Introduction to R for Data Analysis in the Health Sciences: Lecture 2

Amy Willis, Biostatistics, UW

11 October, 2019

Welcome back

- Check out syllabus for *scope*, *grading*, *office hours*, *course policies* and *resources*
- Look at Lecture 1 slides for critical introductory material
- Congratulations on fantastic work on In-Class Exercise and Homework 1!

Norms and ground rules

For all:

- Be respectful and understanding of our diverse experiences
  - Both our *programming experiences* and our *life experiences*
- If comfortable, share your questions and your understandings with the whole class...
  - ...so everyone can benefit from your learning
- Avoid distracting other people during class
  - This includes with gossiping, shopping online, watching TV and using Facebook

For Amy, Serge and Thayer:

- Teach and help inclusively
- Post common questions and their answers on Canvas
- Check in with the class regularly to confirm understanding/following
- Recognise different learning styles and adapt; offer multiple pathways to learning

Normalising struggle

If this is your first time programming in R, it is likely that you will experience frustration and challenge in this course. Know that this is normal and a part of the learning process!

You can do it!

Outline of today plan

- Lecture (data manipulation)
  - Quick review of last week
  - New material: data manipulation
- Break
- In-class exercise

Recap

Last week we talked about the package `tidyverse`...

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------- tidyverse 1
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------- tidyverse_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Recap

. . . and we talked about reading in and storing data. . .

```
fev_data <- "/Users/adwillis/teaching/19-509/datasets/fev.csv" %>%
    read_csv
```

```
## Parsed with column specification:
## cols(
##   seqnbr = col_double(),
##   subjid = col_double(),
##   age = col_double(),
##   fev = col_double(),
##   height = col_double(),
##   sex = col_double(),
##   smoke = col_double()
## )
```

Recap

. . . and we showed how we can filter data. . .

```
is_tall <- fev_data$height > 72
fev_data[is_tall, ]
```

```
## # A tibble: 7 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    401  18841    14  4.27   72.5     1     0
## 2    450  32741    13  4.22   74       1     0
## 3    464  37241    13  4.88   73       1     0
## 4    517  49541    13  5.08   74       1     0
## 5    550  59941    14  4.27   72.5     1     0
## 6    632  37441    17  5.63   73       1     0
## 7    636  44241    16  3.64   73.5     1     0
```

Recap

. . . and take some summary statistics

```
mean(fev_data[is_tall, ]$fev)
```

```
## [1] 4.572429
```

```
sd(fev_data[is_tall, ]$fev)
```

```
## [1] 0.6632754
```

Reminders

- The first line of any R script should be `library(tidyverse)`
- We refer to the rows and columns of a data frame using square brackets, e.g., `fev_data[is_tall, ]`
- We call functions using `function_name(input)` or `input %>% function_name`

Functions and arguments

Functions can have multiple arguments (inputs)

```
some_heights <- c(162, 170, 176, NA)
sd(some_heights)
```

```
## [1] NA
```

```
sd(some_heights, na.rm = TRUE)
```

```
## [1] 7.023769
```

Functions and arguments

Arguments can be referred to by name, or given in the canonical order. Find the canonical order by asking:

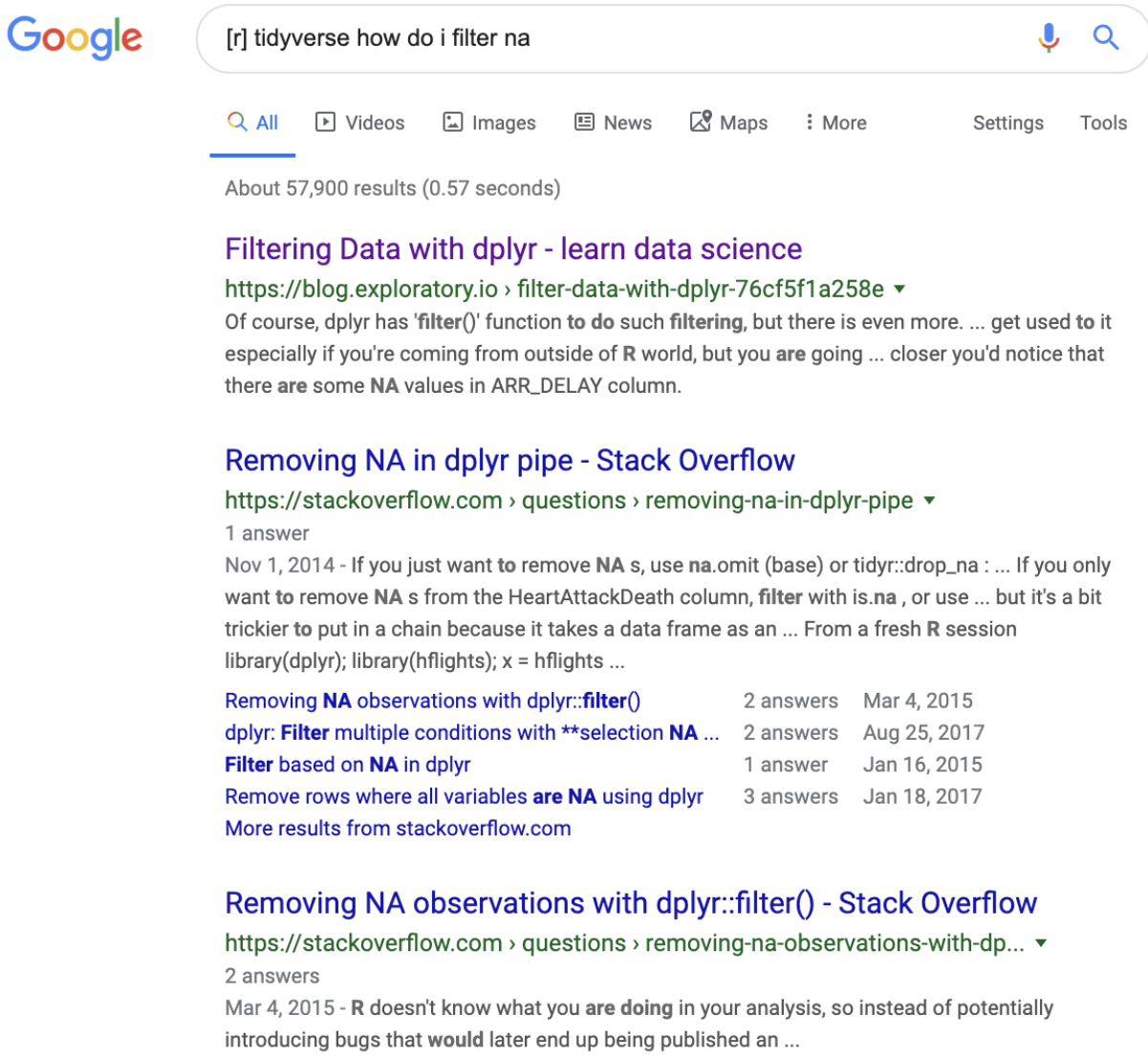
```
?sd
```

Getting help

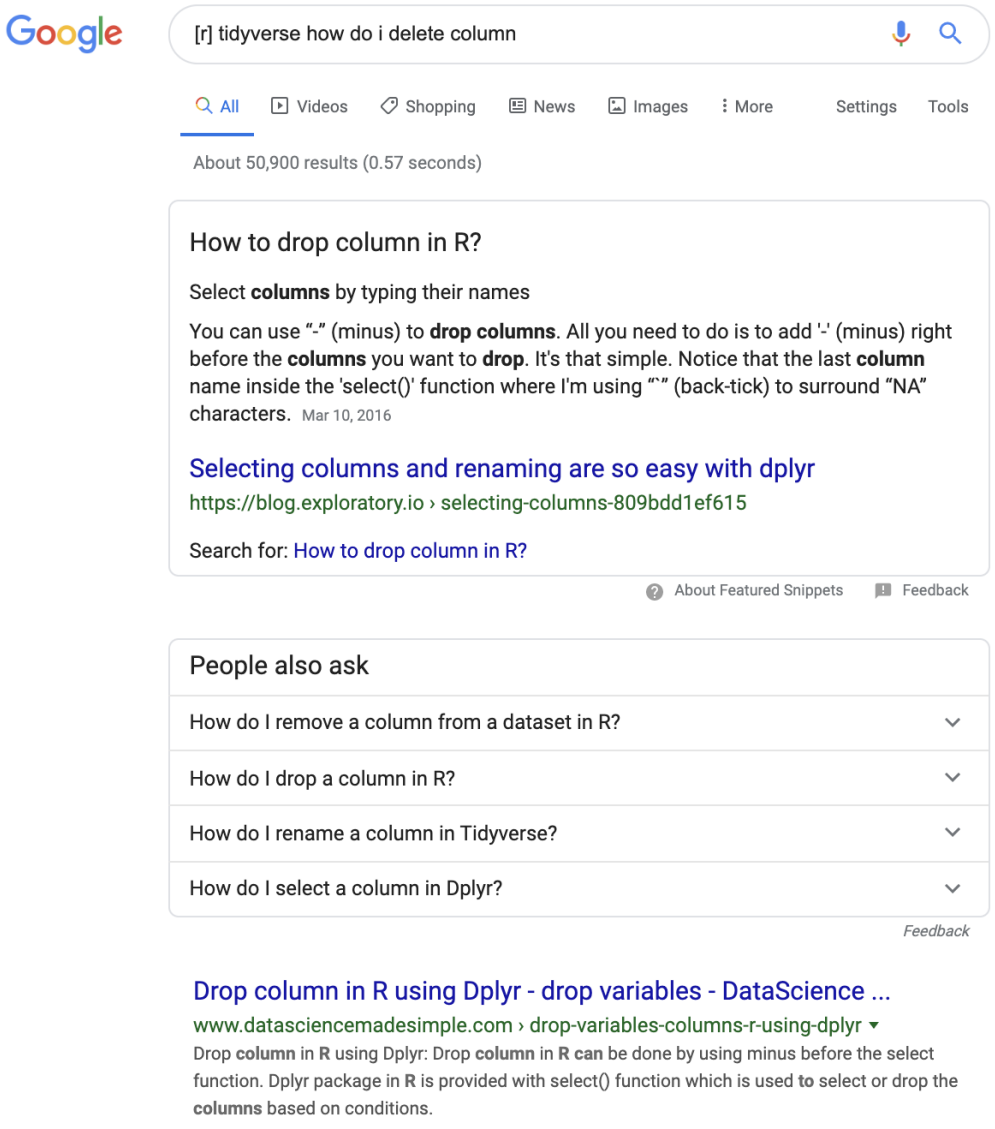
Some functions are well documented and give helpful information. Others are not.

The internet is a fantastic resource!

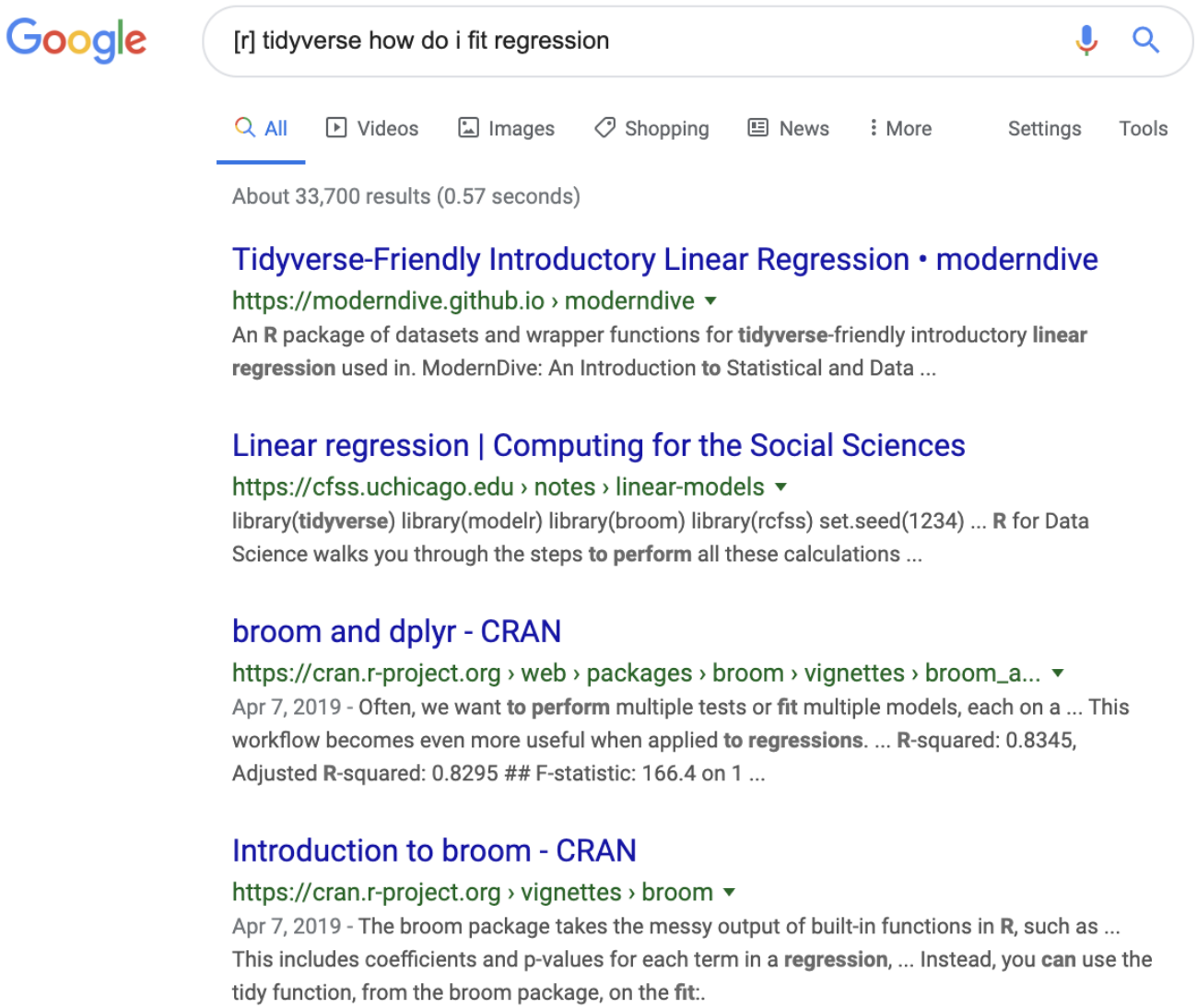
Getting help



Getting help



Getting help



Filtering data

We already know one way to filter data:

```
is_tall <- fev_data$height > 72
fev_data[is_tall, ]
```

```
## # A tibble: 7 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    401  18841    14  4.27   72.5     1     0
## 2    450  32741    13  4.22   74       1     0
## 3    464  37241    13  4.88   73       1     0
## 4    517  49541    13  5.08   74       1     0
## 5    550  59941    14  4.27   72.5     1     0
## 6    632  37441    17  5.63   73       1     0
## 7    636  44241    16  3.64   73.5     1     0
```

Let's learn another way!

Filtering data

Why learn another way?

Easier to remember, faster to write, more robust to errors, highly scalable...

If you don't believe me by the end of the hour, let's chat about it after class!

Filtering data

```
fev_data %>%
  filter(height > 72)
```

```
## # A tibble: 7 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    401  18841    14  4.27   72.5     1     0
## 2    450  32741    13  4.22   74       1     0
## 3    464  37241    13  4.88   73       1     0
## 4    517  49541    13  5.08   74       1     0
## 5    550  59941    14  4.27   72.5     1     0
## 6    632  37441    17  5.63   73       1     0
## 7    636  44241    16  3.64   73.5     1     0
```

Filtering data

```
fev_data %>%
  filter(age == 6)
```

```
## # A tibble: 37 x 7
##    seqnbr subjid   age   fev height   sex smoke
##     <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
##  1      7   1752     6  1.92   58       0     0
##  2      8   1753     6  1.42   56       0     0
##  3     11   1952     6  1.60   53       0     0
##  4     18   3551     6  1.88   53       0     0
##  5     49  10841     6  1.65   55       1     0
##  6     55  12241     6  1.63   54       1     0
##  7     63  14251     6  1.48   51       0     0
##  8     66  14541     6  1.75   57.5     1     0
##  9     80  16151     6  1.72   53       0     0
## 10     82  16252     6  1.70   53       0     0
## # ... with 27 more rows
```

Filtering data

```
fev_data %>%
  filter(age != 20)
```

```
## # A tibble: 654 x 7
##    seqnbr subjid   age   fev height   sex smoke
##     <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
##  1      1    301     9  1.71   57       0     0
##  2      2    451     8  1.72   67.5     0     0
##  3      3    501     7  1.72   54.5     0     0
##  4      4    642     9  1.56   53       1     0
##  5      5    901     9  1.90   57       1     0
##  6      6   1701     8  2.34   61       0     0
##  7      7   1752     6  1.92   58       0     0
##  8      8   1753     6  1.42   56       0     0
##  9      9   1901     8  1.99   58.5     0     0
## 10     10   1951     9  1.94   60       0     0
## # ... with 644 more rows
```

Filtering data

Syntax for filtering rules for continuous data:

- `age == 6`
- `age != 6`
- `fev > 2`
- `fev <= 1.5`: less than or equal to
- `is.na(age)`
- `!is.na(age)`

Filtering data

Syntax for filtering rules for categorical data:

- `sex == "F"` or `sex != "F"`
- `sex %in% c("M", "F")`

Filtering data

You can filter based on multiple criteria, too!

```
fev_data %>%
  filter(age == 14, smoke != 0)
```

```
## # A tibble: 7 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    332   4952    14  2.24     66     0     1
## 2    358  10053    14  3.43     64     0     1
## 3    370  11642    14  3.96     72     1     1
## 4    384  15751    14  3.07     65     0     1
## 5    439  30042    14  4.31     69     1     1
## 6    556  61941    14  2.28     66     1     1
## 7    602  82743    14  4.76     68     1     1
```

Selecting columns

Now we know how to select rows. How do we select columns?

```
fev_data %>%
  select(fev, height, age)
```

```
## # A tibble: 654 x 3
##      fev height   age
##    <dbl>  <dbl> <dbl>
##  1  1.71   57       9
##  2  1.72   67.5     8
##  3  1.72   54.5     7
##  4  1.56   53       9
##  5  1.90   57       9
##  6  2.34   61       8
##  7  1.92   58       6
##  8  1.42   56       6
##  9  1.99   58.5     8
## 10  1.94   60       9
## # ... with 644 more rows
```

Selecting columns

We can also drop columns

```
fev_data %>%
  select(-seqnbr, -subjid)
```

```
## # A tibble: 654 x 5
##      age   fev height   sex smoke
##    <dbl> <dbl>  <dbl> <dbl> <dbl>
##  1     9  1.71   57       0     0
##  2     8  1.72   67.5     0     0
##  3     7  1.72   54.5     0     0
##  4     9  1.56   53       1     0
##  5     9  1.90   57       1     0
##  6     8  2.34   61       0     0
##  7     6  1.92   58       0     0
##  8     6  1.42   56       0     0
##  9     8  1.99   58.5     0     0
## 10     9  1.94   60       0     0
## # ... with 644 more rows
```

Summarising data

Now that we know how to subset our data, how do we summarize it?

Summarising data

How do we summarize our data?

```
fev_data %>%
  filter(age == 14, smoke != 0) %>%
  summarize(mean(fev))
```

```
## # A tibble: 1 x 1
##   `mean(fev)`
##         <dbl>
## 1        3.43
```

Summarising data

You can give a summary variable a name, e.g., `my_mean`:

```
fev_data %>%
  filter(age == 14, smoke != 0) %>%
  summarize(my_mean = mean(fev))
```

```
## # A tibble: 1 x 1
##   my_mean
##     <dbl>
## 1    3.43
```

Summarising data

We can summarize in multiple ways at once!

```
fev_data %>%
  filter(age == 14, smoke != 0) %>%
  summarize(mean(fev), sd(fev))
```

```
## # A tibble: 1 x 2
##   `mean(fev)` `sd(fev)`
##         <dbl>     <dbl>
## 1        3.43     0.976
```

Both `summarise` and `summarize` work!

Summarising data

But what if I want the average FEV for both smokers and non-smokers? Do I have to repeat this?

No – I can create a grouping variable!

Summarising data

Create a grouping variable with `group_by`

```
fev_data %>%
  group_by(smoke)
```

```
## # A tibble: 654 x 7
## # Groups:   smoke [2]
##    seqnbr subjid   age   fev height   sex smoke
##     <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
##  1      1    301     9  1.71   57       0     0
##  2      2    451     8  1.72   67.5     0     0
##  3      3    501     7  1.72   54.5     0     0
##  4      4    642     9  1.56   53       1     0
##  5      5    901     9  1.90   57       1     0
##  6      6   1701     8  2.34   61       0     0
##  7      7   1752     6  1.92   58       0     0
##  8      8   1753     6  1.42   56       0     0
##  9      9   1901     8  1.99   58.5     0     0
## 10     10   1951     9  1.94   60       0     0
## # ... with 644 more rows
```

This doesn't change the dataset in any way (except the groups are listed)

Summarising data

Now when we go to `summarize` there is a summary for each group:

```
fev_data %>%
  group_by(smoke) %>%
  summarize(mean(fev), sd(fev))
```

```
## # A tibble: 2 x 3
##   smoke `mean(fev)` `sd(fev)`
##   <dbl>       <dbl>     <dbl>
## 1     0        2.57     0.851
## 2     1        3.28     0.750
```

Note: `smoke == 1` is smokers. Interesting...

Summarising data

It's hard to interpret differences in mean and standard deviation without knowledge of the sample size in each group!

```
fev_data %>%
  group_by(smoke) %>%
  summarize(mean(fev), sd(fev))
```

```
## # A tibble: 2 x 3
##   smoke `mean(fev)` `sd(fev)`
##   <dbl>       <dbl>     <dbl>
## 1     0        2.57     0.851
## 2     1        3.28     0.750
```

Summarising data

Use `n()` with `summarize` to get the number of observations in each group:

```
fev_data %>%
  group_by(smoke) %>%
  summarize(n = n(), mean = mean(fev), sd = sd(fev))
```

```
## # A tibble: 2 x 4
##   smoke     n  mean    sd
##   <dbl> <int> <dbl> <dbl>
## 1     0   589  2.57 0.851
## 2     1    65  3.28 0.750
```

Summarising data

You can also summarize based on your own variables!

```
fev_data %>%
  group_by(height < 60) %>%
  summarize(n(), mean(fev))
```

```
## # A tibble: 2 x 3
##   `height < 60` `n()` `mean(fev)`
##   <lgl>          <int>       <dbl>
## 1 FALSE            409        3.10
## 2 TRUE             245        1.86
```

Think-pair-share

```
fev_data$sex %>% unique
```

```
## [1] 0 1
```

Suppose I was given this dataset without information about variable coding. We have 2 sexes observed in this dataset – most likely binary male and female. How could we figure out if `sex == 1` is male or female?

```
fev_data %>%
  select(-seqnbr, -subjid) %>%
  names
```

```
## [1] "age"    "fev"    "height" "sex"    "smoke"
```

Discuss with a partner how you might investigate this (2 minutes).

Think-pair-share

[Live demo]

Final data analysis

Solving the smokers-have-better-lung-function paradox:

```
fev_data %>%
  group_by(smoke) %>%
  summarize(n(), mean(fev), sd(fev))
```

```
## # A tibble: 2 x 4
##   smoke `n()` `mean(fev)` `sd(fev)`
##   <dbl> <int>       <dbl>     <dbl>
## 1     0   589        2.57     0.851
## 2     1    65        3.28     0.750
```

This was surprising to me. What could be confounded with smoking status?

Final data analysis: smoking and lung function in children

```
fev_data %>%
  group_by(smoke) %>%
  summarize(mean(age))
```

```
## # A tibble: 2 x 2
##   smoke `mean(age)`
##   <dbl>       <dbl>
## 1     0        9.53
## 2     1       13.5
```

Smokers are typically *older* than non-smokers!

Final data analysis: smoking and lung function in children

```
fev_data %>%
  group_by(age, smoke) %>%
  summarize(n(), mean(fev), sd(fev))
```

```
## # A tibble: 28 x 5
## # Groups:   age [17]
##      age smoke `n()` `mean(fev)` `sd(fev)`
##    <dbl> <dbl> <int>       <dbl>     <dbl>
##  1     3     0     2        1.24     0.235
##  2     4     0     9        1.28     0.353
##  3     5     0    28        1.55     0.308
##  4     6     0    37        1.66     0.223
##  5     7     0    54        1.87     0.335
##  6     8     0    85        2.12     0.391
##  7     9     0    93        2.44     0.478
##  8     9     1     1        1.95    NA
##  9    10     0    76        2.67     0.533
## 10    10     1     5        3.06     0.441
## # ... with 18 more rows
```

Sorting data

```
fev_data %>%
  group_by(age, smoke) %>%
  summarize(n(), mean(fev), sd(fev)) %>%
  arrange(age) # arrange by increasing age
```

```
## # A tibble: 28 x 5
## # Groups:   age [17]
##      age smoke `n()` `mean(fev)` `sd(fev)`
##    <dbl> <dbl> <int>       <dbl>     <dbl>
##  1     3     0     2        1.24     0.235
##  2     4     0     9        1.28     0.353
##  3     5     0    28        1.55     0.308
##  4     6     0    37        1.66     0.223
##  5     7     0    54        1.87     0.335
##  6     8     0    85        2.12     0.391
##  7     9     0    93        2.44     0.478
##  8     9     1     1        1.95    NA
##  9    10     0    76        2.67     0.533
## 10    10     1     5        3.06     0.441
## # ... with 18 more rows
```

Putting it all together

```
fev_data %>%
  group_by(age, smoke) %>%
  summarize(n = n(), # name summary statistics columns...
            mean = mean(fev), sd = sd(fev)) %>%
  filter(n >= 5) %>% # ... so we can filter on them later
  arrange(desc(age)) # arrange by decreasing age
```

```
## # A tibble: 21 x 5
## # Groups:   age [14]
##      age smoke     n  mean    sd
##    <dbl> <dbl> <int> <dbl> <dbl>
##  1    17     0     6  4.65 0.871
##  2    16     0     6  3.75 0.567
##  3    16     1     7  3.60 0.858
##  4    15     0     9  3.92 0.960
##  5    15     1    10  3.09 0.766
##  6    14     0    18  3.64 0.627
##  7    14     1     7  3.43 0.976
##  8    13     0    30  3.53 0.717
##  9    13     1    13  3.38 0.776
## 10    12     0    50  3.24 0.733
## # ... with 11 more rows
```

Summary

Today we learnt 5 `tidyverse` functions that facilitate data analysis:

- `filter()` picks rows based on their valuess
- `select()` picks columns based on their names
- `summarize()` reduces multiple values to a single summary
- `group_by()` groups rows together (useful for `summarize`)
- `arrange()` changes the ordering of the rows

The plan

- 5 minute break
- In-class exercise available via Canvas and github.com/adw96/biost509
  - Designed to be completed by 3:20 p.m.
  - Due today 5 p.m.
  - Place a *yellow sticky note* on your computer to indicate you are stuck, or a *blue sticky note* to indicate you have a non-urgent question
- Homework due next week by 1 p.m. Friday

The TAs and I will walk around to help you; we will help *yellow stickies*, then *blue stickies*. Please put your sticky note high enough that we can see it!