
 7 Charles   M       9
 8 Elizabeth F      10
 9 Joseph    M      10
10 Thomas    M      11
# i 107,963 more rows
```