Homework assignment 1

Owen Bruce

1. Install the babynames package.

```
#install.packages('babynames')
```

2. How many variables and observations does this package contain?

```
library(babynames)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
data(babynames)
glimpse(babynames)
```

```
Rows: 1,924,665
Columns: 5
$ year <dbl> 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880,~
$ sex  <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", ~
$ name <chr> "Mary", "Anna", "Emma", "Elizabeth", "Minnie", "Margaret", "Ida",~
$ n    <int> 7065, 2604, 2003, 1939, 1746, 1578, 1472, 1414, 1320, 1288, 1258,~
$ prop <dbl> 0.07238359, 0.02667896, 0.02052149, 0.01986579, 0.01788843, 0.016~
```

The package contains 5 variables with 1,924,665 observations.

3. Create a data dictionary for each of the variables that includes the variable name, data type, and a description.

```
(data_dictionary <- tibble::tibble(
  variable_name = c("year", "sex", "name", "n", "prop"),
    data_type = c("double", "character", "character", "integer", "double"),
    description = c("The year the data is from",
   "The sex of the baby",
   "The baby name",
   "The number of babies born with that name in the specific year",
   "n divided by total number of applicants in that year,
    which means proportions are of people of that sex with that name born in
   that year")))
```

```
# A tibble: 5 x 3
  variable_name data_type description
  <chr>         <chr>     <chr>
1 year          double    "The year the data is from"
2 sex           character "The sex of the baby"
3 name          character "The baby name"
4 n             integer   "The number of babies born with that name in the spec~
5 prop          double    "n divided by total number of applicants in that year~
```

4. What is the range of years covered in babynames?

```
range(babynames$year)
```

```
[1] 1880 2017
```

The years covered range from 1880 to 2017.

5. Create an object from the babynames package that does not include the variable n.

```
baby_no_n <- babynames |> dplyr::select(!n)
glimpse(baby_no_n)
```

```
Rows: 1,924,665
Columns: 4
$ year <dbl> 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880,~
$ sex  <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", ~
$ name <chr> "Mary", "Anna", "Emma", "Elizabeth", "Minnie", "Margaret", "Ida",~
$ prop <dbl> 0.07238359, 0.02667896, 0.02052149, 0.01986579, 0.01788843, 0.016~
```

6. What is one reason for not including n, but keeping the variable prop?

Assessing names by the proportion (prop) instead of the frequency (n) allows for more accurate comparisons of popularity between years, as the increasing population will result in greater frequencies of all names. Proportion normalizes population.

7. Using the object created in Question 5, what was the most popular name for both sexes in: a) the 2nd millennium? and b) the 3rd millennium?

```r
#select the 2nd millennium by removing the 3rd millennium
baby_no_n |> dplyr::filter(year < 2000) |>
  #consolidate the names
  dplyr::group_by(name) |>
  #add all of the proportions together
  #(I think this results in the most popular name?)
  dplyr::summarize(cumulative_prop = sum(prop)) |>
  #arrange in descending order
  dplyr::arrange(desc(cumulative_prop))
```

```
# A tibble: 68,676 x 2
   name    cumulative_prop
   <chr>             <dbl>
 1 John               5.25
 2 James              4.52
 3 Mary               4.51
 4 William            4.34
 5 Robert             3.79
 6 Charles            2.50
 7 Michael            2.27
 8 Joseph             2.22
 9 George             2.10
10 David              2.09
# i 68,666 more rows
```

```r
#select the 3rd millennium
baby_no_n |> dplyr::filter(year >= 2000) |>
  #consolidate the names
  dplyr::group_by(name) |>
  #add all of the proportions together
  dplyr::summarize(cumulative_prop = sum(prop)) |>
  #arrange in descending order
  dplyr::arrange(desc(cumulative_prop))
```

```
# A tibble: 67,063 x 2
   name    cumulative_prop
   <chr>             <dbl>
 1 Jacob             0.199
 2 Michael           0.179
 3 Emma              0.171
 4 Emily             0.163
 5 Ethan             0.157
 6 Matthew           0.157
 7 William           0.157
 8 Joshua            0.155
 9 Olivia            0.153
10 Daniel            0.151
# i 67,053 more rows
```

John was the most popular name in the 2nd millennium, and Jacob is the most popular name in the 3rd millennium so far.

8. What were the most popular names beginning with the letters Q, V, and X between 2000 and 2012?

```r
#select the years between 2000 and 2012
baby_no_n |> dplyr::filter(year >=2000 & year <=2012) |>
  #select the names starting with Q, V, and X using the stringr package
  dplyr::filter(stringr::str_starts(name, "Q") |
                stringr::str_starts(name, "V") |
                stringr::str_starts(name, "X")) |>
  #consolidate the names
  dplyr::group_by(name) |>
  #add all of the proportions together
  dplyr::summarize(cumulative_prop = sum(prop)) |>
  #arrange in descending order
  dplyr::arrange(desc(cumulative_prop))
```

```
# A tibble: 1,095 x 2
   name     cumulative_prop
   <chr>              <dbl>
 1 Victoria          0.0523
 2 Xavier            0.0328
 3 Vanessa           0.0255
 4 Victor            0.0236
 5 Vincent           0.0220
 6 Valeria           0.0169
 7 Valerie           0.0138
 8 Quinn             0.0116
 9 Vivian            0.0104
10 Veronica          0.0102
# i 1,085 more rows
```

The most popular names are shown above (Victoria, Xavier, Vanessa, etc)

9. Create a new object that retains all the variables of the babynames package, but create a new column that contains the decade each year is a part of named decade.

```r
#I originally started making an incredibly long case_when() call but decided
#there must be an easier way
#and found the floor() function
babynames_decade <- babynames |> dplyr::mutate(decade = floor(year/10) * 10)
slice(babynames_decade, 14990)
```

```
# A tibble: 1 x 6
   year sex   name      n     prop decade
  <dbl> <chr> <chr> <int>    <dbl>  <dbl>
1  1886 M     Amil      5 0.000042   1880
```

10. What is the mean and median number of female and male babies in each decade?

```r
#Mean Female:
babynames_decade |> dplyr::filter(sex == "F") |>
  dplyr::group_by(decade, year) |>
  dplyr::summarise(total_n = sum(n)) |>
  dplyr::group_by(decade) |>
  dplyr::summarize(mean_n = mean(total_n))
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 14 x 2
   decade   mean_n
    <dbl>    <dbl>
 1   1880  131269.
 2   1890  222156.
 3   1900  292759.
 4   1910  815631.
 5   1920 1195247.
 6   1930 1066217.
 7   1940 1448386.
 8   1950 1923218.
 9   1960 1826330.
10   1970 1545145.
11   1980 1717197.
12   1990 1800055.
13   2000 1846090.
14   2010 1759262.
```

```r
#Mean Male:
babynames_decade |> dplyr::filter(sex == "M") |>
  dplyr::group_by(decade, year) |>
  dplyr::summarise(total_n = sum(n)) |>
  dplyr::group_by(decade) |>
  dplyr::summarize(mean_n = mean(total_n))
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 14 x 2
   decade   mean_n
    <dbl>    <dbl>
 1   1880  109542.
 2   1890  114096.
 3   1900  135761.
 4   1910  667529.
 5   1920 1101954.
 6   1930 1056471.
 7   1940 1488444.
 8   1950 2020988.
 9   1960 1926507.
10   1970 1652341.
11   1980 1845544.
```

```
12   1990 1947317.
13   2000 1994394.
14   2010 1892570
```

```
#Median Female:
babynames_decade |> dplyr::filter(sex == "F") |>
  dplyr::group_by(decade, year) |>
  dplyr::summarise(total_n = sum(n)) |>
  dplyr::group_by(decade) |>
  dplyr::summarize(median_n = median(total_n))
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 14 x 2
   decade median_n
    <dbl>    <dbl>
 1   1880  131038.
 2   1890  227972
 3   1900  293461
 4   1910  872722.
 5   1920 1199632.
 6   1930 1064269
 7   1940 1372882.
 8   1950 1948446.
 9   1960 1829906.
10   1970 1526470.
11   1980 1704064.
12   1990 1779225
13   2000 1834728.
14   2010 1760622
```

```
#Median Male:
babynames_decade |> dplyr::filter(sex == "M") |>
  dplyr::group_by(decade, year) |>
  dplyr::summarise(total_n = sum(n)) |>
  dplyr::group_by(decade) |>
  dplyr::summarize(median_n = median(total_n))
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 14 x 2
   decade median_n
    <dbl>    <dbl>
 1   1880  110536.
 2   1890  114263
 3   1900  132735
 4   1910  751680.
 5   1920 1104540.
 6   1930 1042556.
 7   1940 1403910.
 8   1950 2048517
 9   1960 1927008
10   1970 1632478
11   1980 1827704
12   1990 1925422.
13   2000 1982272
14   2010 1894522
```

11. In which decade(s) and year(s), was:

a) your name the most popular? Years:

```
#Two interpretations: Which years does my name have the highest popularity:
babynames_decade |> dplyr::filter(name=="Owen") |> dplyr::arrange(desc(prop))
```

```
# A tibble: 179 x 6
    year sex   name      n    prop decade
   <dbl> <chr> <chr> <int>   <dbl>  <dbl>
 1  2016 M     Owen  10282 0.00510   2010
 2  2017 M     Owen   9312 0.00474   2010
 3  2015 M     Owen   9587 0.00470   2010
 4  2014 M     Owen   9143 0.00447   2010
 5  2013 M     Owen   8755 0.00434   2010
 6  2012 M     Owen   8689 0.00429   2010
 7  2011 M     Owen   8329 0.00411   2010
 8  2010 M     Owen   8174 0.00398   2010
 9  2009 M     Owen   8139 0.00384   2000
10  2006 M     Owen   8170 0.00373   2000
# i 169 more rows
```

```
#Which year is my name the most popular name (probably none of them):
babynames_decade |> dplyr::group_by(year) |> dplyr::filter(n == max(n)) |>
  dplyr::filter(name == "Owen")
```

```
# A tibble: 0 x 6
# Groups:   year [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

```
#as the output is blank, it is never the most popular
```

Decades:

```
#Approach 1:
babynames_decade |> dplyr::filter(name=="Owen") |> dplyr::group_by(decade, name) |>
  dplyr::summarize(cumulative_prop = sum(prop)) |>
  dplyr::arrange(desc(cumulative_prop))
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 14 x 3
# Groups:   decade [14]
   decade name  cumulative_prop
    <dbl> <chr>           <dbl>
 1   2010 Owen          0.0358
 2   2000 Owen          0.0290
 3   1880 Owen          0.00642
 4   1890 Owen          0.00618
 5   1900 Owen          0.00588
 6   1910 Owen          0.00558
 7   1920 Owen          0.00518
 8   1990 Owen          0.00475
 9   1930 Owen          0.00397
10   1940 Owen          0.00264
11   1980 Owen          0.00211
12   1950 Owen          0.00194
13   1970 Owen          0.00178
14   1960 Owen          0.00148
```

```
#Approach 2:
babynames_decade |> dplyr::group_by(decade) |> dplyr::filter(n == max(n)) |>
  dplyr::filter(name == "Owen")
```

```
# A tibble: 0 x 6
# Groups:   decade [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

b) your supervisor's name the most popular? Years:

```
babynames_decade |> dplyr::filter(name=="Anthea") |> dplyr::arrange(desc(prop))
```

```
# A tibble: 67 x 6
    year sex   name        n       prop decade
   <dbl> <chr> <chr>   <int>      <dbl>  <dbl>
 1  1977 F     Anthea     18 0.0000109    1970
 2  1978 F     Anthea     17 0.0000103    1970
 3  1973 F     Anthea     16 0.0000103    1970
 4  1981 F     Anthea     18 0.0000101    1980
 5  1982 F     Anthea     18 0.00000992   1980
 6  1990 F     Anthea     20 0.00000974   1990
 7  1974 F     Anthea     15 0.00000958   1970
 8  1983 F     Anthea     17 0.0000095    1980
 9  2000 F     Anthea     18 0.00000902   2000
10  1980 F     Anthea     16 0.00000899   1980
# i 57 more rows
```

```
#2:
babynames_decade |> dplyr::group_by(year) |> dplyr::filter(n == max(n)) |>
  dplyr::filter(name == "Anthea")
```

```
# A tibble: 0 x 6
# Groups:   year [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

Decades:

```
babynames_decade |> dplyr::filter(name=="Anthea") |>
  dplyr::group_by(decade, name) |>
  dplyr::summarize(cumulative_prop = sum(prop)) |>
  dplyr::arrange(desc(cumulative_prop))
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 9 x 3
# Groups:   decade [9]
  decade name    cumulative_prop
   <dbl> <chr>             <dbl>
1   1970 Anthea      0.0000801
2   1980 Anthea      0.0000751
3   1990 Anthea      0.0000573
4   2000 Anthea      0.0000573
5   1960 Anthea      0.0000566
6   2010 Anthea      0.0000358
7   1950 Anthea      0.0000242
8   1940 Anthea      0.00000894
9   1930 Anthea      0.00000452
```

```r
#2:
babynames_decade |> dplyr::group_by(decade) |>
  dplyr::filter(n == max(n)) |> dplyr::filter(name == "Anthea")
```

```
# A tibble: 0 x 6
# Groups:   decade [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

c) Mike's kids' names, Jack and Scott, the most popular? Years:

```r
#1.
(jack_years <- babynames_decade |> dplyr::filter(name == "Jack") |>
   dplyr::arrange(desc(prop)))
```

```
# A tibble: 256 x 6
    year sex   name      n   prop decade
   <dbl> <chr> <chr> <int>  <dbl>  <dbl>
 1  1927 M     Jack  12795 0.0110   1920
 2  1930 M     Jack  12431 0.0110   1930
 3  1929 M     Jack  12167 0.0110   1920
 4  1928 M     Jack  12494 0.0109   1920
 5  1931 M     Jack  11477 0.0107   1930
 6  1926 M     Jack  12201 0.0107   1920
 7  1925 M     Jack  12010 0.0104    1920
 8  1924 M     Jack  11924 0.0102    1920
 9  1932 M     Jack  10719 0.00998   1930
10  1923 M     Jack  11191 0.00988   1920
# i 246 more rows
```

```
(scott_years <- babynames_decade |> dplyr::filter(name == "Scott") |>
    dplyr::arrange(desc(prop)))
```

```
# A tibble: 196 x 6
    year sex   name      n   prop decade
   <dbl> <chr> <chr> <int>  <dbl>  <dbl>
 1  1971 M     Scott 30918 0.0170   1970
 2  1969 M     Scott 28687 0.0157   1960
 3  1970 M     Scott 28591 0.0150   1970
 4  1963 M     Scott 30415 0.0147   1960
 5  1968 M     Scott 26031 0.0147   1960
 6  1962 M     Scott 30707 0.0146   1960
 7  1967 M     Scott 25543 0.0144   1960
 8  1966 M     Scott 26033 0.0143   1960
 9  1964 M     Scott 28507 0.0141   1960
10  1972 M     Scott 22864 0.0137   1970
# i 186 more rows
```

```
#2.
babynames_decade |> dplyr::group_by(year) |> dplyr::filter(n == max(n)) |>
  dplyr::filter(name == "Jack")
```

```
# A tibble: 0 x 6
# Groups:   year [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

```
babynames_decade |> dplyr::group_by(year) |> dplyr::filter(n == max(n)) |>
  dplyr::filter(name == "Scott")
```

```
# A tibble: 0 x 6
# Groups:   year [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

Decades:

```r
(jack_decades <- babynames_decade |>
   dplyr::filter(name == "Jack") |>
   dplyr::group_by(decade, name) |>
   dplyr::summarize(cumulative_prop = sum(prop)) |>
   dplyr::arrange(desc(cumulative_prop))
)
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 14 x 3
# Groups:   decade [14]
   decade name  cumulative_prop
    <dbl> <chr>           <dbl>
 1   1920 Jack          0.103
 2   1930 Jack          0.0894
 3   1910 Jack          0.0708
 4   1900 Jack          0.0627
 5   1940 Jack          0.0497
 6   2000 Jack          0.0449
 7   1890 Jack          0.0376
 8   2010 Jack          0.0333
 9   1950 Jack          0.0307
10   1880 Jack          0.0228
11   1990 Jack          0.0184
12   1960 Jack          0.0180
13   1970 Jack          0.0109
14   1980 Jack          0.00834
```

```r
(scott_decades <- babynames_decade |>
    dplyr::filter(name == "Scott") |>
    dplyr::group_by(decade, name) |>
    dplyr::summarize(cumulative_prop = sum(prop)) |>
    dplyr::arrange(desc(cumulative_prop))
)
```

```
`summarise()` has grouped output by 'decade'. You can override using the
`.groups` argument.
```

```
# A tibble: 14 x 3
# Groups:   decade [14]
   decade name  cumulative_prop
    <dbl> <chr>           <dbl>
 1   1960 Scott         0.137
 2   1970 Scott         0.114
 3   1980 Scott         0.0568
 4   1950 Scott         0.0534
 5   1990 Scott         0.0237
 6   2000 Scott         0.00768
 7   1940 Scott         0.00742
 8   1880 Scott         0.00412
 9   1890 Scott         0.00309
10   2010 Scott         0.00285
11   1900 Scott         0.00237
12   1910 Scott         0.00199
13   1930 Scott         0.00182
14   1920 Scott         0.00168
```

```
#2.
babynames_decade |> dplyr::group_by(decade) |> dplyr::filter(n == max(n)) |>
  dplyr::filter(name == "Jack")
```

```
# A tibble: 0 x 6
# Groups:   decade [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

```
babynames_decade |> dplyr::group_by(decade) |> dplyr::filter(n == max(n)) |>
  dplyr::filter(name == "Scott")
```

```
# A tibble: 0 x 6
# Groups:   decade [0]
# i 6 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>,
#   decade <dbl>
```

The names in this question appear to never have been the most popular name for any given year or decade, meaning that approach one is likely the correct interpretation of the question.