CaseStudy_Babynames

Ni

6/11/2020

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
setwd("~/Dropbox/Coursera/RStudio/Data")
Babynames<-read.csv("BabyNames.csv")
#glimpse(Babynames)
```

```
# Remonve column and change the column name count to number
babynames<-Babynames %>%
  select(-rank) %>%
  rename(number=count)
```

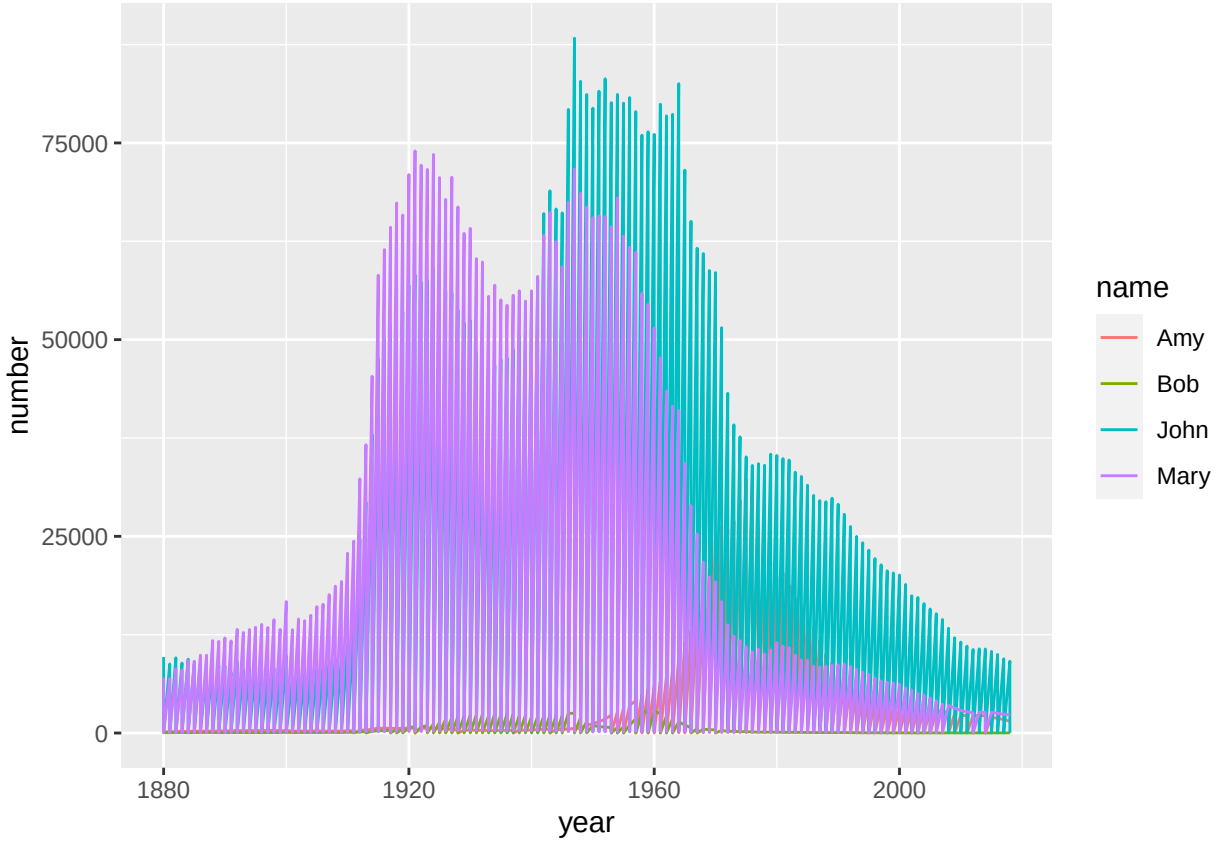
Find the most common name

```
most_com<-babynames%>%
  group_by(name)%>%
  top_n(1,number)
```

```
# Filtering and arranging
babynames_filter<-babynames %>%
  filter(year==2012) %>%
  arrange(desc(number))
```

Visualizing names with ggplot2

```
Babynames_selected<-babynames%>%
    filter(name %in% c("John","Mary","Bob","Amy"))
# Call ggplot with this object
ggplot(Babynames_selected,aes(x=year,y=number,color=name))+
  geom_line()
```




```
babynames_yeartotal<-babynames %>%
  select(name,year,number) %>%
  # The group_by tells dplyr that we only want to add up within each year
  group_by(year) %>%
  # mutate create a new column, the total number of baby born in that year
  mutate(year_total=sum(number))
  # Notice that from the header that the table is still grouped by year,
  # which could affect other verbs you will use in the future.
  # In particular, it can make other mutates or filters slower to run,
  # especially there are lots of groups in the table
head(babynames_yeartotal)
```

```
## # A tibble: 6 x 4
## # Groups:   year [3]
##   name   year number year_total
##   <chr> <int>  <int>      <int>
## 1 John   1880   9655     201484
## 2 Mary   1880   7065     201484
```

```
## 3 John   1881   8769     192696
## 4 Mary   1881   6919     192696
## 5 John   1882   9557     221533
## 6 Mary   1882   8148     221533
```

```
babynames_fraction<-babynames %>%
  select(name,year,number) %>%
  # The group_by tells dplyr that we only want to add up within each year
  group_by(year) %>%
  # mutate create a new column, the total number of baby born in that year
  mutate(year_total=sum(number)) %>%
  # so it's good practice to use ungroup()
  ungroup() %>%
  # now we can calculate the fraction
  mutate(fraction=number/year_total)
head(babynames_fraction)
```

```
## # A tibble: 6 x 5
##   name   year number year_total fraction
##   <chr> <int>  <int>      <int>    <dbl>
## 1 John   1880   9655     201484   0.0479
## 2 Mary   1880   7065     201484   0.0351
## 3 John   1881   8769     192696   0.0455
## 4 Mary   1881   6919     192696   0.0359
## 5 John   1882   9557     221533   0.0431
## 6 Mary   1882   8148     221533   0.0368
```

```
# You can see a different visualization
babynames_filterfraction<-babynames_fraction%>%
    filter(name =="John")
head(babynames_filterfraction)
```

```
## # A tibble: 6 x 5
##   name   year number year_total fraction
##   <chr> <int>  <int>      <int>    <dbl>
## 1 John   1880   9655     201484   0.0479
## 2 John   1881   8769     192696   0.0455
## 3 John   1882   9557     221533   0.0431
## 4 John   1883   8894     216946   0.0410
## 5 John   1884   9388     243462   0.0386
## 6 John   1885   8756     240854   0.0364
```

```
# Call ggplot with this object
#ggplot(babynames_fraction,aes(x=year,y=fraction))+
  #geom_line()
```

Find the year each name is most common

```
babynames_yearmostcom<-babynames %>%
  group_by(year) %>%
```

```
  mutate(year_total=sum(number)) %>%
  ungroup() %>%
  mutate(fraction=number/year_total) %>%
  group_by(name) %>%
  top_n(1,fraction)
head(babynames_yearmostcom)
```

```
## # A tibble: 6 x 6
## # Groups:   name [6]
##   name    sex    year number year_total fraction
##   <chr>   <chr> <int>  <int>      <int>    <dbl>
## 1 John    M      1880   9655     201484   0.0479
## 2 Mary    F      1891  11703     286672   0.0408
## 3 Robert  M      1937  61829    2130370   0.0290
## 4 Linda   F      1948  96209    3452217   0.0279
## 5 Lisa    F      1965  60266    3626029   0.0166
## 6 Michael M      1969  85208    3476215   0.0245
```

Add columns name_total and name_max for each name

```
nameTotalandnameMax<-babynames %>%
  group_by(name) %>%
  mutate(name_total = sum(number),
         name_max = max(number))
head(nameTotalandnameMax)
```

```
## # A tibble: 6 x 6
## # Groups:   name [2]
##   name  sex    year number name_total name_max
##   <chr> <chr> <int>  <int>      <int>    <int>
## 1 John  M      1880   9655    5146274    88318
## 2 Mary  F      1880   7065    4140687    73982
## 3 John  M      1881   8769    5146274    88318
## 4 Mary  F      1881   6919    4140687    73982
## 5 John  M      1882   9557    5146274    88318
## 6 Mary  F      1882   8148    4140687    73982
```

```
# This is also called normalizing
names_normalized<-babynames %>%
    group_by(name) %>%
    mutate(name_total = sum(number),
         name_max = max(number)) %>%
# ungroup the table
    ungroup() %>%
# Add the fraction_max column containing the number by the name maximum
  mutate(fraction_max = number / name_max)
```

```
# Filter names
names_filtered <- names_normalized %>%
  filter(name %in% c("Alice","Tom"))
```

```
# Visualize these names over time
ggplot(names_filtered, aes(x = year, y = fraction_max, color = name)) +
  geom_line()
```

Window function

- How do we visualize the biggest change within each name?
- To do so, need to find differences between each pair of consecutive years.
- The window function takes a vector, and return another vector of the same length
- lag()

```
# For example
v<-c(1,3,6,14)
# lag this vector, meaning move each vecotor to the right by one
lag(v)
```

```
## [1] NA  1  3  6
```

```
# NA 1 3 6
# NA is missing
# followed by 1,3 and 6, the item just prior to it in the original vector
# Compare consecutive steps
# What is each value once we've subtracted the previous one?
v-lag(v)
```

```
## [1] NA  2  3  8
```

Now that we know how to calculate the difference between consecutive values in a vector, we can use that in a grouped mutate to find the changes in the popularity of one name in consecutive years

```
filterJohn1<-babynames_fraction %>%
  filter(name=="John")  %>%
  arrange(year)
head(filterJohn1)
```

```
## # A tibble: 6 x 5
##   name   year number year_total fraction
##   <chr> <int>  <int>      <int>    <dbl>
## 1 John   1880   9655     201484 0.0479
## 2 John   1880     46     201484 0.000228
## 3 John   1881   8769     192696 0.0455
## 4 John   1881     26     192696 0.000135
## 5 John   1882   9557     221533 0.0431
## 6 John   1882     40     221533 0.000181
```

You can see the fraction of babies born each year that are named John, you can also compare between year

```
filterJohn2<-babynames_fraction %>%
  filter(name=="John")  %>%
  arrange(year) %>%
# Use the lag window function
  mutate(difference=fraction-lag(fraction))
head(filterJohn2)
```

```
## # A tibble: 6 x 6
##   name   year number year_total fraction difference
##   <chr> <int>  <int>      <int>    <dbl>      <dbl>
## 1 John   1880   9655     201484 0.0479      NA
## 2 John   1880     46     201484 0.000228    -0.0477
## 3 John   1881   8769     192696 0.0455       0.0453
## 4 John   1881     26     192696 0.000135    -0.0454
## 5 John   1882   9557     221533 0.0431       0.0430
## 6 John   1882     40     221533 0.000181    -0.0430
```

Notice the first observation is missing because there is no previous year.

After that, we can see the name Matthew went up or down within each year.

The biggest jump of that name

What if we want to know the biggest jump that the name Matthew took in popularity?

We could sort in descending order of the difference column.

```
filterJohn3<-babynames_fraction %>%
  filter(name=="John")  %>%
  arrange(year) %>%
# Use the lag window function
  mutate(difference=fraction-lag(fraction)) %>%
# To see the biggest jump
  arrange(desc(difference))
head(filterJohn3)
```

```
## # A tibble: 6 x 6
##   name   year number year_total fraction difference
##   <chr> <int>  <int>      <int>    <dbl>      <dbl>
## 1 John   1881   8769     192696   0.0455     0.0453
## 2 John   1882   9557     221533   0.0431     0.0430
## 3 John   1883   8894     216946   0.0410     0.0408
## 4 John   1884   9388     243462   0.0386     0.0384
## 5 John   1885   8756     240854   0.0364     0.0362
## 6 John   1886   9026     255317   0.0354     0.0352
```

We can see the biggest jumps when John got much more popular, were in 1881, 1882 and 1883.

The changes within every name

```
changesInEveryName<-babynames_fraction %>%
  arrange(name, year)  %>%
  mutate(difference = fraction - lag(fraction))  %>%
  group_by(name)  %>%
  arrange(desc(difference))
head(changesInEveryName)
```

```
## # A tibble: 6 x 6
## # Groups:   name [2]
##   name     year number year_total fraction difference
##   <chr>   <int>  <int>      <int>    <dbl>      <dbl>
## 1 John     1880   9655     201484   0.0479     0.0479
## 2 William  1880   9532     201484   0.0473     0.0473
## 3 John     1881   8769     192696   0.0455     0.0453
## 4 William  1881   8524     192696   0.0442     0.0441
## 5 John     1882   9557     221533   0.0431     0.0430
## 6 William  1882   9298     221533   0.0420     0.0418
```

Using the ratio to describe the frequency of a name

```
addRatioCol<-babynames_fraction %>%
# Arrange the data in order of name, then year
  arrange(name, year)  %>%
# Group the data by name
```

```
  group_by(name)  %>%
# Add a ratio column that contains the ratio between each year
  mutate(ratio = fraction / lag(fraction))
head(addRatioCol)
```

```
## # A tibble: 6 x 6
## # Groups:   name [1]
##   name   year number year_total   fraction ratio
##   <chr> <int>  <int>      <int>      <dbl> <dbl>
## 1 Aaban  2007      5    3994007 0.00000125 NA
## 2 Aaban  2009      6    3815638 0.00000157  1.26
## 3 Aaban  2010      9    3690700 0.00000244  1.55
## 4 Aaban  2011     11    3651914 0.00000301  1.24
## 5 Aaban  2012     11    3650462 0.00000301  1.00
## 6 Aaban  2013     14    3637310 0.00000385  1.28
```

```
babynames_ratios_filtered<-babynames_fraction %>%
# Arrange the data in order of name, then year
  arrange(name, year)  %>%
# Group the data by name
  group_by(name)  %>%
# Add a ratio column that contains the ratio between each year
# A ratio column to describe the ratio of the frequency of a baby name between consecutive years to des
  mutate(ratio = fraction / lag(fraction)) %>%
# filter
  filter(fraction >= 0.00001)%>%
# Extract the largest ratio from each name
  top_n(1, ratio) %>%
# Sort the ratio column in descending order
  arrange(desc(ratio)) %>%
# Filter for fractions greater than or equal to 0.001
  filter(fraction >= 0.001)
head(babynames_ratios_filtered)
```

```
## # A tibble: 6 x 6
## # Groups:   name [6]
##   name      year number year_total fraction ratio
##   <chr>    <int>  <int>      <int>    <dbl> <dbl>
## 1 Sophia    2010  20639    3690700  0.00559 2371.
## 2 Olivia    2001  13978    3741451  0.00374 2352.
## 3 Luke      2014  10503    3696311  0.00284 2067.
## 4 Isabella  2002  12166    3736042  0.00326 2031.
## 5 Emma      2016  19471    3652968  0.00533 1966.
## 6 Isaac     2011   9597    3651914  0.00263 1940.
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

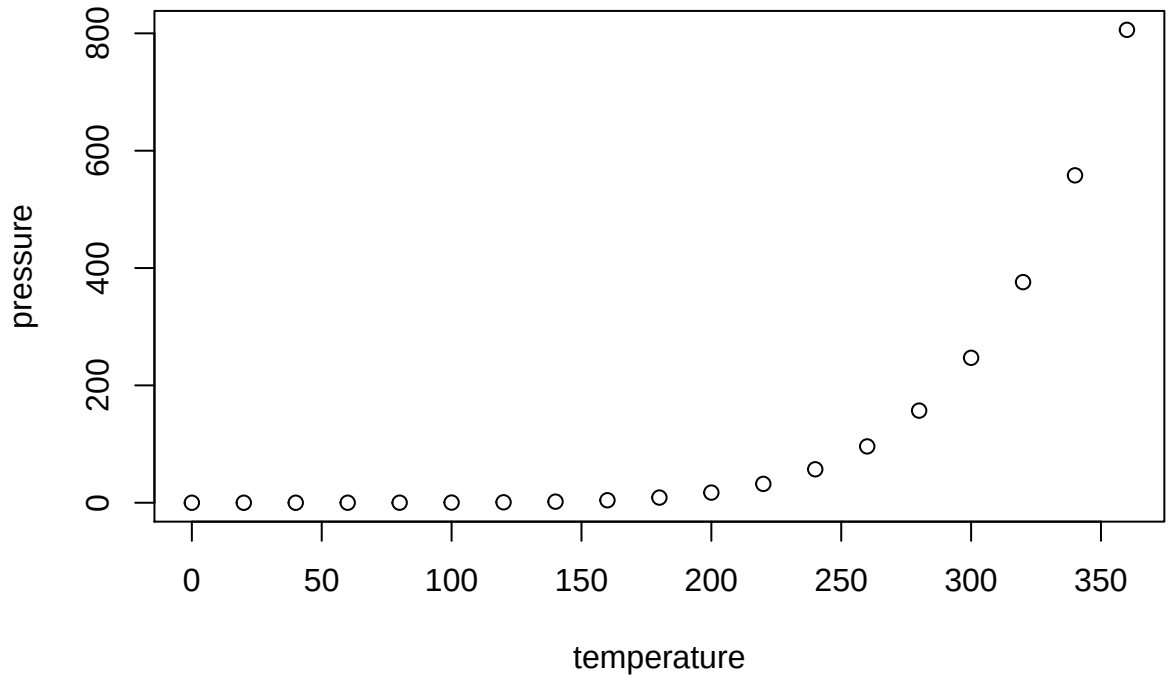
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.