How to work with List Columns

August 2018

Garrett Grolemund

rstd.io/list-columns

RStudio

nested data frame

Species	data
setosa	<tibble [50 x 4]>
versicolor	<tibble [50 x 4]>
virginica	<tibble [50 x 4]>

n_iris

"cell" contents

Sepal.L	Sepal.W	Petal.L	Petal.W
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

n_iris$data[[1]]

Sepal.L	Sepal.W	Petal.L	Petal.W
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5

n_iris$data[[2]]

Sepal.L	Sepal.W	Petal.L	Petal.W
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2

n_iris$data[[3]]

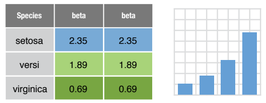
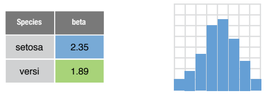
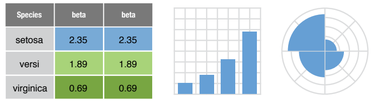
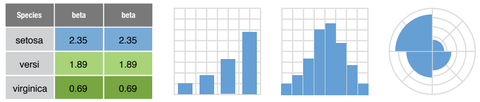
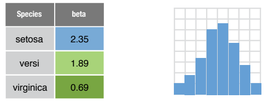
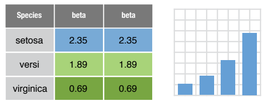
```r
library(babynames)
filter(babynames, name == "Mary")
```

```
   year sex   name      n   prop
1  1880 F     Mary   7065 0.0724
2  1880 F     Anna   2604 0.0267
3  1880 F     Emma   2003 0.0205
4  1880 M     John   9655 0.0815
5  1880 M     James  5927 0.0501
6  1880 M     Mary     27 0.000228
7  1881 F     Anna   2698 0.0273
```

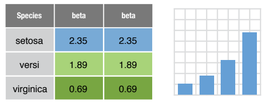
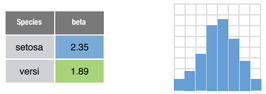
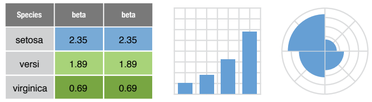
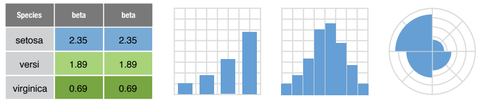
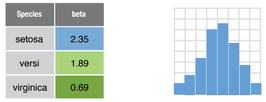
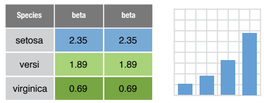
```r
library(babynames)
filter(babynames, name == "Mary")
```

```
   year sex   name      n   prop
1  1880 F     Mary   7065 0.0724
2  1880 M     Mary     27 0.000228
3  1881 F     Mary   6919 0.0700
4  1881 M     Mary     29 0.000268
5  1882 F     Mary   8148 0.0704
6  1882 M     Mary     30 0.000246
7  1883 F     Mary   8012 0.0667
```

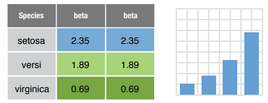
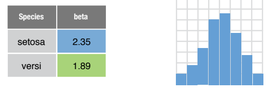
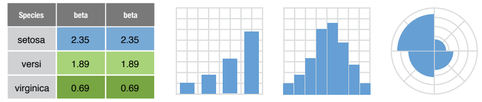
```r
library(babynames)
babynames
```

	year	sex	name	n	prop	list?
1	1880	F	Mary	7065	0.0724	
2	1880	F	Anna	2604	0.0267	
3	1880	F	Emma	2003	0.0205	
4	1880	M	John	9655	0.0815	
5	1880	M	James	5927	0.0501	
6	1880	M	Mary	27	0.000228	
7	1881	F	Anna	2698	0.0273	

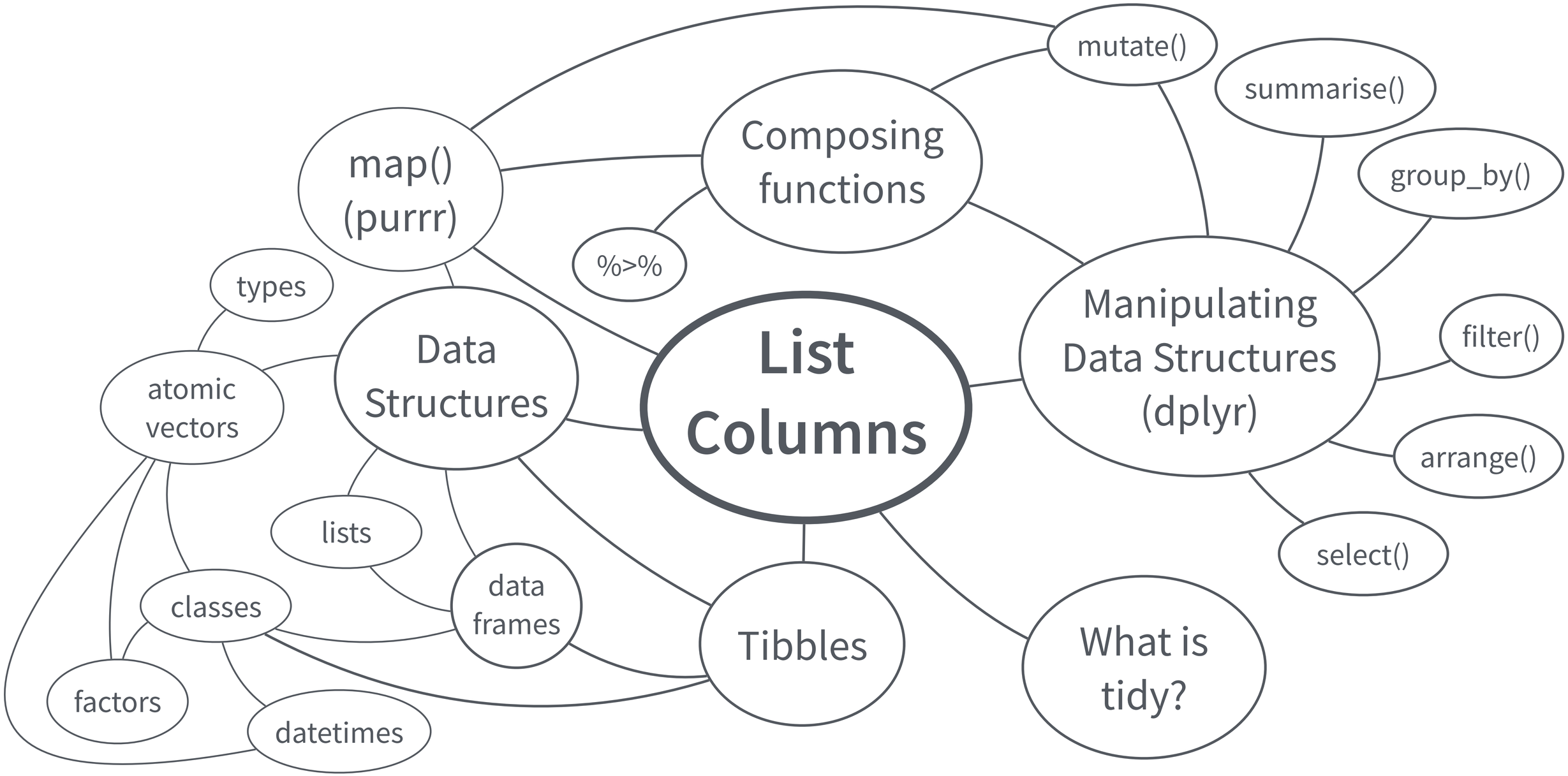
```r
library(babynames)
filter(babynames, name == "Mary")
```

	year	sex	name	n	prop	list?
1	1880	F	Mary	7065	0.0724	
2	1880	F	Anna	2604	0.0267	
3	1880	F	Emma	2003	0.0205	
4	1880	M	John	9655	0.0815	
5	1880	M	James	5927	0.0501	
6	1880	M	Mary	27	0.000228	
7	1881	F	Anna	2698	0.0273	

```r
library(babynames)
filter(babynames, name == "Mary")
```

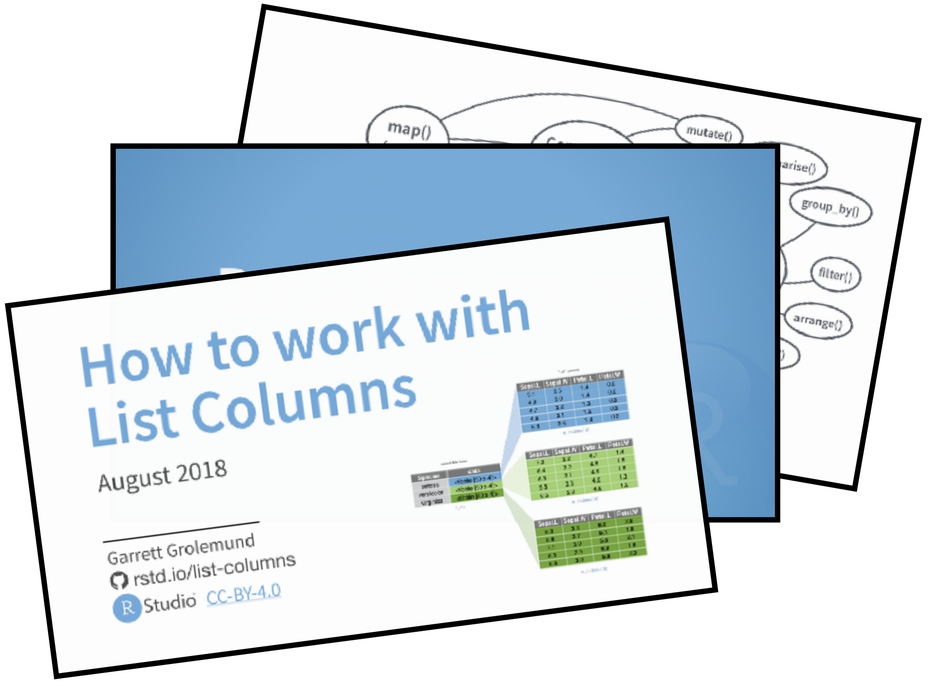
	year	sex	name	n	prop	list?
1	1880	F	Mary	7065	0.0724	
2	1880	M	Mary	27	0.000228	
3	1881	F	Mary	6919	0.0700	
4	1881	M	Mary	29	0.000268	
5	1882	F	Mary	8148	0.0704	
6	1882	M	Mary	30	0.000246	
7	1883	F	Mary	8012	0.0667	

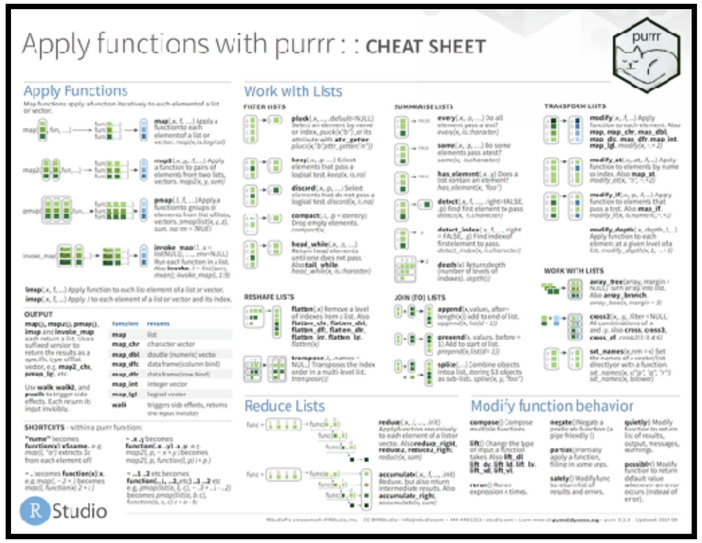
List Columns



1. Data Structures in R
2. Data frames, tibbles, and list columns
3. Single table verbs +
4. Composing functions
5. map() functions
6. **Case Study**
7. Tao of Tidy

rstd.io/list-columns





code.Rmd

slides.pdf

rstd.io/purrr-cheatsheet
rstd.io/dplyr-cheatsheet

Data Structures in R

```
1
```

```
[1] 1
```

```
c(1, 2, 3.14)
```

```
[1] 1.00 2.00 3.14
```

```
is.vector(1)
```

```
TRUE
```

```
is.vector(c(1, 2, 3.14))
```

```
TRUE
```

```
typeof(c(1L, 2L, 3L))
[1] "integer"

typeof(c(1, 2, 3.14))
[1] "double"
```

```
typeof(c("a", "b", "c"))
[1] "character"

typeof(c(TRUE, FALSE))
[1] "logical"
```

```
x <- c(1L, 2L, 3L)
x
[1] 1 2 3
```

```r
x <- c(1L, 2L, 3L)
class(x) <- "Date"
x
```

```
"1970-01-02" "1970-01-03" "1970-01-04"
```

```
x <- c(1L, 2L, 3L)
levels(x) <- c("Blue", "Brown", "Green")
class(x) <- "factor"
x
[1] Blue  Brown Green
Levels: Blue Brown Green
```

```
dim(x) <- c(3, 1)
x
     [,1]
[1,] Blue
[2,] Brown
[3,] Green
Levels: Blue Brown Green
```

```
(y <- list(a = c(1, 2, 3.14),
           b = c("a", "b", "c"),
           c = c(TRUE, FALSE, FALSE)))
$a
[1] 1.00 2.00 3.14
$b
[1] "a" "b" "c"
$c
[1]  TRUE FALSE FALSE
```

```
typeof(y)
```

```
[1] "list"
```

```
is.vector(y)
```

```
[1] TRUE
```

```
class(y) <- "data.frame"
rownames(y) <- c("1", "2", "3")
y
```

```
     a b     c
1 1.00 a  TRUE
2 2.00 b FALSE
3 3.14 c FALSE
```

Should this work?

```
y$d <- list(p = 1:3, q = TRUE, r = 0L)
y
     a b     c         d
1 1.00 a  TRUE 1, 2, 3
2 2.00 b FALSE      TRUE
3 3.14 c FALSE         0
```

Yes!

```
y$d
$p
[1] 1 2 3
$q
[1] TRUE
$r
[1] 0
```

Data frames, Tibbles, and List Columns

```r
data.frame(a = c(1, 2, 3.14),
           b = c("a", "b", "c"),
           c = c(TRUE, FALSE, FALSE))
```

```
     a b     c
1 1.00 a  TRUE
2 2.00 b FALSE
3 3.14 c FALSE
```

```r
data.frame(list(a = c(1, 2, 3.14),
                b = c("a", "b", "c"),
                c = c(TRUE, FALSE, FALSE)))
```

```
     a b     c
1 1.00 a  TRUE
2 2.00 b FALSE
3 3.14 c FALSE
```

```r
data.frame(list(a = c(1, 2, 3.14),
                b = c("a", "b", "c"),
                c = c(TRUE, FALSE, FALSE)))
```

```
     a b     c
1 1.00 a  TRUE
2 2.00 b FALSE
3 3.14 c FALSE
```

```r
data.frame(a = c(1, 2, 3.14),
           b = c("a", "b", "c"),
           c = c(TRUE, FALSE, FALSE),
           d = list(p = 1:3, q = TRUE, r = 0L))
```

```
     a b     c d.p  d.q d.r
1 1.00 a  TRUE   1 TRUE   0
2 2.00 b FALSE   2 TRUE   0
3 3.14 c FALSE   3 TRUE   0
```

```r
z <- data.frame(a = c(1, 2, 3.14),
                b = c("a", "b", "c"),
                c = c(TRUE, FALSE, FALSE))
```

```
     a b     c
1 1.00 a  TRUE
2 2.00 b FALSE
3 3.14 c FALSE
```

```
z$d <- list(p = 1:30, q = TRUE, r = 0L)
z
```

```
     a b     c
1 1.00 a  TRUE
2 2.00 b FALSE
3 3.14 c FALSE


                                        d
1 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
2
                                     TRUE
3
                                        0
```

```
library(tibble)
class(z) <- c("tbl_df", "tbl", "data.frame")
z
# A tibble: 3 x 4
      a b      c     d
  <dbl> <fct> <lgl> <list>
1  1    a     TRUE  <int [30]>
2  2    b     FALSE <lgl [1]>
3  3.14 c     FALSE <int [1]>
```

```r
library(tibble)
as_tibble(z)
```

```
# A tibble: 3 x 4
      a b      c     d
  <dbl> <fct> <lgl> <list>
1  1    a     TRUE  <int [30]>
2  2    b     FALSE <lgl [1]>
3  3.14 c     FALSE <int [1]>
```

```r
data.frame(a = c(1, 2, 3.14),
           b = c("a", "b", "c"),
           c = c(TRUE, FALSE, FALSE),
           d = list(p = 1:3, q = TRUE, r = 0L))
```

```
     a b     c d.p  d.q d.r
1 1.00 a  TRUE   1 TRUE   0
2 2.00 b FALSE   2 TRUE   0
3 3.14 c FALSE   3 TRUE   0
```

```
tibble(a = c(1, 2, 3.14),
       b = c("a", "b", "c"),
       c = c(TRUE, FALSE, FALSE),
       d = list(p = 1:3, q = TRUE, r = 0L))
```

```
# A tibble: 3 x 4
      a b     c     d
  <dbl> <chr> <lgl> <list>
1  1    a     TRUE  <int [3]>
2  2    b     FALSE <lgl [1]>
3  3.14 c     FALSE <int [1]>
```

```
tribble(~a,   ~b,     ~c,          ~d,
        1,  "a",   TRUE,   p = 1:3,
        2,  "b",  FALSE, q = TRUE,
     3.14,  "c",  FALSE,    r = 0L)
# A tibble: 3 x 4
      a b     c     d
  <dbl> <chr> <lgl> <list>
1  1    a     TRUE  <int [3]>
2  2    b     FALSE <lgl [1]>
3  3.14 c     FALSE <int [1]>
```

The real benefit?

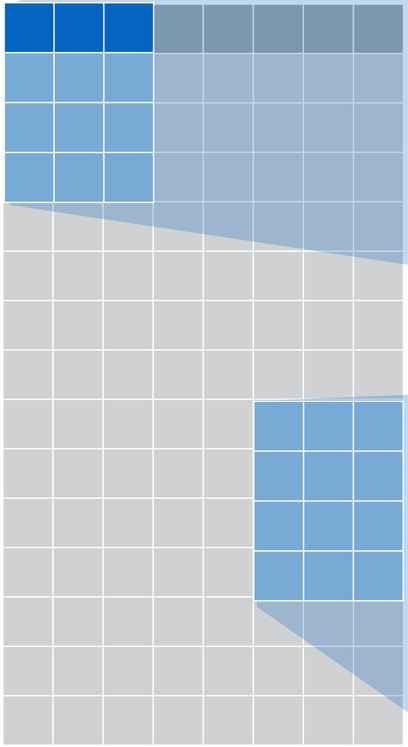
```
as_tibble(babynames)
```

```
# A tibble: 126,888 x 5
    year sex   name          n   prop
   <dbl> <chr> <chr>     <int>  <dbl>
 1  1880 F     Mary       7065 0.0724
 2  1880 F     Anna       2604 0.0267
 3  1880 F     Emma       2003 0.0205
 4  1880 F     Elizabeth  1939 0.0199
 5  1880 F     Minnie     1746 0.0179
 6  1880 F     Margaret   1578 0.0162
 7  1880 F     Ida        1472 0.0151
 8  1880 F     Alice      1414 0.0145
 9  1880 F     Bertha     1320 0.0135
10  1880 F     Sarah      1288 0.0132
# ... with 126,878 more rows
```

The real benefit?

```
as.data.frame(babynames)
```

```
        year sex            name     n         prop
1       1880   F            Mary  7065 7.238433e-02
2       1880   F            Anna  2604 2.667923e-02
3       1880   F            Emma  2003 2.052170e-02
4       1880   F       Elizabeth  1939 1.986599e-02
5       1880   F          Minnie  1746 1.788861e-02
6       1880   F        Margaret  1578 1.616737e-02
7       1880   F             Ida  1472 1.508135e-02
8       1880   F           Alice  1414 1.448711e-02
9       1880   F          Bertha  1320 1.352404e-02
10      1880   F           Sarah  1288 1.319618e-02
11      1880   F           Annie  1258 1.288882e-02
12      1880   F           Clara  1226 1.256096e-02
13      1880   F            Ella  1156 1.184378e-02
14      1880   F        Florence  1063 1.089095e-02
15      1880   F            Cora  1045 1.070653e-02
16      1880   F          Martha  1040 1.065530e-02
```



A large table to display

```
# A tibble: 234 × 6
   manufacturer      model displ
          <chr>      <chr> <dbl>
1          audi         a4   1.8
2          audi         a4   1.8
3          audi         a4   2.0
4          audi         a4   2.0
5          audi         a4   2.8
6          audi         a4   2.8
7          audi         a4   3.1
8          audi a4 quattro   1.8
9          audi a4 quattro   1.8
10         audi a4 quattro   2.0
# ... with 224 more rows, and 3
#   more variables: year <int>,
#   cyl <int>, trans <chr>
```

tibble display

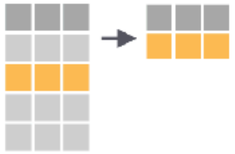
```
156 1999   6   auto(l4)
157 1999   6   auto(l4)
158 2008   6   auto(l4)
159 2008   8   auto(s4)
160 1999   4 manual(m5)
161 1999   4   auto(l4)
162 2008   4 manual(m5)
163 2008   4 manual(m5)
164 2008   4   auto(l4)
165 2008   4   auto(l4)
166 1999   4   auto(l4)
 [ reached getOption("max.print")
-- omitted 68 rows ]
```

data frame display

Single table verbs +

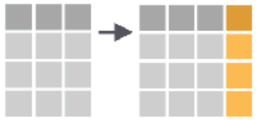
dplyr single table verbs

arrange(.data, …)
Order rows by values of a column (low to high), use with **desc()** to order from high to low.



filter(.data, …)
Extract rows that meet logical criteria.

select(.data, …)
Extract columns by name.



mutate(.data, …)
Compute new column(s).

summarise(.data, …)
Compute table of summaries. Use **group_by()** to compute groupwise summaries.

tidyr nest()

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virginica	6.3	3.3	6.0	2.5
virginica	5.8	2.7	5.1	1.9
virginica	7.1	3.0	5.9	2.1
virginica	6.3	2.9	5.6	1.8
virginica	6.5	3.0	5.8	2.2

→

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virginica	6.3	3.3	6.0	2.5
virginica	5.8	2.7	5.1	1.9
virginica	7.1	3.0	5.9	2.1
virginica	6.3	2.9	5.6	1.8
virginica	6.5	3.0	5.8	2.2

→

nested data frame

Species	data
setosa	<tibble [50 x 4]>
versicolor	<tibble [50 x 4]>
virginica	<tibble [50 x 4]>

n_iris

"cell"

Sepal.L	Sepal.W	Petal.L	Petal.W
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

n_iris$data[[1]]

Sepal.L	Sepal.W	Petal.L	Petal.W
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5

n_iris$data[[2]]

Sepal.L	Sepal.W	Petal.L	Petal.W
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2

n_iris$data[[3]]

tidyr unnest()

nested data frame

Species	data
setosa	<tibble [50 x 4]>
versicolor	<tibble [50 x 4]>
virginica	<tibble [50 x 4]>

→

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virginica	6.3	3.3	6.0	2.5
virginica	5.8	2.7	5.1	1.9
virginica	7.1	3.0	5.9	2.1
virginica	6.3	2.9	5.6	1.8
virginica	6.5	3.0	5.8	2.2

Composing functions

%>%

pipes

x %>% f(y)
becomes **f(x, y)**

%>%

```
babynames        filter(_______, name == "Mary")
```

```
babynames %>% filter(name == "Mary")
filter(babynames, name == "Mary")
```

babynames

```
# A tibble: 126,888 x 5
    year sex   name          n   prop
   <dbl> <chr> <chr>     <int>  <dbl>
 1  1880 F     Mary       7065 0.0724
 2  1880 F     Anna       2604 0.0267
 3  1880 F     Emma       2003 0.0205
 4  1880 F     Elizabeth  1939 0.0199
 5  1880 F     Minnie     1746 0.0179
 6  1880 F     Margaret   1578 0.0162
 7  1880 F     Ida        1472 0.0151
 8  1880 F     Alice      1414 0.0145
 9  1880 F     Bertha     1320 0.0135
10  1880 F     Sarah      1288 0.0132
# ... with 126,878 more rows
```

```r
babynames %>%
  select(-prop)
```

```
# A tibble: 1,858,689 x 4
    year sex   name          n
   <dbl> <chr> <chr>     <int>
 1  1880 F     Mary       7065
 2  1880 F     Anna       2604
 3  1880 F     Emma       2003
 4  1880 F     Elizabeth  1939
 5  1880 F     Minnie     1746
 6  1880 F     Margaret   1578
 7  1880 F     Ida        1472
 8  1880 F     Alice      1414
 9  1880 F     Bertha     1320
10  1880 F     Sarah      1288
# ... with 1,858,679 more rows
```

```r
babynames %>%
  select(-prop) %>%
  filter(!is.na(n))
```

```
# A tibble: 1,858,689 x 4
    year sex   name          n
   <dbl> <chr> <chr>     <int>
 1  1880 F     Mary       7065
 2  1880 F     Anna       2604
 3  1880 F     Emma       2003
 4  1880 F     Elizabeth  1939
 5  1880 F     Minnie     1746
 6  1880 F     Margaret   1578
 7  1880 F     Ida        1472
 8  1880 F     Alice      1414
 9  1880 F     Bertha     1320
10  1880 F     Sarah      1288
# ... with 1,858,679 more rows
```

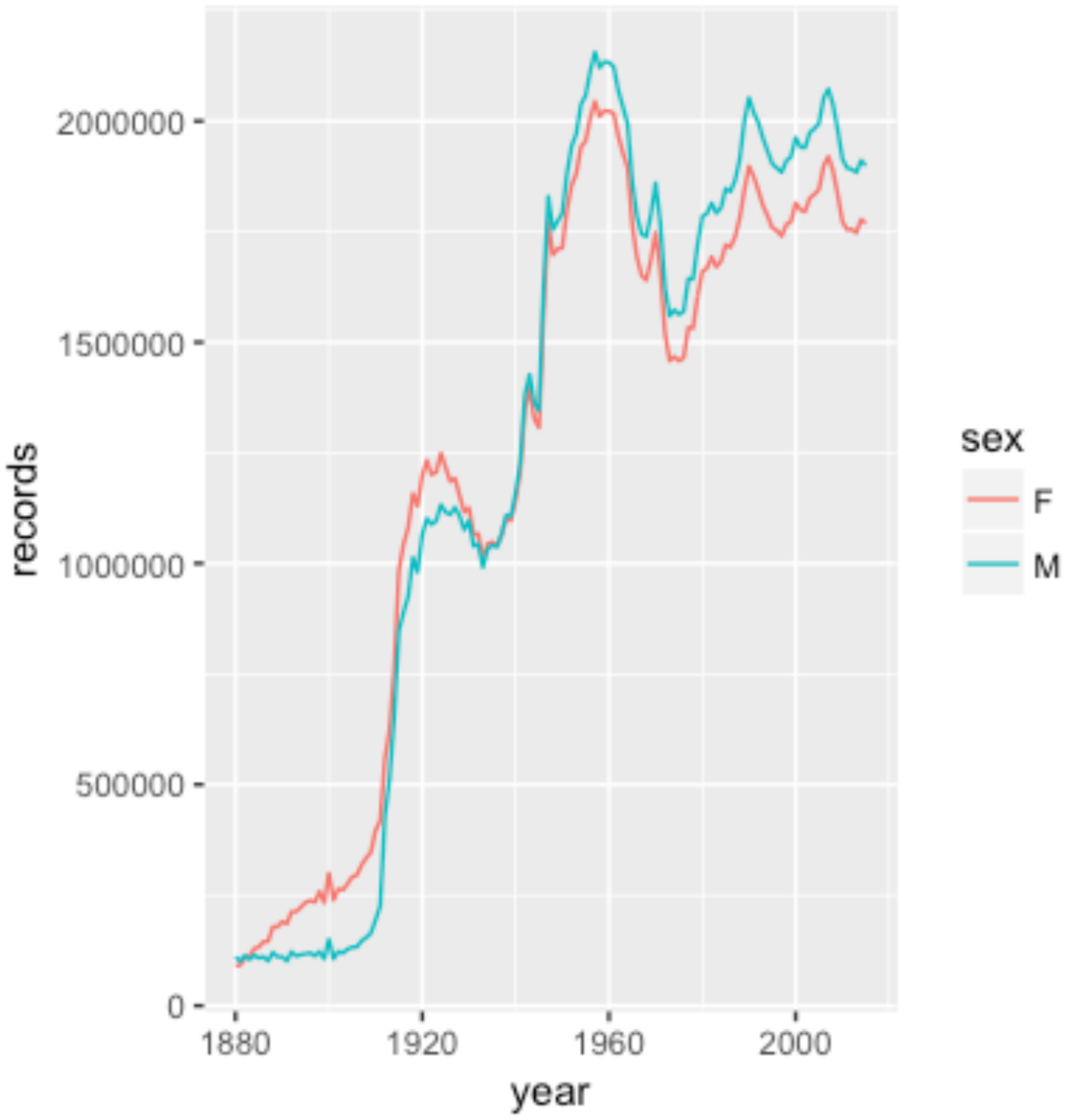
```r
babynames %>%
  select(-prop) %>%
  filter(!is.na(n)) %>%
  group_by(year, sex)
```

```
# A tibble: 1,858,689 x 4
# Groups:   year, sex [272]
    year sex   name          n
   <dbl> <chr> <chr>     <int>
 1  1880 F     Mary       7065
 2  1880 F     Anna       2604
 3  1880 F     Emma       2003
 4  1880 F     Elizabeth  1939
 5  1880 F     Minnie     1746
 6  1880 F     Margaret   1578
 7  1880 F     Ida        1472
 8  1880 F     Alice      1414
 9  1880 F     Bertha     1320
10  1880 F     Sarah      1288
# ... with 1,858,679 more rows
```

```r
babynames %>%
  select(-prop) %>%
  filter(!is.na(n)) %>%
  group_by(year, sex) %>%
  summarise(N = sum(n))
```

```
# A tibble: 272 x 3
# Groups:   year [?]
    year sex         N
   <dbl> <chr>   <int>
 1  1880 F       90992
 2  1880 M      110490
 3  1881 F       91953
 4  1881 M      100743
 5  1882 F      107848
 6  1882 M      113686
 7  1883 F      112318
 8  1883 M      104627
 9  1884 F      129020
10  1884 M      114443
# ... with 262 more rows
```

```
babynames %>%
  select(-prop) %>%
  filter(!is.na(n)) %>%
  group_by(year, sex) %>%
  summarise(N = sum(n)) %>%
  ggplot() +
    geom_line(mapping =
      aes(x = year,
          y = N,
          color = sex))
```



```
y
   a b     c        d
1.00 a  TRUE 1, 2, 3
2.00 b FALSE    TRUE
3.14 c FALSE       0
```

```
y
   a b     c         d
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```r
y %>% mutate(asq = sqrt(a))
```

```
   a b     c         d
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
   a b     c         d  asq
1.00 a  TRUE <int [3]> 1.00
2.00 b FALSE <lgl [1]> 1.41
3.14 c FALSE <int [1]> 1.77
```

```
y %>% mutate(asq = )
```

```
   a b     c         d
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

→

```
   a b     c         d  asq
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
   a b     c         d  asq
1.00 a  TRUE <int [3]> 1.00
2.00 b FALSE <lgl [1]> 1.41
3.14 c FALSE <int [1]> 1.77
```

```
y %>% mutate(asq = )
   a b     c         d
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

→

```
   a b     c         d  asq
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
      1.00
sqrt( 2.00 )
      3.14
```

→

```
      1.00
      1.41
      1.77
```

```
   a b     c         d  asq
1.00 a  TRUE <int [3]> 1.00
2.00 b FALSE <lgl [1]> 1.41
3.14 c FALSE <int [1]> 1.77
```

```
y %>% mutate(asq = )
```

```
   a b     c         d
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
   a b     c         d  asq
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
sqrt( 1.00 )
      2.00
      3.14
```

```
1.00
1.41
1.77
```

```
   a b     c         d  asq
1.00 a  TRUE <int [3]> 1.00
2.00 b FALSE <lgl [1]> 1.41
3.14 c FALSE <int [1]> 1.77
```

```
y %>% mutate(asq = sqrt(a))
```

works with tables

works with structure stored in a column (double vector)

```r
y %>% mutate(dsq = sqrt(d))
```

```
   a b     c         d
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
y %>% mutate(dsq = )
   a b     c         d                 a b     c         d  dsq
1.00 a  TRUE <int [3]>            1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>  ------>   2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>            3.14 c FALSE <int [1]>
```

```
y %>% mutate(dsq = )
   a b     c         d
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

→

```
   a b     c         d  dsq
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
       <int [3]>
sqrt(  <lgl [1]>  )
       <int [1]>
```

→ Error!

```
y %>% mutate(dsq = )
   a b     c         d                 a b     c         d  dsq
1.00 a  TRUE <int [3]>            1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>  ------>   2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>            3.14 c FALSE <int [1]>

     <int [3]>                <    [ ]>
map(<lgl [1]> ,      )  ---->   <    [ ]>
     <int [1]>                <    [ ]>
```

```
y %>% mutate(dsq = )
   a b     c         d                 a b     c         d  dsq
1.00 a  TRUE <int [3]>            1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>  ------>   2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>            3.14 c FALSE <int [1]>
     <int [3]>                <    [ ]>
map(<lgl [1]>, sqrt)  ------>  <    [ ]>
     <int [1]>                <    [ ]>
```

```
y %>% mutate(dsq = )
```

```
   a b     c        d
1.00 a  TRUE 1, 2, 3
2.00 b FALSE     TRUE
3.14 c FALSE        0
```

→

```
   a b     c         d  dsq
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
     <int [3]>                <    [ ]>
map(<lgl [1]>, sqrt)  →       <    [ ]>
     <int [1]>                <    [ ]>
```

```
sqrt(c(1, 2, 3)) → 1.00 1.41 1.73
sqrt(TRUE) → 1
sqrt(0) → 0
```

```
y %>% mutate(dsq = )
```

```
   a b     c        d                  a b     c         d  dsq
1.00 a  TRUE 1, 2, 3                 1.00 a  TRUE <int [3]>
2.00 b FALSE     TRUE                2.00 b FALSE <lgl [1]>
3.14 c FALSE        0                3.14 c FALSE <int [1]>
```

```
     <int [3]>                    <dbl [3]>
map(<lgl [1]>, sqrt)              <dbl [1]>
     <int [1]>                    <dbl [1]>
```

```
sqrt(c(1, 2, 3)) 1.00 1.41 1.73
sqrt(TRUE) 1
sqrt(0) 0
```

```
y %>% mutate(dsq = )
```

```
   a b     c        d
1.00 a  TRUE 1, 2, 3
2.00 b FALSE     TRUE
3.14 c FALSE        0
     <int [3]>
     <lgl [1]>
     <int [1]>
```

```
map( , sqrt)
```

```
   a b     c         d   dsq
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
     <dbl [3]>
     <dbl [1]>
     <dbl [1]>
```

```
sqrt(c(1, 2, 3))
sqrt(TRUE)
sqrt(0)
```

```
y %>% mutate(dsq = )
```

```
   a b     c        d
1.00 a  TRUE 1, 2, 3
2.00 b FALSE     TRUE
3.14 c FALSE        0
```

```
     <int [3]>
map(<lgl [1]>, sqrt)
     <int [1]>
```

```
sqrt(c(1, 2, 3))
sqrt(TRUE)
sqrt(0)
```

```
   a b     c         d   dsq
1.00 a  TRUE <int [3]>
2.00 b FALSE <lgl [1]>
3.14 c FALSE <int [1]>
```

```
<dbl [3]>
<dbl [1]>
<dbl [1]>
```

```
   a b     c         d   dsq
1.00 a  TRUE <int [3]> <dbl [3]>
2.00 b FALSE <lgl [1]> <dbl [1]>
3.14 c FALSE <int [1]> <dbl [1]>
```

```
y %>% mutate(dsq = map(d, sqrt))
```

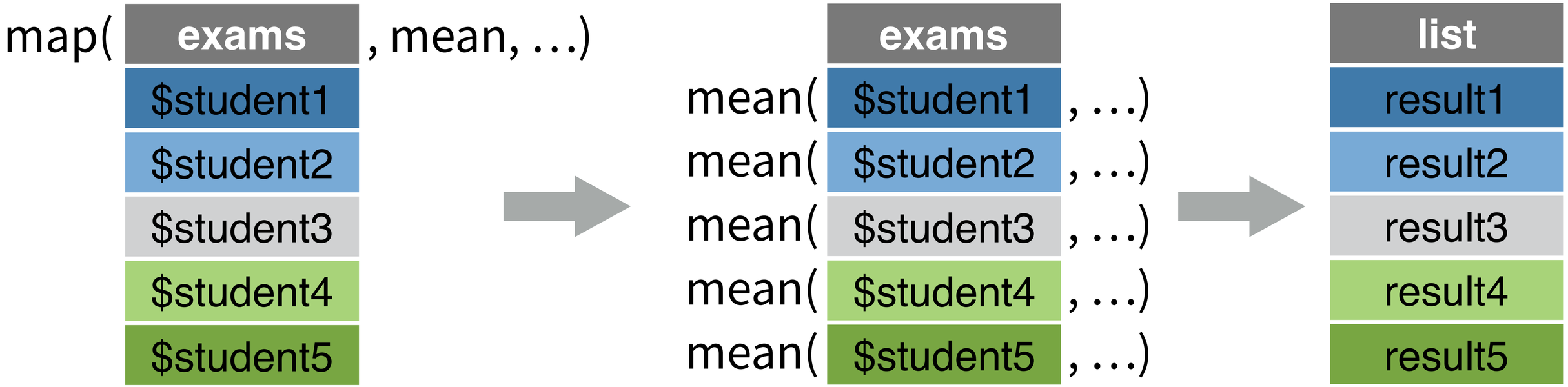
works with tables

works with contents of column (list)

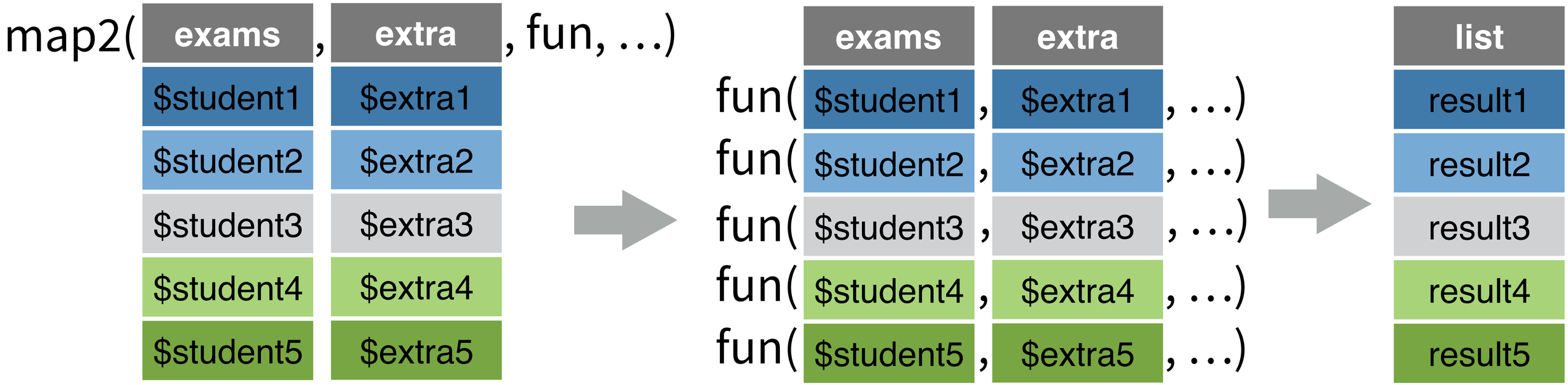
works with contents of list (atomic vectors)

map() functions

map()


map(exams , mean, …)
$student1
$student2
$student3
$student4
$student5
exams
mean($student1 , …)
mean($student2 , …)
mean($student3 , …)
mean($student4 , …)
mean($student5 , …)
list
result1
result2
result3
result4
result5

map2()


map2(
exams
extra
, fun, …)
$student1
$student2
$student3
$student4
$student5
$extra1
$extra2
$extra3
$extra4
$extra5
exams
extra
fun($student1 , $extra1 , …)
fun($student2 , $extra2 , …)
fun($student3 , $extra3 , …)
fun($student4 , $extra4 , …)
fun($student5 , $extra5 , …)
list
result1
result2
result3
result4
result5

pmap()

Map over three or more lists. Put the lists into a list of list whose names match argument names in the function.

pmap(list(, ,), fun, …)

exams	extra	more
$student1	$extra1	$more1
$student2	$extra2	$more2
$student3	$extra3	$more3
$student4	$extra4	$more4
$student5	$extra5	$more5

	exams		extra		more	
fun(	$student1	,	$extra1	,	$more1	, …)
fun(	$student2	,	$extra2	,	$more2	, …)
fun(	$student3	,	$extra3	,	$more3	, …)
fun(	$student4	,	$extra4	,	$more4	, …)
fun(	$student5	,	$extra5	,	$more5	, …)

map and walk functions

single list	two lists	n lists	returns results as
map()	map2()	pmap()	list
map_chr()	map2_chr()	pmap_chr()	character vector
map_dbl()	map2_dbl()	pmap_dbl()	double vector
map_int()	map2_int()	pmap_int()	integer vector
map_lgl()	map2_lgl()	pmap_lgl()	logical vector
map_df()	map2_df()	pmap_df()	data frame
walk()	walk2()	pwalk()	side effect

List Column Case Study

Babynames that appeared each year

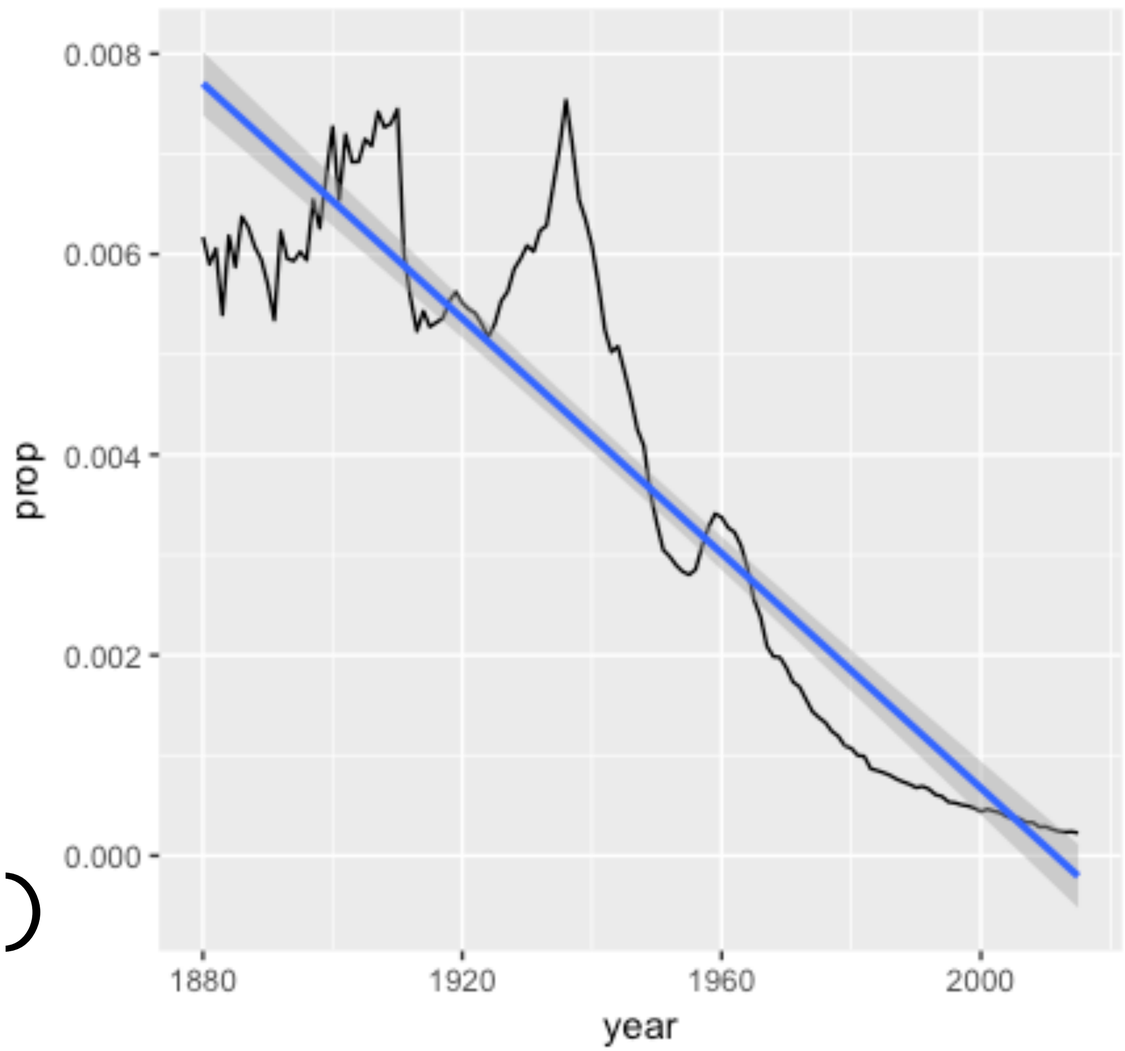
```
everpresent <- babynames %>%
  group_by(name, sex) %>%
  summarise(years = n()) %>%
  ungroup() %>%
  filter(years == max(years))

babynames <- babynames %>%
  semi_join(everpresent)
```

```r
joe <- babynames %>%
  filter(name == "Joe",
         sex == "M")
joe
```

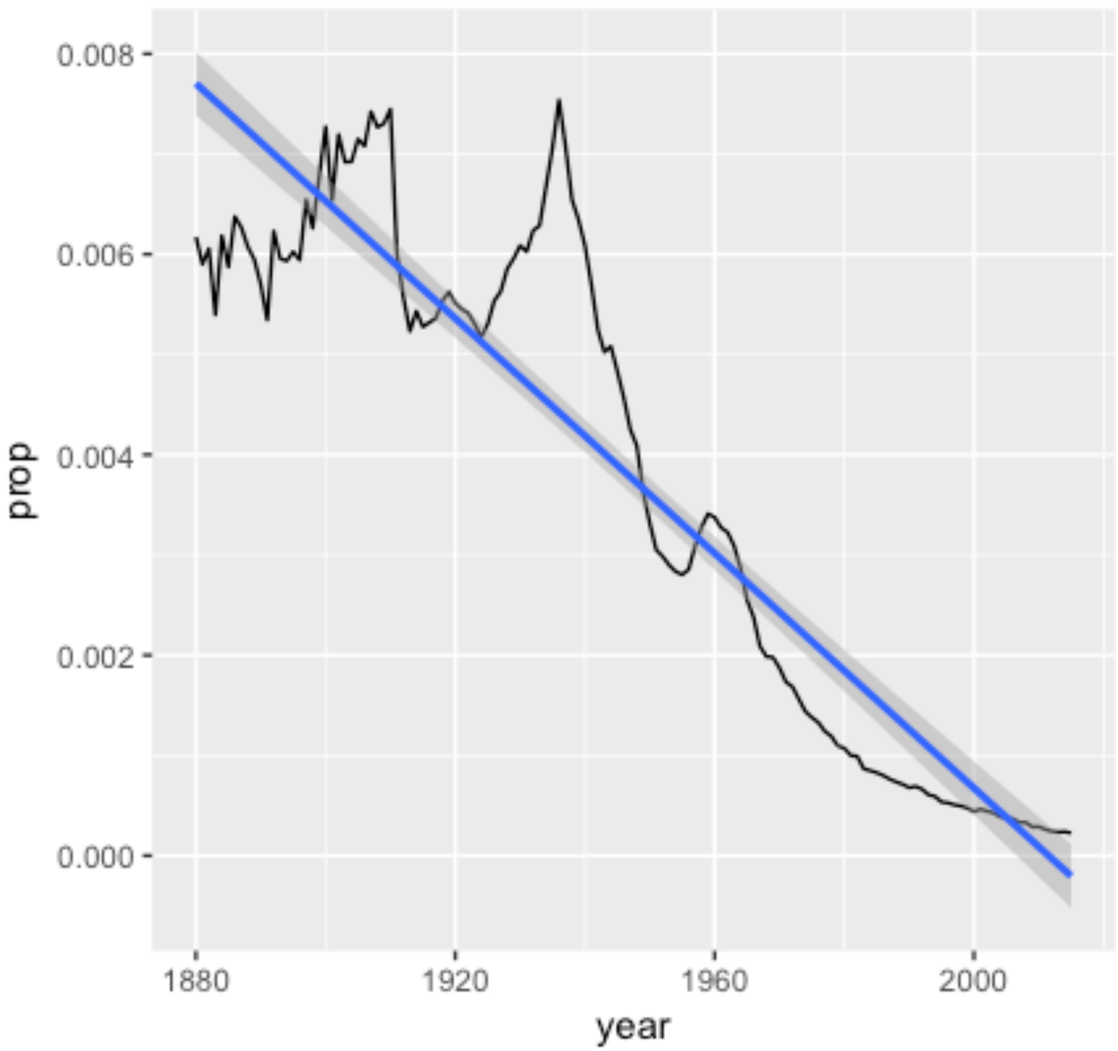
```
# A tibble: 136 x 5
    year sex   name      n    prop
   <dbl> <chr> <chr> <int>   <dbl>
 1  1880 M     Joe     731 0.00617
 2  1881 M     Joe     639 0.00590
 3  1882 M     Joe     739 0.00606
 4  1883 M     Joe     607 0.00540
 5  1884 M     Joe     759 0.00618
 6  1885 M     Joe     681 0.00587
 7  1886 M     Joe     759 0.00638
 8  1887 M     Joe     685 0.00627
 9  1888 M     Joe     789 0.00607
10  1889 M     Joe     708 0.00595
# ... with 126 more rows
```

```
joe <- babynames %>%
  filter(name == "Joe",
         sex == "M")
joe  %>%
  ggplot(mapping =
    aes(x = year,
        y = prop)) +
    geom_line() +
    geom_smooth(method = lm)
```



What is the slope?

What is the R-squared? (fit)



```r
joe_mod <- lm(prop ~ year, data = joe)
joe_mod
```

```
Call:
lm(formula = prop ~ year, data = joe)

Coefficients:
(Intercept)         year
  1.178e-01   -5.857e-05
```

```r
joe_mod <- lm(prop ~ year, data = joe)
coef(joe_mod)
```

```
  (Intercept)          year
 1.178179e-01 -5.857169e-05
```

```r
joe_mod <- lm(prop ~ year, data = joe)
pluck(coef(joe_mod), "year")
```

```
[1] -5.857169e-05
```

```r
joe_mod <- lm(prop ~ year, data = joe)
pluck(coef(joe_mod), "year")
```

```
[1] -5.857169e-05
```

```r
library(broom)
glance(joe_mod)
```

```
  r.squared adj.r.squared        sigma statistic
1 0.8584798     0.8574236 0.0009405581  812.8611 9.
```

```r
joe_mod <- lm(prop ~ year, data = joe)
pluck(coef(joe_mod), "year")
```

```
[1] -5.857169e-05
```

```r
library(broom)
pluck(glance(joe_mod), "r.squared")
```

```
[1] 0.8584798
```

babynames

```
# A tibble: 126,888 x 5
    year sex   name           n   prop
   <dbl> <chr> <chr>      <int>  <dbl>
 1  1880 F     Mary        7065 0.0724
 2  1880 F     Anna        2604 0.0267
 3  1880 F     Emma        2003 0.0205
 4  1880 F     Elizabeth   1939 0.0199
 5  1880 F     Minnie      1746 0.0179
 6  1880 F     Margaret    1578 0.0162
 7  1880 F     Ida         1472 0.0151
 8  1880 F     Alice       1414 0.0145
 9  1880 F     Bertha      1320 0.0135
10  1880 F     Sarah       1288 0.0132
# ... with 126,878 more rows
```

```r
babynames %>%
  group_by(name, sex)
```

```
# A tibble: 126,888 x 5
# Groups:   name, sex [933]
    year sex   name          n   prop
   <dbl> <chr> <chr>     <int>  <dbl>
 1  1880 F     Mary       7065 0.0724
 2  1880 F     Anna       2604 0.0267
 3  1880 F     Emma       2003 0.0205
 4  1880 F     Elizabeth  1939 0.0199
 5  1880 F     Minnie     1746 0.0179
 6  1880 F     Margaret   1578 0.0162
 7  1880 F     Ida        1472 0.0151
 8  1880 F     Alice      1414 0.0145
 9  1880 F     Bertha     1320 0.0135
10  1880 F     Sarah      1288 0.0132
# ... with 126,878 more rows
```

```r
babynames %>%
  group_by(name, sex) %>%
  nest()
```

```
# A tibble: 933 x 3
   name      sex   data
   <chr>     <chr> <list>
 1 Mary      F     <tibble [136 × 3]>
 2 Anna      F     <tibble [136 × 3]>
 3 Emma      F     <tibble [136 × 3]>
 4 Elizabeth F     <tibble [136 × 3]>
 5 Minnie    F     <tibble [136 × 3]>
 6 Margaret  F     <tibble [136 × 3]>
 7 Ida       F     <tibble [136 × 3]>
 8 Alice     F     <tibble [136 × 3]>
 9 Bertha    F     <tibble [136 × 3]>
10 Sarah     F     <tibble [136 × 3]>
# ... with 923 more rows
```

Sanity check: What is in one of these cells?

```
babynames %>%
  group_by(name, sex) %>%
  nest() %>%
  pluck("data") %>%
  pluck(1)
```

```
# A tibble: 136 x 3
    year     n   prop
   <dbl> <int>  <dbl>
 1  1880  7065 0.0724
 2  1881  6919 0.0700
 3  1882  8148 0.0704
 4  1883  8012 0.0667
 5  1884  9217 0.0670
 6  1885  9128 0.0643
 7  1886  9889 0.0643
 8  1887  9888 0.0636
 9  1888 11754 0.0620
10  1889 11648 0.0616
# ... with 126 more rows
```

```r
babynames %>%
  group_by(name, sex) %>%
  nest() %>%
  mutate(
    model = map(data,
      ~lm(prop ~ year,
          data = .x))
  )
```

```
# A tibble: 933 x 4
   name      sex   data                 model
   <chr>     <chr> <list>               <list>
 1 Mary      F     <tibble [136 × 3]> <S3: lm>
 2 Anna      F     <tibble [136 × 3]> <S3: lm>
 3 Emma      F     <tibble [136 × 3]> <S3: lm>
 4 Elizabeth F     <tibble [136 × 3]> <S3: lm>
 5 Minnie    F     <tibble [136 × 3]> <S3: lm>
 6 Margaret  F     <tibble [136 × 3]> <S3: lm>
 7 Ida       F     <tibble [136 × 3]> <S3: lm>
 8 Alice     F     <tibble [136 × 3]> <S3: lm>
 9 Bertha    F     <tibble [136 × 3]> <S3: lm>
10 Sarah     F     <tibble [136 × 3]> <S3: lm>
# ... with 923 more rows
```

```r
babynames %>%
  group_by(name, sex) %>%
  nest() %>%
  mutate(
    model = map(data,
      ~lm(prop ~ year,
          data = .x)),
    slope = map_dbl(model,
      ~pluck(coef(.x),
             "year"))
  )
```

```
# A tibble: 933 x 5
   name      sex   data                 model          slope
   <chr>     <chr> <list>               <list>         <dbl>
 1 Mary      F     <tibble [136 × 3]> <S3: lm> -0.000577
 2 Anna      F     <tibble [136 × 3]> <S3: lm> -0.000179
 3 Emma      F     <tibble [136 × 3]> <S3: lm> -0.0000657
 4 Elizabeth F     <tibble [136 × 3]> <S3: lm> -0.0000725
 5 Minnie    F     <tibble [136 × 3]> <S3: lm> -0.0000966
 6 Margaret  F     <tibble [136 × 3]> <S3: lm> -0.000173
 7 Ida       F     <tibble [136 × 3]> <S3: lm> -0.0000862
 8 Alice     F     <tibble [136 × 3]> <S3: lm> -0.000110
 9 Bertha    F     <tibble [136 × 3]> <S3: lm> -0.0000948
10 Sarah     F     <tibble [136 × 3]> <S3: lm>  0.00000845
# ... with 923 more rows
```

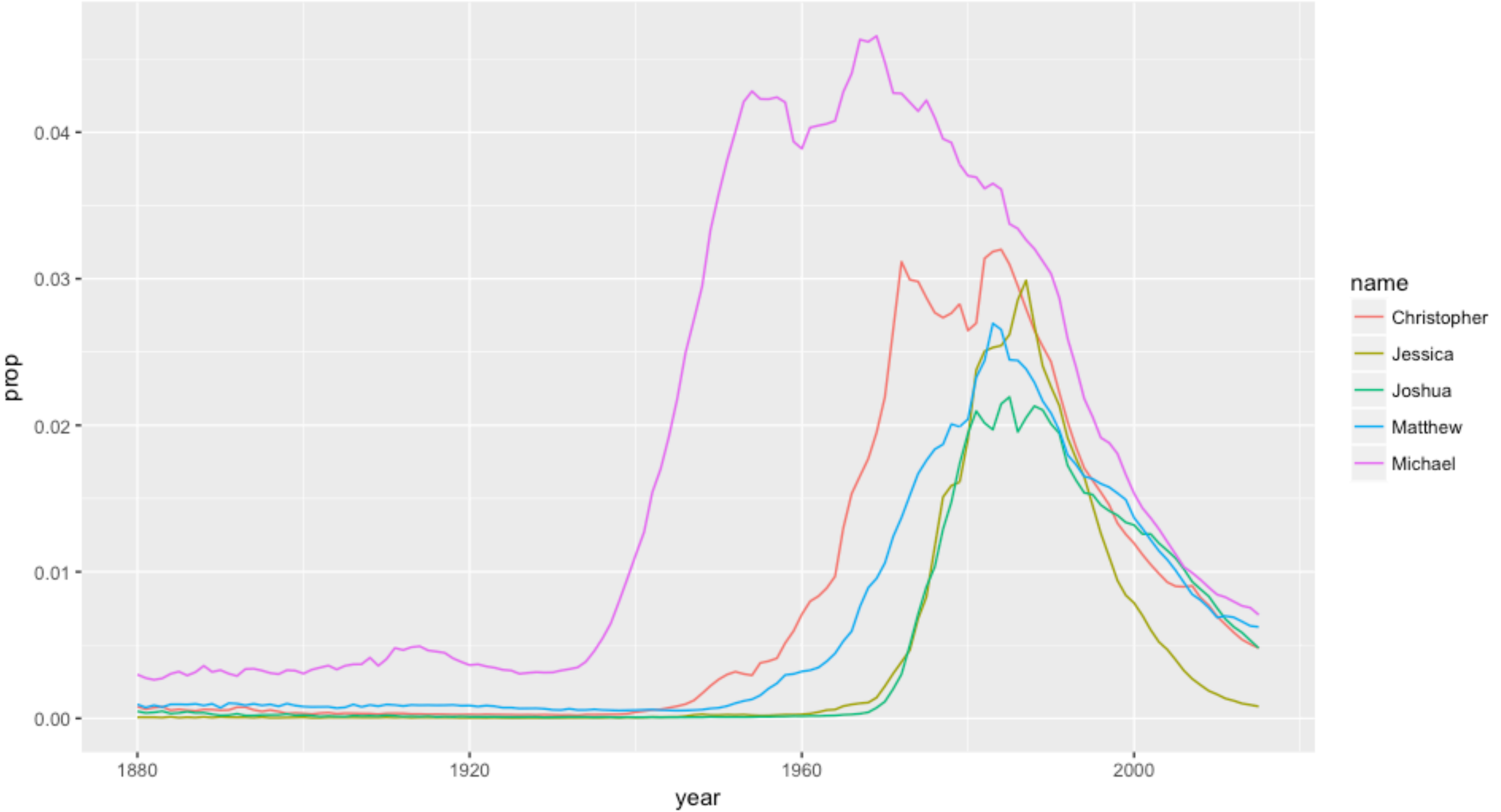
```r
babymods <- babynames %>%
  group_by(name, sex) %>%
  nest() %>%
  mutate(
    model = map(data,
      ~lm(prop ~ year,
          data = .x)),
    slope = map_dbl(model,
      ~pluck(coef(.x),
             "year")),
    r_squared = map_dbl(model,
      ~pluck(glance(.x),"r.squared")))
```

```
# A tibble: 933 x 6
   name      sex   data                 model          slope r_squared
   <chr>     <chr> <list>               <list>         <dbl>     <dbl>
 1 Mary      F     <tibble [136 × 3]> <S3: lm> -0.000577     0.914
 2 Anna      F     <tibble [136 × 3]> <S3: lm> -0.000179     0.708
 3 Emma      F     <tibble [136 × 3]> <S3: lm> -0.0000657    0.230
 4 Elizabeth F     <tibble [136 × 3]> <S3: lm> -0.0000725    0.704
 5 Minnie    F     <tibble [136 × 3]> <S3: lm> -0.0000966    0.644
 6 Margaret  F     <tibble [136 × 3]> <S3: lm> -0.000173     0.803
 7 Ida       F     <tibble [136 × 3]> <S3: lm> -0.0000862    0.719
 8 Alice     F     <tibble [136 × 3]> <S3: lm> -0.000110     0.901
 9 Bertha    F     <tibble [136 × 3]> <S3: lm> -0.0000948    0.756
10 Sarah     F     <tibble [136 × 3]> <S3: lm>  0.00000845   0.00705
# ... with 923 more rows
```

```
# A tibble: 933 x 6
   name      sex   data                 model          slope r_squared
   <chr>     <chr> <list>               <list>         <dbl>     <dbl>
 1 Mary      F     <tibble [136 × 3]> <S3: lm> -0.000577     0.914
 2 Anna      F     <tibble [136 × 3]> <S3: lm> -0.000179     0.708
 3 Emma      F     <tibble [136 × 3]> <S3: lm> -0.0000657    0.230
 4 Elizabeth F     <tibble [136 × 3]> <S3: lm> -0.0000725    0.704
 5 Minnie    F     <tibble [136 × 3]> <S3: lm> -0.0000966    0.644
 6 Margaret  F     <tibble [136 × 3]> <S3: lm> -0.000173     0.803
 7 Ida       F     <tibble [136 × 3]> <S3: lm> -0.0000862    0.719
 8 Alice     F     <tibble [136 × 3]> <S3: lm> -0.000110     0.901
 9 Bertha    F     <tibble [136 × 3]> <S3: lm> -0.0000948    0.756
10 Sarah     F     <tibble [136 × 3]> <S3: lm>  0.00000845   0.00705
# ... with 923 more rows
```

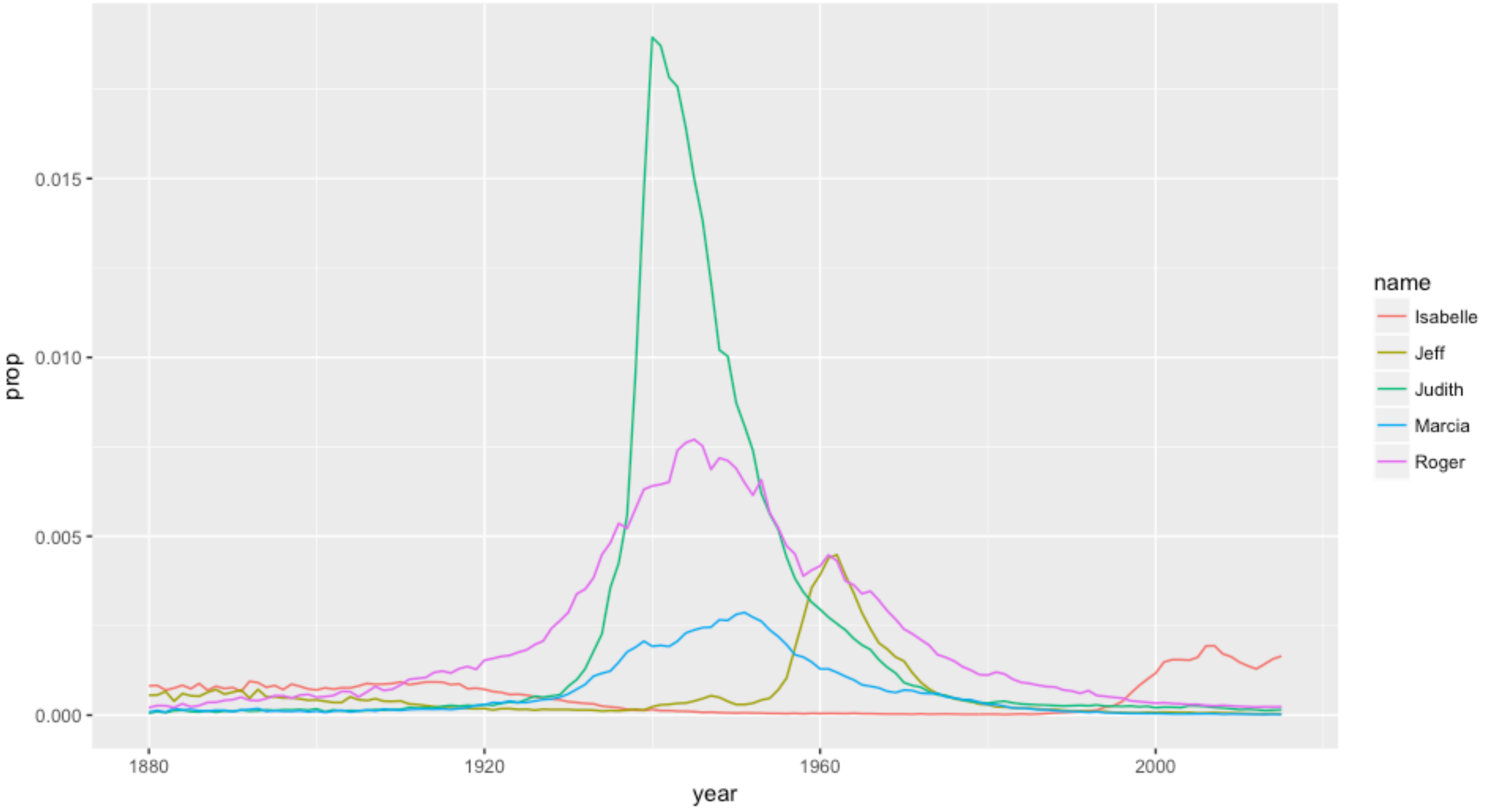
Which names increased the most?

```
babymods %>%
  arrange(desc(slope)) %>%
  head(5) %>%
  unnest(data) %>%
  ggplot(mapping = aes(x = year, y = prop)) +
    geom_line(mapping = aes(color = name))
```


0.04
0.03
0.02
0.01
0.00
prop
1880
1920
1960
2000
year
name
Christopher
Jessica
Joshua
Matthew
Michael

Which names were the least linear?

```
babymods %>%
  arrange(r_squared) %>%
  head(5) %>%
  unnest(data) %>%
  ggplot(mapping = aes(x = year, y = prop)) +
    geom_line(mapping = aes(color = name))
```


0.015
0.010
0.005
0.000
prop
1880
1920
1960
2000
year
name
Isabelle
Jeff
Judith
Marcia
Roger

The tao of tidy

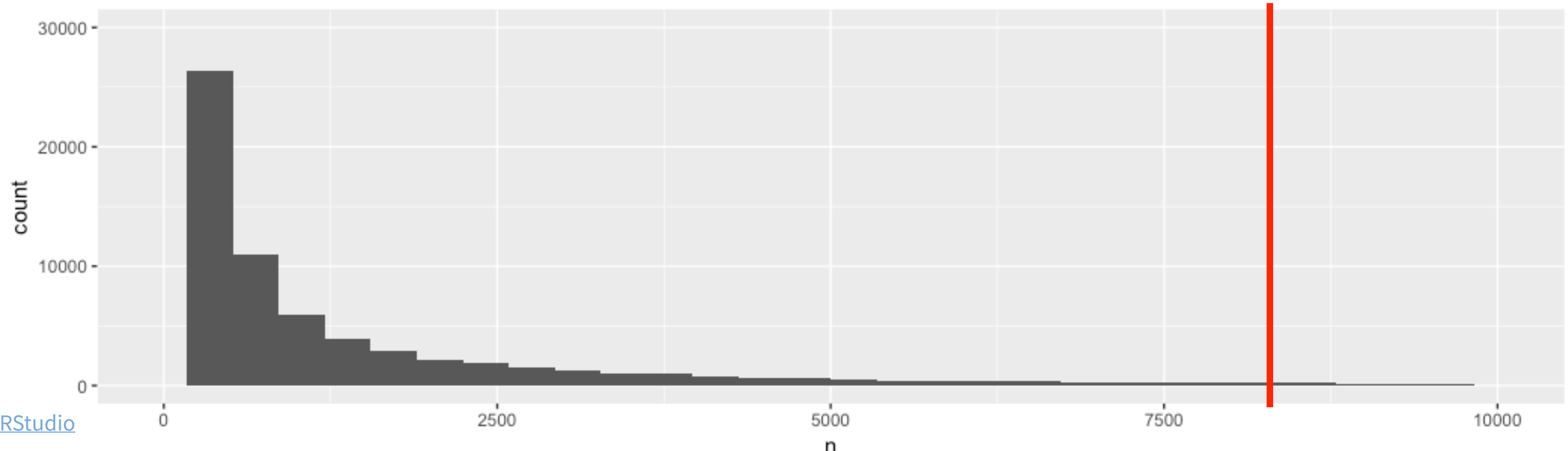
"Data are not just numbers,
they are numbers with a context."

- George Cobb and David Moore (1997)

Two types of context

1. Other values of the **same variable** (all of them)

```
sum(babynames$n)
194086403
```

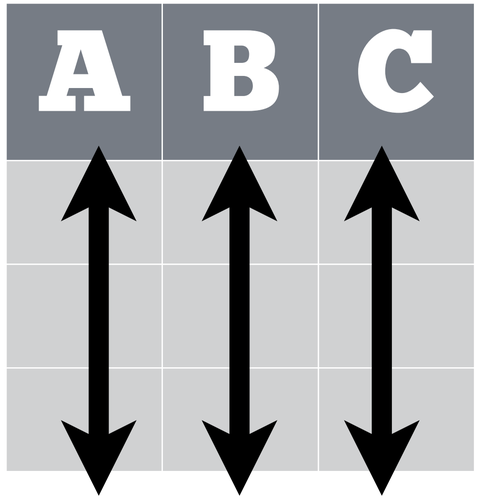


Two types of context

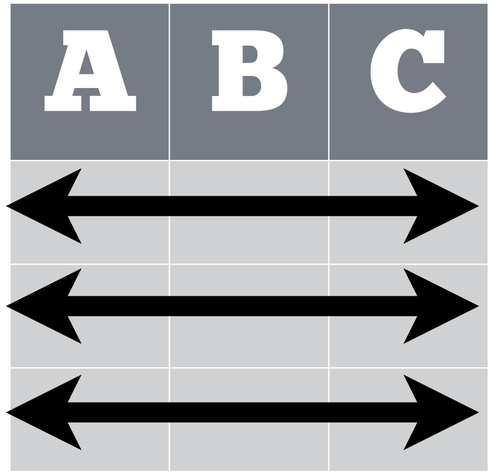
1. Other values of the **same variable** (all of them)
2. Other values of the **same observation**

$$prop = n \ x \ 100?$$

Tidy Data



&

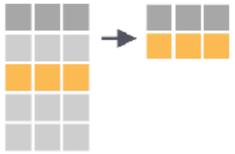


Each **variable** is in its own **column**

Each **observation**, or **case**, is in its own **row**

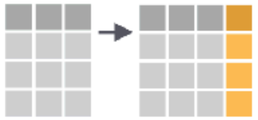
The tidyverse creates a system for working with tidy data

arrange(.data, …)
Order rows by values of a column (low to high), use with **desc()** to order from high to low.



filter(.data, …)
Extract rows that meet logical criteria.

select(.data, …)
Extract columns by name.



mutate(.data, …)
Compute new column(s).

summarise(.data, …)
Compute table of summaries. Use **group_by()** to compute groupwise summaries.

List Columns

You can use this system to organize more than numbers.

How to work with List Columns

August 2018

Garrett Grolemund

rstd.io/list-columns

RStudio

nested data frame

Species	data
setosa	<tibble [50 x 4]>
versicolor	<tibble [50 x 4]>
virginica	<tibble [50 x 4]>

n_iris

"cell" contents

Sepal.L	Sepal.W	Petal.L	Petal.W
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

n_iris$data[[1]]

Sepal.L	Sepal.W	Petal.L	Petal.W
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5

n_iris$data[[2]]

Sepal.L	Sepal.W	Petal.L	Petal.W
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2

n_iris$data[[3]]