Outline for today

- Brief reviews of homeworks 7 and 8
- Quick review of the final study guide
- One topic briefly noted: *isotonic regression*
- The *grammar of graphics* and `ggplot2`

Homework 7

```
> d <- read.csv("NYC.csv")
> contrasts(d$store) <- contr.sum
> contrasts(d$word) <- contr.sum
> contrasts(d$emphasis) <- contr.sum
> r <- glm(r ~ store + word + emphasis, data = d,
+          family = binomial)
```

```
> summary(r)
...
Coefficients:

            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.93588    0.10283  -9.102  < 2e-16 ***
store1      -1.34852    0.16928  -7.966 1.64e-15 ***
store2       0.45423    0.12180   3.729 0.000192 ***
word1        0.50065    0.08763   5.713 1.11e-08 ***
emphasis1    0.16457    0.08948   1.839 0.065877 .
...
```

I prefer the functions of `multcomp` over `TukeyHSD`.

```
> library(multcomp)
> pairs <- glht(r, linfct = mcp(store = "Tukey"))
> summary(pairs)
...
                       Estimate Std. Error z value Pr(>|z|)
Macy's - Klein's == 0    1.8028     0.2617   6.890   <1e-04 ***
Saks - Klein's == 0      2.2428     0.2820   7.954   <1e-04 ***
Saks - Macy's == 0       0.4400     0.1950   2.256   0.0604 .
...
```

So we find that w.r.t. to *r*-usage, S. Klein's < [Macy's = Saks] at α = .05.

Homework 8

```
> library(lme4)
> library(lmtest)
> d <- read.table("elp.tsv", header = TRUE)
> d$RT <- log(d$RT)
> d$subjID <- as.factor(d$subjID)
> d$trial <- with(d, trial - mean(trial))
> d$length <- scale(with(d, length * length))
> d$OLD <- scale(d$OLD)
```

```
> d$sbtlx.freq <- log(d$sbtlx.freq + 1)
> d$sbtlx.basefreq <- log(d$sbtlx.basefreq + 1)
> d$sbtlx.freq <- with(d, sbtlx.freq - mean(sbtlx.freq))
> d$sbtlx.basefreq <- with(d, sbtlx.basefreq -
+                            mean(sbtlx.basefreq))
> if (with(d, cor(sbtlx.freq, sbtlx.basefreq) > .5)) {
+     d$sbtlx.basefreq <- residuals(lm(
+         sbtlx.basefreq ~ sbtlx.freq, data = d))
+     stopifnot(all.equal(with(d,
+         cor(sbtlx.freq, sbtlx.basefreq)), 0))
+ }
```

```
> r <- lmer(RT ~ trial + length + OLD + sbtlx.freq +
+       sbtlx.basefreq + (1 | subjID) + (1 | word), data = d)
> summary(r)
...
Fixed effects:
                 Estimate Std. Error t value

(Intercept)     6.389e+02  9.366e+00   68.21
trial          -1.543e-02  5.695e-04  -27.10
length          5.051e+00  6.817e-01    7.41
OLD             4.446e+01  1.714e+00   25.95
sbtlx.freq     -2.860e+01  3.491e-01  -81.91
sbtlx.basefreq -1.343e+01  5.592e-01  -24.02
...
```

	Coef.	S.E.	χ^2	$p(\chi^2)$
(Intercept)	6.558	0.00		
Trial number	0.000	0.00	1081.80	$< .001$
Squared length	0.022	0.00	148.13	$< .001$
OLD	0.036	0.00	402.48	$< .001$
Word frequency	−0.037	0.00	6468.10	$< .001$
Base frequency	−0.017	0.00	612.32	$< .001$

Further reading

The *best* resource on mixed effects modeling (and regression in general) thus far is:

Gelman, A. and Hill, J. 2007. *Data analysis using regression and multilevel/hierarchical models*. Cambridge: Cambridge University Press.

The book is quite large but I've never had a serious need for the most advanced material (e.g., in chapters 16-24), and some of that content is now obsolete.

The authors teach this material regularly in the statistics department at Columbia, part of our graduate consortium...

Final study guide

Briefly noted: isotonic regression

Monotonic and isotonic functions

A function *f* over (ordinal or interval) variables is said to be (*monotonically*) *increasing* if for all pairs x_1 and $x_2 \in X$, $x_1 \leq x_2$ implies $f(x_1) \leq f(x_2)$.

Likewise it is said to be (*monotonically*) *decreasing* when for all pairs $x_1 \leq x_2$ implies $f(x_1) \geq f(x_2)$.

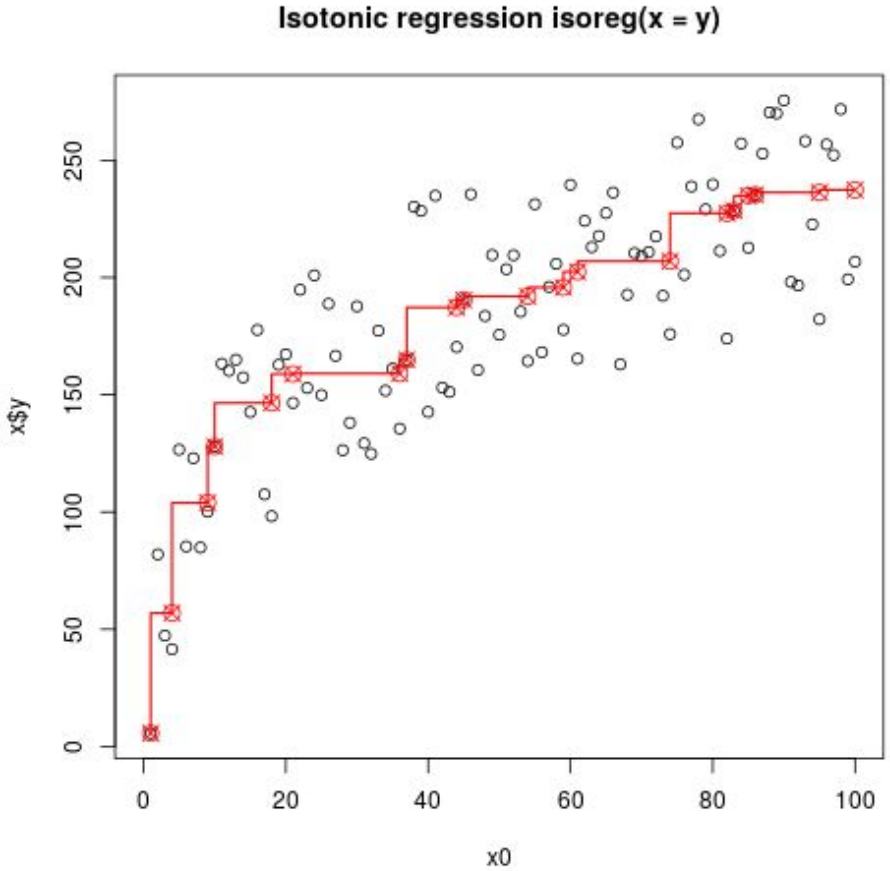
Isotonic functions are the union of monotonically increasing or decreasing functions.

Linear implies isotonic, but not the other way around.

Isotonic regression

Isotonic regression fits a isotonic *step function* to a sequence of data, minimizing the sum of squared errors. Unlike linear regression, it admits at most one (numeric) independent variable. This is not normally used for inference, but rather for data visualization.

```
> n <- 50
> y <- sample(-50:50, n) + 50 * log1p(1:n)
> plot(isoreg(y))
```


Isotonic regression isoreg(x = y)
x$y
x0
0
50
100
150
200
250
0
20
40
60
80
100

Uses of isotonic regression

The fitting procedure minimizes the sum of squared errors, but instead of a line, produces a series of "knots" (steps).

We can extract the fitted step function using `fitted`; this is useful for automatically determining whether the relationship between variables is increasing or decreasing.

The `residuals` can also be used to control for an isotonic relationship.

The grammar of graphics and `ggplot2`

Bibliographic note

The following is closely based on the tutorial in chapter 2 of Wickham 2009, which I recommend reading (there's a link to a digital version of the book on the course website).

R graphics library

R ships with so-called "base" graphics functions such as `plot`, `hist`, etc. Many R objects (such as `lm` results) can also be `plot`ted.

There have been many attempts to add richer, easier-to-use graphics libraries to R, most notably `lattice` and `ggplot2`.

Introducing `ggplot2`

`ggplot2` is a third-party R library that attempts to implement Wilkinson's (2005) *grammar of graphics*. In what sense is it like a grammar?

- It subdivides the work of specifying graphs to separate "modules". These modules freely combine, making it easy to learn new pieces of the stack.
- Graphs are built up iteratively, piece by piece. (This is supposed to mirror the process by which we go from visualization to description to inference.)

"Without the grammar, there is no underlying theory, so most graphics packages are just a big collection of special cases." (Wickham 2009)

```
> install.packages("ggplot2", dependencies = TRUE)
```

The X^0s of `ggplot2`

- **Data**, usually expressed as a `data.frame` (though vectors also can be used)
- **Aesthetics**, mappings of variables onto attributes (axis, color, shape, size)
- **Layers**, including
  - **Geoms**, the geometric objects that make up a plot
  - **Stats**, visualizable summaries of the data
- **Scales**, including axes, coordinates, gridlines, and transformations
- **Facets**, an optional division of the data into subsets and subgraphs
- **Themes**, font face and size, background, default colors

All `ggplot2` graphs require a data frame, an aesthetic specification, and at least one layer; scales, facets, and themes have default specifications.

What `ggplot2` is not

- Does not tell you what or how to plot (I recommend sketching a bit).
- Does not do 3-d graphs (but those are useless anyways).
- Does not make interactive graphs, but can be used to make graph "movies".

Example data set

In what follows, I use examples based on the `mpg` data. This data set, included with R, includes fuel economy numbers for popular car makes 1999-2008.

There are two "DVs",

- `cty` (average "city" mileage, in miles per gallon), and
- hwy (average "highway" mileage, in miles per gallon)

and several "IVs", including:

- `displ` (displacement, i.e., the size of the engine, in liters),
- `cyl` (number of cylinders), and
- `class` (`suv`, `compact`, etc.).

Specifying data and aesthetics

The `ggplot` command wraps the data and defines the default `aes`thetic mapping.

```
> g <- ggplot(mpg, aes(x = displ, y = hwy))
```

We can also safely omit the `x =` and `y =` parts, thus

```
> g <- ggplot(mpg, aes(displ, hwy))
```

This indicates that the *X*-axis variable is displacement and the *Y*-axis variable is highway efficiency. Yet it does no visible work, and does not produce a plot.

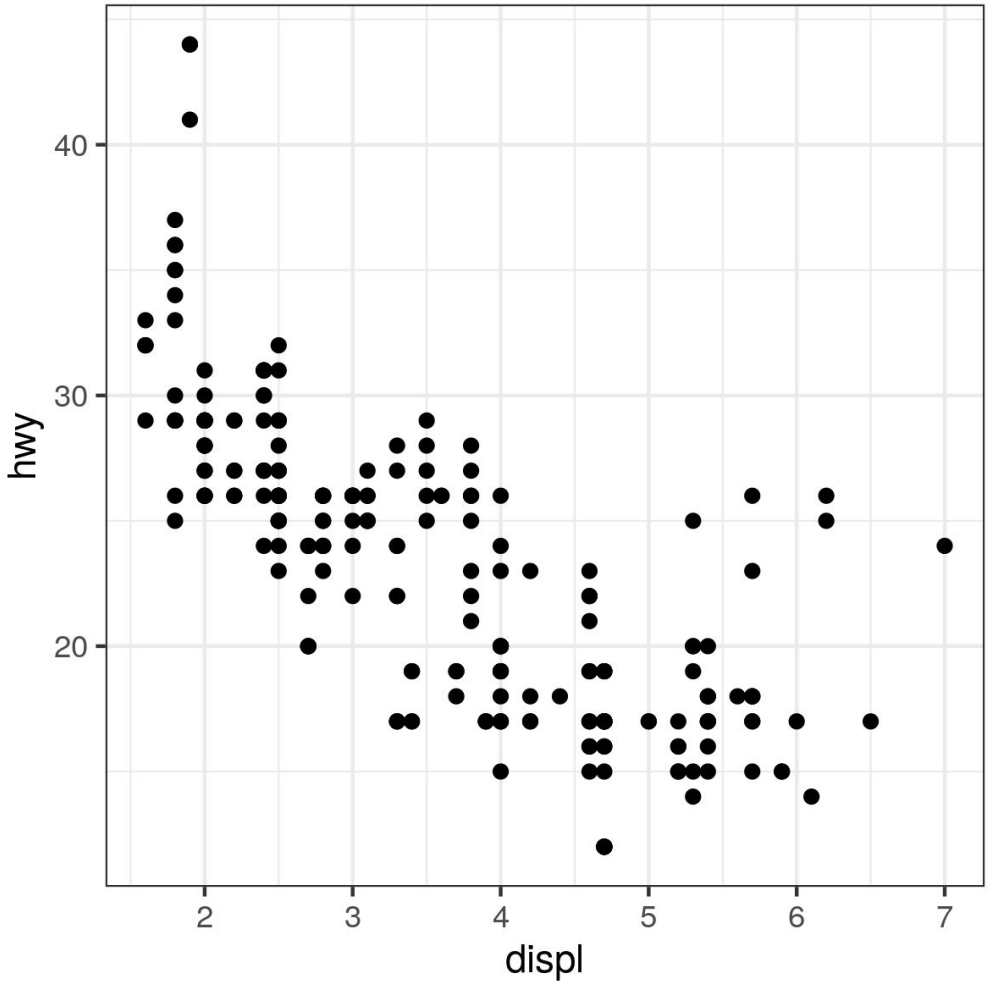
(We usually put the DV on the *Y*-axis, though we'll see an exception below.)

Adding a geom

A `ggplot2` graph minimally consists of data, an aesthetic mapping, and at least one layer. Here, we'll generate a scatterplot by adding (literally) a `geom_point` layer.

```
> p <- g + geom_point()
```

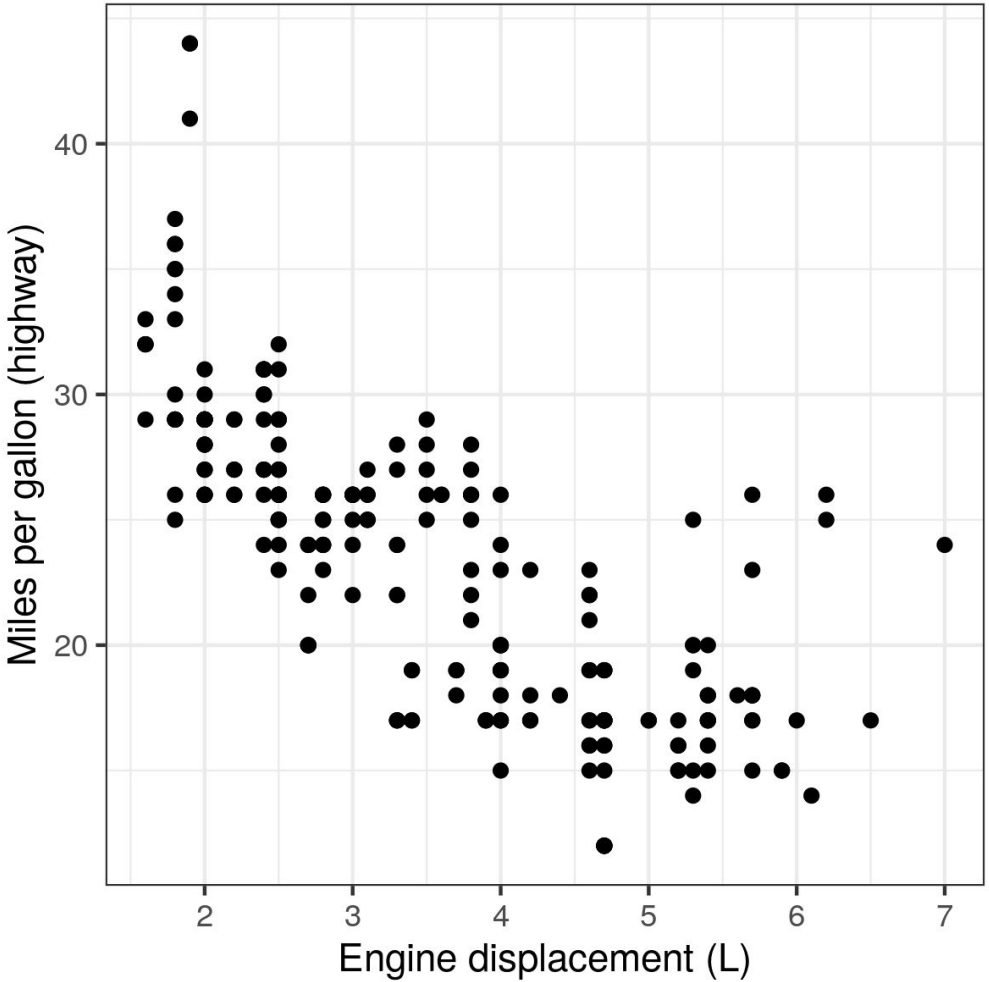
This, at last, produces a scatter plot.


hwy
40
30
20
2
3
4
5
6
7
displ

Labeling the axes

By default `ggplot2` uses the variable names to label the axes. That's not publication-worthy! We'll relabel them ourselves.

```
p <- p + xlab("Engine displacement (L)") +
+        ylab("Miles per gallon (highway)")
```


Miles per gallon (highway)
40
30
20
2
3
4
5
6
7
Engine displacement (L)

Notes thus far

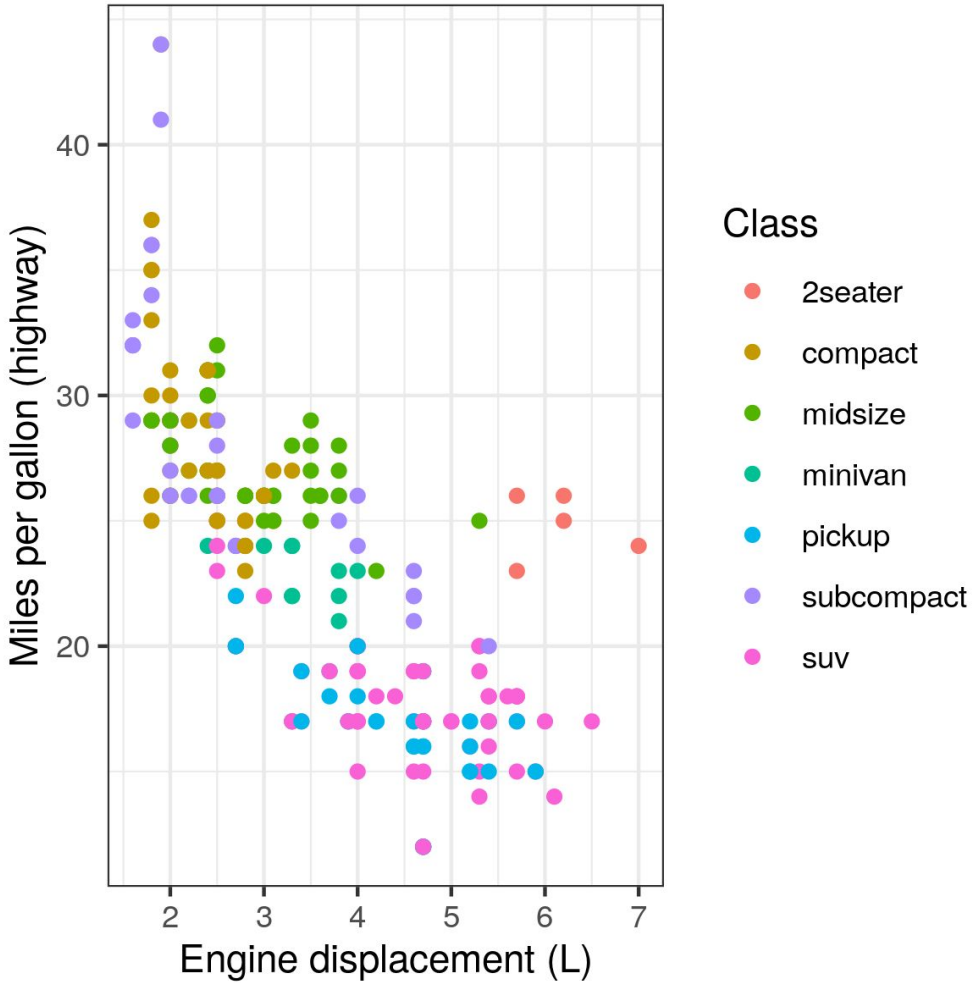
- We can assign the intermediate pieces of a graph (like the data/aesthetic mapping) to variables if we intend to reuse them in other plots in the same session or program.
- In an interactive session, a graph will not appear on-screen until you `print` it.
- To save a graph to disk, use the `ggsave` function:

```
> ggsave("scatter.png", p, dpi = 300, width = 4, height = 4)
```

Adding more aesthetics (1/)

For categorical IVs, we can use (discrete) color or shape.

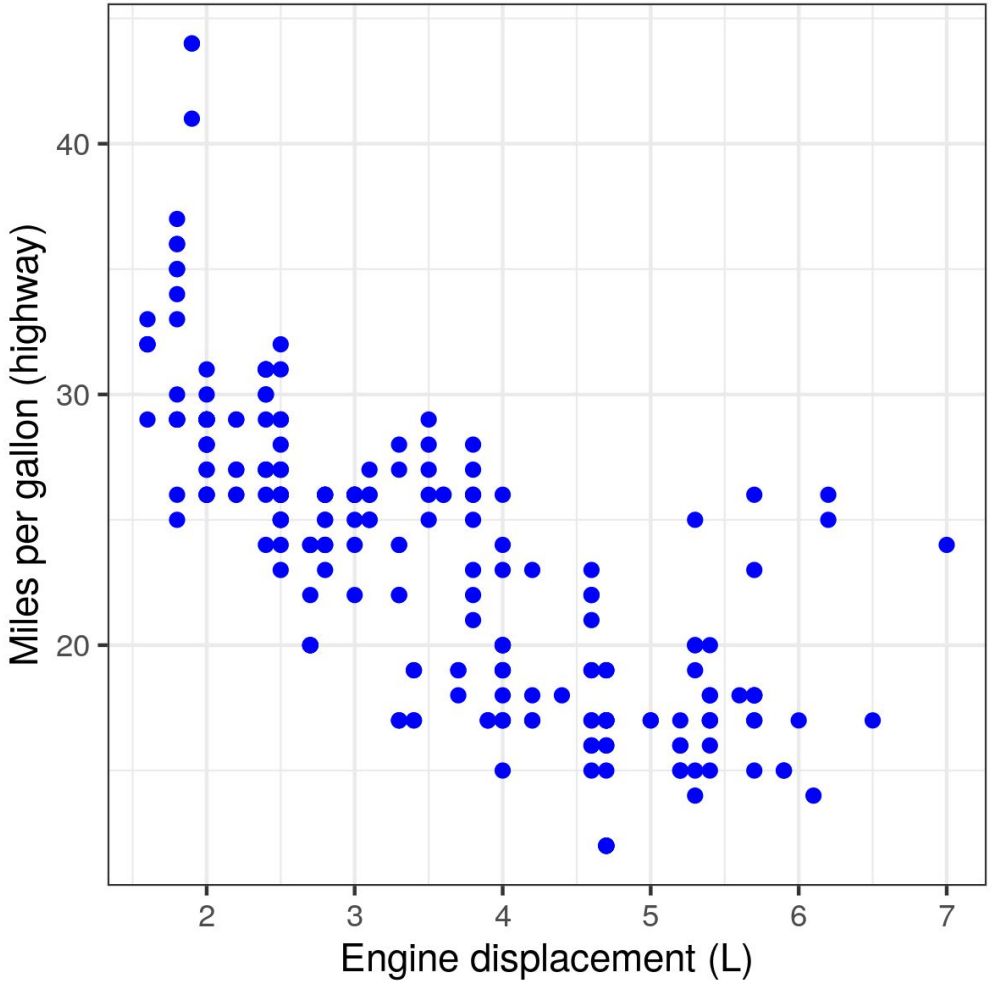
```
> q <- ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
+       geom_point() +
+       xlab("Engine displacement (L)") +
+       ylab("Miles per gallon (highway)") +
+       scale_color_discrete("Class")
```


Miles per gallon (highway)
40
30
20
2
3
4
5
6
7
Engine displacement (L)
Class
2seater
compact
midsize
minivan
pickup
subcompact
suv

Fixed color

If you just want a single fixed color, use the following:

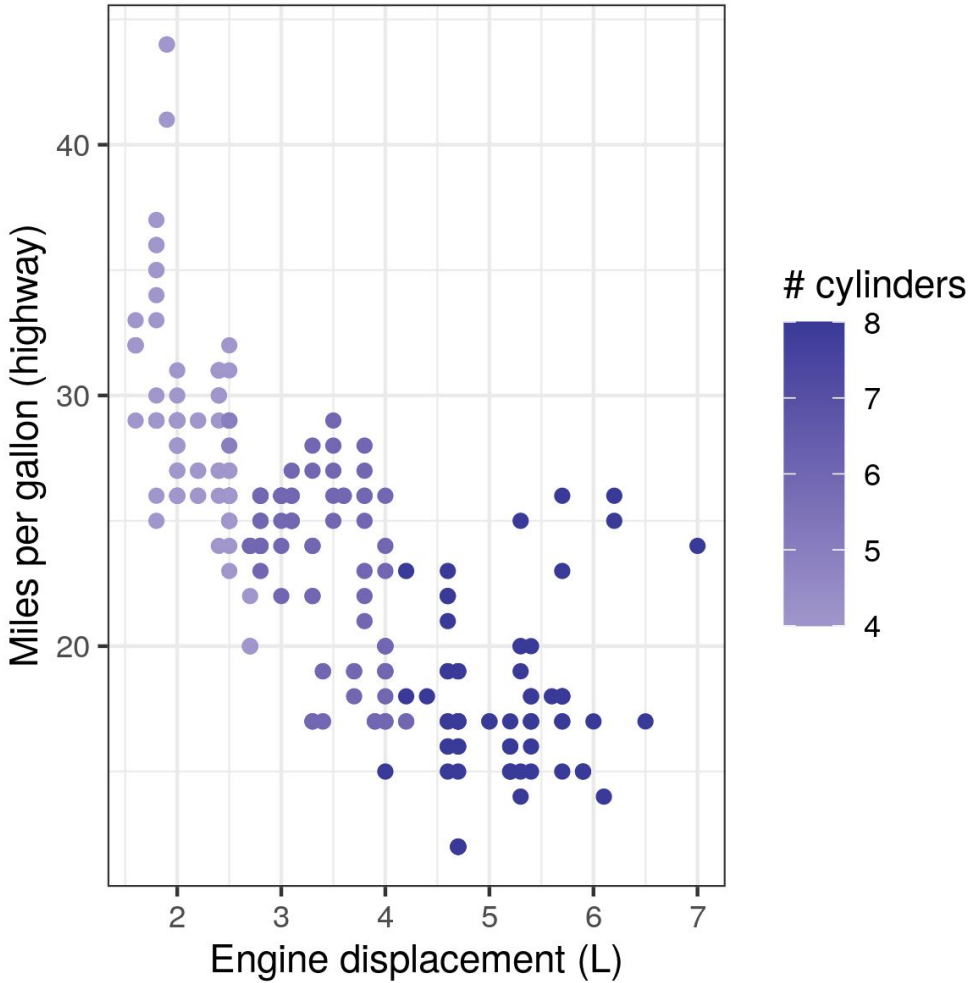
```
> q <- ggplot(mpg, aes(x = displ, y = hwy)) +
+       geom_point(color = "blue") +
+       xlab("Engine displacement (L)") +
+       ylab("Miles per gallon (highway)") +
+       scale_color_discrete("Class")
```


Miles per gallon (highway)
40
30
20
2
3
4
5
6
7
Engine displacement (L)

Adding more aesthetics (2/)

For continuous IVs, we can use color or size.

```
> q <- ggplot(mpg, aes(x = displ, y = hwy, color = cyl)) +
+       geom_point() +
+       xlab("Engine displacement (L)") +
+       ylab("Miles per gallon (highway)") +
+       scale_color_gradient2("# cylinders")
```


Miles per gallon (highway)
40
30
20
2
3
4
5
6
7
Engine displacement (L)
cylinders
8
7
6
5
4

Color and colorblindness

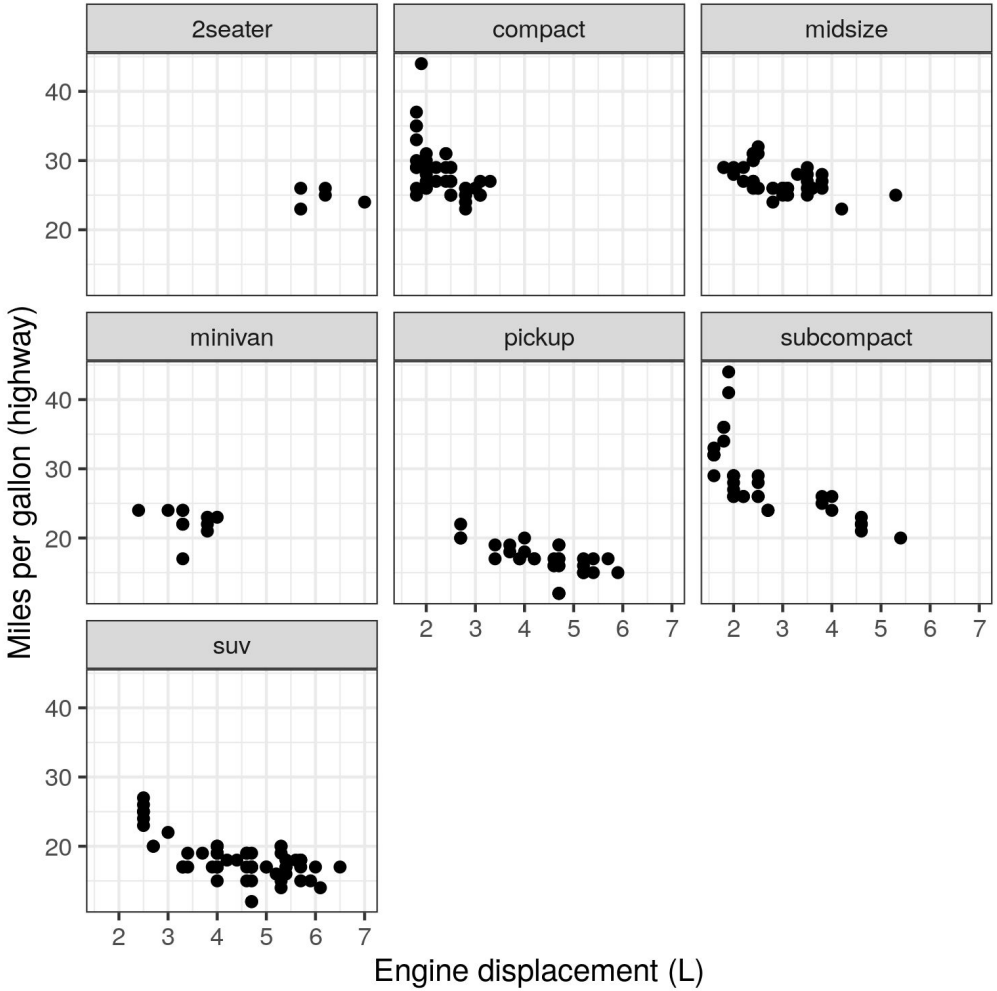
- Colorblindness is far more common than you think; about 4.5% of the population is red-green colorblind, so be careful.
- The ColorBrewer website is a good place to experiment with various types of (colorblind-safe) color palettes; R, and `ggplot2` in particular, support most of these schemes via the `RColorBrewer` third-party library.

Faceting

It's very easy to put too much information on a single graph. Instead we may prefer to partition the data into subgraphs with the same aesthetics and layers.

- `facet_wrap` specifies a "1-d" partition of subgraphs along some categorical IV. This ribbon of graphs appears "wraps around" to form an square.
- `facet_grid` specifies a "2-d" partition of subgraphs along two categorical IVs.

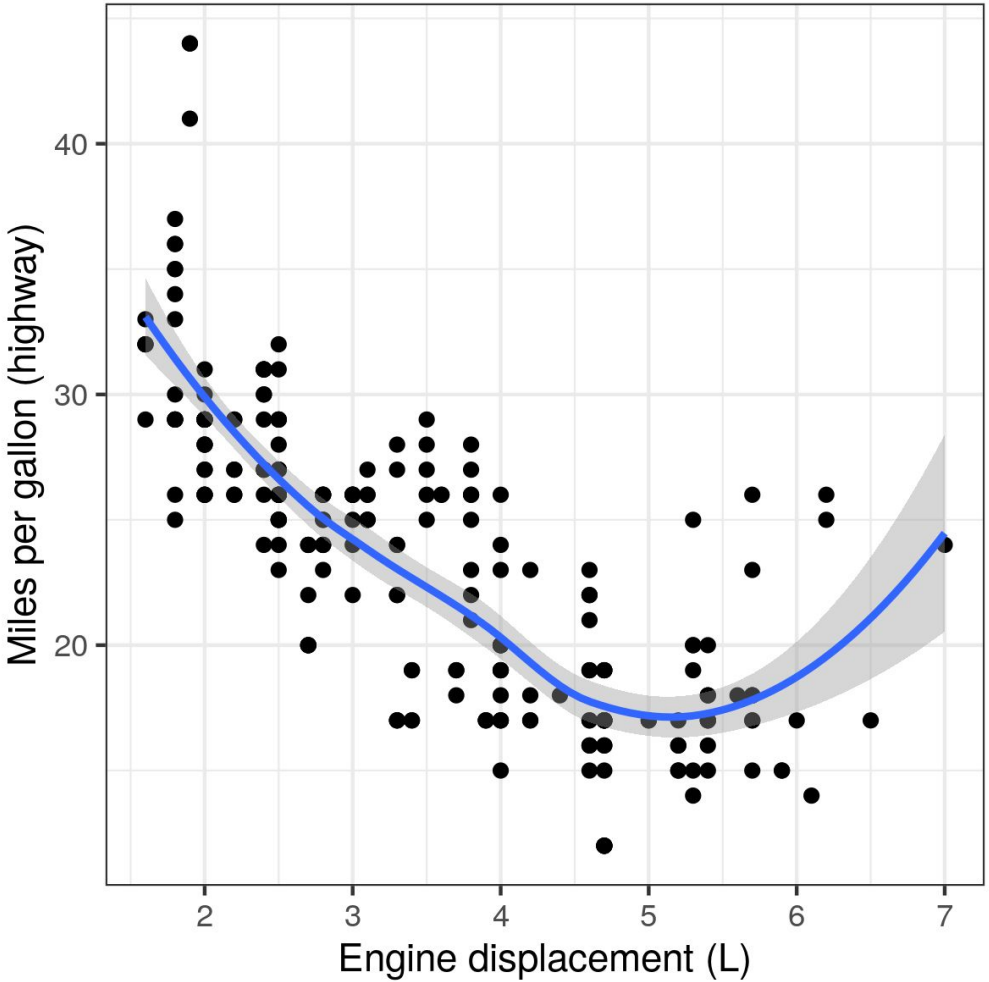
```
> q <- ggplot(mpg, aes(x = displ, y = hwy)) +
+       geom_point() +
+       facet_wrap(~ class) +
+       xlab("Engine displacement (L)") +
+       ylab("Miles per gallon (highway)")
```


2seater
compact
midsize
minivan
pickup
subcompact
suv
Miles per gallon (highway)
Engine displacement (L)
20
30
40
2
3
4
5
6
7

Smoothers

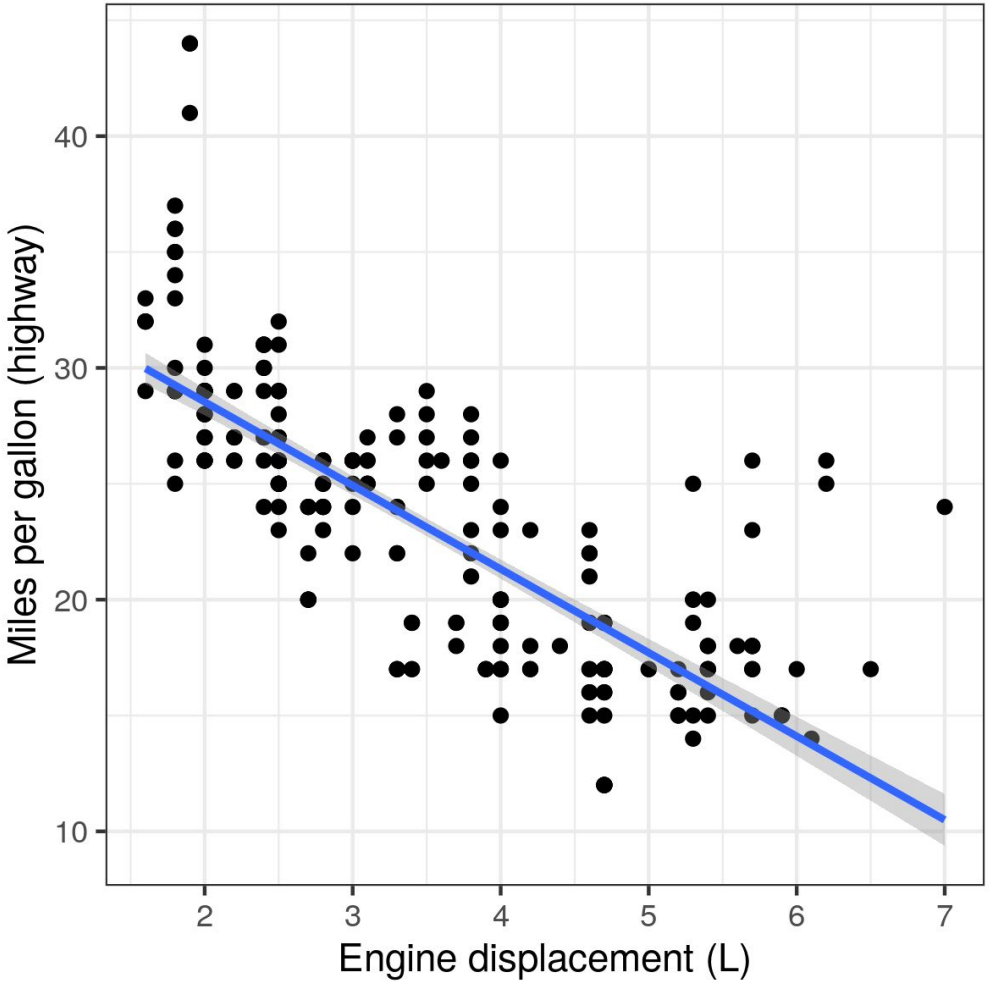
We can also add a sort of line to the scatter plot using `geom_smooth`.

```
> q <- p + geom_smooth() +
+       xlab("Engine displacement (L)") +
+       ylab("Miles per gallon (highway)")
```


Miles per gallon (highway)
40
30
20
2
3
4
5
6
7
Engine displacement (L)

Smoother notes

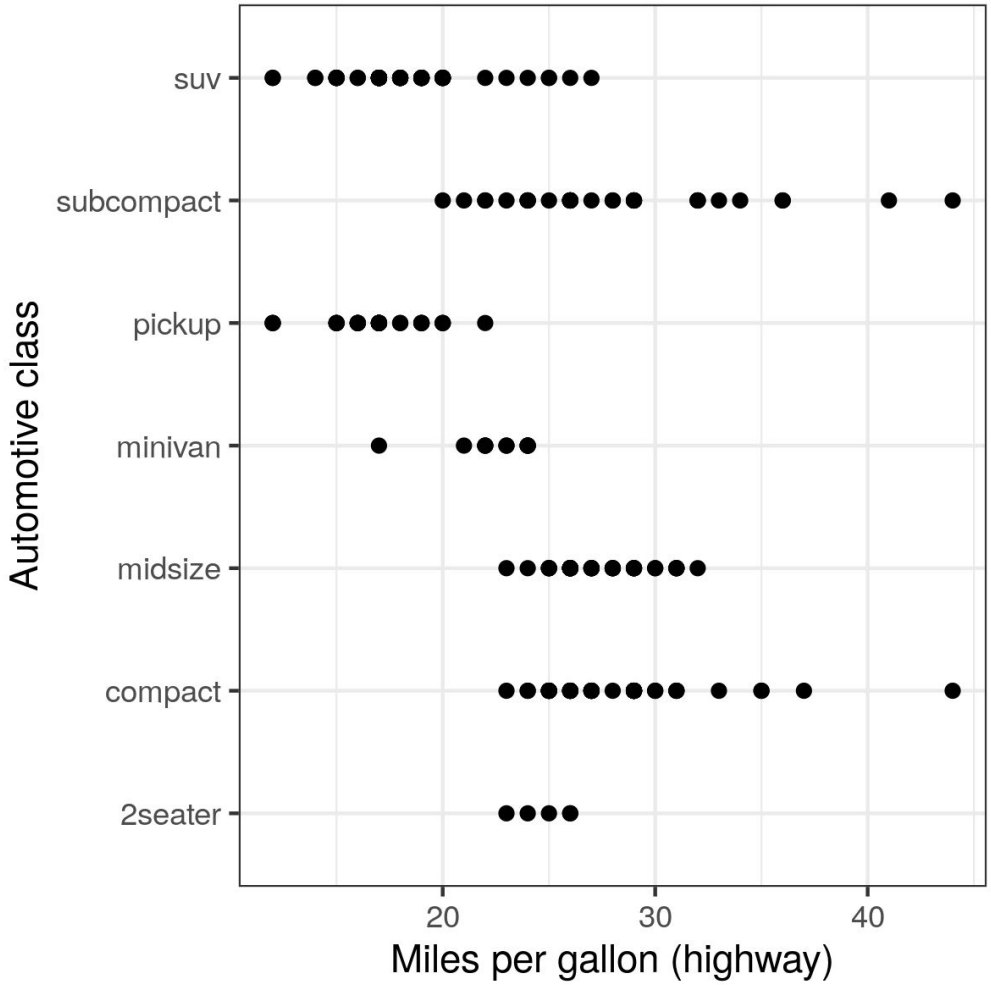
- By default, `geom_smooth` uses a non-linear smoother called LOESS (short for "locally estimated scatterplot smoothing"). With LOESS, it is important to set the `span` hyperparameter to control for "wigglyiness"; here's how I do that.
- By default, `geom_smooth` indicates standard errors by adding a sort of "halo" around the smoothing curve or line; to disable this, use `geom_smooth(se = FALSE)`.
- If you actually desire a regression line instead of a curve, use `geom_smooth(method = "rlm")`; several other methods are available.


Miles per gallon (highway)
40
30
20
10
2
3
4
5
6
7
Engine displacement (L)

Categorical/continuous plots

When working with a continuous DV and a categorical (partic. many-leveled) IV, it can be desirable to make a scatter plot in which the categorical attribute is displayed on the *y*-axis, where the labels are easier to read. This is known as a *dotplot*.

```
> r <- ggplot(mpg, aes(x = hwy, y = class)) +
+       geom_point() +
+       xlab("Miles per gallon (highway)") +
+       ylab("Automotive class")
```

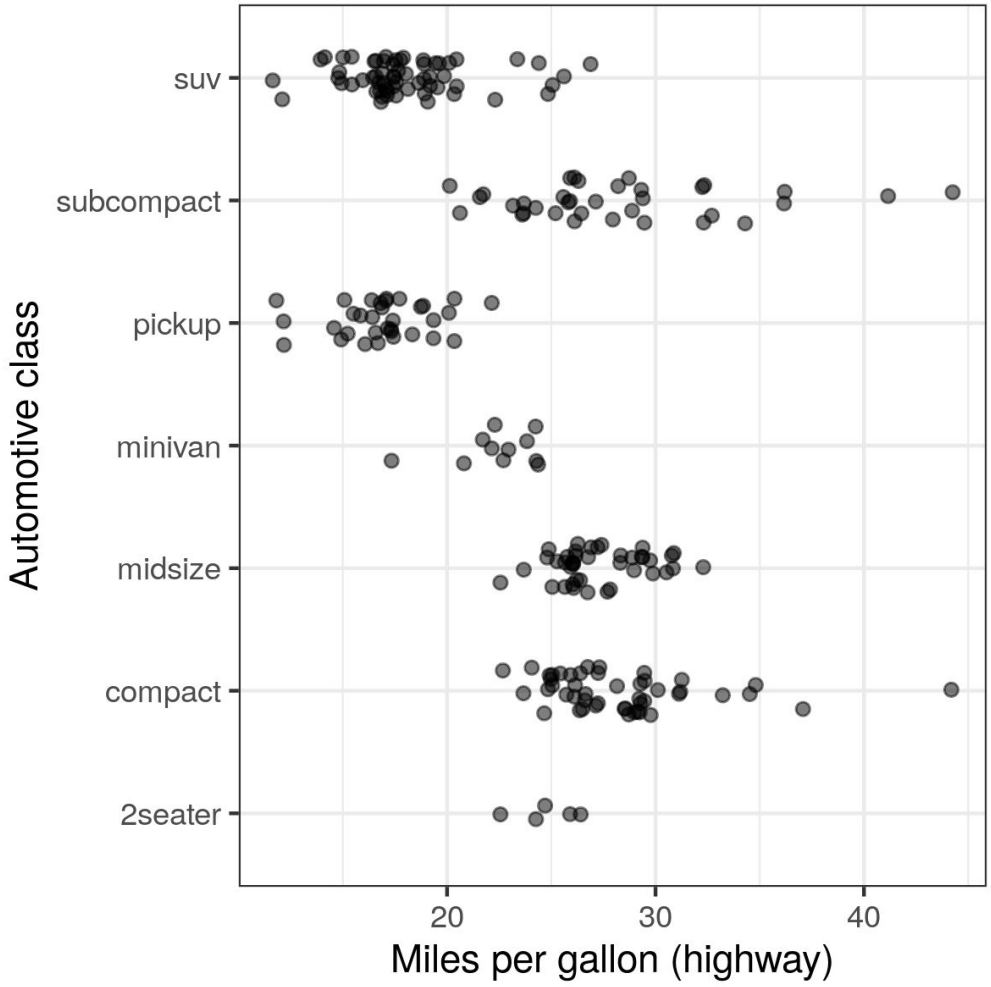

suv
subcompact
pickup
minivan
midsize
compact
2seater
Automotive class
20
30
40
Miles per gallon (highway)

Adding jitter

In such graphics, however, it is difficult to discern where points overlap. One way around this is to add *jitter* (horizontal and vertical noise) to the plot. We can optionally control the degree of jitter along both axes, and set alpha (i.e., saturation) to a value < 1 so that overlap becomes more visible.

```
> r <- ggplot(mpg, aes(x = hwy, y = class)) +
+       geom_jitter(alpha = .5, width = .5, height = .3) +
+       xlab("Miles per gallon (highway)") +
+       ylab("Automotive class")
```

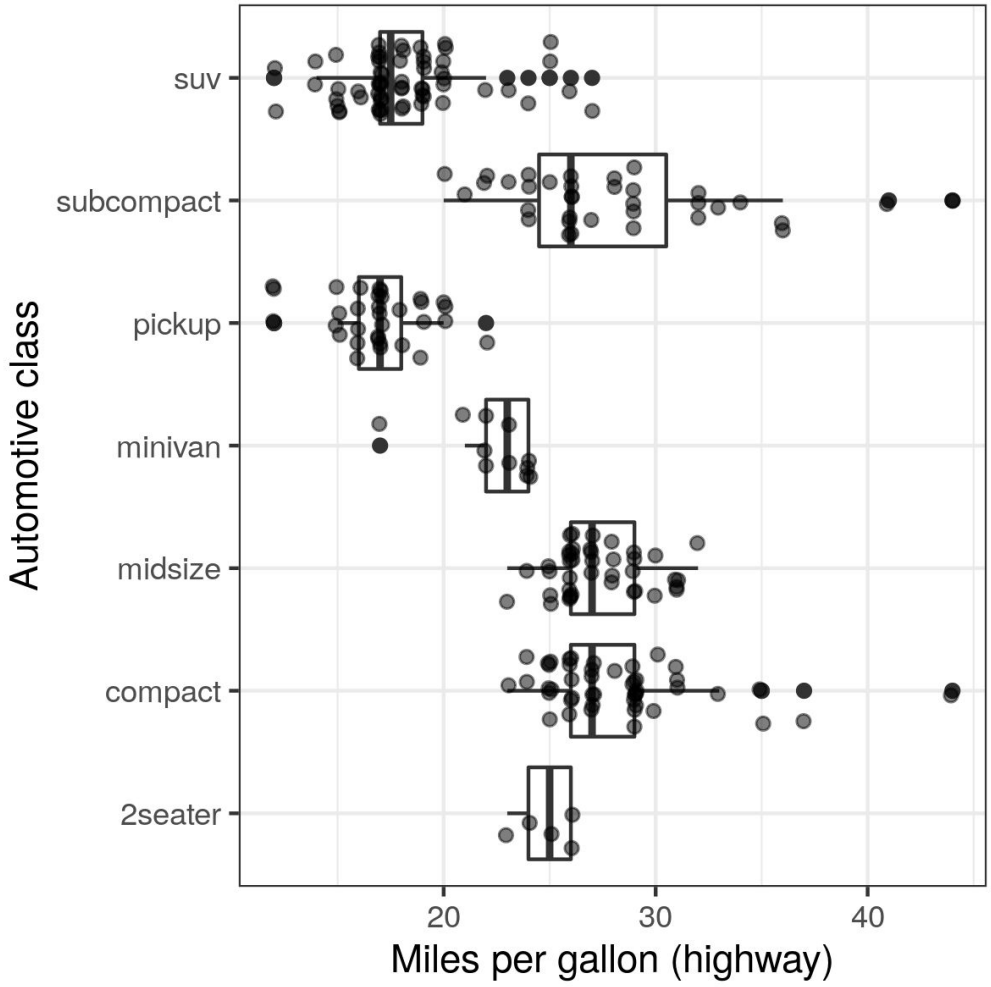
Alternatives to jitter include *stacks* and *beeswarms*, also available in `ggplot2`.


Automotive class
suv
subcompact
pickup
minivan
midsize
compact
2seater
20
30
40
Miles per gallon (highway)

Box plots

To add a non-parametric summary (the three quartiles) to each level of a category, we just add a boxplot before the jittered points. Note two complexities: order matters, and we have to flip the axis to get R to lay it out properly.

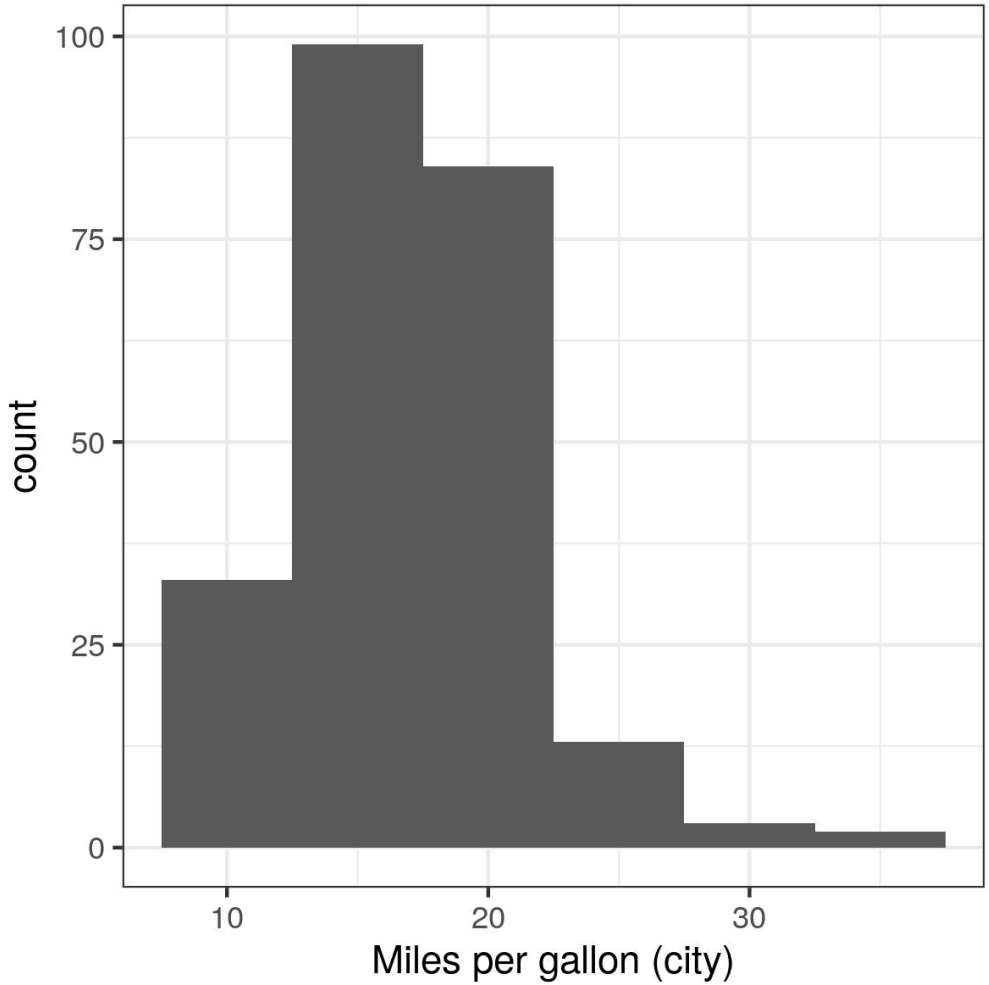
```
> r <- ggplot(mpg, aes(x = class, y = hwy)) +
+       geom_boxplot() +
+       geom_jitter(alpha = .5, width = .3, height = .1) +
+       scale_x_discrete() +
+       coord_flip() +
+       xlab("Automotive class") +
+       ylab("Miles per gallon (highway)")
```

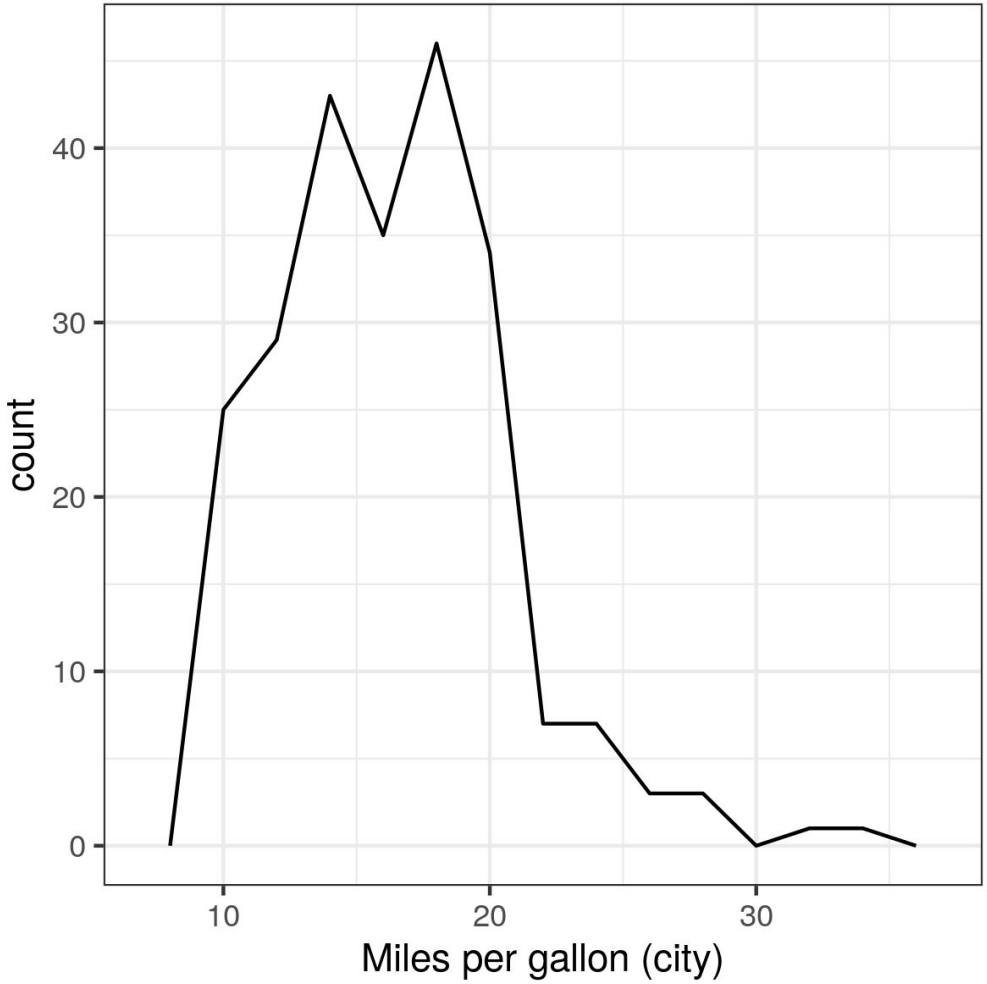

suv
subcompact
pickup
minivan
midsize
compact
2seater
Automotive class
20
30
40
Miles per gallon (highway)

Histograms

And of course we can do histograms and various forms of density plots.

```
> r <- ggplot(mpg, aes(cty)) +
+       geom_histogram(binwidth = 5) +
+       xlab("Miles per gallon (city)")
> r <- ggplot(mpg, aes(cty)) +
+       geom_freqpoly(binwidth = 2) +
+       xlab("Miles per gallon (city)")
```


100
75
50
25
0
count
10
20
30
Miles per gallon (city)


40
30
20
10
0
count
10
20
30
Miles per gallon (city)

Briefly noted

- `geom_dotplot` makes dot plots with stacks, beeswarms, etc., but insists on placing the categorical attribute on the *X*-axis (ick).
- `geom_text` gives you a scatter plot with textual labels instead of points and can be used to make word clouds (yay).
- `geom_tile` can be used to make a table of shaded/colored tiles.

`ggplot2` gotchas

- Some functionality depends on you having previously loaded libraries it needs, though the error message will usually say as much.
- It's very easy to make an ugly plot.

Unsolicited advice

- The default `ggplot2` theme is too "inky": use `theme_bw()` instead.
  - To set it globally for your R session use `theme_set(theme_bw())`.
- Set axes labels with `xlab` and `ylab` rather than using the variable name.
  - The same goes for the legend, and categorical labels appearing on the graph.
- **Do not** set a graph title when writing for publication; use the caption to explain what the graph is, and what all the abbreviations mean, instead.
- Match the font face and size of the graph to that of the rest of the paper.
- **Do not** resize the graph in LaTex or Word; instead, set the size (in inches or pixels) from R and then use the original size.
- Use a 300-DPI PNG as your preferred format unless otherwise directed.
- Go easy on color.

One last thing...

Statistics in Python

Basically everything I've talked about in this class can be done in Python, too:

- basic summary statistics with the (built-in) `statistics` module
- one- and two-sample tests and reference distributions (binomial, χ^2, t) with the (third-party) `scipy.stats` module
- (generalized) linear (mixed-effects) model fitting and interpretation with the (third-party) `statmodels` module
- `ggplot2`-compatible graphing with the (third-party) `plotnine` module

I myself often prefer these over their R equivalents.

Questions? Please take them to email, or Slack.