Unit 1: Introduction to Data

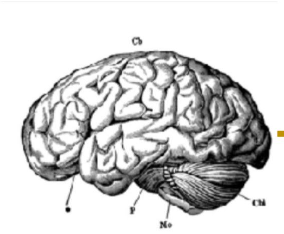
4. Using the tidyverse

1/31/2022

Key ideas

1. R is an awesome language for rapid prototyping
2. dplyr verbs are a useful framework for transforming data (munging)
3. Every r chunk is a paragraph, every line of code is a sentence, pipes are periods.

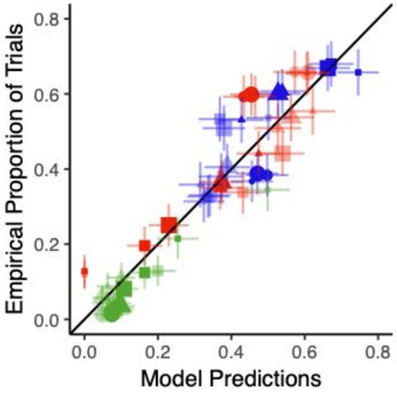
From an idea to a statistical model



A vague idea

Specifying, clarifying, testing, iterating, ...

computation



time

We should optimize for human thought, not computation

From an idea to a statistical model

R has most of the features you know and love:

- Iterative control structure (e.g. `for`, `while`)
- Functional programming (e.g. `map`)
- Objects (e.g. `structure`)

But, the best thing about R (in my opinion) is the suite of packages in the `tidyverse`

tidyverse
:
%>%
Ceci n'est pas un pipe.
TIBBLE
dplyr
ggplot2
readr
purrr

Very basics of R

```
foo <- “hello"
```

This is how you assign values to variables.
You can use = instead of <-, but you shouldn’t!
The asymmetry between the sides is clearer

```
foo <- c(1,2,3,4)
```

`c` is for concatenate. You use it to make lists

```
5 >= 7
```

returns `FALSE`.

```
is.character(“the”)
```

returns `TRUE`.

magrittr: the pipe operator (%>%)

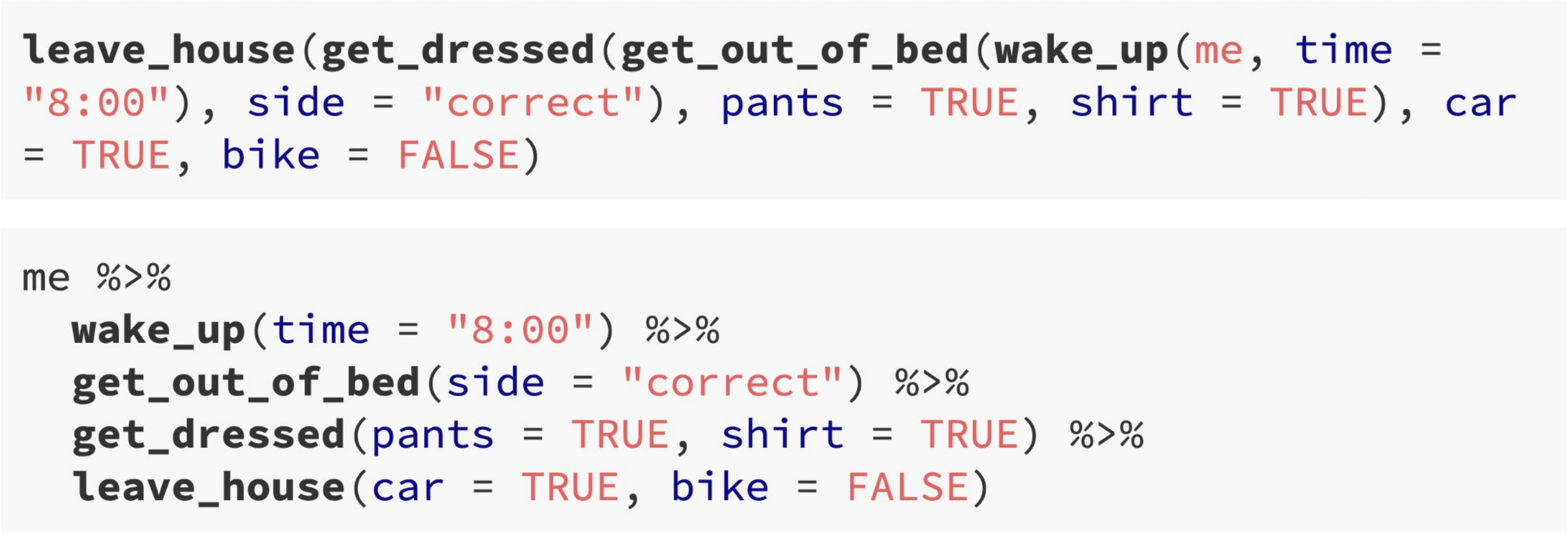
```
foo <- “hello"

bar <- paste(foo, “world")

baz <- paste(bar, “from 85309"

baz returns “hello world from 85309”


baz <- paste(paste(“hello”, “world”), “from 85309")

   y <- f(g(x))
```

magrittr: the pipe operator (%>%)

```
foo <- “hello"

bar <- paste(foo, “world")

baz <- paste(bar, “from 85309"

baz returns “hello world from 85309”

baz <- “hello” %>%
          paste(“world”) %>%
          paste(“from 85309")
```

```
 y <- x %>% g %>% f
```

magrittr: the pipe operator (%>%)



```r
leave_house(get_dressed(get_out_of_bed(wake_up(me, time =
"8:00"), side = "correct"), pants = TRUE, shirt = TRUE), car
= TRUE, bike = FALSE)
```

```r
me %>%
  wake_up(time = "8:00") %>%
  get_out_of_bed(side = "correct") %>%
  get_dressed(pants = TRUE, shirt = TRUE) %>%
  leave_house(car = TRUE, bike = FALSE)
```

Thanks, @andrewheiss!

tibble: a human readable, general data structure

```
    days <- c(“monday”, “tuesday”, “wednesday”)
meanings <- c(“moon”, “Tiu”, “Woden”)
 origins <- tibble(day = days,
                   meaning = meanings)
```

```
origins$meaning

origins %>%
    pull(meaning)
```

Returns c(“moon”, “Tiu”, “Woden”)

day	*meaning*
monday	*moon*
tuesday	*Tiu*
wednesday	*Woden*

dplyr: verbs for working with data

`group_by` whatever you’re going to do next, do it separately for each group.

`select` keep just a subset of the columns in a tibble

`filter` keep just the rows whose values in one or more columns match a truth condition (`day == “monday"`)

`mutate` apply an operation to one or more columns (`as.numeric`, `log`, etc)

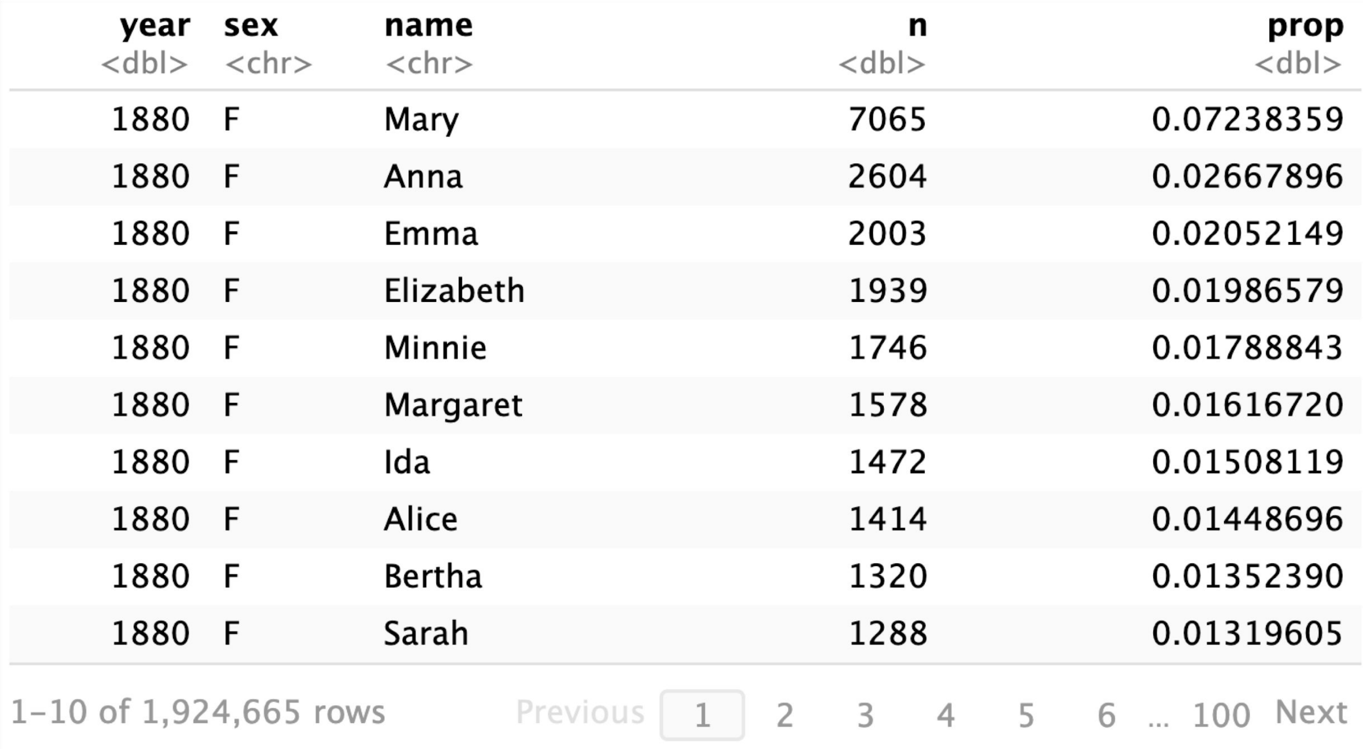
`summarise` apply an operation to one or more columns that produces a single number (`sum`, `mean`, etc)

Transform Data with

Slides from “Remaster the tidyverse” by Garret Grolemund

https://github.com/rstudio-education/remaster-the-tidyverse

```
install.packages(“babynames”)
library(babynames)
babynames
```



year	sex	name	n	prop
<dbl>	<chr>	<chr>	<dbl>	<dbl>
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720
1880	F	Ida	1472	0.01508119
1880	F	Alice	1414	0.01448696
1880	F	Bertha	1320	0.01352390
1880	F	Sarah	1288	0.01319605

1-10 of 1,924,665 rows

How to isolate?

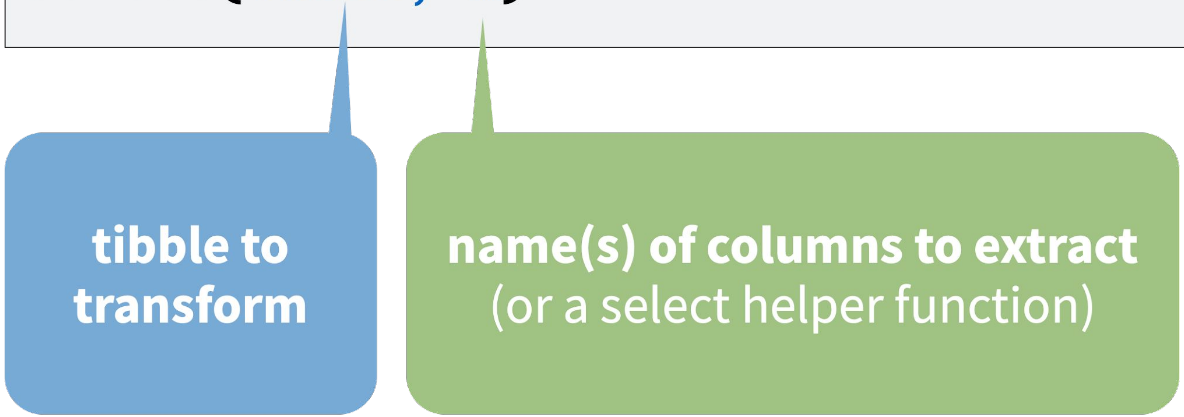
year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081
1881	M	William	8524	0.0787
1881	M	James	5442	0.0503
1881	M	Charles	4664	0.0431
1881	M	Garrett	7	0.0001
1881	M	Gideon	7	0.0001

year	sex	name	n	prop
1880	M	Garrett	13	0.0001
1881	M	Garrett	7	0.0001
…	…	Garrett	…	…

select()

Extract columns by name.

```
select(.data, …)
```



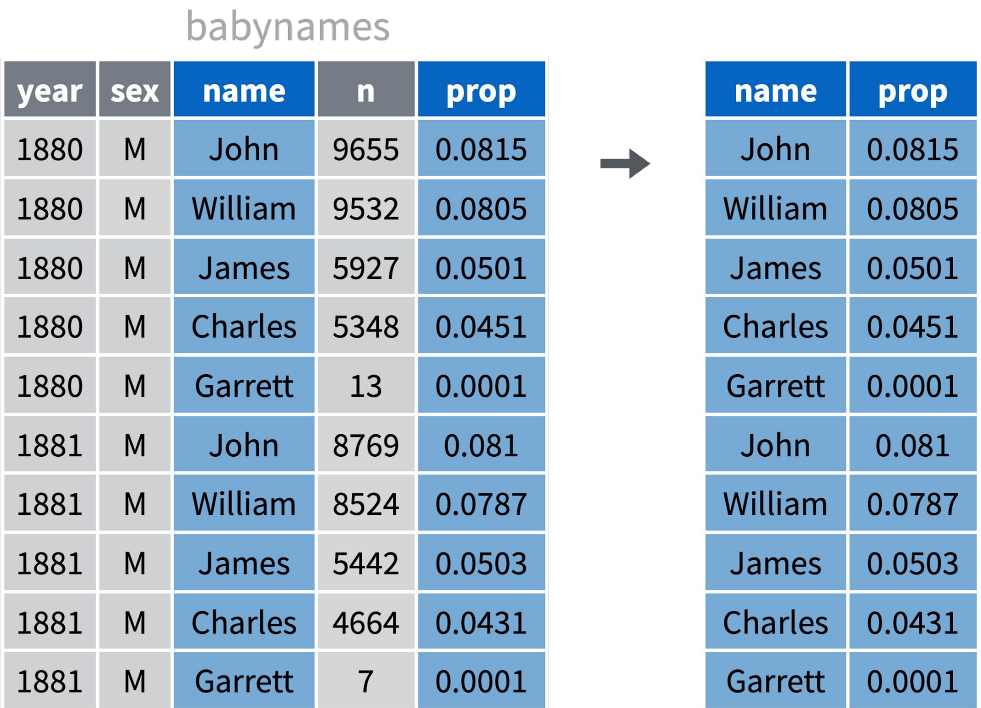
tibble to transform

name(s) of columns to extract
(or a select helper function)

select()

Extract columns by name.

```
select(babynames, name, prop)
```



babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081
1881	M	William	8524	0.0787
1881	M	James	5442	0.0503
1881	M	Charles	4664	0.0431
1881	M	Garrett	7	0.0001

name	prop
John	0.0815
William	0.0805
James	0.0501
Charles	0.0451
Garrett	0.0001
John	0.081
William	0.0787
James	0.0503
Charles	0.0431
Garrett	0.0001

select() helpers

: - Select range of columns

```
select(mpg, cty:class)
```

- - Select every column but

```
select(mpg, -c(cty, hwy))
```

starts_with() - Select columns that start with…

```
select(mpg, starts_with("c"))
```

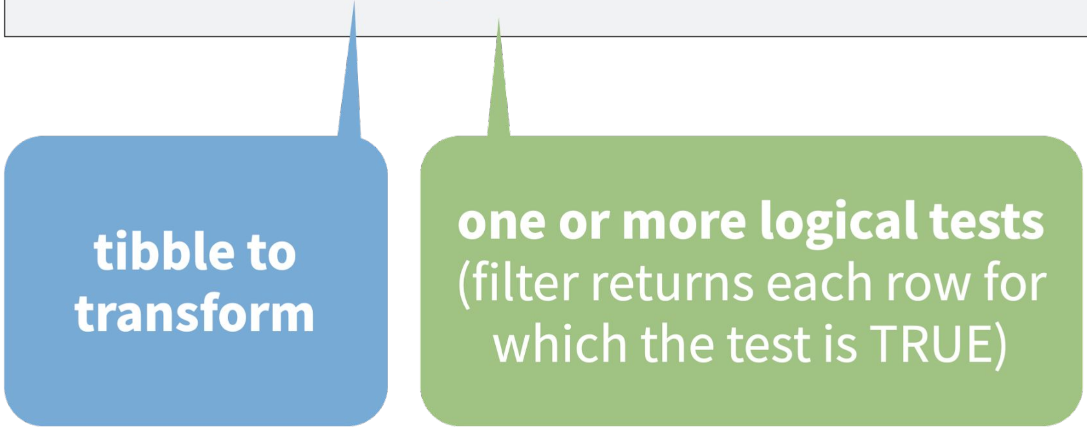
ends_with() - Select columns that end with…

```
select(mpg, ends_with("y"))
```

filter()

Extract rows that meet logical criteria.

```
filter(.data, … )
```



tibble to transform

one or more logical tests (filter returns each row for which the test is TRUE)

filter()

Extract rows that meet logical criteria.

```
filter(babynames, name == "Garrett")
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081
1881	M	William	8524	0.0787
1881	M	James	5442	0.0503

→

year	sex	name	n	prop
1880	M	Garrett	13	0.0001
1881	M	Garrett	7	0.0001
…	…	Garrett	…	…

Logical tests

?Comparison

`x < y`	Less than
`x > y`	Greater than
`x == y`	Equal to
`x <= y`	Less than or equal to
`x >= y`	Greater than or equal to
`x != y`	Not equal to
`x %in% y`	Group membership
`is.na(x)`	Is NA
`!is.na(x)`	Is not NA

```r
x <- 1
x >= 2
# FALSE
```

```r
x <- c(1, 2, 3)
x >= 2
# FALSE TRUE TRUE
```

```
filter(babynames, prop >= 0.08)
#    year   sex    name     n       prop
# 1  1880     M    John  9655 0.08154630
# 2  1880     M William  9531 0.08049899
# 3  1881     M    John  8769 0.08098299
```

```
filter(babynames, name == "Sea")
#    year   sex  name     n         prop
# 1  1982     F   Sea     5 2.756771e-06
# 2  1985     M   Sea     6 3.119547e-06
# 3  1986     M   Sea     5 2.603512e-06
# 4  1998     F   Sea     5 2.580377e-06
```

Two common mistakes

1. Using = instead of ==

```
filter(babynames, name = "Sea")
filter(babynames, name == "Sea")
```

2. Forgetting quotes

```
filter(babynames, name == Sea)
filter(babynames, name == "Sea")
```

filter()

Extract rows that meet *every* logical criteria.

```
filter(babynames, name == "Garrett", year == 1880)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081
1881	M	William	8524	0.0787
1881	M	James	5442	0.0503
1881	M	Charles	4664	0.0431

→

year	sex	name	n	prop
1880	M	Garrett	13	0.0001

Boolean operators

?base::Logic

a & b	and
a | b	or
xor(a,b)	exactly or
!a	not
( )	To group tests . & evaluates before |

Two more common mistakes

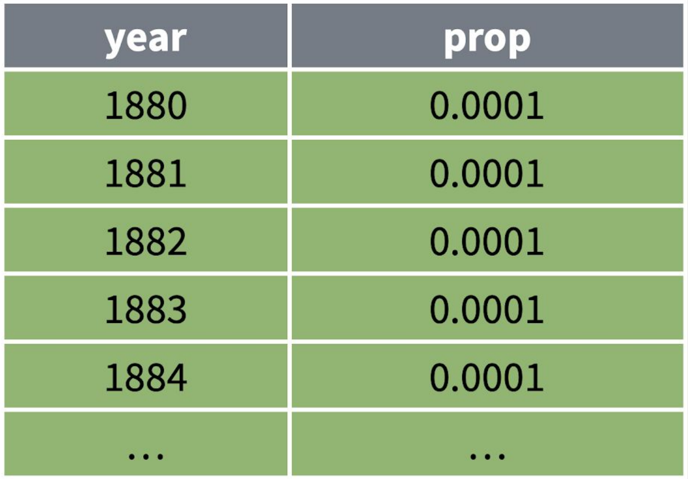
3. Collapsing multiple tests into one

```
filter(babynames, 10 < n < 20)
filter(babynames, 10 < n, n < 20)
```

4. Stringing together many tests (when you could use %in%)

```
filter(babynames, n == 5 | n == 6 | n == 7 | n == 8)
filter(babynames, n %in% c(5, 6, 7, 8))
```

```
babynames %>%
  filter(name == "Garrett", sex == "M") %>%
  select(year, prop)
```



year	prop
1880	0.0001
1881	0.0001
1882	0.0001
1883	0.0001
1884	0.0001
…	…

Deriving information

summarise() - summarise **variables**

group_by() - group **cases**

mutate() - create new **variables**

summarise()

Compute table of summaries.

```
babynames %>%
     summarise(total = sum(n),
               max = max(n))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081
1881	M	William	8524	0.0787

→

total	max
348120517	99686

group_by()

Groups cases by common values.

```
babynames %>%
  group_by(sex) %>%
  summarise(total = sum(n))
```

sex	total
F	172371079
M	175749438

ungroup()

Removes grouping criteria from a data frame.

```
babynames %>%
  group_by(sex) %>%
  ungroup() %>%
  summarise(total = sum(n))
```

total
348120517

mutate()

Create new columns.

```
babynames %>%
  mutate(percent = round(prop*100, 2))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081

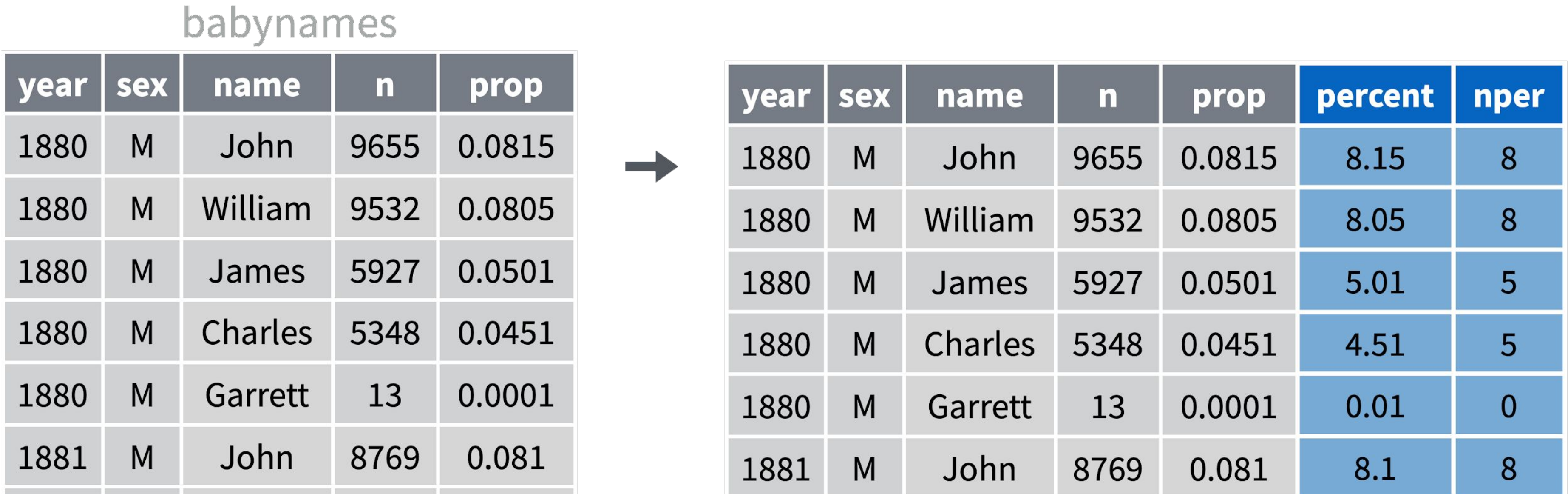
→

year	sex	name	n	prop	percent
1880	M	John	9655	0.0815	8.15
1880	M	William	9532	0.0805	8.05
1880	M	James	5927	0.0501	5.01
1880	M	Charles	5348	0.0451	4.51
1880	M	Garrett	13	0.0001	0.01
1881	M	John	8769	0.081	8.1

mutate()

Create new columns.

```
babynames %>%
  mutate(percent = round(prop*100, 2), nper = round(percent))
```



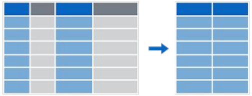
babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081

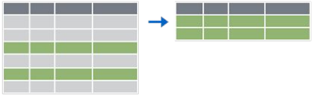
→

year	sex	name	n	prop	percent	nper
1880	M	John	9655	0.0815	8.15	8
1880	M	William	9532	0.0805	8.05	8
1880	M	James	5927	0.0501	5.01	5
1880	M	Charles	5348	0.0451	4.51	5
1880	M	Garrett	13	0.0001	0.01	0
1881	M	John	8769	0.081	8.1	8

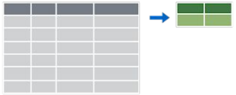
Recap: Single table verbs



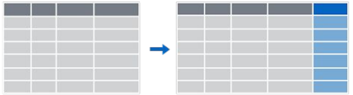
Extract variables with **select()**



Extract cases with **filter()**



Make tables of summaries with **summarise()**.



Make new variables, with **mutate()**.

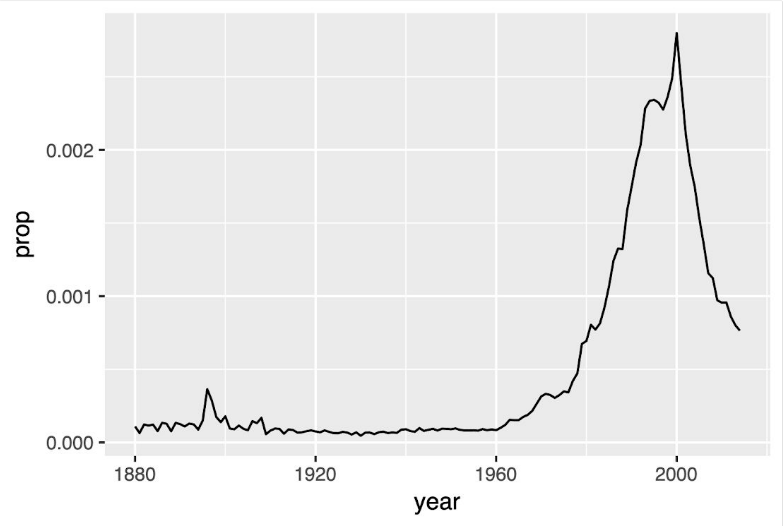
ggplot2: A grammar of graphics

ggplot()

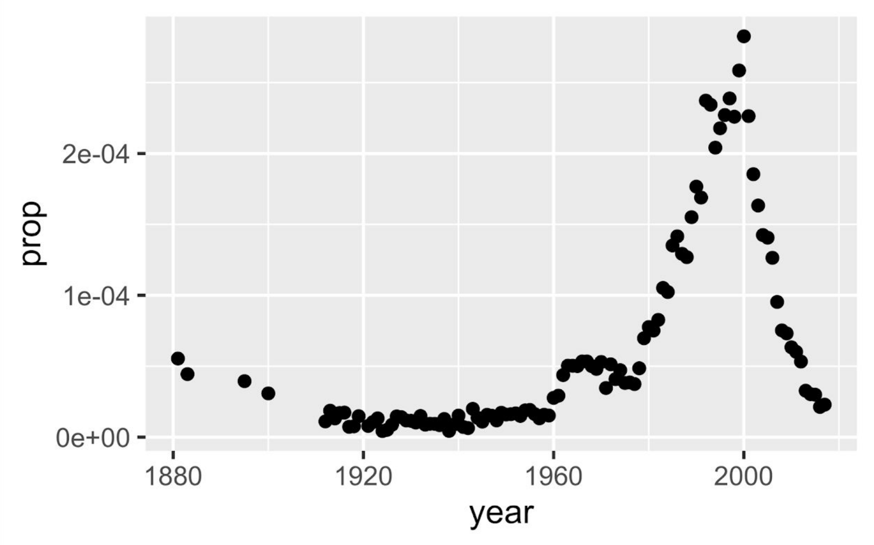
Key idea: You can compose a plot the same way you compose a sentence by following a grammar.

```
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>) +
   <GEOM_FUNCTION>() +
   <GEOM_FUNCTION>() +
   ...
```

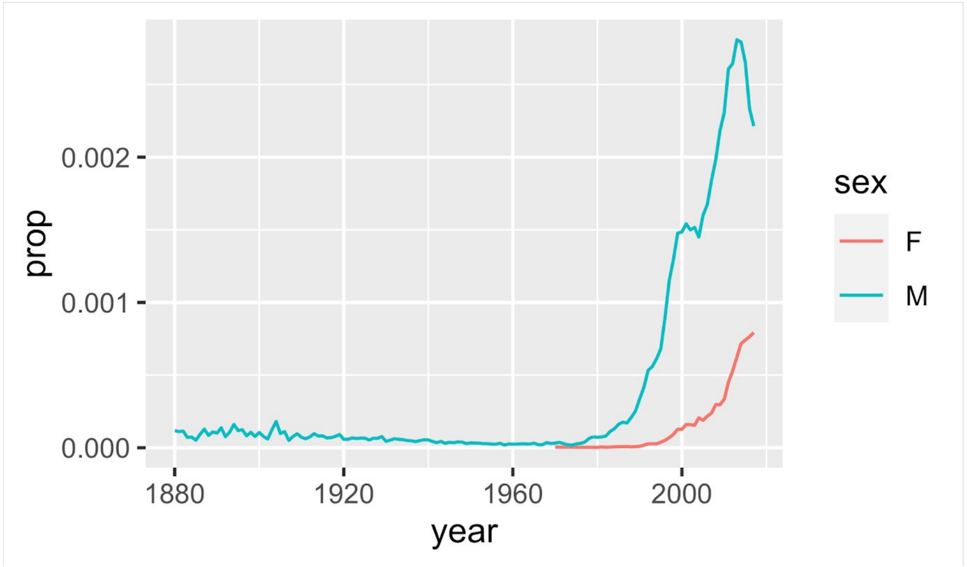
```
babynames %>%
  filter(name == "Garrett", sex == "M") %>%
  ggplot(aes(x = year, y = prop) +
    geom_line()
```



```
babynames %>%
  filter(name == "Garrett", sex == "M") %>%
  ggplot(aes(x = year, y = prop) +
    geom_point()
```



```
babynames %>%
  filter(name == "Parker") %>%
  ggplot(aes(x = year, y = prop, color = sex) +
    geom_line()
```



Key ideas

1. R is an awesome language for rapid prototyping
2. dplyr verbs are a useful framework for transforming data (munging)
3. Every r chunk is a paragraph, every line of code is a sentence, pipes are periods.