

Introduction to ggplot2

R Pruim

CVC 2016

Goals

What I will try to do

- ▶ give a tour of `ggplot2`
- ▶ explain how to think about plots the `ggplot2` way
- ▶ prepare/encourage you to learn more later

What I can't do in one session

- ▶ show every bell and whistle
- ▶ make you an expert at using `ggplot2`

Set up

```
require(mosaic)
require(lubridate) # package for working with dates
data(Births78)      # restore fresh version of Births78
head(Births78, 3)
```

##		date	births	dayofyear	wday
##	1	1978-01-01	7701	1	Sun
##	2	1978-01-02	7527	2	Mon
##	3	1978-01-03	8825	3	Tues

The grammar of graphics

geom: the geometric “shape” used to display data (glyph)

- ▶ bar, point, line, ribbon, text, etc.

aesthetic: an attribute controlling how geom is displayed

- ▶ x position, y position, color, fill, shape, size, etc.

scale: conversion of raw data to visual display

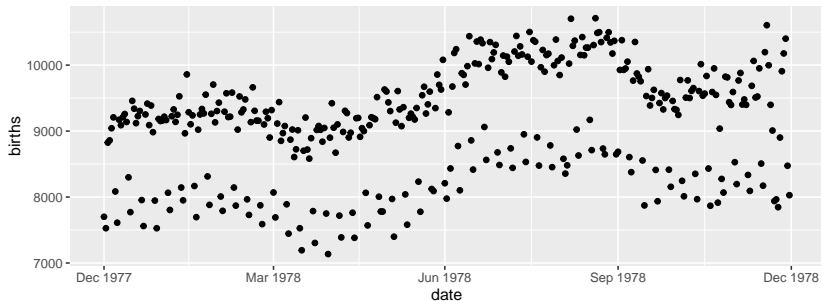
- ▶ particular assignment of colors, shapes, sizes, etc.

guide: helps user convert visual data back into raw data (legends, axes)

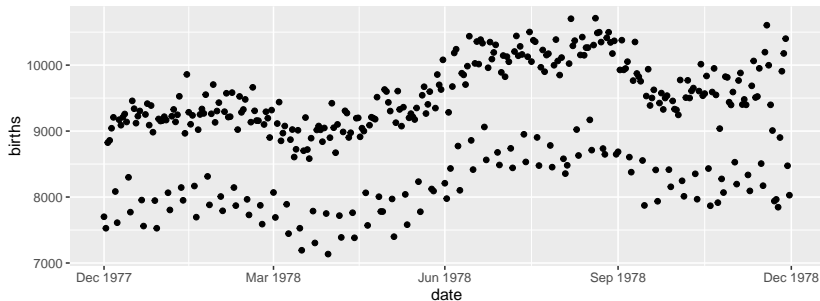
stat: a transformation applied to data before geom gets it

- ▶ example: histograms work on binned data

How do we make this plot?

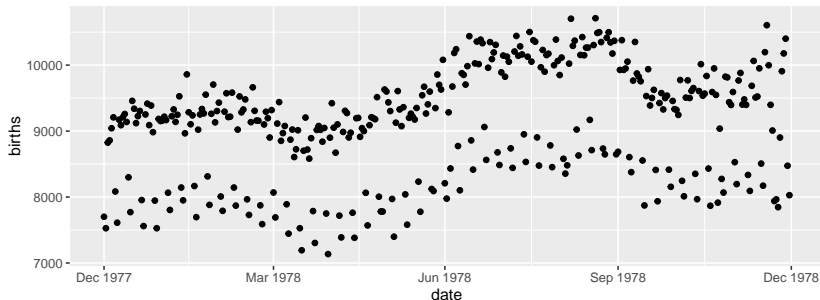


How do we make this plot?



Two Questions:

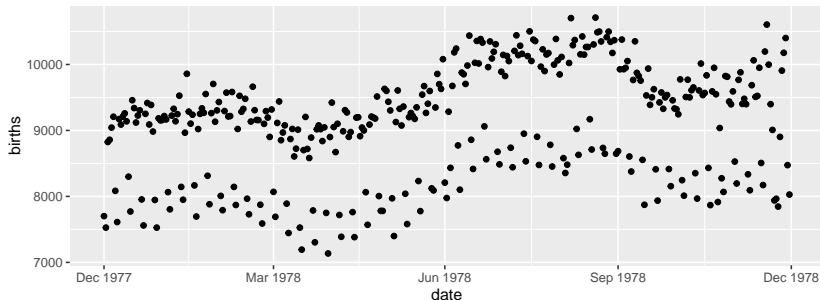
How do we make this plot?



Two Questions:

1. What do we want R to do? (What is the goal?)
2. What does R need to know?

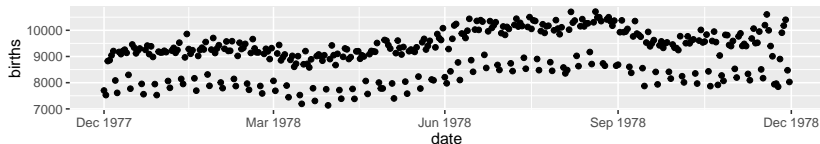
How do we make this plot?



Two Questions:

1. Goal: scatterplot = a plot with points
2. What does R need to know?
 - ▶ data source: Births78
 - ▶ aesthetics:
 - ▶ `date` -> `x`
 - ▶ `births` -> `y`
 - ▶ default color (same for all points)

How do we make this plot?



1. Goal: scatterplot = a plot with points

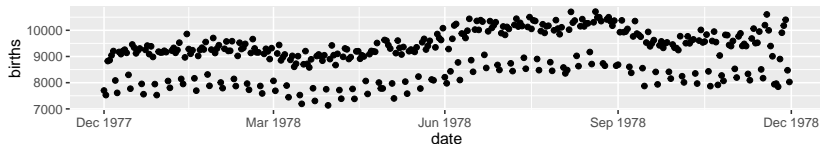
- ▶ `ggplot() + geom_point()`

2. What does R need to know?

- ▶ data source: `data = Births78`

- ▶ aesthetics: `aes(x = date, y = births)`

How do we make this plot?



1. Goal: scatterplot = a plot with points

- ▶ `ggplot() + geom_point()`

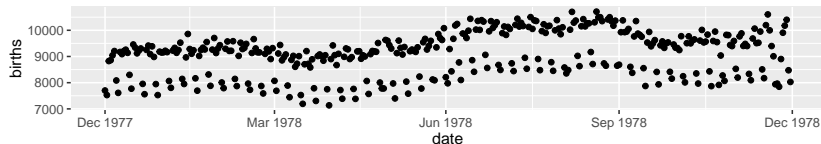
2. What does R need to know?

- ▶ data source: `data = Births78`

- ▶ aesthetics: `aes(x = date, y = births)`

```
ggplot(data = Births78, aes(x = date, y = births)) +  
  geom_point()
```

How do we make this plot?



1. Goal: scatterplot = a plot with points

- ▶ `ggplot() + geom_point()`

2. What does R need to know?

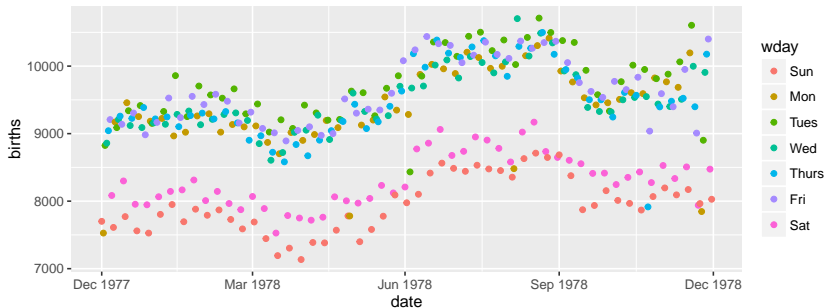
- ▶ data source: `data = Births78`

- ▶ aesthetics: `aes(x = date, y = births)`

```
ggplot(data = Births78, aes(x = date, y = births)) +  
  geom_point()
```

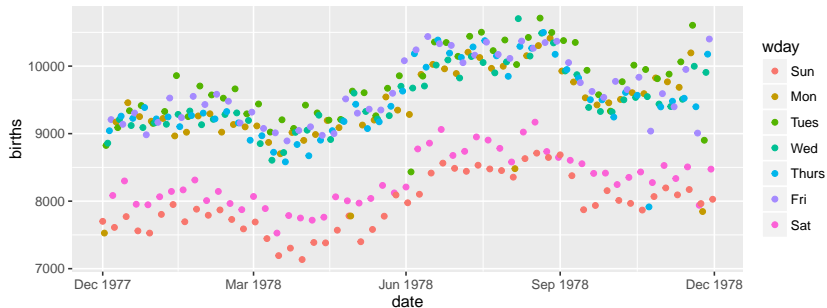
```
ggplot() +  
  geom_point(data = Births78, aes(x = date, y = births))
```

How do we make this plot?



What has changed?

How do we make this plot?



What has changed?

- new aesthetic: mapping color to day of week

Adding day of week to the data set

The `wday()` function in the `lubridate` package computes the day of the week from a date.

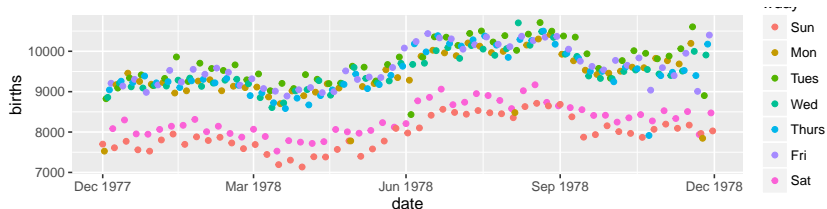
```
Births78 <-  
  Births78 %>%  
  mutate(wday = wday(date, label = TRUE))
```

Adding day of week to the data set

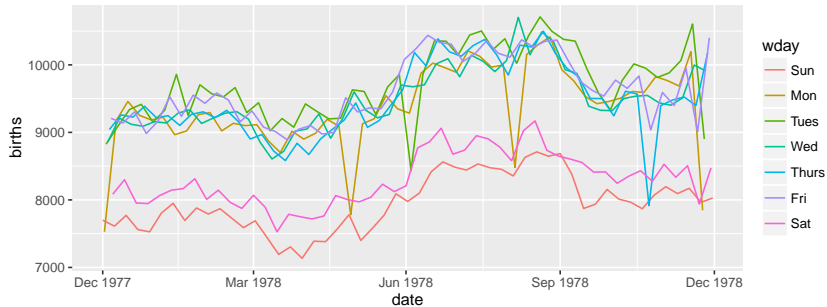
The `wday()` function in the `lubridate` package computes the day of the week from a date.

```
Births78 <-  
  Births78 %>%  
  mutate(wday = wday(date, label = TRUE))
```

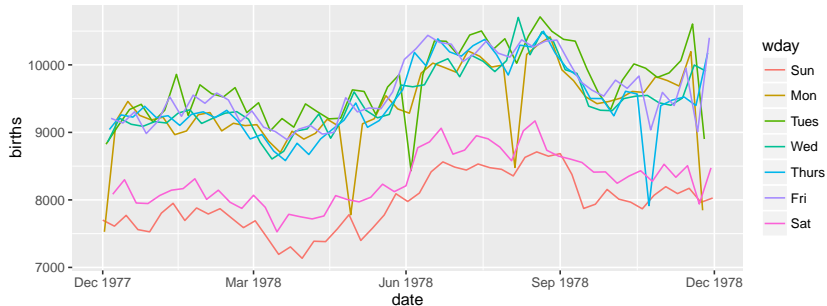
```
ggplot(data = Births78) +  
  geom_point(aes(x = date, y = births, color = wday))
```



How do we make this plot?



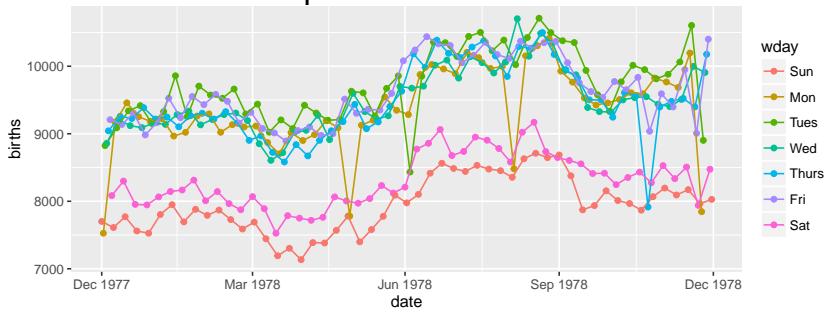
How do we make this plot?



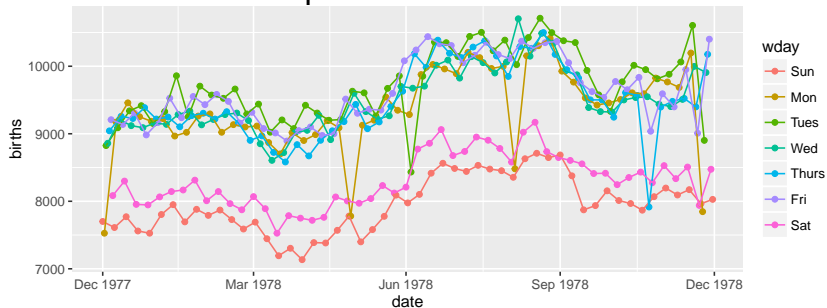
This time we use lines instead of dots

```
ggplot(data = Births78) +  
  geom_line(aes(x = date, y = births, color = wday))
```

How do we make this plot?



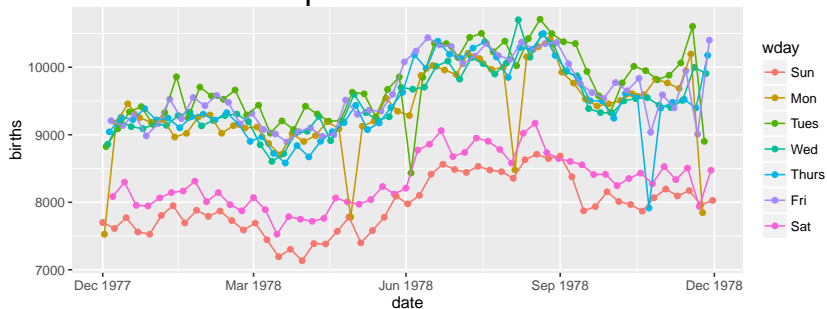
How do we make this plot?



This time we have two **layers**, one with points and one with lines

```
ggplot(data = Births78,  
       aes(x = date, y = births, color = wday)) +  
  geom_point() + geom_line()
```

How do we make this plot?



This time we have two **layers**, one with points and one with lines

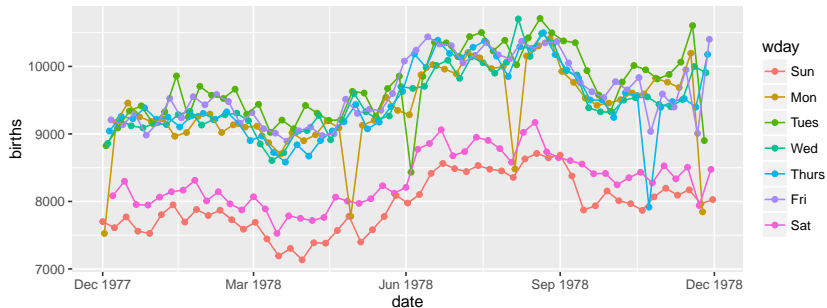
```
ggplot(data = Births78,  
       aes(x = date, y = births, color = wday)) +  
  geom_point() + geom_line()
```

- The layers are placed one on top of the other: the points are *below* and the lines are *above*.
- data and aes specified in `ggplot()` affect all geoms

Alternative Syntax

```
Births78 %>%
```

```
  ggplot(aes(x = date, y = births, color = wday)) +  
  geom_point() +  
  geom_line()
```



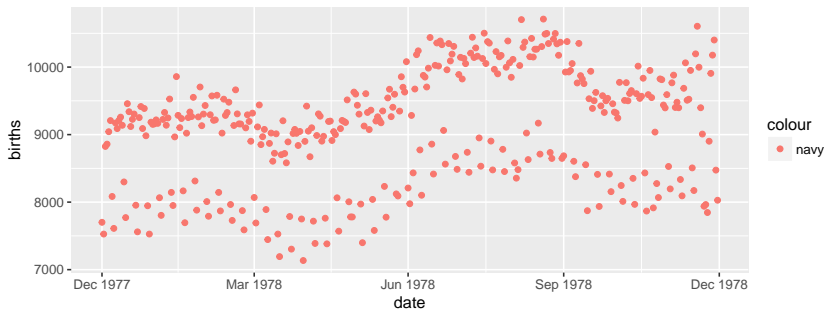
What does this do?

```
Births78 %>%  
  ggplot(aes(x = date, y = births, color = "navy")) +  
  geom_point()
```

What does this do?

```
Births78 %>%
```

```
  ggplot(aes(x = date, y = births, color = "navy")) +  
  geom_point()
```



This is *mapping* the color aesthetic to a new variable with only one value (“navy”).

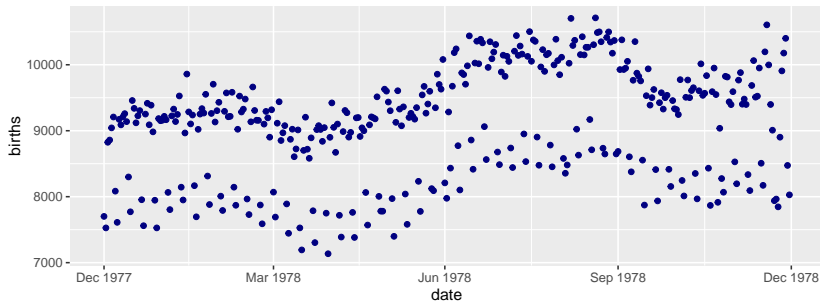
So all the dots get set to the same color, but it's not navy.

Setting vs. Mapping

If we want to *set* the color to be navy for all of the dots, we do it this way:

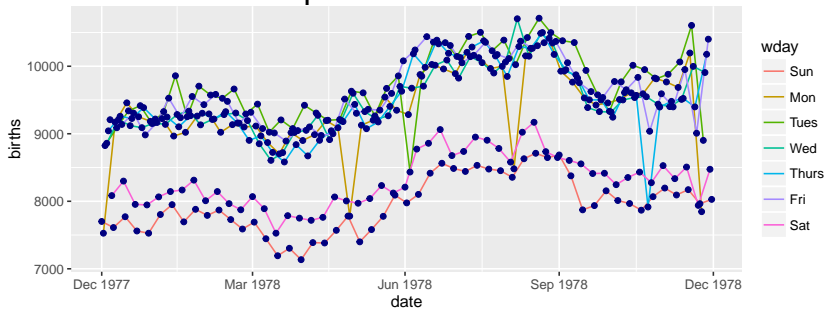
```
Births78 %>%
```

```
  ggplot(aes(x = date, y = births)) +      # map these  
  geom_point(color = "navy")              # set this
```

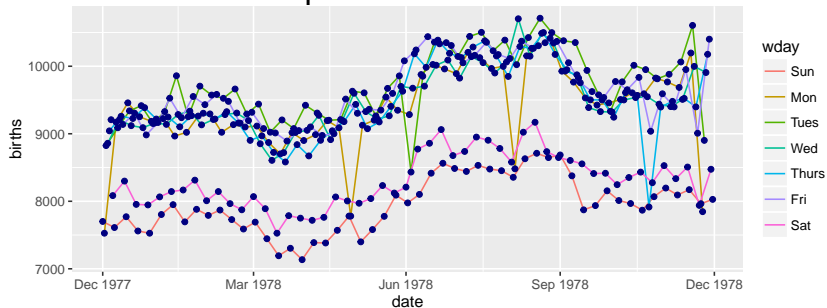


- Note that `color = "navy"` is now outside of the aesthetics list. That's how `ggplot2` distinguishes between mapping and setting.

How do we make this plot?



How do we make this plot?



```
Births78 %>%
```

```
ggplot(aes(x = date, y = births)) +  
  geom_line(aes(color = wday)) +           # map color here  
  geom_point(color = "navy")              # set color here
```

- ▶ `ggplot()` establishes the default data and aesthetics for the geoms, but each geom may change these defaults.
- ▶ good practice: put into `ggplot()` the things that affect all (or most) of the layers; rest in `geom_blah()`

Other geoms

```
apropos("^geom_") %>% head(21)
```

[1]	"geom_abline"	"geom_area"	"geom_bar"
[4]	"geom_bin2d"	"geom_blank"	"geom_boxplot"
[7]	"geom_contour"	"geom_count"	"geom_crossbar"
[10]	"geom_curve"	"geom_density"	"geom_density_2d"
[13]	"geom_density2d"	"geom_dotplot"	"geom_errorbar"
[16]	"geom_errorbarh"	"geom_freqpoly"	"geom_hex"
[19]	"geom_histogram"	"geom_hline"	"geom_jitter"

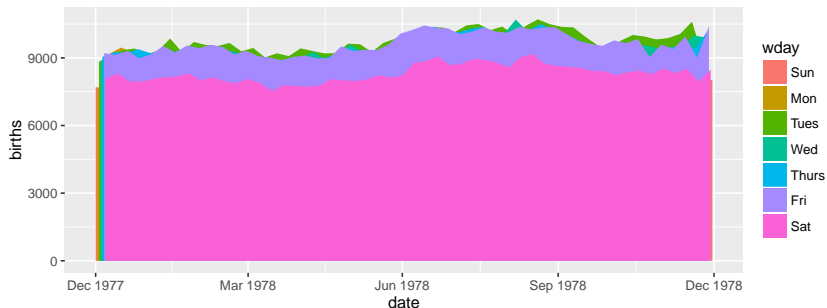
help pages will tell you their aesthetics, default stats, etc.

```
?geom_area # for example
```

Let's try geom_area

```
Births78 %>%
```

```
  ggplot(aes(x = date, y = births, fill = wday)) +  
  geom_area()
```

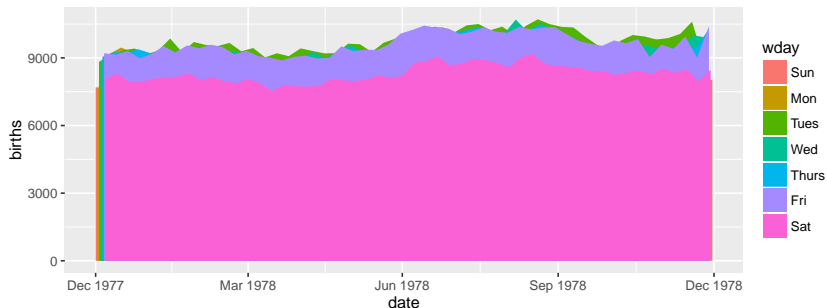


This is not a good plot

Let's try geom_area

```
Births78 %>%
```

```
  ggplot(aes(x = date, y = births, fill = wday)) +  
  geom_area()
```



This is not a good plot

- ▶ overplotting is hiding much of the data
- ▶ extending y-axis to 0 may or may not be desirable.

Side note: what makes a plot good?

Most (all?) graphics are intended to help us make comparisons

- ▶ How does something change over time?
- ▶ Do my treatments matter? How much?
- ▶ Do men and women respond the same way?

Key plot metric: Does my plot make the comparisons I am interested in

- ▶ easily, and
- ▶ accurately?

Time for some different data

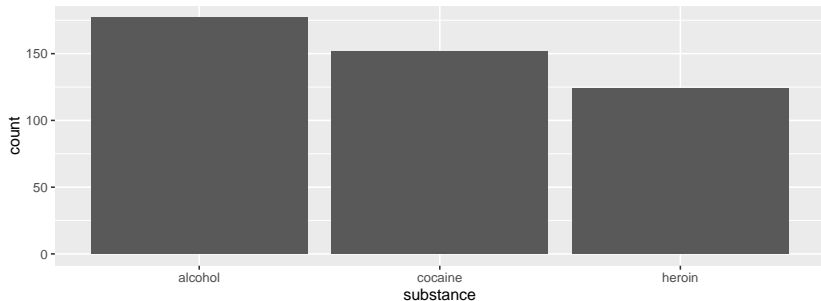
HELPrct: Health Evaluation and Linkage to Primary care
randomized clinical trial

?HELPrct

Subjects admitted for treatment for addiction to one of three
substances.

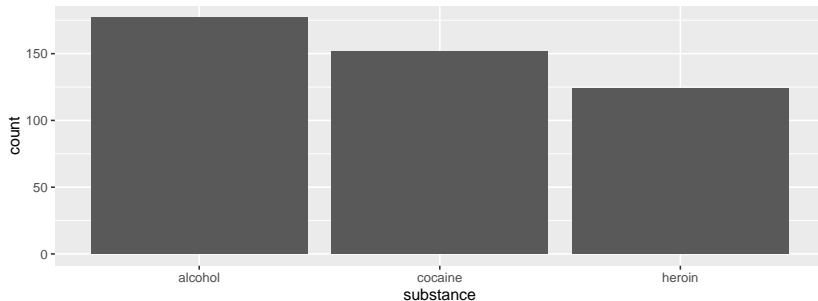
Why are these people in the study?

```
HELPrct %>%  
  ggplot(aes(x = substance)) +  
  geom_bar()
```



Why are these people in the study?

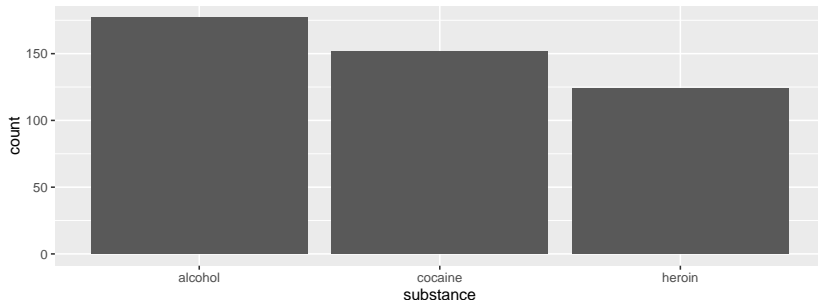
```
HELPrct %>%  
  ggplot(aes(x = substance)) +  
  geom_bar()
```



► Hmm. What's up with y?

Why are these people in the study?

```
HELPrct %>%  
  ggplot(aes(x = substance)) +  
  geom_bar()
```



- ▶ Hmm. What's up with y?
 - ▶ `stat_bin()` is being applied to the data before the `geom_bar()` gets to do its thing. Binning creates the y values.

Data Flow

org data $\xrightarrow{\text{stat}}$ statified $\xrightarrow{\text{aesthetics}}$ aesthetic data $\xrightarrow{\text{scales}}$ scaled data

Simplifications:

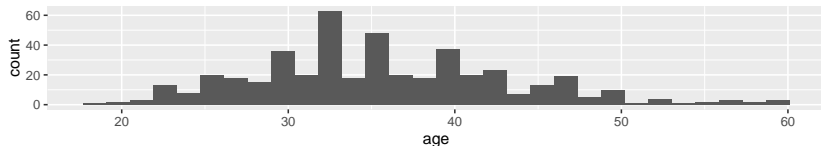
- ▶ Aesthetics get computed twice, once before the stat and again after. Examples: bar charts, histograms
- ▶ We need to look at the aesthetics to figure out which variable to bin
 - ▶ then the stat does the binning
 - ▶ bin counts become part of the aesthetics for geom: `y = ..count..`
- ▶ This process happens *in each layer*
- ▶ `stat_identity()` is the “do nothing” stat.

How old are people in the HELP study?

How old are people in the HELP study?

```
HELPrct %>%  
  ggplot(aes(x = age)) +  
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with  
## `binwidth`.
```

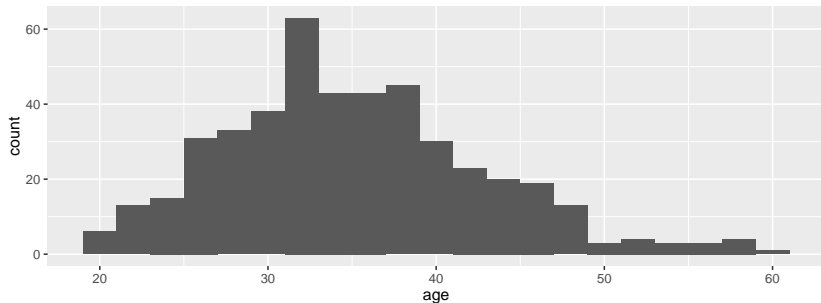


Notice the messages

- ▶ `stat_bin`: Histograms are not mapping the raw data but binned data.
`stat_bin()` performs the data transformation.
- ▶ `binwidth`: a default binwidth has been selected, but we should really choose our own.

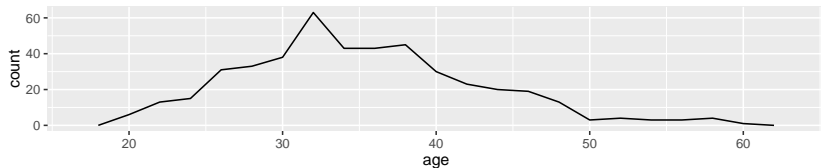
Setting the binwidth manually

```
HELPrct %>%  
  ggplot(aes(x = age)) +  
  geom_histogram(binwidth = 2)
```

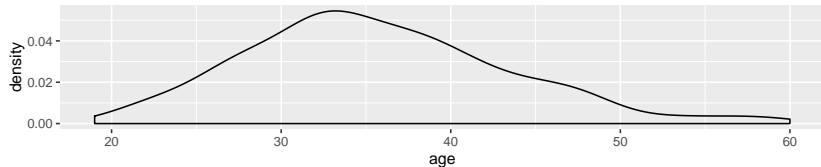


How old are people in the HELP study? – Other geoms

```
HELPrct %>%  
  ggplot(aes(x = age)) +  
  geom_freqpoly(binwidth = 2)
```



```
HELPrct %>%  
  ggplot(aes(x = age)) +  
  geom_density()
```



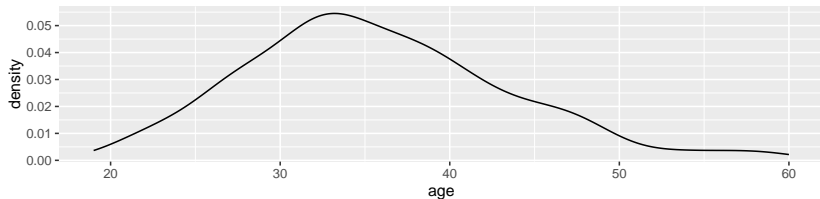
Selecting stat and geom manually

Every geom comes with a default stat

- ▶ for simple cases, the stat is `stat_identity()` which does nothing
- ▶ we can mix and match geoms and stats however we like

```
HELPrct %>%
```

```
  ggplot(aes(x = age)) +  
  geom_line(stat = "density")
```

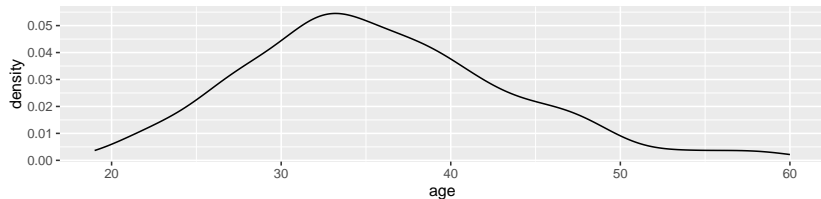


Selecting stat and geom manually

Every stat comes with a default geom, every geom with a default stat

- ▶ we can specify stat instead of geom, if we prefer
- ▶ we can mix and match geoms and stats however we like

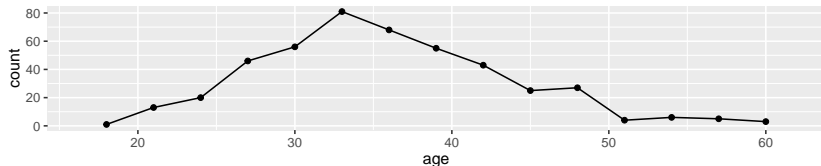
```
HELPrct %>%  
  ggplot(aes(x = age)) +  
  stat_density( geom = "line")
```



More combinations

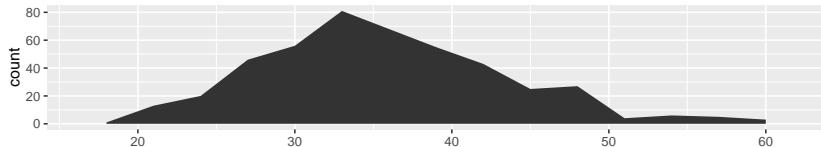
```
HELPrct %>%
```

```
  ggplot(aes(x = age)) +  
  geom_point(stat = "bin", binwidth = 3) +  
  geom_line(stat = "bin", binwidth = 3)
```



```
HELPrct %>%
```

```
  ggplot(aes(x = age)) +  
  geom_area(stat = "bin", binwidth = 3)
```



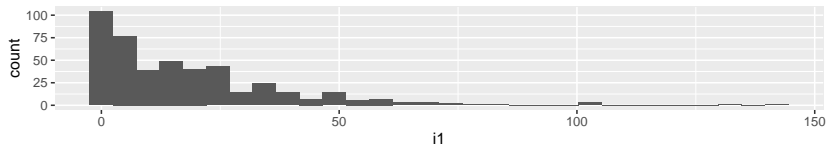
Your turn: How much do they drink? (i1)

Create a plot that shows the distribution of the average daily alcohol consumption in the past 30 days (i1).

How much do they drink? (i1)

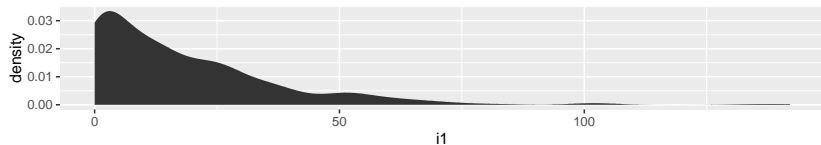
```
HELPrct %>%
```

```
  ggplot(aes(x = i1)) + geom_histogram()
```



```
HELPrct %>%
```

```
  ggplot(aes(x = i1)) + geom_area(stat = "density")
```



Covariates: Adding in more variables

Q. How does alcohol consumption (or age, your choice) differ by sex and substance (alcohol, cocaine, heroin)?

Decisions:

- ▶ How will we display the variables: `i1` (or age), `sex`, `substance`
- ▶ What comparisons are we most interested in?

Give it a try.

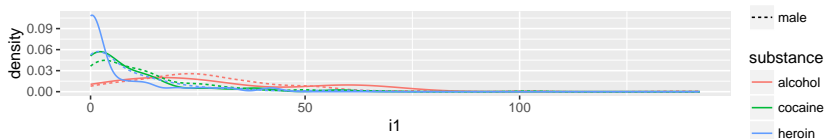
- ▶ Note: I'm cheating a bit. You may want to do some things I haven't shown you yet. (Feel free to ask.)

Covariates: Adding in more variables

Using color and linetype:

```
HELPrct %>%
```

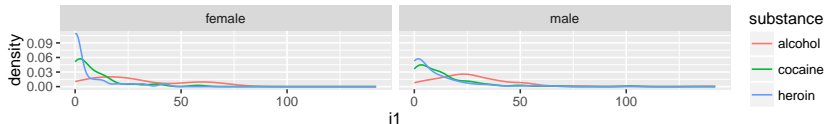
```
  ggplot(aes(x = i1, color = substance, linetype = sex)) +  
  geom_line(stat = "density")
```



Using color and facets

```
HELPrct %>%
```

```
  ggplot(aes(x = i1, color = substance)) +  
  geom_line(stat = "density") + facet_grid( . ~ sex )
```

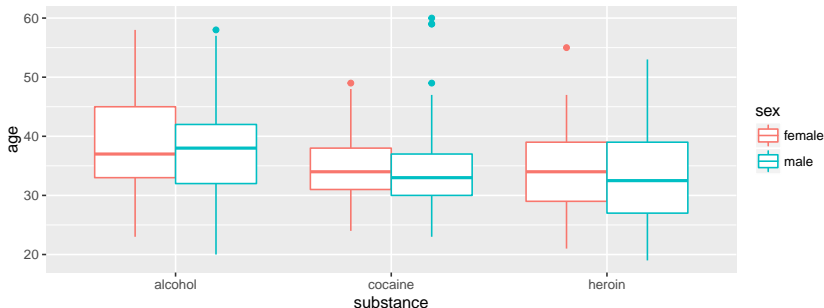


Boxplots

Boxplots use `stat_quantile()` which computes a five-number summary (roughly the five quartiles of the data) and uses them to define a “box” and “whiskers”. The quantitative variable must be `y`, and there must be an additional `x` variable.

```
HELPrct %>%
```

```
  ggplot(aes(x = substance, y = age, color = sex)) +  
  geom_boxplot()
```

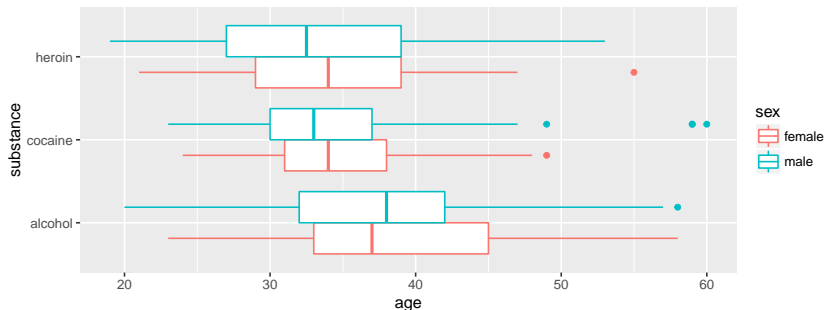


Horizontal boxplots

Horizontal boxplots are obtained by flipping the coordinate system:

```
HELPrct %>%
```

```
  ggplot(aes(x = substance, y = age, color = sex)) +  
  geom_boxplot() +  
  coord_flip()
```



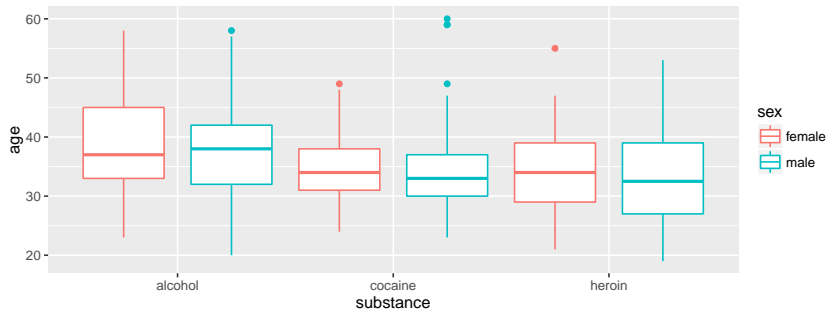
- `coord_flip()` may be used with other plots as well to reverse the roles of x and y on the plot.

Give me some space

We've triggered a new feature: `dodge` (for dodging things left/right). We can control how much if we set the `dodge` manually.

```
HELPrct %>%
```

```
  ggplot(aes(x = substance, y = age, color = sex)) +  
  geom_boxplot(position = position_dodge(width = 1))
```

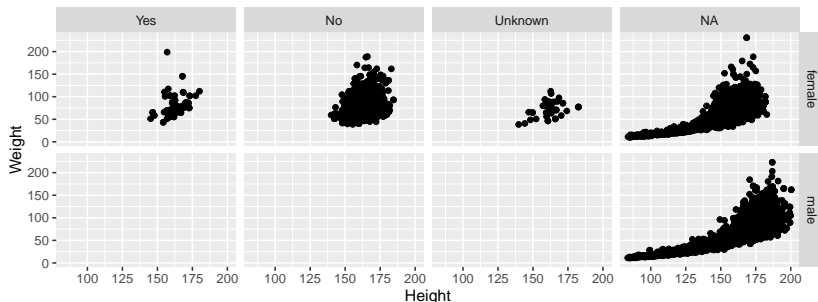


Issues with bigger data

```
require(NHANES)
dim(NHANES)
```

```
## [1] 10000    76
```

```
NHANES %>% ggplot(aes(x = Height, y = Weight)) +  
  geom_point() + facet_grid( Gender ~ PregnantNow )
```



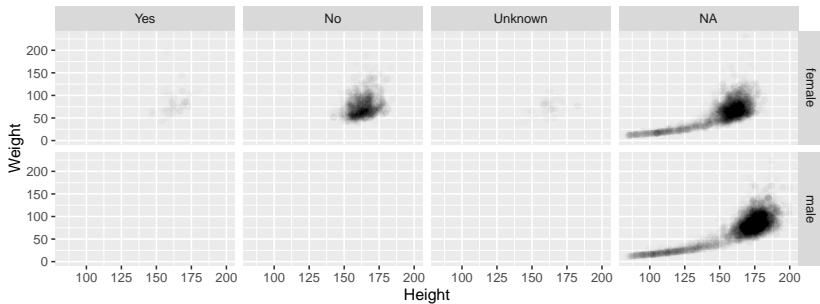
- Although we can see a generally positive association (as we would expect), the overplotting may be hiding information.

Using alpha (opacity)

One way to deal with overplotting is to set the opacity low.

NHANES %>%

```
ggplot(aes(x = Height, y = Weight)) +  
  geom_point(alpha = 0.01) + facet_grid( Gender ~ Pregnant)
```

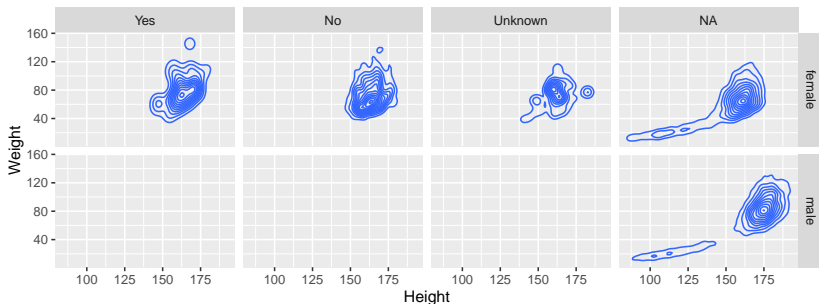


geom_density2d

Alternatively (or simultaneously) we might prefer a different geom altogether.

NHANES %>%

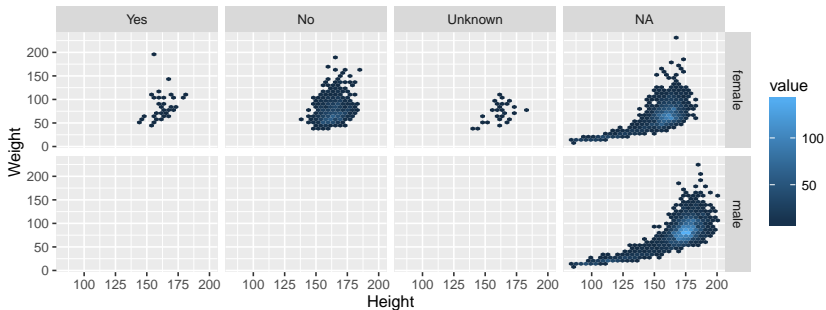
```
ggplot(aes(x = Height, y = Weight)) +  
  geom_density2d() + facet_grid( Gender ~ PregnantNow )
```



geom_hex

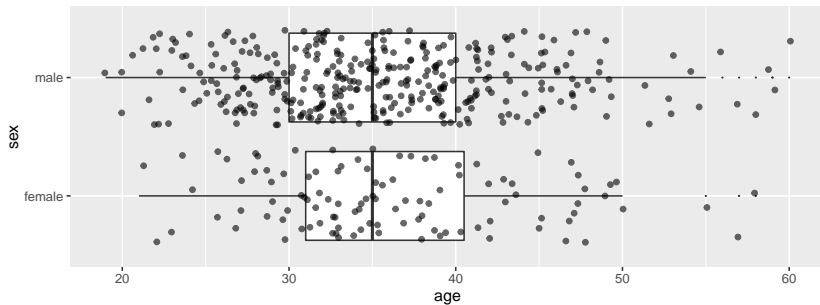
NHANES %>%

```
ggplot(aes(x = Height, y = Weight)) +  
  geom_hex() + facet_grid(Gender ~ PregnantNow)
```



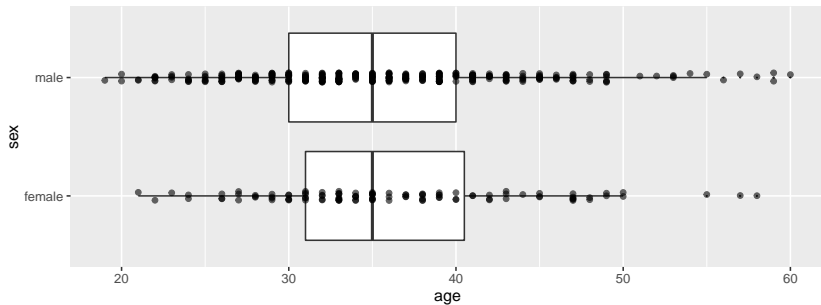
Multiple layers

```
ggplot( data = HELPrct, aes(x = sex, y = age)) +  
  geom_boxplot(outlier.size = 0) +  
  geom_jitter(alpha = .6) +  
  coord_flip()
```



Multiple layers

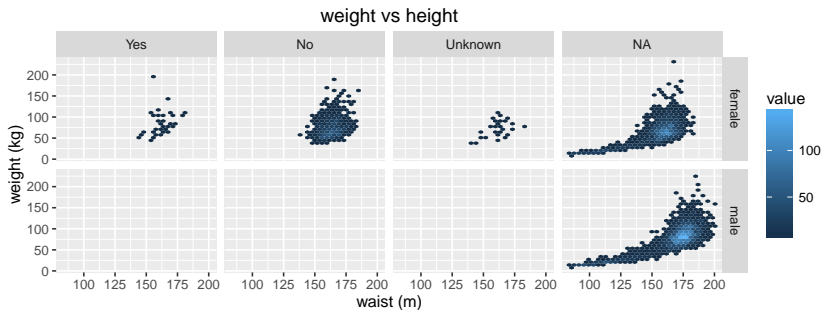
```
ggplot( data = HELPrct, aes(x = sex, y = age)) +  
  geom_boxplot(outlier.size = 0) +  
  geom_point(alpha = .6, position = position_jitter(width =  
  coord_flip())
```



Labeling

NHANES %>%

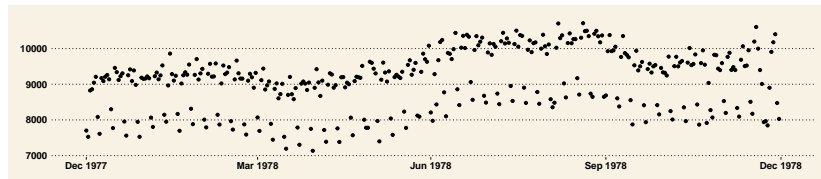
```
ggplot(aes(x = Height, y = Weight)) +  
  geom_hex() + facet_grid( Gender ~ PregnantNow ) +  
  labs(x = "waist (m)", y = "weight (kg)",  
       title = "weight vs height")
```



Things I haven't mentioned (much)

- ▶ scales (fine tuning mapping from data to plot)
- ▶ guides (so reader can map from plot to data)
- ▶ coords (`coord_flip()` is good to know about)
- ▶ themes (for customizing appearance)

```
require(ggthemes)
ggplot(data = Births78, aes(x = date, y = births)) +
  geom_point() + theme_wsj() # wall street journal
```



Things I haven't mentioned (much)

- ▶ scales (fine tuning mapping from data to plot)
- ▶ guides (so reader can map from plot to data)
- ▶ coords (`coord_flip()` is good to know about)
- ▶ themes (for customizing appearance)

```
require(xkcd)
ggplot(data = Births78, aes(x = date, y = births, colour = 
  geom_smooth(se = FALSE) + theme_xkcd()
```

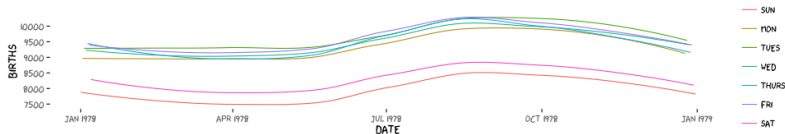
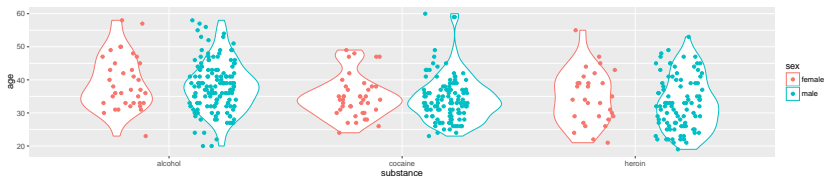


Figure 1: births

Things I haven't mentioned (much)

- ▶ scales (fine tuning mapping from data to plot)
- ▶ guides (so reader can map from plot to data)
- ▶ coords (coord_flip() is good to know about)
- ▶ themes (for customizing appearance)
- ▶ position (position_dodge() can be used for side by side bars)

```
ggplot(data = HELPrct,  
       aes(x = substance, y = age, color = sex)) +  
  geom_violin(position = position_dodge()) +  
  geom_point(aes(color = sex, fill = sex),  
            position = position_jitterdodge())
```



Things I haven't mentioned (much)

- ▶ scales (fine tuning mapping from data to plot)
- ▶ guides (so reader can map from plot to data)
- ▶ themes (for customizing appearance)
- ▶ position (`position_dodge()`, `position_jitterdodge()`, `position_stack()`, etc.)

A little bit of everything

```
ggplot(data = HELPrct,
       aes(x = substance, y = age, color = sex)) +
  geom_boxplot(position = position_dodge(width = 1)) +
  geom_point(aes(fill = sex), alpha = .5,
            position = position_jitterdodge(dodge.width = 1)) +
  facet_wrap(~homeless)
```



A short cut

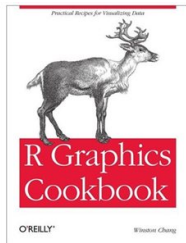
`mpplot(dataframe)` provides an interactive plotting tool

`mpplot(HELPrct)`

- ▶ quickly make several plots from a data frame
- ▶ can show the expression so you can learn how to do it or copy and paste into another document
- ▶ `ggplot2` or `lattice`

Want to learn more?

- ▶ docs.ggplot2.org/
- ▶ Winston Chang's: *R Graphics Cookbook*



What's around the corner?

`ggvis`

- ▶ dynamic graphics (brushing, sliders, tooltips, etc.)
- ▶ uses Vega (D3) to animate plots in a browser
- ▶ similar structure to `ggplot2` but different syntax and names

Dynamic documents

- ▶ combination of `RMarkdown`, `ggvis`, and `shiny`