All Training materials are provided "as is" and without warranty and RStudio disclaims any and all express and implied warranties including without limitation the implied warranties of title, fitness for a particular purpose, merchantability and noninfringement.

Studio

Transforming Data

Turn data into information with dplyr and base R

Garrett Grolemund

Master Instructor, RStudio

August 2014

1. Baby names data
2. Subsetting data sets
3. Transform and reorder data
4. Join data sets
5. Groupwise operations

Baby names data

Baby names

Top 1000 male and female baby names in the US, from 1880 to 2008.

258,000 records (1000 * 2 * 129)

But only five variables: year, name, soundex, sex and prop.

```
library(ggplot2)
```

Make sure the data sets are in your working directory

Prevent R from reading in strings as factors (the default)

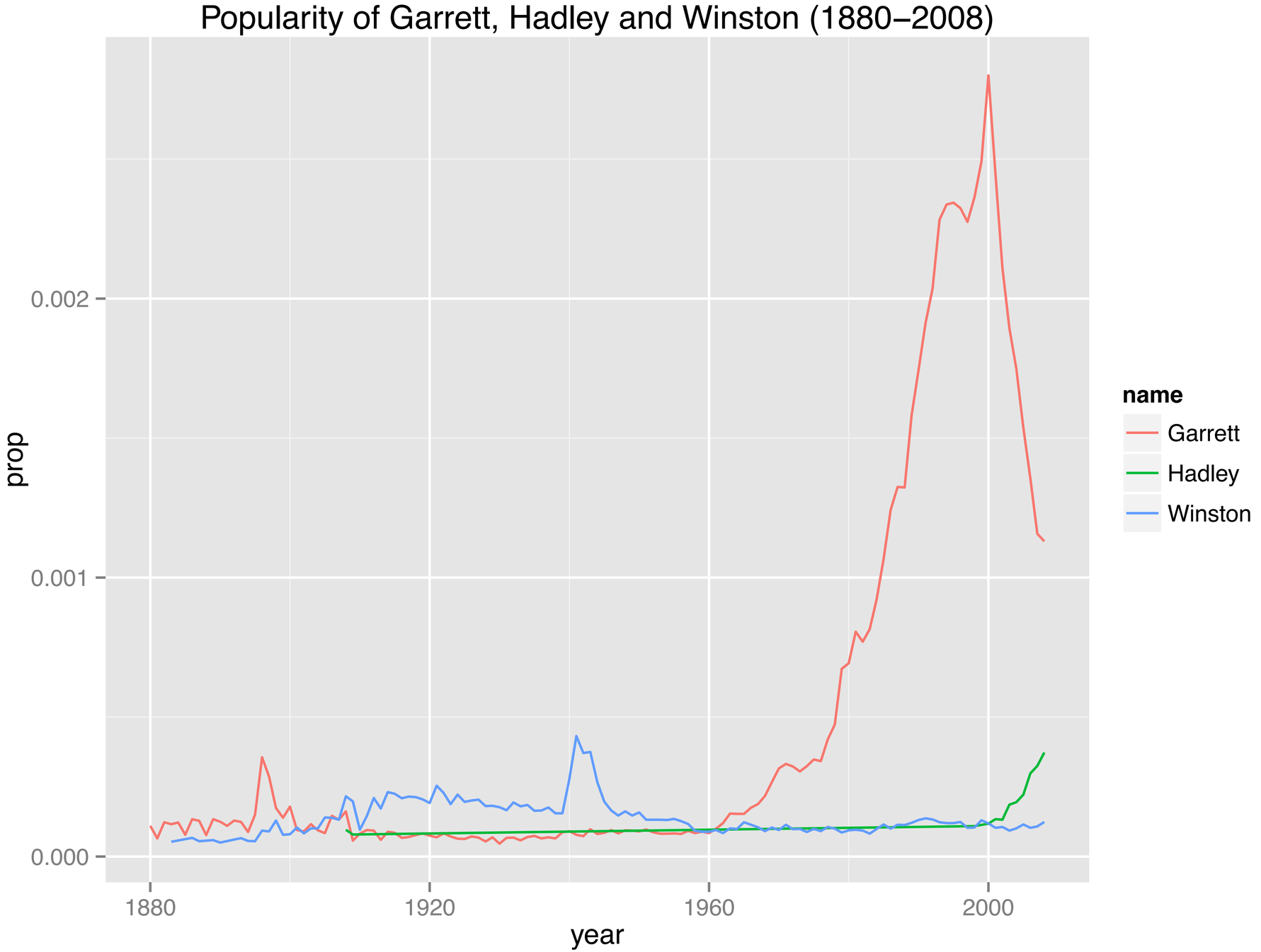
```
options(stringsAsFactors = FALSE)

bnames <- read.csv("data/bnames.csv.bz2")
births <- read.csv("data/births.csv")
```

```
head(bnames)
```

```
  year    name      prop sex soundex
1 1880    John 0.081541 boy    J500
2 1880 William 0.080511 boy    W450
3 1880   James 0.050057 boy    J520
4 1880 Charles 0.045167 boy    C642
5 1880  George 0.043292 boy    G620
```

```
tail(bnames)
```

```
        year     name      prop  sex soundex
257996 2008 Carleigh 0.000128 girl    C642
257997 2008    Iyana 0.000128 girl    I500
257998 2008   Kenley 0.000127 girl    K540
257999 2008   Sloane 0.000127 girl    S450
258000 2008  Elianna 0.000127 girl    E450
```



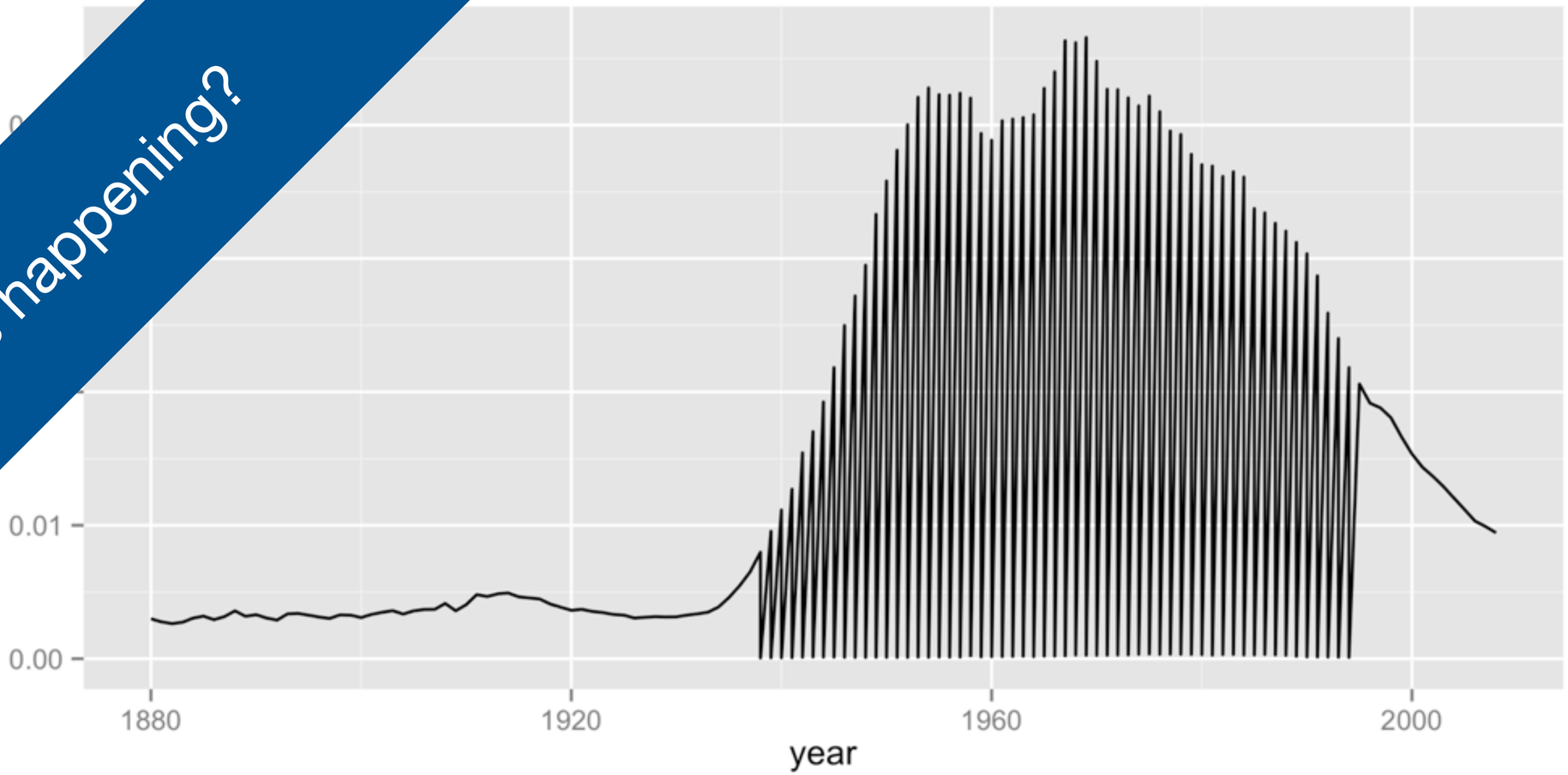
Your turn

Use logical subsetting to extract your name from the dataset. Plot the trend over time.

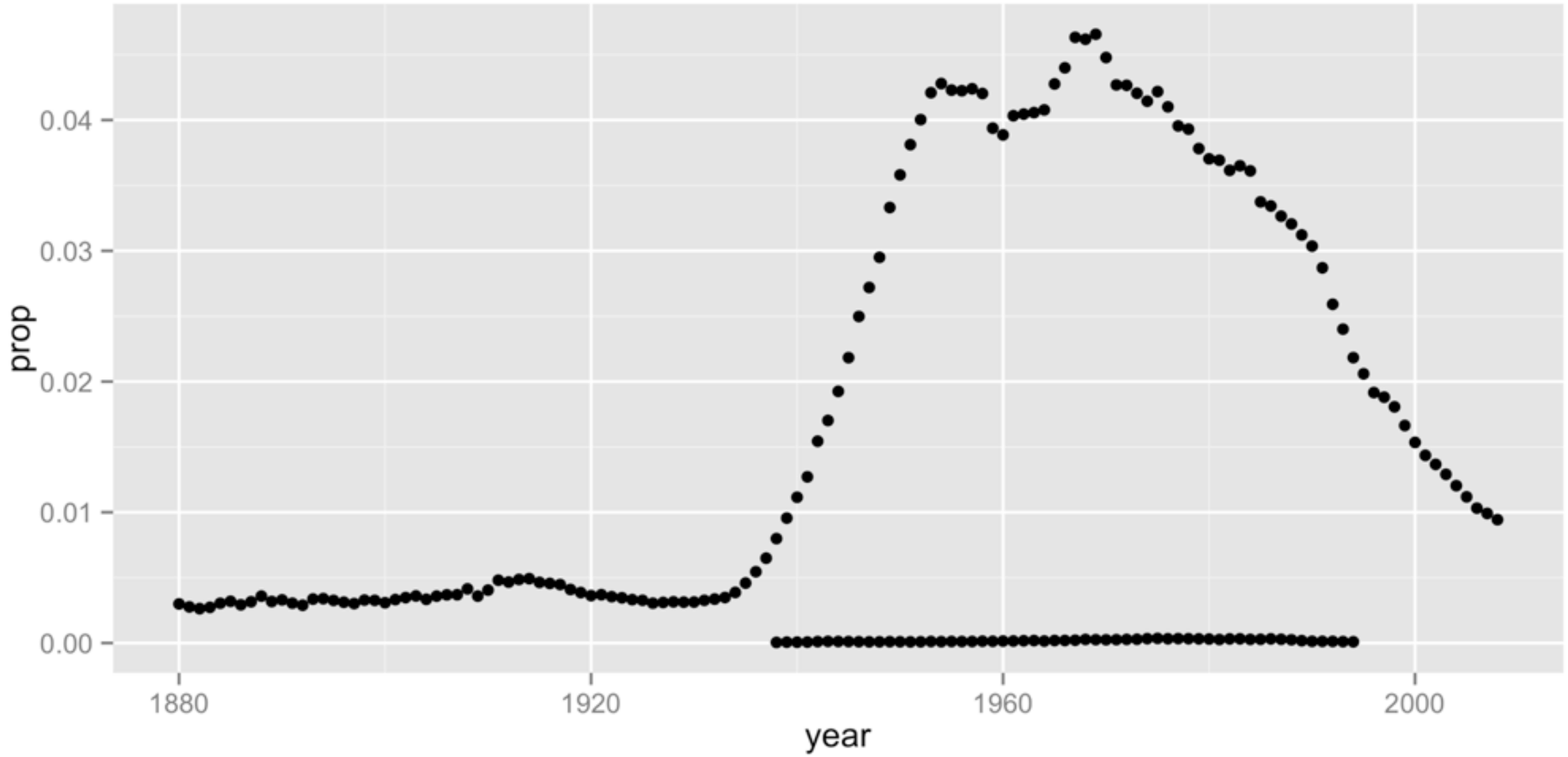
What geom should you use? Do you need any extra aesthetics?

```
garrett <- bnames[bnames$name == "Garrett", ]

qplot(year, prop, data = garrett, geom = "line")
```

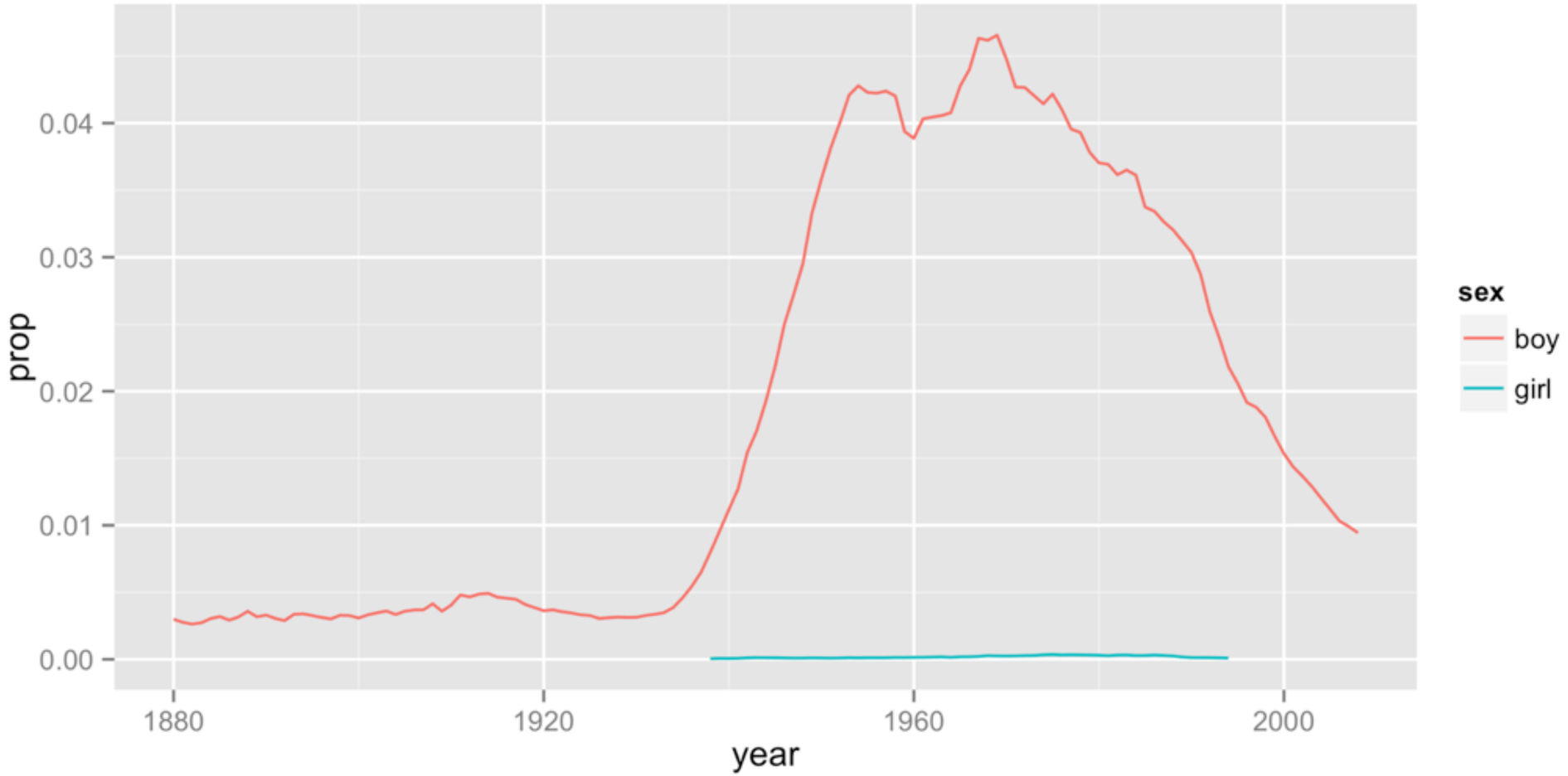
What is happening?




```
michael <- bnames[bnames$name == "Michael", ]
qplot(year, prop, data = michael, geom = "line")
```

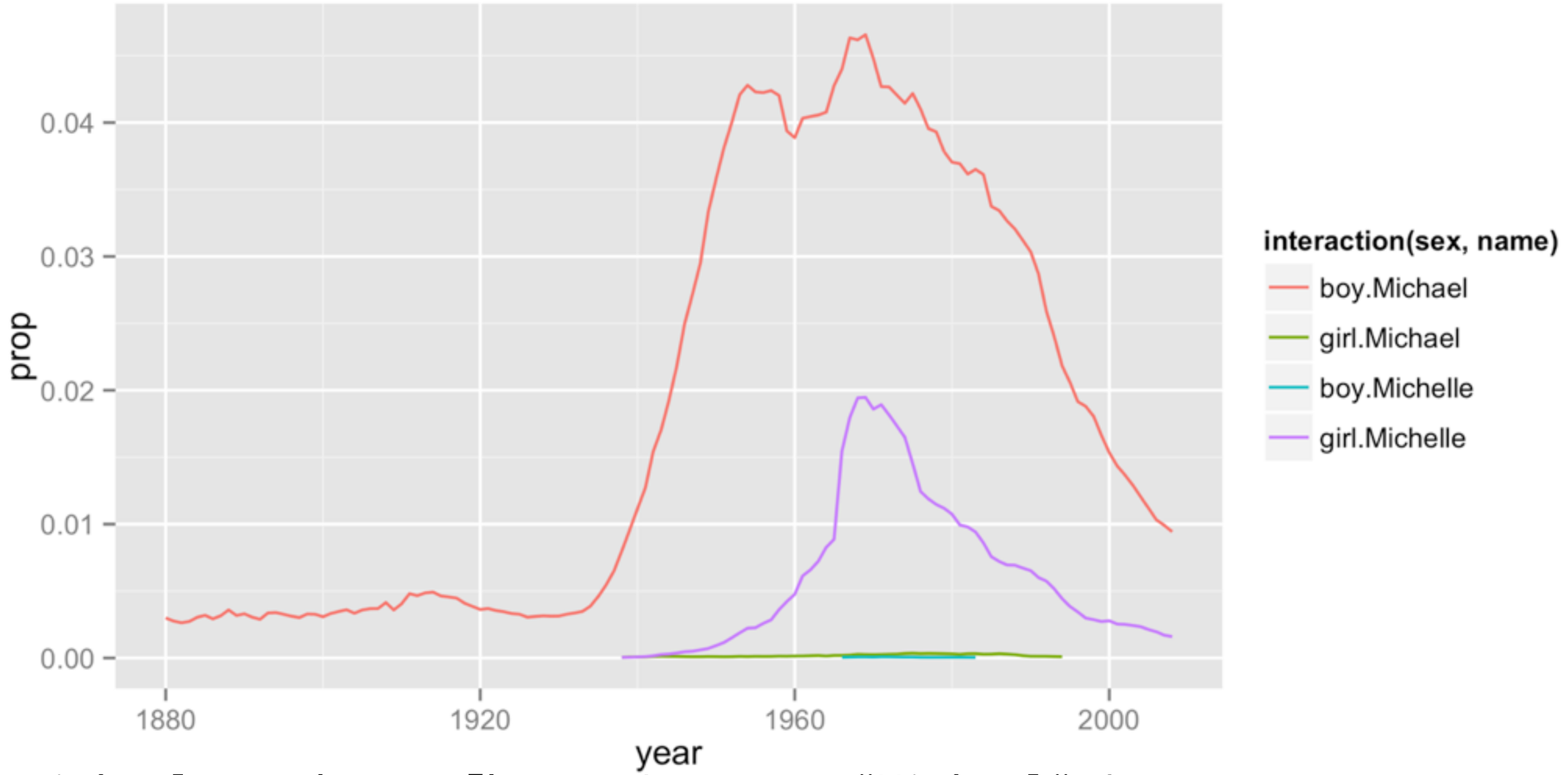



```
qplot(year, prop, data = michael, geom = "point")
```



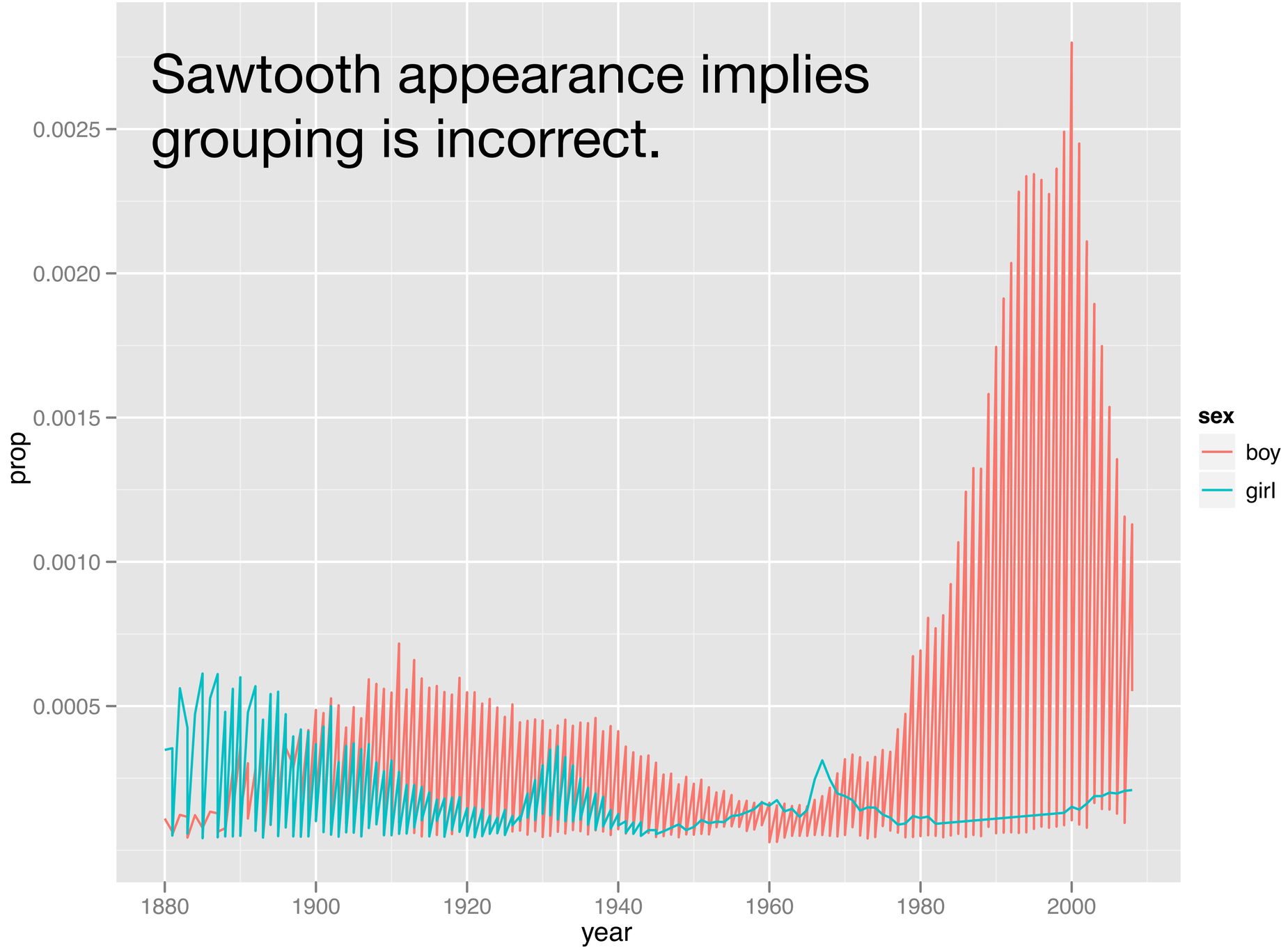

```
qplot(year, prop, data = michael, geom = "line",
  color = sex)
```

creates a different colored line for each group of sex (male, female)




```
michaels <- bnames[bnames$name == "Michael" |
                   bnames$name == "Michelle", ]
qplot(year, prop, data = michaels, geom = "line",
  color = interaction(sex, name))
```

use interaction to group on the combination of two variables



dplyr

dplyr

An R package to manipulate data

- easier
- faster!

```
# install.packages("dplyr")
library(dplyr)
```

filter
select
mutate
summarise
arrange

Structure

- First argument is a data frame
- Subsequent arguments say what to do with data frame
- Always return a data frame
- (Never modify in place)

```
tbl <- data.frame(
  color = c("blue", "black", "blue", "blue", "black"),
  value = 1:5)
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

color	value
blue	1
blue	3
blue	4

```
filter(tbl, color == "blue")
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

color	value
blue	1
blue	4

```
filter(tbl, value %in% c(1, 4))
```

Quiz

Use filter to find all of the girls names from a year that ends in `00

```
filter(bnames, sex == "girl" & (year == 1900 | year ==
2000))
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

color
blue
black
blue
blue
black

```
select(tbl, color)
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

value
1
2
3
4
5

```
select(tbl, -color)
```

- **filter**: pick rows by criteria
- **select**: pick columns by name

tbl

color	value
4	1
1	2
5	3
3	4
2	5

color	value
1	2
2	5
3	4
4	1
5	3

```
arrange(tbl, color)
```

tbl

color	value
4	1
1	2
5	3
3	4
2	5

color	value
5	3
4	1
3	4
2	5
1	2

```
arrange(tbl, desc(color))
```

Your turn

Reorder the rows from highest to lowest `prop`.

Which name was the most popular in a single year?

In what year was your name the most popular (hint: use the data set with just your name)

```
arrange(bnames, desc(prop)) # John in 1880
arrange(garrett, desc(prop)) # 2000
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

color	value	double
blue	1	2
black	2	4
blue	3	6
blue	4	8
black	5	10

```
mutate(tbl, double = 2 * value)
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

color	value	double	quadruple
blue	1	2	4
black	2	4	8
blue	3	6	12
blue	4	8	16
black	5	10	20

```
mutate(tbl, double = 2 * value,
       quadruple = 2 * double)
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

total
15

```
summarise(tbl, total = sum(value))
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

total	avg
15	3

```
summarise(tbl, total = sum(value),
          avg = mean(value))
```

- **mutate**: add new variables
- **summarise**: reduce variables to values
- **arrange**: reorder rows

Your turn

With the data frame containing your name:

1. Add a new column to the data that changes the `prop` to a percentage
2. Create a summary that displays the `min`, `mean`, and `max prop` for your name.

```
mutate(garrett, perc = prop * 100)

summarise(garrett,
  min = min(prop),
  mean = mean(prop),
  max = max(prop))
```

Summary

- **filter**: pick rows matching criteria
- **select**: pick columns by name
- **arrange**: reorder rows
- **mutate**: add new variables
- **summarise**: reduce variables to values

Joining data sets

Your turn

Why might prop be a bad way to compare names across different years?

Births

The number of boys and girls born each year from 1880 - 2009

260 records (2 * 130)

Three variables: year, sex, births

```
head(bnames)
  year     name      prop sex soundex
1 1880     John 0.081541 boy    J500
2 1880  William 0.080511 boy    W450
3 1880    James 0.050057 boy    J520
4 1880  Charles 0.045167 boy    C642
5 1880   George 0.043292 boy    G620
6 1880    Frank 0.027380 boy    F652
```

```
head(births)
  year sex births
1 1880 boy 118405
2 1881 boy 108290
3 1882 boy 122034
4 1883 boy 112487
5 1884 boy 122745
6 1885 boy 115948
```

How would you combine these data sets? Describe a strategy.

Joining datasets

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

+

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

= ?

```
x <- data.frame(
  name = c("John", "Paul", "George", "Ringo", "Stuart", "Pete"),
  instrument = c("guitar", "bass", "guitar", "drums", "bass",
     "drums"))

y <- data.frame(
  name = c("John", "Paul", "George", "Ringo", "Brian"),
  band = c("TRUE", "TRUE", "TRUE",  "TRUE", "FALSE"))
```

x

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

+

y

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

=

name	instrument	band
John	guitar	T
Paul	bass	T
George	guitar	T
Ringo	drums	T
Stuart	bass	NA
Pete	drums	NA

```
left_join(x, y, by = "name")
```

x

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

+

y

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

=

name	instrument	band
John	guitar	T
Paul	bass	T
George	guitar	T
Ringo	drums	T

```
inner_join(x, y, by = "name")
```

x

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

+

y

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

=

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums

```
semi_join(x, y, by = "name")
```

x

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

+

y

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

=

name	instrument
Stuart	bass
Pete	drums

```
anti_join(x, y, by = "name")
```

Type	Action
left_join	Include all of x, and matching rows of y
inner_join	Include rows of x that appear in y, and matching rows of y
semi_join	Include rows of x that appear in y
anti_join	Include rows of x that *do not* appear in y

Your turn

Combine bnames with births, and then create a new column that shows the total number of babies born each year for each name.

```
bnames2 <- left_join(bnames, births, by = c("year",
"sex"))

bnames2 <- mutate(bnames2, n = prop * births)
bnames2

bnames2 <- mutate(bnames2, n = round(prop * births))
bnames2
```

bnames2

```
##    year     name      prop sex soundex births    n
## 1  1880     John 0.081541 boy    J500 118405 9655
## 2  1880  William 0.080511 boy    W450 118405 9533
## 3  1880    James 0.050057 boy    J520 118405 5927
## 4  1880  Charles 0.045167 boy    C642 118405 5348
## 5  1880   George 0.043292 boy    G620 118405 5126
## 6  1880    Frank 0.027380 boy    F652 118405 3242
## 7  1880   Joseph 0.022229 boy    J210 118405 2632
## 8  1880   Thomas 0.021401 boy    T520 118405 2534
## 9  1880    Henry 0.020641 boy    H560 118405 2444
## 10 1880   Robert 0.020404 boy    R163 118405 2416
## ..  ...      ...      ... ...     ...    ...  ...
```

Group wise operations

Total number of people per name

	name	total
1	Aaden	959
2	Aaliyah	39665
3	Aarav	219
4	Aaron	509464
5	Ab	25
6	Abagail	2682
7	Abb	16
8	Abbey	14348
9	Abbie	16622
10	Abbigail	6800

Do we have enough information to calculate this?

Your turn

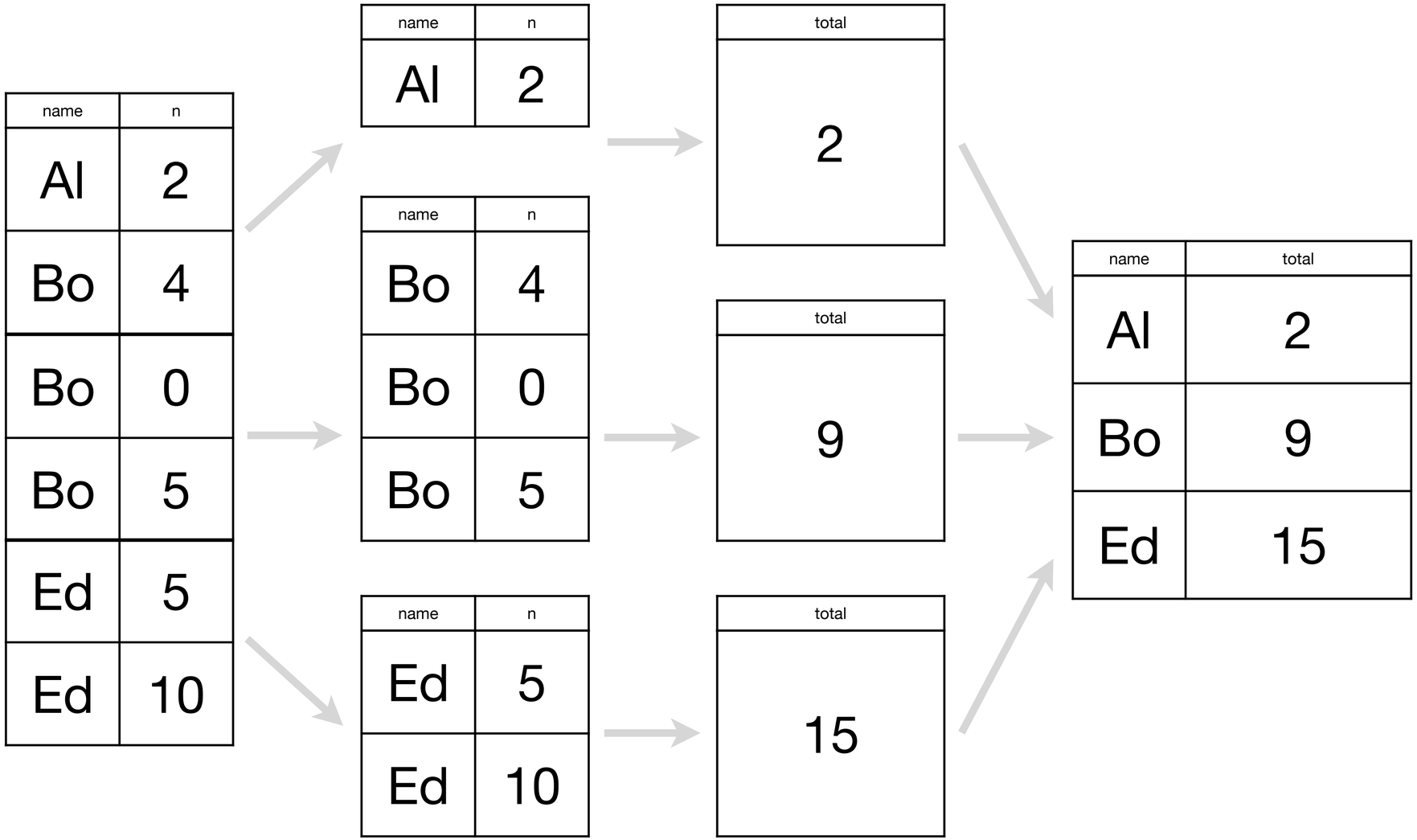
Calculate the total for a single name (e.g, your name).

Then devise a strategy for calculating the total for *every* name.

	name	total
1	Aaden	959
2	Aaliyah	39665
3	Aarav	219
4	Aaron	509464
5	Ab	25
6	Abagail	2682
7	Abb	16
8	Abbey	14348
9	Abbie	16622
10	Abbigail	6800

```
garrett <- filter(bnames2, name == "Garrett")
sum(garrett$n)

# Or
summarise(garrett, total = sum(n))

# But how could we do this for every name?
```

Split Apply Combine




```
summarise(by_name, total = sum(n))
```

group_by

```
group_by(bnames2, name)
## Source: local dat
## Groups: name

##    year    name       prop sex soundex births    n
## 1  1880    John 0.081541 boy    J500 118405 9655
## 2  1880 William 0.080511 boy    W450 118405 9533
## 3  1880   James 0.050057 boy    J520 118405 5927
## 4  1880 Charles 0.045167 boy    C642 118405 5348
## 5  1880  George 0.043292 boy    G620 118405 5126
## 6  1880   Frank 0.027380 boy    F652 118405 3242
## 7  1880  Joseph 0.022229 boy    J210 118405 2632
## 8  1880  Thomas 0.021401 boy    T520 118405 2534
## 9  1880   Henry 0.020641 boy    H560 118405 2444
## 10 1880  Robert 0.020404 boy    R163 118405 2416
## ..  ...     ...      ... ...     ...    ...  ...
```

mutate, summarise, and arrange will execute groupwise on this variable

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

total
15

```
summarise(tbl, total = sum(value))
```

tbl

color	value
blue	1
black	2
blue	3
blue	4
black	5

color	total
blue	8
black	7

```
by_color <- group_by(tbl, color)
summarise(by_color, total = sum(value))
```

```
by_name <- group_by(bnames2, name)
totals <- summarise(by_name, total = sum(n))
totals
```

```
## Source: local data frame [6,782 x 2]

##         name  total
## 1      Aaden    959
## 2    Aaliyah  39665
## 3      Aarav    219
## 4      Aaron 509464
## 5         Ab     25
## 6    Abagail   2682
## 7        Abb     16
## 8      Abbey  14348
## ..       ...    ...
```

tbl

tbl_df

tbl is a special case of data frame that can be manipulated more easily

```
bnames <- tbl_df(bnames)
births <- tbl_df(births)

class(bnames)
## [1] "tbl_df"     "tbl"          "d
```

Always start by making your data tbls

R will always show just the part of the tbl that fits the console

Console ~/Dropbox (RStudio)/RStudio/rstudio-training/in-person-intro/Two-

```
> tbl_df(diamonds)
Source: local data frame [53,940 x 10]

   carat       cut color clarity depth table price
1   0.23     Ideal     E     SI2  61.5    55   326
2   0.21   Premium     E     SI1  59.8    61   326
3   0.23      Good     E     VS1  56.9    65   327
4   0.29   Premium     I     VS2  62.4    58   334
5   0.31      Good     J     SI2  63.3    58   335
6   0.24 Very Good     J    VVS2  62.8    57   336
7   0.24 Very Good     I    VVS1  62.3    57   336
8   0.26 Very Good     H     SI1  61.9    55   337
9   0.22      Fair     E     VS2  65.1    61   337
10  0.23 Very Good     H     VS1  59.4    61   338
..   ...       ...   ...     ...   ...   ...   ...
Variables not shown: x (dbl), y (dbl), z (dbl)
>
```

Console ~/Dropbox (RStudio)/RStudio/rstudio-training/

```
> tbl_df(diamonds)
Source: local data frame [53,940 x 10]

   carat       cut color clarity
1   0.23     Ideal     E     SI2
2   0.21   Premium     E     SI1
3   0.23      Good     E     VS1
4   0.29   Premium     I     VS2
5   0.31      Good     J     SI2
6   0.24 Very Good     J    VVS2
7   0.24 Very Good     I    VVS1
8   0.26 Very Good     H     SI1
9   0.22      Fair     E     VS2
10  0.23 Very Good     H     VS1
..   ...       ...   ...     ...
Variables not shown: depth (dbl),
  table (dbl), price (int), x
  (dbl), y (dbl), z (dbl)
>
```

Use `View()` to see more

Interactions

Interactions

Use multiple variables to create groups based on the interaction of variables

```
group_by(bnames2, name, sex)
## Source: local data fram
## Groups: name, sex

##    year    name      prop sex soundex births     n
## 1  1880    John 0.081541 boy    J500 118405 9655
## 2  1880 William 0.080511 boy    W450 118405 9533
## 3  1880   James 0.050057 boy    J520 118405 5927
## 4  1880 Charles 0.045167 boy    C642 118405 5348
## ..  ...     ...      ... ...     ...    ...  ...
```

dplyr will treat each unique combination of these values as a separate group

Interactions

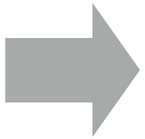
```
by_name <- group_by(bnames2, name)
group_by(by_name, sex)
## Source: local data frame [258,000 x 7]
## Groups: name, sex

##    year     name      prop sex soundex births    n
## 1  1880     John 0.081541 boy    J500 118405 9655
## 2  1880  William 0.080511 boy    W450 118405 9533
## 3  1880    James 0.050057 boy    J520 118405 5927
## 4  1880  Charles 0.045167 boy    C642 118405 5348
## ..  ...      ...      ... ...     ...    ...  ...
```

```
name_sex <- group_by(bnames2, name, sex)
totals2 <- summarise(name_sex, total = sum(n))
totals2
## Source: local data frame [7,455 x 3]
## Groups: name
##
##        name  sex  total
## 1     Aaden  boy    959
## 2  Aaliyah girl  39665
## 3     Aarav  boy    219
## 4     Aaron  boy 508094
## 5     Aaron girl   1370
## 6        Ab  boy     25
## 7  Abagail girl   2682
## ..      ...  ...    ...
```

summarise returns a data frame that has one less level of grouping

“Unwrap” groups with summarise

```
name_sex <- group_by(bnames2, name, sex)
totals2 <- summarise(name_sex, total = sum(n))
totals2
## Source: local data frame [7,455 x 3]
## Groups: name
##
##       name  sex  total
## 1    Aaden  boy    959
## 2  Aaliyah girl  39665
## 3    Aarav  boy    219
## 4    Aaron  boy 508094
## 5    Aaron girl   1370
## 6       Ab  boy     25
## 7  Abagail girl   2682
## ..     ...  ...    ...
```



```
summarise(totals2, total = sum(total))
Source: local data frame [6,782 x 2]

       name  total
1     Aaden    959
2   Aaliyah  39665
3     Aarav    219
4     Aaron 509464
5        Ab     25
6   Abagail   2682
7       Abb     16
8     Abbey  14348
9     Abbie  16622
10 Abbigail   6800
..      ...    ...
```

Ungroup

Use `ungroup` to remove group specifications.

```
by_name_sex <- group_by(bnames2, name, sex)
ungroup(by_name_sex)
## Source: local data frame [258,000 x 7]

##    year    name     prop sex soundex births    n
## 1  1880    John 0.081541 boy    J500 118405 9655
## 2  1880 William 0.080511 boy    W450 118405 9533
## 3  1880   James 0.050057 boy    J520 118405 5927
## 4  1880 Charles 0.045167 boy    C642 118405 5348
## ..  ...     ...      ... ...     ...    ...  ...
```

Summary functions

- `min(x), median(x), max(x), quantile(x, p)`
- `n(), nth(), n_distinct()`
- `sd(x), var(x), iqr(x), mad(x)`

Practice

Your turn

Calculate the total number of babies in each soundex

```
by_soundex <- group_by(bnames2, soundex)
stotals <- summarise(by_soundex, total = sum(n))
stotals
```

```
## Source: local data frame [1,392 x 2]

##    soundex  total
## 1     A000     11
## 2     A100 193837
## 3     A120  15652
## 4     A124 256458
## 5     A130     11
## 6     A134   5181
## 7     A135    901
## 8     A140  30815
## ..     ...    ...
```

```
# What is the most popular sound?
arrange(stotals, desc(total))
```

```
# What is the most popular sound?
arrange(stotals, desc(total))
## Source: local data frame [1,392 x 2]

##    soundex   total
## 1     J500 9991737
## 2     M240 5823791
## 3     M600 5553703
## 4     J520 5524958
## 5     R163 5047182
## 6     W450 4116109
## 7     C623 4016919
## 8     J200 3859240
## 9     D500 3774287
## 10    D130 3506995
## ..     ...     ...
```

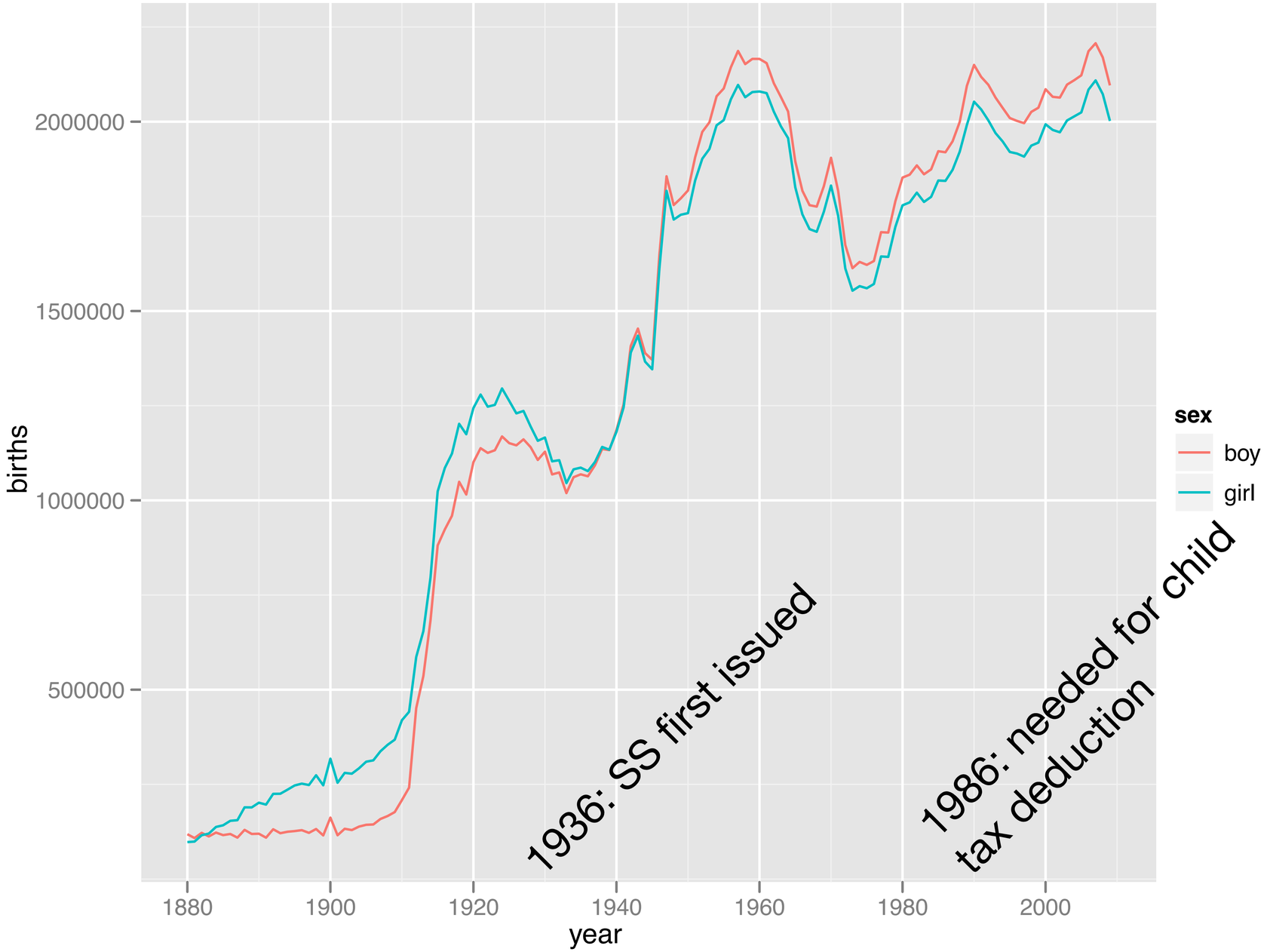
```
# What is the most popular sound?
j500 <- filter(bnames, soundex == "J500")
unique(j500$name)
##  [1] "John"     "Jim"      "Juan"     "Jimmie"   "Johnnie"  "Johnny"
##  [7] "Johnie"   "Jean"     "June"     "Jonah"    "Jennie"   "Jimmy"
## [13] "Johny"    "Jonnie"   "Johney"   "Jamie"    "Jon"      "Joan"
## [19] "Jan"      "Jame"     "Jaime"    "Jamey"    "Jaimie"   "Jammie"
## [25] "Jayme"    "Juwan"    "Johan"    "Jaheim"   "Jahiem"   "Jaheem"
## [31] "Jane"     "Janie"    "Johanna"  "Joanna"   "Jannie"   "Jenny"
## [37] "Jeanne"   "Johannah" "Juana"    "Junie"    "Jinnie"   "Jeanie"
## [43] "Jeannie"  "Junia"    "Janey"    "Jeane"    "Joanne"   "Joann"
## [49] "Jayne"    "Jana"     "Janna"    "Jann"     "Joni"     "Joanie"
## [55] "Jeanna"   "Jami"     "Johnna"   "Jeana"    "Jonna"    "Jena"
## [61] "Jenni"    "Jenna"    "Janae"    "Jaimee"   "Janay"    "Joana"
## [67] "Janiya"   "Johana"   "Jamya"    "Janiyah"  "Janiah"   "Jamiya"
```

Your turn

Calculate the total number of boys and the total number of girls for each year

```
year_sex <- group_by(bnames2, year, sex)
ytotals <- summarise(year_sex, births = sum(n))
ytotals
```

```
Source: local data frame [258 x 3]
Groups: year

   year  sex births
1  1880  boy 110207
2  1880 girl  91227
3  1881  boy 100763
4  1881 girl  92204
5  1882  boy 113194
6  1882 girl 107713
7  1883  boy 104487
..  ...  ...    ...
```




```
qplot(year, total, data = ytotals, geom = "line", color = sex)
```

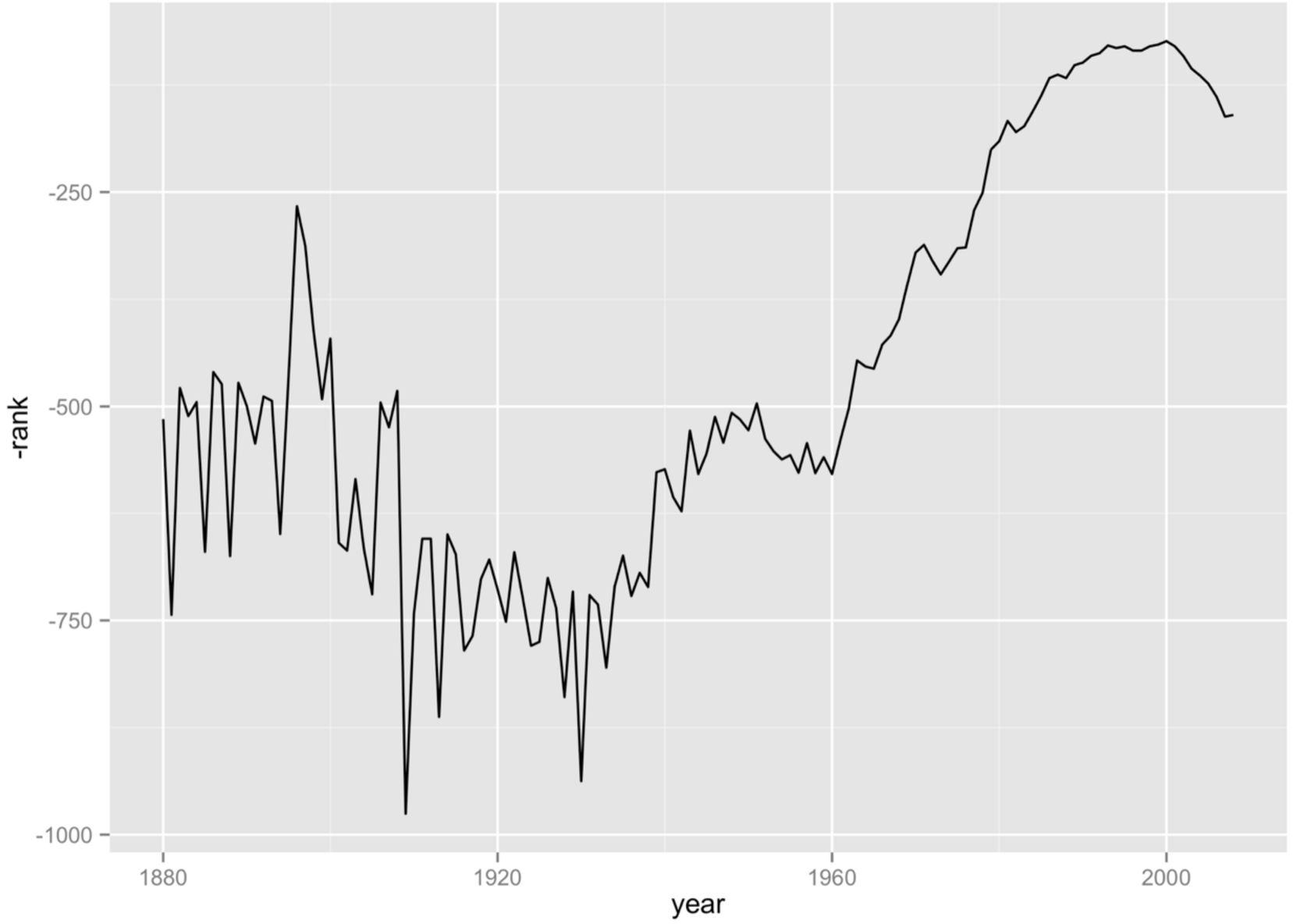
Your turn

Calculate the rank of each name within each year (and within each gender).

Hint: create a new variable equal to rank(desc(prop))

```
year_sex <- group_by(bnames2, year, sex)
ranks <- mutate(year_sex, rank = rank(desc(prop)))
ranks
```

```
## Source: local data frame [258,000 x 8]
## Groups: year, sex

##    year    name     prop sex soundex births    n rank
## 1  1880    John 0.081541 boy    J500 118405 9655    1
## 2  1880 William 0.080511 boy    W450 118405 9533    2
## 3  1880   James 0.050057 boy    J520 118405 5927    3
## 4  1880 Charles 0.045167 boy    C642 118405 5348    4
## 5  1880  George 0.043292 boy    G620 118405 5126    5
## 6  1880   Frank 0.027380 boy    F652 118405 3242    6
## 7  1880  Joseph 0.022229 boy    J210 118405 2632    7
## .. ...     ...      ... ...     ...    ...  ...  ...
```




```
garrett <- filter(ranks, name == "Garrett")
qplot(year, -rank, data = garrett, geom = "line")
```

```
# What names were ranked number one?
ones <- filter(ranks, rank == 1)
ones <- select(ones, year, name, sex)
ones
## Source: local data frame [258 x 3]
## Groups: year, sex

##    year name sex
## 1  1880 John boy
## 2  1881 John boy
## 3  1882 John boy
## 4  1883 John boy
## 5  1884 John boy
## 6  1885 John boy
## 7  1886 John boy
## 8  1887 John boy
## 9  1888 John boy
## 10 1889 John boy
..  ...  ... ...
```

```
# What names were ranked number one?
library(reshape2)
dcast(ones, year ~ sex, value = “name”)
##     year     boy     girl
## 1   1880    John     Mary
## 2   1881    John     Mary
## 3   1882    John     Mary
## 4   1883    John     Mary
## 5   1884    John     Mary
## 6   1885    John     Mary
## 7   1886    John     Mary
## 8   1887    John     Mary
## 9   1888    John     Mary
## 10  1889    John     Mary
```

Where next?

Other data sources

dplyr creates a consistent syntax for manipulating data from various sources

- data frame, data table, data cube
- PostgreSQL, Greenplum, redshift
- MySQL, MariaDB
- SQLite
- MonetDB, BigQuery
- *Oracle*, *SQL Server*

Read dplyr's built in vignettes to learn more

```
browseVignettes(package = "dplyr")
```