Introduction to R for Data Analysis in the Health Sciences: Lecture 4

Amy Willis, Biostatistics, UW

25 October, 2019

Today

- Perspective on tidyverse
- R Projects
- Plotting
  - base `R`
  - `ggplot2`, which I'm just going to call "ggplot"

Many thanks to those of you who submitted anonymous feedback; In-Class Exercise 4 will ask everyone for some feedback about the course (pacing, clarity, etc.)

Why are we talking about tidyverse?

Streamline analysis: compare the following

```r
# no tidyverse
with(subset(seizure_data_frame, Trt == 0), mean(BL))
with(subset(seizure_data_frame, Trt == 0), sd(BL))
with(subset(seizure_data_frame, Trt == 1), mean(BL))
with(subset(seizure_data_frame, Trt == 1), sd(BL))

# with tidyverse
library(tidyverse)
seizure_data_frame %>%
  group_by(Trt) %>%
  summarise(mean(BL), sd(BL))
```

How about with 5 categories? 20 categories?

Why learn tidyverse?

Streamline data manipulation

```r
# no tidyverse
seizure_data_frame$TotalSeizures <-
  seizure_data_frame$Y1 + seizure_data_frame$Y2 +
  seizure_data_frame$Y3 + seizure_data_frame$Y4

# with tidyverse
seizure_data_frame %>%
  mutate(TotalSeizures = Y1 + Y2 + Y3 + Y4)
```

Why learn tidyverse?

`read_csv` $>>$ `read.csv`

- tibbles instead of data frames
  - Default printing is 10 lines, not 10,000 lines
  - Tells you the column types (factor or character; integer or double)
  - Significantly easier summaries
- Tells you about ambiguities; makes intelligent guesses
- Formats dates nicely
- Doesn't convert strings to factors

Why learn tidyverse?

Without tidyverse:

```r
inflamm <- read.table("datasets/inflamm.txt",header=T)
with(inflamm, table(male,diab2))
```

```
##      diab2
## male    0    1
##     0 2457  410
##     1 1694  382
```

```r
inflamm2 <- subset(inflamm, !is.na(diab2))
nfemale <- with(inflamm2,sum(male==0))
nDiabfemale <- with(inflamm2,sum(male==0 & diab2==1))
pDiabfemale <- nDiabfemale/nfemale
pDiabfemale
```

```
## [1] 0.1430066
```

Why learn tidyverse?

With tidyverse:

```r
library(tidyverse)
inflam <- read_tsv("datasets/inflamm.txt")
inflam %>%
  group_by(male, diab2) %>%
  summarise(n())
```

```
## # A tibble: 6 x 3
## # Groups:   male [2]
##    male diab2 `n()`
##   <dbl> <dbl> <int>
## 1     0     0  2457
## 2     0     1   410
## 3     0    NA    37
## 4     1     0  1694
## 5     1     1   382
## 6     1    NA    20
```

Why learn tidyverse?

```
inflam %>%
  filter(!is.na(diab2)) %>%
  group_by(male, diab2) %>%
  summarise(n = n()) %>%
  mutate(prop = n/sum(n))
```

```
## # A tibble: 4 x 4
## # Groups:   male [2]
##    male diab2     n  prop
##   <dbl> <dbl> <int> <dbl>
## 1     0     0  2457 0.857
## 2     0     1   410 0.143
## 3     1     0  1694 0.816
## 4     1     1   382 0.184
```

Why learn tidyverse?

High-quality graphics with `ggplot2`! More later…

Workflow versus product

Many of you have been submitting homeworks with the first line `setwd("/Users/adwillis/BIOST509")`... which is fine... but not “best practice”.

Introducing a better alternative: RStudio Projects

Workflow versus product

Your workflow:

- Your operating system
- Your code editor (e.g. RStudio)
- The name of your home directory (mine is `/Users/adwillis`; yours might be `C:/Users/yourname/`)
- The code you ran yesterday
- The programs you have installed at this time

Workflow versus product

Your product:

- The raw data
- The code that needs to be run on the raw data to get results
  - including required packages
- The results/report that you produce at the end of your analysis

Workflow versus product

"Any R script you write should be written assuming that it will be run from a fresh R process with working directory set to the project directory. It creates everything it needs, in its own workspace or folder, and it touches nothing it did not create. . . " – Jenny Bryan

Workflow versus product

Avoid hardwiring your workflow into your product:

- Every distinct intellectual unit you work on should have its own folder
  - every homework, data analysis, every method, every research project, every class
- You can type `install.packages` into your console (that's workflow!), but use `library` in a script (that's product!)

Workflow versus product

Avoid hardwiring your workflow into your product. This allows your code to be

- Portable
  - other computers
  - robust to your own reorganisation
- Polite
  - Avoids overwriting or interfering with other projects
  - Even your computer setup may change in a year!

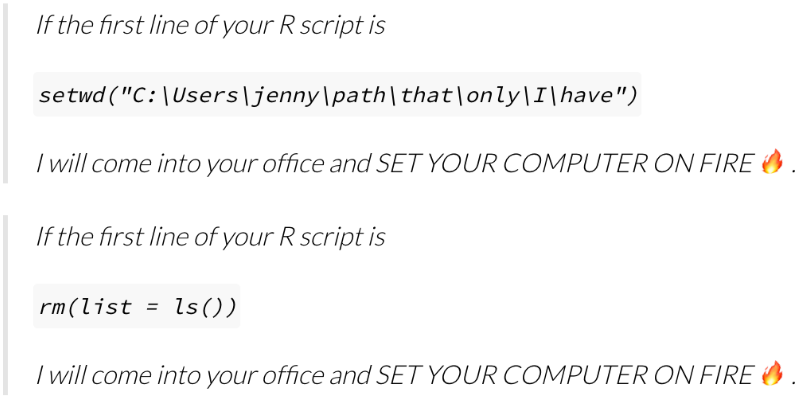
RStudio Projects

RStudio Projects are an alternative to `setwd` that avoids hardwiring your workflow into your product

- Create a project with File -> New Project
- RStudio creates `myproj.Rproj`, which goes in your working directory for that project
- Double-click on your `Rproj` file to open a fresh RStudio instance in the correct working directory
  - Has the capacity to load your `RData`
- You can run multiple R sessions using multiple RStudio Projects at once

Prof. Jenny Bryan feels very strongly about this

`https://www.tidyverse.org/articles/2017/12/workflow-vs-script/`



> *If the first line of your R script is*
>
> ```
> setwd("C:\Users\jenny\path\that\only\I\have")
> ```
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥 .*

> *If the first line of your R script is*
>
> ```
> rm(list = ls())
> ```
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥 .*

Prof. Jenny Bryan feels very strongly about this

This certainly violates our class norms (“inclusive teaching”, “recognise different learning styles”), so we are not going to do it!

But I encourage you to use `File -> New Project` instead of `setwd()`.

Graphical communication in practice

Graphical communication is critical for both *exploring* and *explaining* data

Today we will discuss two options

- base graphics in `R`
- `ggplot2`

Base `R` graphics are great for quickly viewing and checking your data, but publication-quality figures are easier to create in `ggplot`

Plotting data with base R

We will start by revisiting the "FEV" lung function dataset

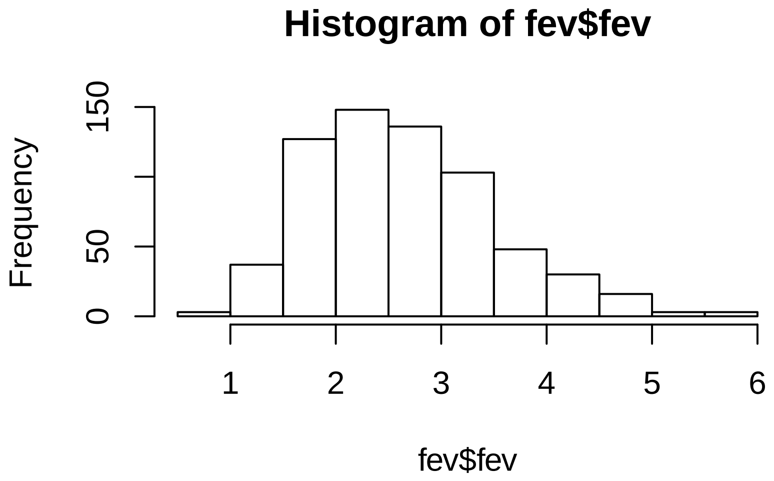
```
library(tidyverse)
fev <- read_csv("datasets/fev.csv")
```

```
## 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()
## )
```

Plotting histograms

We can draw a histogram of the column `fev` using `hist`

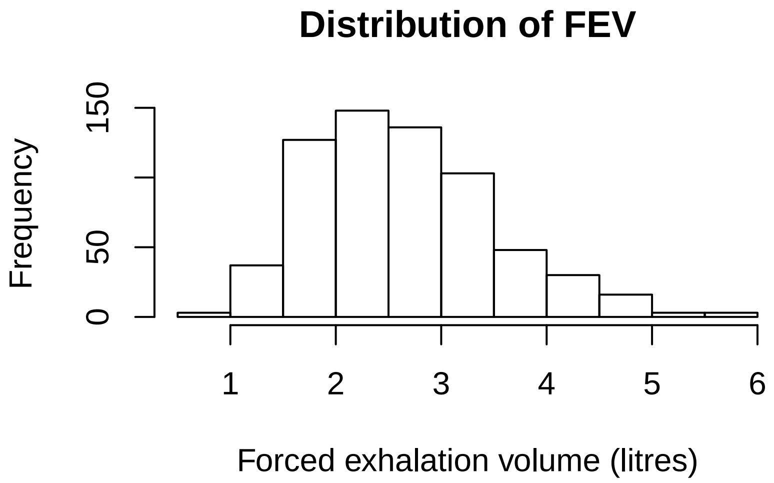
```
hist(fev$fev)
```



Plotting histograms

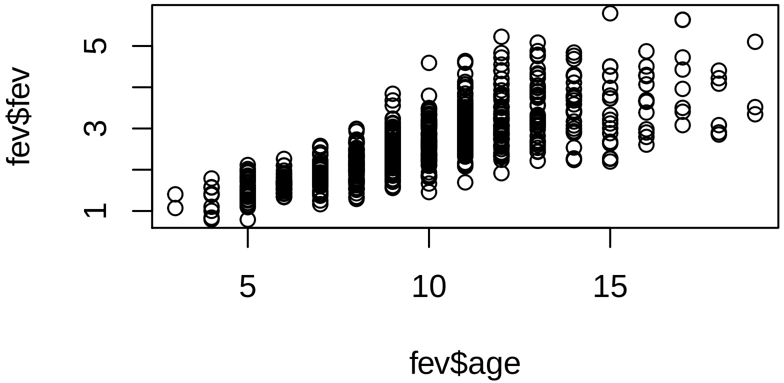
We can add labels to the x-axis with `xlab`, and a title with `main`

```r
hist(fev$fev, xlab="Forced exhalation volume (litres)",
     main = "Distribution of FEV")
```



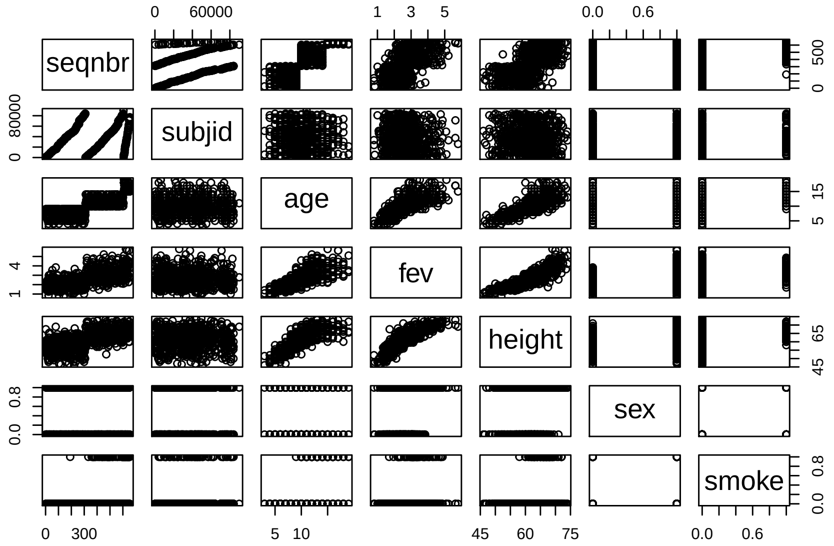
Plotting scatterplots

```
plot(x=fev$age, fev$fev)
```



Plotting scatterplots

```r
plot(fev)
```



Plotting scatterplots

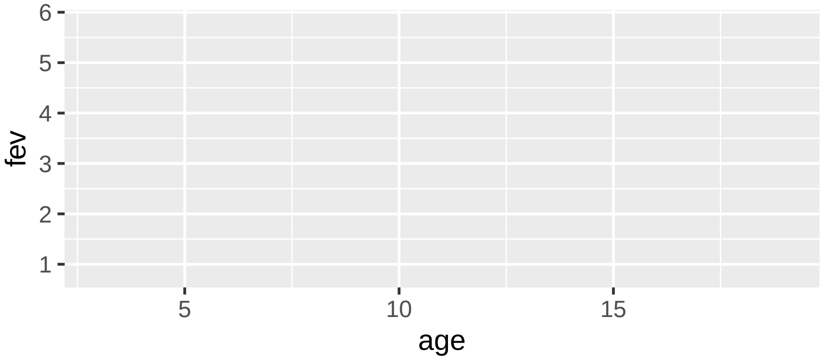
```r
pdf("fev_all_by_all.pdf")
plot(fev)
dev.off()
```

```
## pdf
##   2
```

ggplot:

The first step to using `ggplot` is creating a blank canvas

```
library(tidyverse)
fev %>%
  ggplot(aes(x = age, y = fev))
```



ggplot:

```r
fev %>%
  ggplot(aes(x = age, y = fev))
```

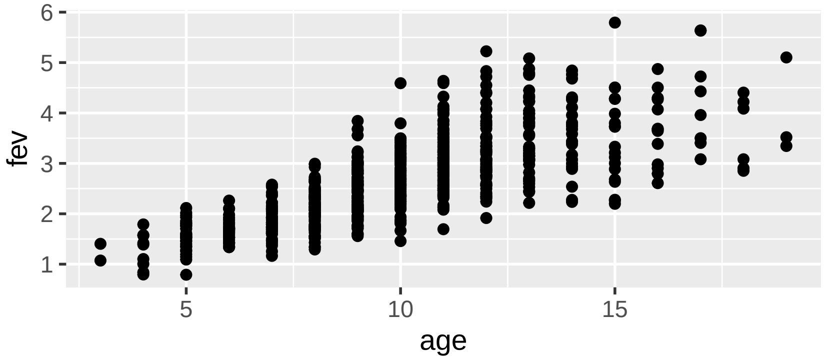
We have created a blank canvas, with `fev` as the input data. `age` will be the variable on the x-axis and `fev` will be the variable on the y-axis.

`aes` stands for “aesthetic”

ggplot: scatterplots

We then *add* points to the canvas using `geom_point`

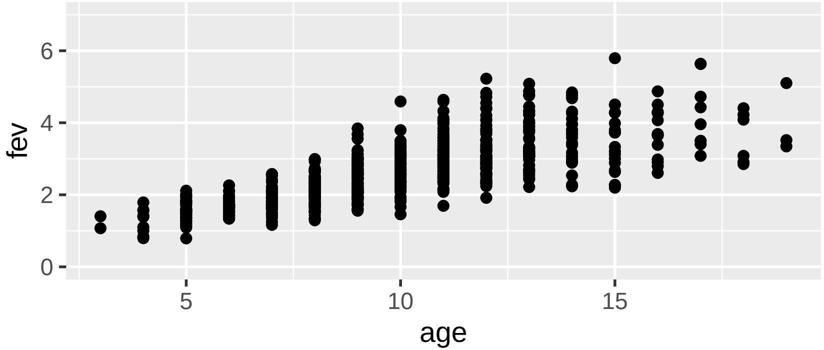
```
fev %>%
  ggplot(aes(x = age, y = fev)) +
  geom_point()
```



ggplot: Scatterplots

Modify the plot by adding elements to it; for example, the limits (range) of the y-axis…

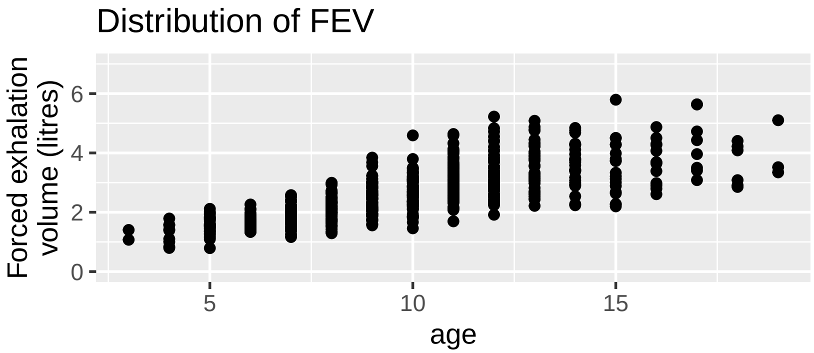
```
fev %>%
  ggplot(aes(x = age, y = fev)) +
  geom_point() +
  ylim(0, 7)
```



ggplot: Scatterplots

... labels on the y-axis, and a title:

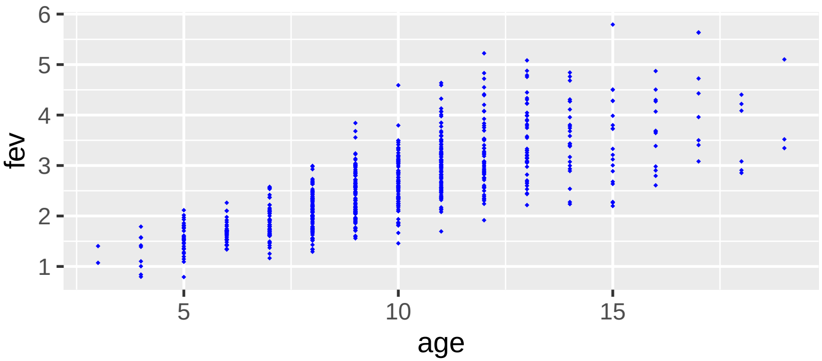
```
fev %>%
  ggplot(aes(x = age, y = fev)) +
  geom_point() +
  ylim(0, 7) +
  ylab("Forced exhalation\nvolume (litres)") +
  ggtitle("Distribution of FEV")
```



ggplot: Scatterplots

Suppose we want all points to be half as big as the default, with triangles as the plotting character, and all points to be blue:

```
fev %>%
  ggplot(aes(x = age, y = fev)) +
  geom_point(cex = 0.5, pch = 18, col = "blue")
```



We want this to affect *all points*, so we add them as an argument to `geom_point`

`ggplot`: Scatterplots

- `cex`: character expansion
- `pch`: plotting character
- `col`: colour

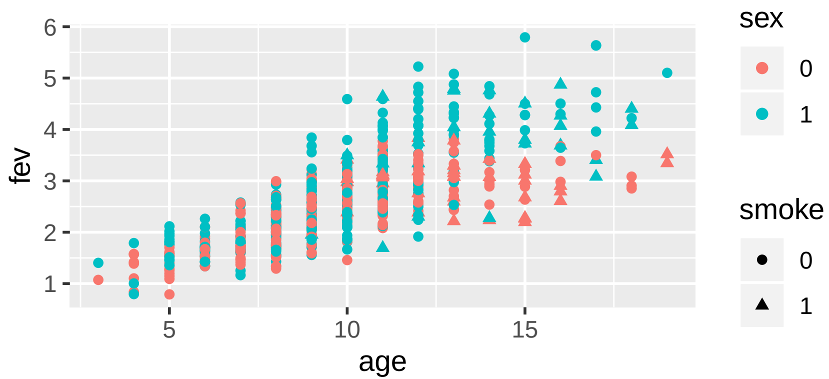
ggplot: Scatterplots

But how do we colour by sex or smoking status?

ggplot: Scatterplots

We add the variables as aesthetic elements:

```
fev %>%
  mutate(sex = as.character(sex),
         smoke = as.character(smoke)) %>%
  ggplot(aes(x = age, y = fev)) +
  geom_point(aes(col = sex, pch = smoke))
```

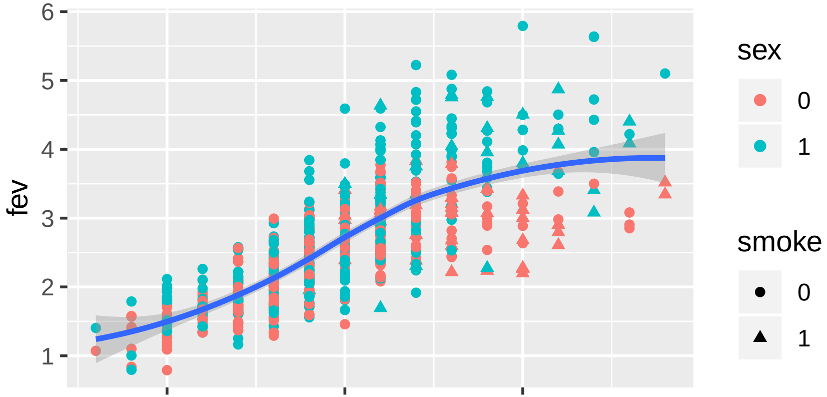


ggplot: Scatterplots

aes in ggplot affects all plotting elements. aes in a specific plotting element only affects that element.

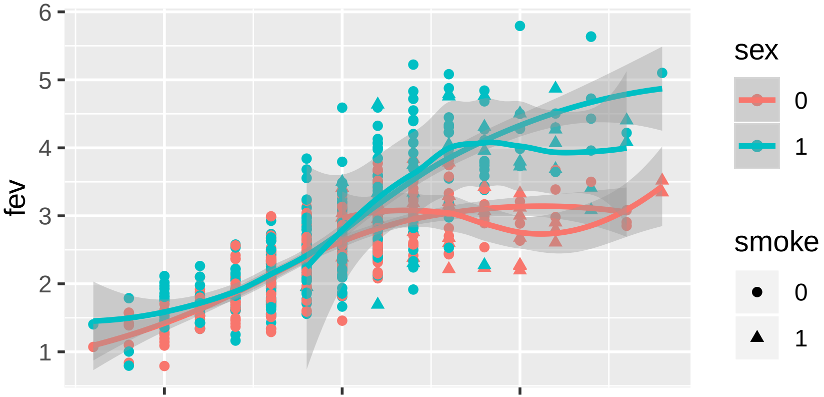
ggplot: Scatterplots

```
fev %>%
  mutate(sex = as.character(sex),
         smoke = as.character(smoke)) %>%
  ggplot(aes(x = age, y = fev)) +
  geom_point(aes(col = sex, pch = smoke)) +
  geom_smooth()
```



ggplot: Scatterplots

```
fev %>%
  mutate(sex = as.character(sex),
         smoke = as.character(smoke)) %>%
  ggplot(aes(x = age, y = fev, col = sex, pch = smoke)) +
  geom_point() +
  geom_smooth()
```

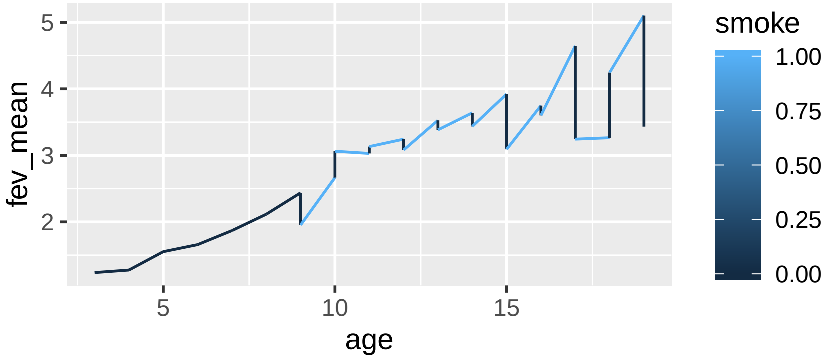


ggplot: Line plots

```
fev_average <- fev %>%
  group_by(age, smoke) %>%
  summarise(fev_mean = mean(fev))
```

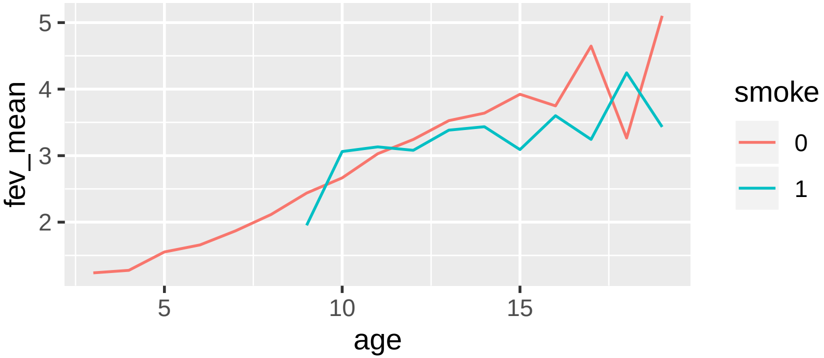
ggplot: Line plots

```r
fev_average %>%
  ggplot(aes(x = age, y = fev_mean)) +
  geom_line(aes(col = smoke))
```



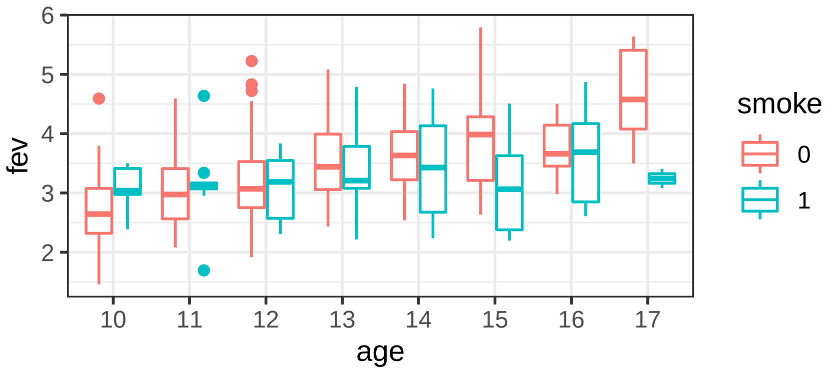
ggplot: Line plots

```r
fev_average %>%
  mutate(smoke = as.character(smoke)) %>%
  ggplot(aes(x = age, y = fev_mean)) +
  geom_line(aes(col = smoke))
```



ggplot: Boxplots

```
fev %>%
  filter(age < 18 & age > 9) %>%
  mutate(sex = as.character(sex),
         smoke = as.character(smoke),
         age = as.factor(age)) %>%
  ggplot(aes(x = age, y = fev, col = smoke)) +
  geom_boxplot() + theme_bw()
```



ggplot: Saving

You can save the last plot using ggsave

```
ggsave("age_vs_fev.pdf")
```

```
## Saving 10 x 7 in image
```

I recommend the package ggpubr for tiling multiple plots on the same image

ggplot: Saving

You can toggle the size as follows:

```r
ggsave("age_vs_fev.pdf", width=10, height=5)
```

ggplot

- `ggplot` is a function available in the `ggplot2` package, which is in the `tidyverse`
- Graphics are built in layers: a plot is initialised, *then* data is then drawn, *then* annotations are added.
- Annotations include
    - ranges
    - labels
    - legends
    - backgrounds

Summary

- Consider using RStudio Projects instead of `setwd`
- Easy plots: `hist` and `plot`
- Beautiful plots: `ggplot`
  - `ggplot` takes time to learn and master, but is very flexible and produces very high quality images

Next week: Fitting regression models in `R` (including "What are factors?")

The plan

- 5 minute break
- In-class exercise available via Canvas
    - Designed to be completed by 3:20 p.m.
    - Due today 6:30 p.m.
    - Feedback form: `https://forms.gle/g54koz2YNq41C5r68`
    - *Yellow sticky note* = urgent; *blue sticky note* = non-urgent
- Homework due next week by 1 p.m. Friday
    - Content advisory: Use of force by police; optional questions on data analysis through a racial equity lens
- Office hours next week as always: Tuesday (Serge), Wednesday (me), Thursday (Thayer)
- Give me feedback at any time via `https://www.get3sixty.com/:h87nqf`