

Stat405

Advanced data manipulation

Hadley Wickham

1. Introduction to ggplot2

2. Data frames

3. Writing functions

4. Data manipulation

5. ggplot2 theory

6. Special data

7. Data cleaning

8. Advanced programming

9. Professional development



Unit 1:
Basic Skills



Unit 2:
Intermediate Skills



Unit 3:
Expert Skills

1. Baby names data
2. Slicing and dicing
3. Merging 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(plyr)
```

```
library(ggplot2)
```

```
options(stringsAsFactors = FALSE)
```

```
bnames <- read.csv("bnames2.csv.bz2")
```

```
births <- read.csv("http://stat405.had.co.nz/data/births.csv")
```

```
> head(bnames, 20)
```

	year	name	soundex	prop	sex
1	1880	John	J500	0.081541	boy
2	1880	William	W450	0.080511	boy
3	1880	James	J520	0.050057	boy
4	1880	Charles	C642	0.045167	boy
5	1880	George	G620	0.043292	boy
6	1880	Frank	F652	0.027380	boy
7	1880	Joseph	J210	0.022229	boy
8	1880	Thomas	T520	0.021401	boy
9	1880	Henry	H560	0.020641	boy
10	1880	Robert	R163	0.020404	boy
11	1880	Edward	E363	0.019965	boy
12	1880	Harry	H600	0.018175	boy
13	1880	Walter	W436	0.014822	boy
14	1880	Arthur	A636	0.013504	boy
15	1880	Fred	F630	0.013251	boy
16	1880	Albert	A416	0.012609	boy
17	1880	Samuel	S540	0.008648	boy
18	1880	David	D130	0.007339	boy
19	1880	Louis	L200	0.006993	boy
20	1880	Joe	J000	0.006174	boy

```
> tail(bnames, 20)
```

	year	name	soundex	prop	sex
257981	2008	Miya	M000	0.000130	girl
257982	2008	Rory	R600	0.000130	girl
257983	2008	Desirae	D260	0.000130	girl
257984	2008	Kianna	K500	0.000130	girl
257985	2008	Laurel	L640	0.000130	girl
257986	2008	Neveah	N100	0.000130	girl
257987	2008	Amaris	A562	0.000129	girl
257988	2008	Hadassah	H320	0.000129	girl
257989	2008	Dania	D500	0.000129	girl
257990	2008	Hailie	H400	0.000129	girl
257991	2008	Jamiya	J500	0.000129	girl
257992	2008	Kathy	K300	0.000129	girl
257993	2008	Laylah	L400	0.000129	girl
257994	2008	Riya	R000	0.000129	girl
257995	2008	Diya	D000	0.000128	girl
257996	2008	Carleigh	C642	0.000128	girl
257997	2008	Iyana	I500	0.000128	girl
257998	2008	Kenley	K540	0.000127	girl
257999	2008	Sloane	S450	0.000127	girl
258000	2008	Elianna	E450	0.000127	girl

Your turn

Extract your name from the dataset. Plot the trend over time.

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

```
garret <- subset(bnames, name == "Garret")
```

```
hadley <- subset(bnames, name == "Hadley")
```

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

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


Your turn

Use the `soundex` variable to extract all names that sound like yours. Plot the trend over time.

Do you have any difficulties? Think about grouping.

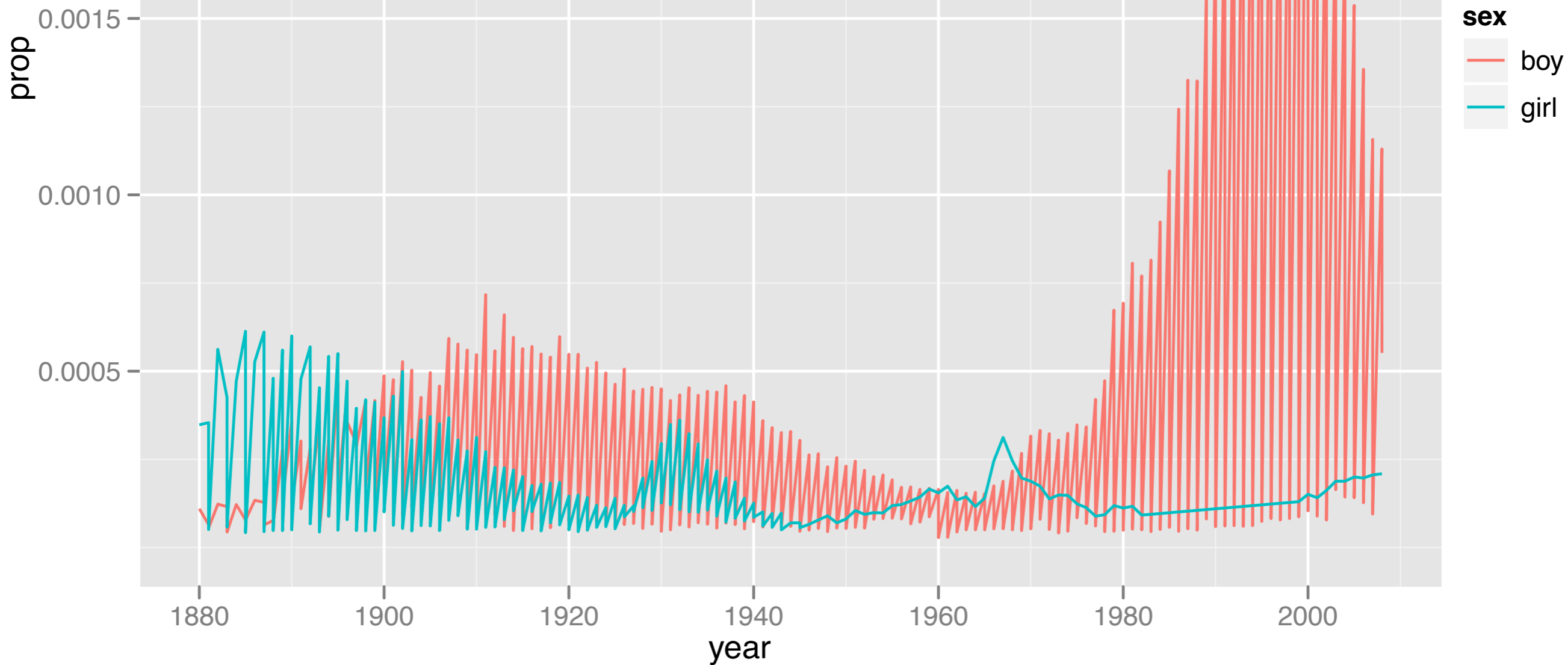
```
glike <- subset(bnames, soundex == "G630")
qplot(year, prop, data = glike)
qplot(year, prop, data = glike, geom = "line")

qplot(year, prop, data = glike, geom = "line",
       colour = sex)

qplot(year, prop, data = glike, geom = "line",
       colour = sex) + facet_wrap(~ name)

qplot(year, prop, data = glike, geom = "line",
       colour = sex, group = interaction(sex, name))
```

Sawtooth appearance implies grouping is incorrect.



Slicing and dicing

Revision

Recall the four functions that filter rows, create summaries, add new variables and rearrange the rows.

You have 30 seconds!

Function	Package
<code>subset</code>	<code>base</code>
<code>summarise</code>	<code>plyr</code>
<code>mutate</code>	<code>plyr</code>
<code>arrange</code>	<code>plyr</code>

They all have similar syntax. The first argument is a data frame, and all other arguments are interpreted in the context of that data frame. Each returns a data frame.

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

color	value
blue	1
blue	3
blue	4

```
subset(df, color == "blue")
```

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

double
2
4
6
8
10

`summarise(df, double = 2 * value)`

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

total
15

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

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(df, double = 2 * value)
```

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(df, color)`

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(df, desc(color))
```

Your turn

In which year was your name most popular? Least popular?

Reorder the data frame containing your name from highest to lowest popularity.

Add a new column that gives the number of babies per thousand with your name.

```
summarise(garrett,  
  least = year[prop == min(prop)],  
  most = year[prop == max(prop)])
```

```
# OR
```

```
summarise(garrett,  
  least = year[which.min(prop)],  
  most = year[which.max(prop)])
```

```
arrange(garrett, desc(prop))
```

```
mutate(garrett, per1000 = round(1000 * prop))
```

Brainstorm

Thinking about the data, what are some of the trends that you might want to explore? What additional variables would you need to create? What other data sources might you want to use?

Pair up and brainstorm for 2 minutes.

External	Internal
<p>Biblical names Hurricanes Ethnicity Famous people</p>	<p>First/last letter Length Vowels Rank Sounds-like</p>

join

ddply

Merging data

Combining 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

=

?

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

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

`join(x, y, type = "left")`

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

`join(x, y, type = "left")`

Try it

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
Brian	NA	F

`join(x, y, type = "right")`

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
Brian	NA	F

`join(x, y, type = "right")`

Try it

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

`join(x, y, type = "inner")`

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

`join(x, y, type = "inner")`

Try it

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
Brian	NA	F

`join(x, y, type = "full")`

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
Brian	NA	F

`join(x, y, type = "full")`

Try it

Type	Action
"left"	Include all of x, and matching rows of y
"right"	Include all of y, and matching rows of x
"inner"	Include only rows in both x and y
"full"	Include all rows

Your turn

Convert from proportions to absolute numbers by combining bnames with births, and then performing the appropriate calculation.

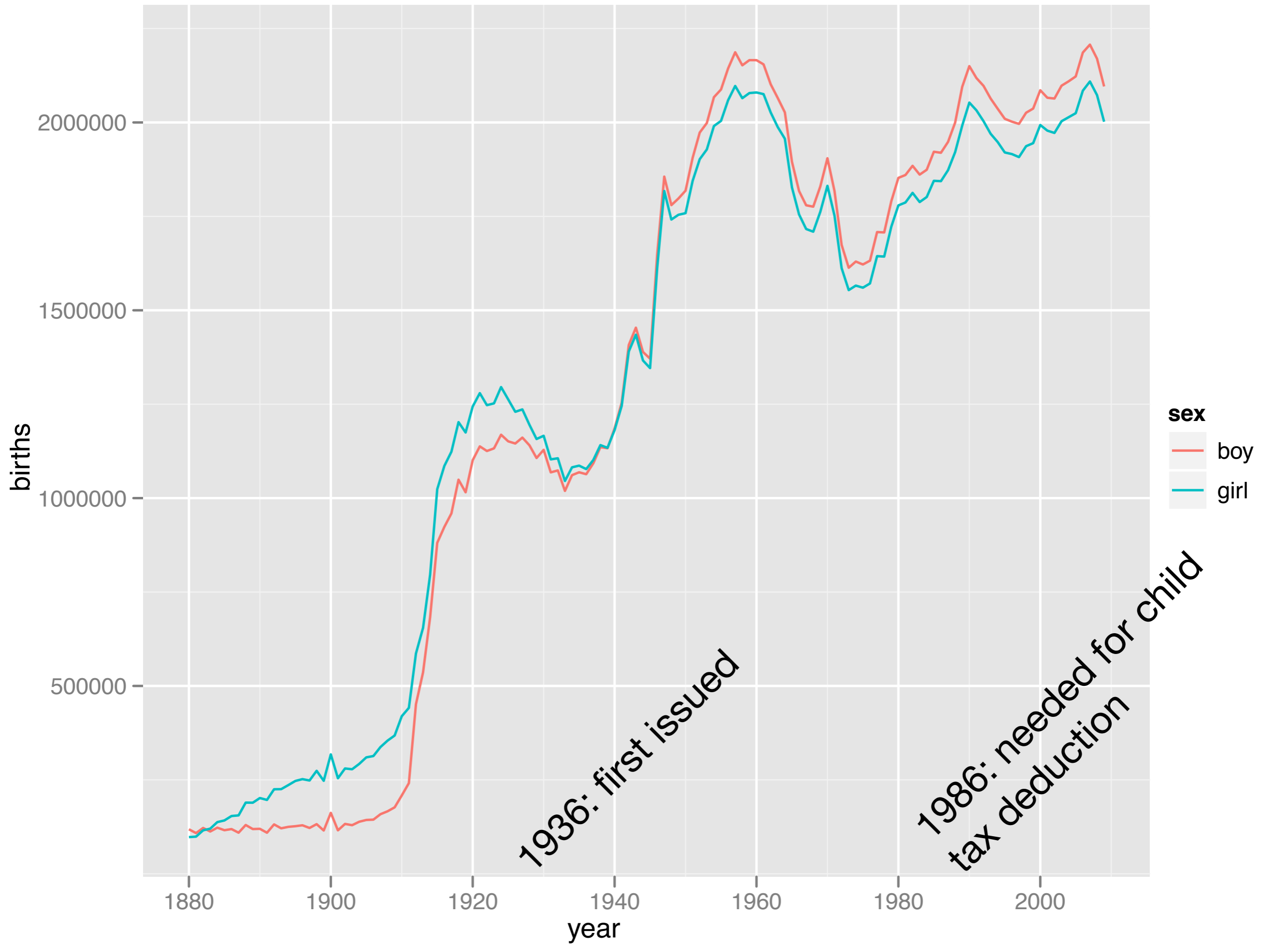
```
bnames2 <- join(bnames, births,  
  by = c("year", "sex"))  
tail(bnames2)
```

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

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

```
# Births database does not contain all births!
```

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



1936: first issued

1986: needed for child tax deduction

sex
— boy
— girl

How would we combine these?

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

+

Name	band	instrument
John	T	vocals
Paul	T	vocals
George	T	backup
Ringo	T	backup
Brian	F	manager

=

?

```
members$instrument <- c("vocals", "vocals", "backup",  
"backup", "manager")
```

How would we combine these?

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

+

Name	band	instrument
John	T	vocals
Paul	T	vocals
George	T	backup
Ringo	T	backup
Brian	F	manager

=

Try it

```
members$instrument <- c("vocals", "vocals", "backup",  
"backup", "manager")
```

```
join(what_played, members, type = "full")
```

```
# :(
```

```
?join
```

```
join(what_played, members, by = "name", type = "full")
```

```
# :(
```

```
names(members)[3] <- "instrument2"
```

```
join(what_played, members, type = "full")
```

Group wise operations

Number of people

How do we compute the number of people with each name over all years ?
It's pretty easy if you have a single name.
(e.g. how many people with your name were born over the entire 128 years)

How would you do it?

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

```
# Or
```

```
summarise(garrett, n = sum(n))
```

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

```
# Split
```

```
pieces <- split(bnames2, list(bnames$name))
```

```
# Apply
```

```
results <- vector("list", length(pieces))
```

```
for(i in seq_along(pieces)) {
```

```
  piece <- pieces[[i]]
```

```
  results[[i]] <- summarise(piece,
```

```
    name = name[1], n = sum(n))
```

```
}
```

```
# Combine
```

```
result <- do.call("rbind", results)
```

```
# Or equivalently
```

```
counts <- ddply(bnames2, "name", summarise,  
  n = sum(n))
```


Input data

Way to split
up input

Or equivalent

```
counts <- dply(bnames2, "name", summarise,  
  n = sum(n))
```

2nd argument to
summarise()

Function to apply to
each piece

x	y
a	2
a	4
b	0
b	5
c	5
c	10

Split

x	y
a	2
a	4
b	0
b	5
c	5
c	10



x	y
a	2
a	4



x	y
b	0
b	5



x	y
c	5
c	10

Split

Apply

x	y
a	2
a	4
b	0
b	5
c	5
c	10



x	y
a	2
a	4



3



x	y
b	0
b	5



2.5



x	y
c	5
c	10



7.5

Split

Apply

Combine

x	y
a	2
a	4
b	0
b	5
c	5
c	10

x	y
a	2
a	4

x	y
b	0
b	5

x	y
c	5
c	10

3

2.5

7.5

x	y
a	3
b	2.5
c	7.5

Your turn

Repeat the same operation, but use soundex instead of name. What is the most common sound? What name does it correspond to? (Hint: use join)

```
counts <- dply(bnames2, "soundex", summarise,  
  n = sum(n))  
counts <- arrange(counts, desc(n))  
  
# Combine with names. When there are multiple  
# possible matches, picks first  
counts <- join(  
  counts, bnames2[, c("soundex", "name")],  
  by = "soundex", match = "first")  
head(counts, 100)  
  
subset(bnames, soundex == "L600")
```