R Graphics with Ggplot2: Day 1

October,13, 2017

ggplot2 is an add-on package to R. It is an alternative to base graphics that has become very popular, to the point where it is recommended/preferred unless you have old code that already uses a different graphing package.

ggplot2 documentation is available at docs.ggplot2.org

Why Ggplot?

- Wilkinson, *Grammar of Graphics* (1999)
- Ggplot2 is an implementation of GoG for R
- Benefits:
  - handsome default settings
  - snap-together building block approach
  - automatic legends, colours, facets
  - statistical overlays: regressions lines and smoothers (with confidence intervals)
- Drawbacks:
  - it can be hard to get it to look *exactly* the way you want
  - requires having the input data in a certain format

Ggplot Building Blocks

- data: 2D table (`data.frame`) of *variables*
- *aesthetics*: map variables to visual attributes (e.g., position)
- *geoms*: graphical representation of data (points, lines, etc.)
- *stats*: statistical transformations to get from data to points in the plot(binning, summarizing, smoothing)
- *scales*: control *how* to map a variable to an aesthetic
- *facets*: juxtapose mini-plots of data subsets, split by variable(s)
- *guides*: axes, legend, etc. reflect the variables and their values

Idea: independently specify and combine the blocks to create the plot you want.

Getting started

There are at least three things we have to specify to create a plot:

1. data
2. aesthetic mappings from data variables to visual properties
3. a layer describing how to draw those properties

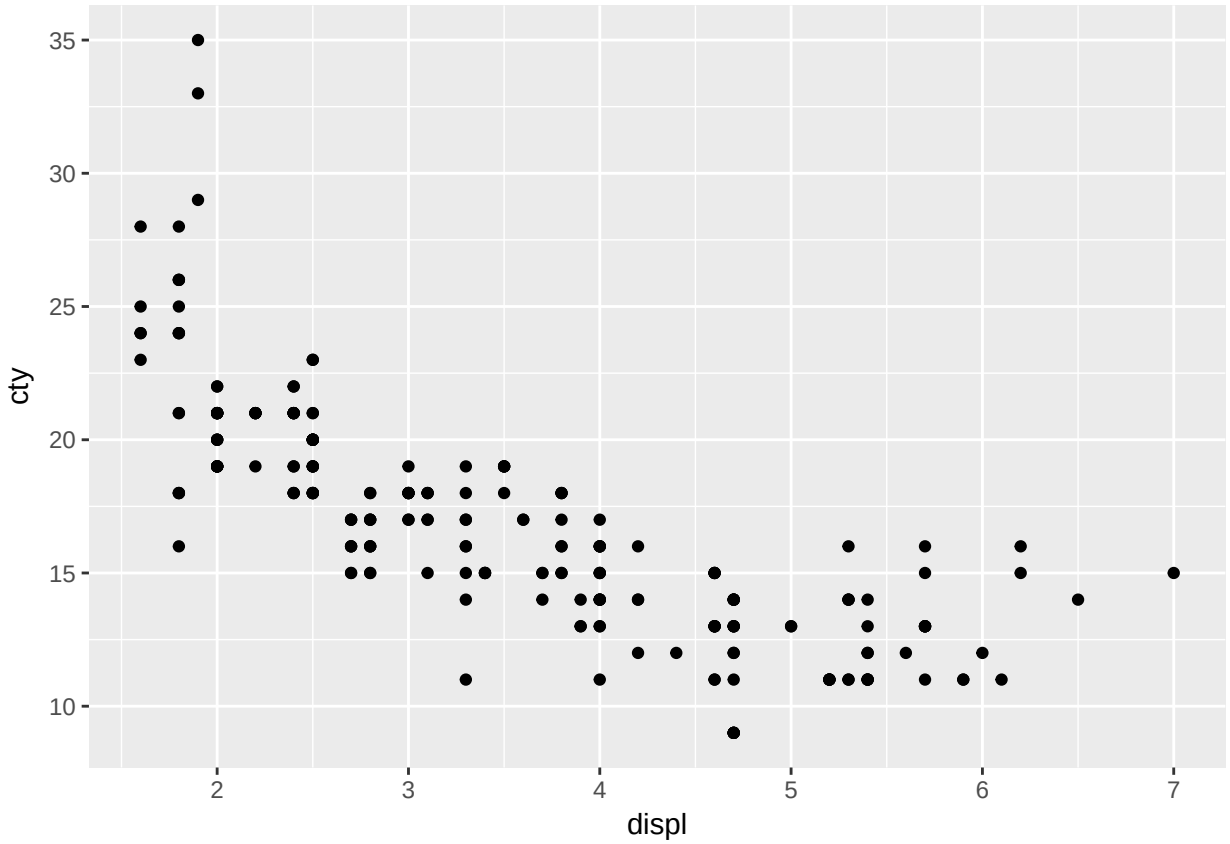
(We are using the `mpg` dataset in the next few examples. It contains information about the fuel economy of a sample of car models in 1999 and 2008. See its help page for details of its variables.)

```
library(ggplot2)
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl      trans   drv   cty   hwy    fl
##          <chr> <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>
## 1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p
```

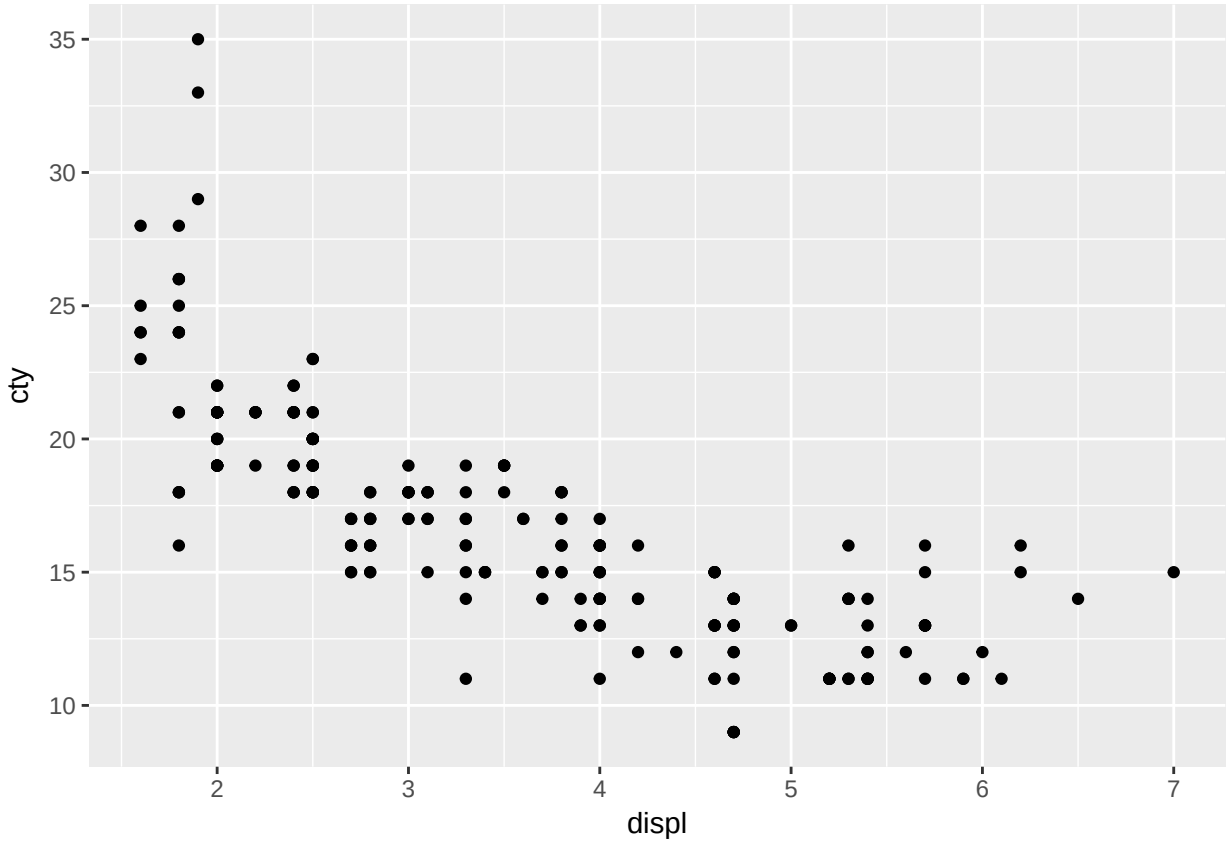
```
## 2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p
## 3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p
## 4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p
## 5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p
## 6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p
## # ... with 1 more variables: class <chr>
```

```
ggplot(mpg, aes(x = displ, y = cty)) + geom_point()
```



Unlike base graphics ggplot creates an R object that can saved for later use. Like all R objects, it must be printed to be viewed.

```
p1 <- ggplot(mpg, aes(x = displ, y = cty)) + geom_point()
p1
```

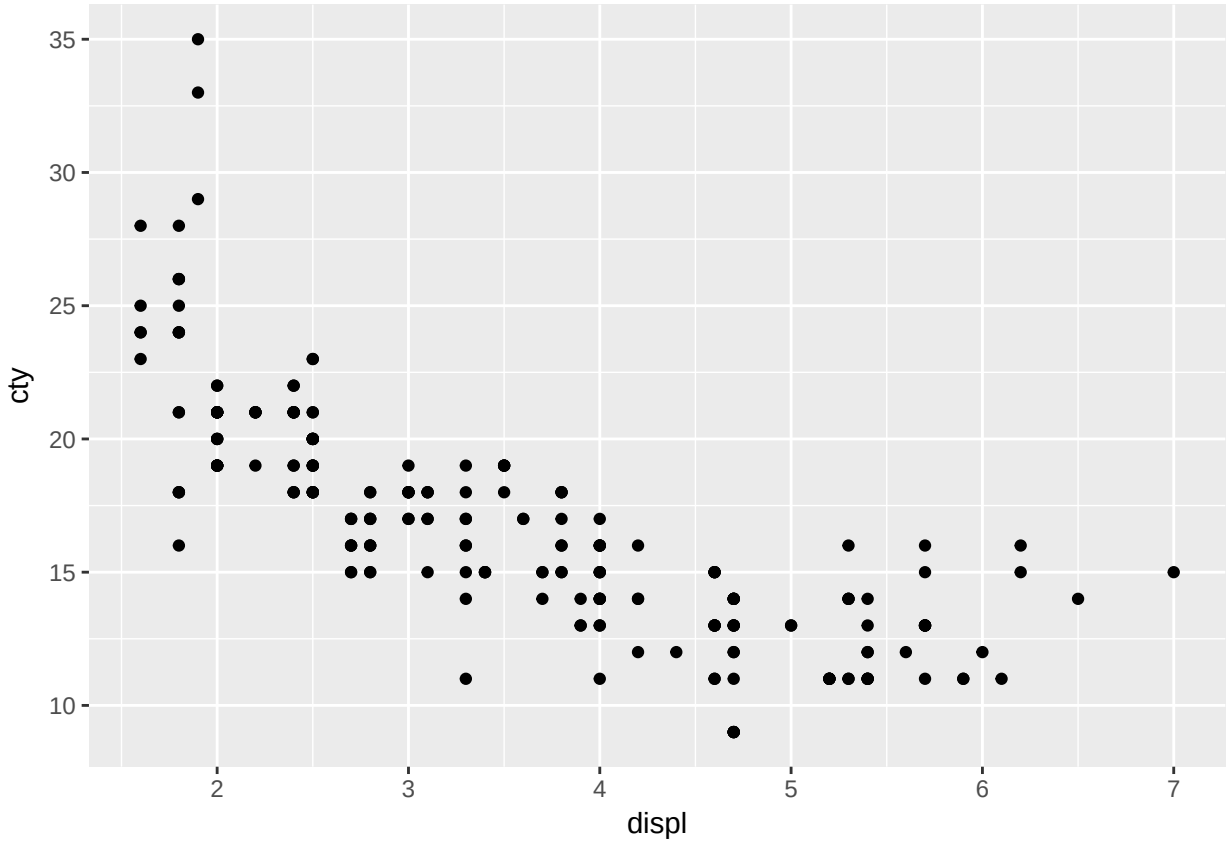



```
class(p1)
```

```
## [1] "gg"     "ggplot"
```

For convenience, the first two arguments to `aes` are implicitly `x` and `y` and don't need to be specified:

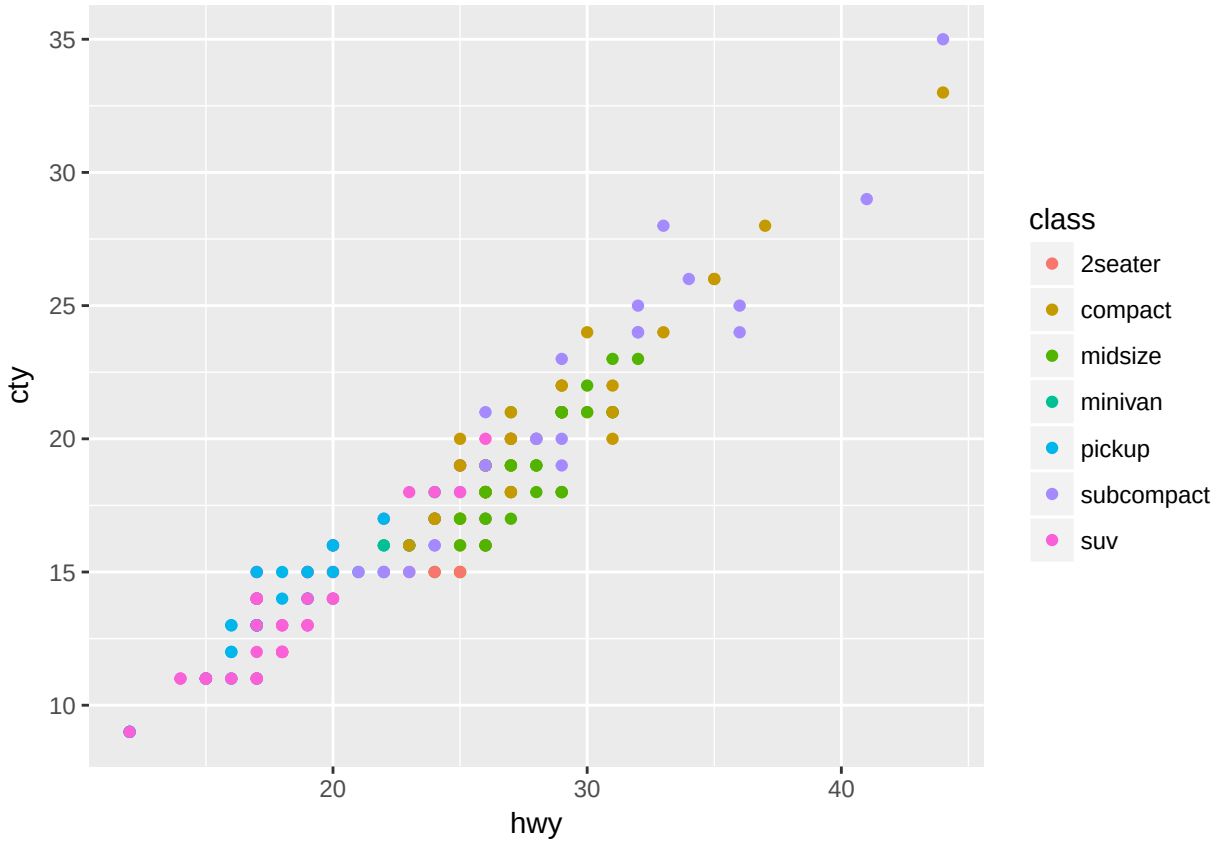
```
ggplot(mpg, aes(displ, cty)) + geom_point()
```



Adding colours, shapes and sizes

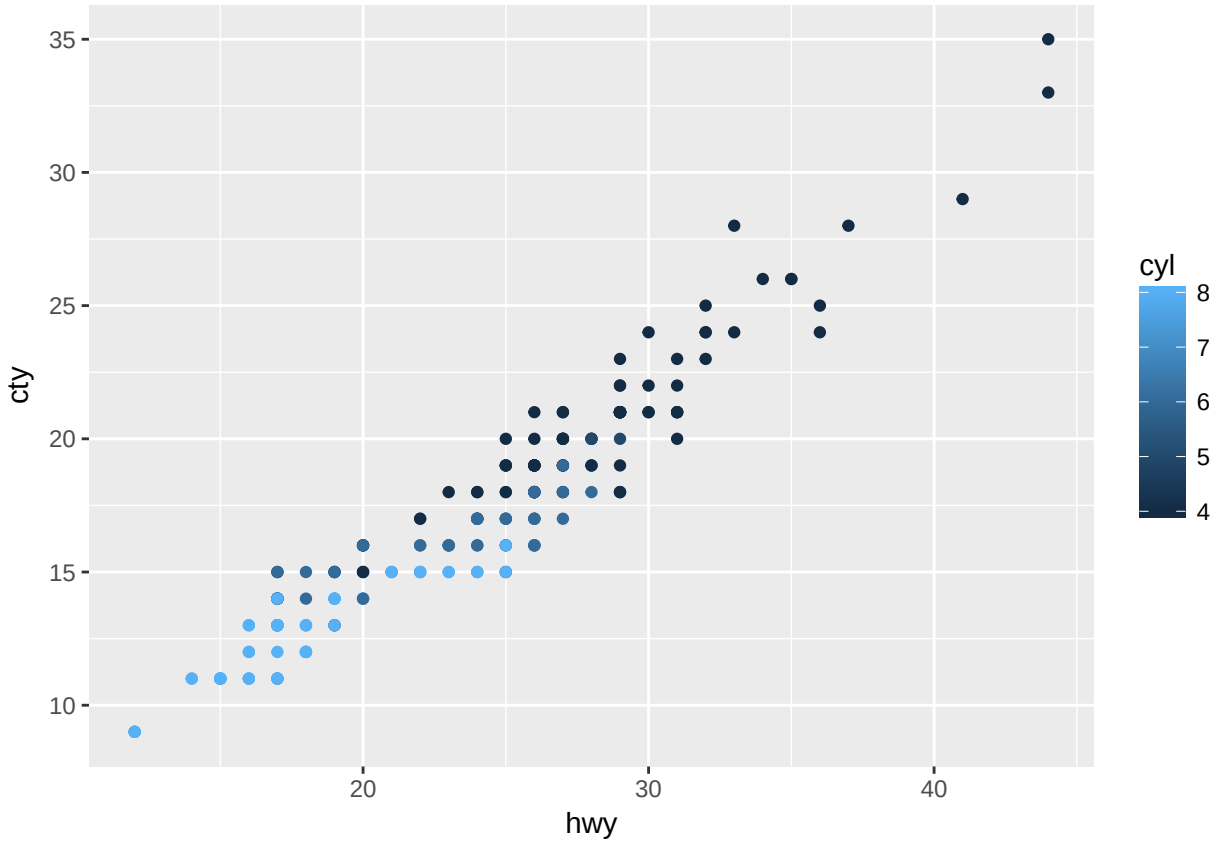
We can add colour to the plot simply by specifying the `colour` aesthetic:

```
ggplot(mpg, aes(hwy,cty, colour = class)) + geom_point()
```

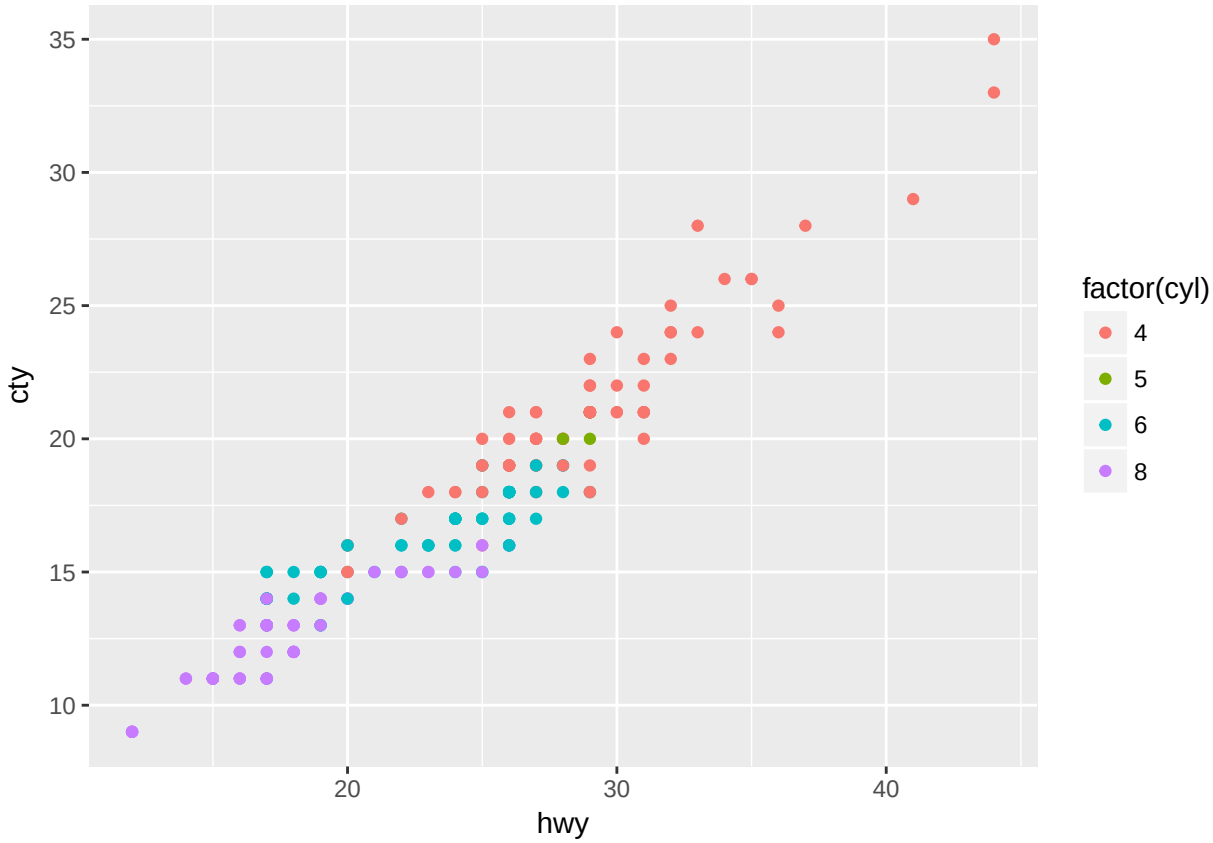


Note that if the colour variable needs to be discrete (a factor or text), or ggplot2 will use a continous colour scale.

```
ggplot(mpg, aes(x = hwy, y = cty, colour = cyl)) + geom_point()
```

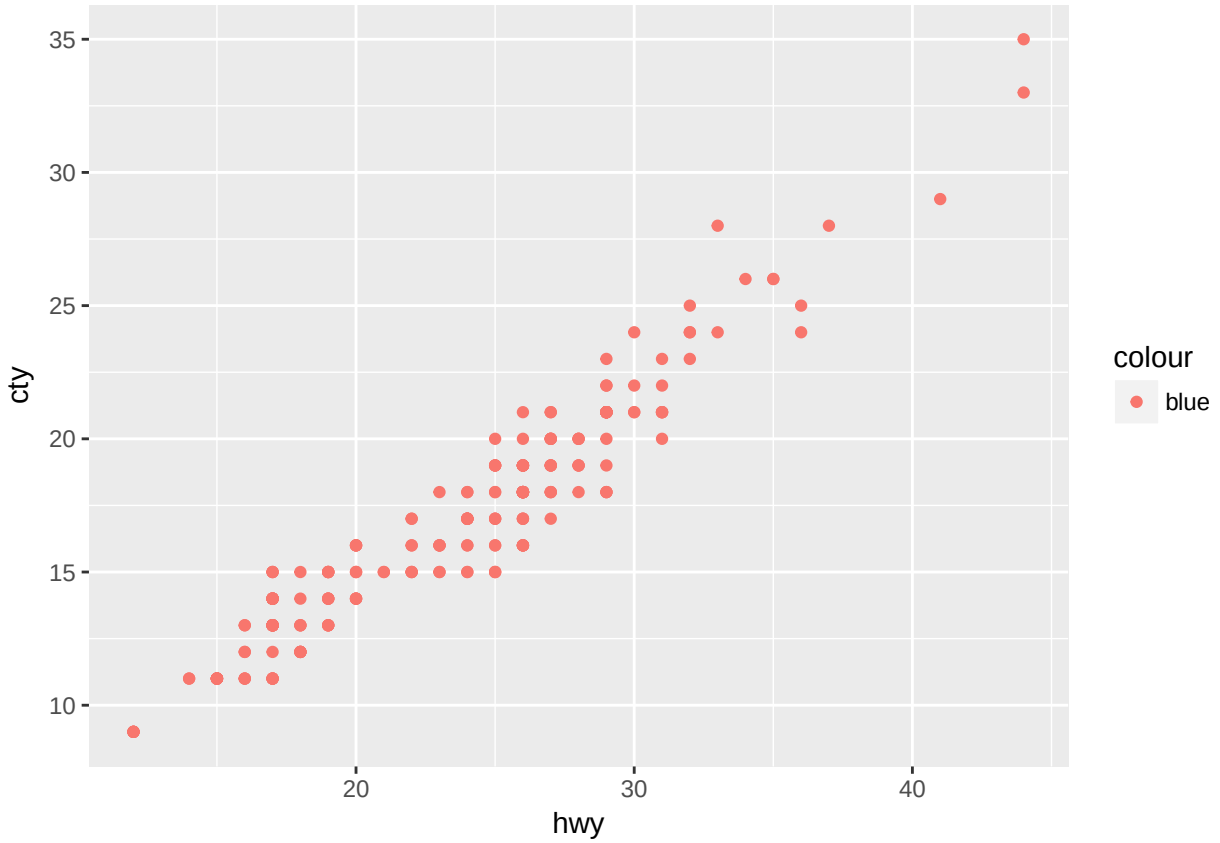



```
ggplot(mpg, aes(x = hwy, y = cty, colour = factor(cyl))) + geom_point()
```

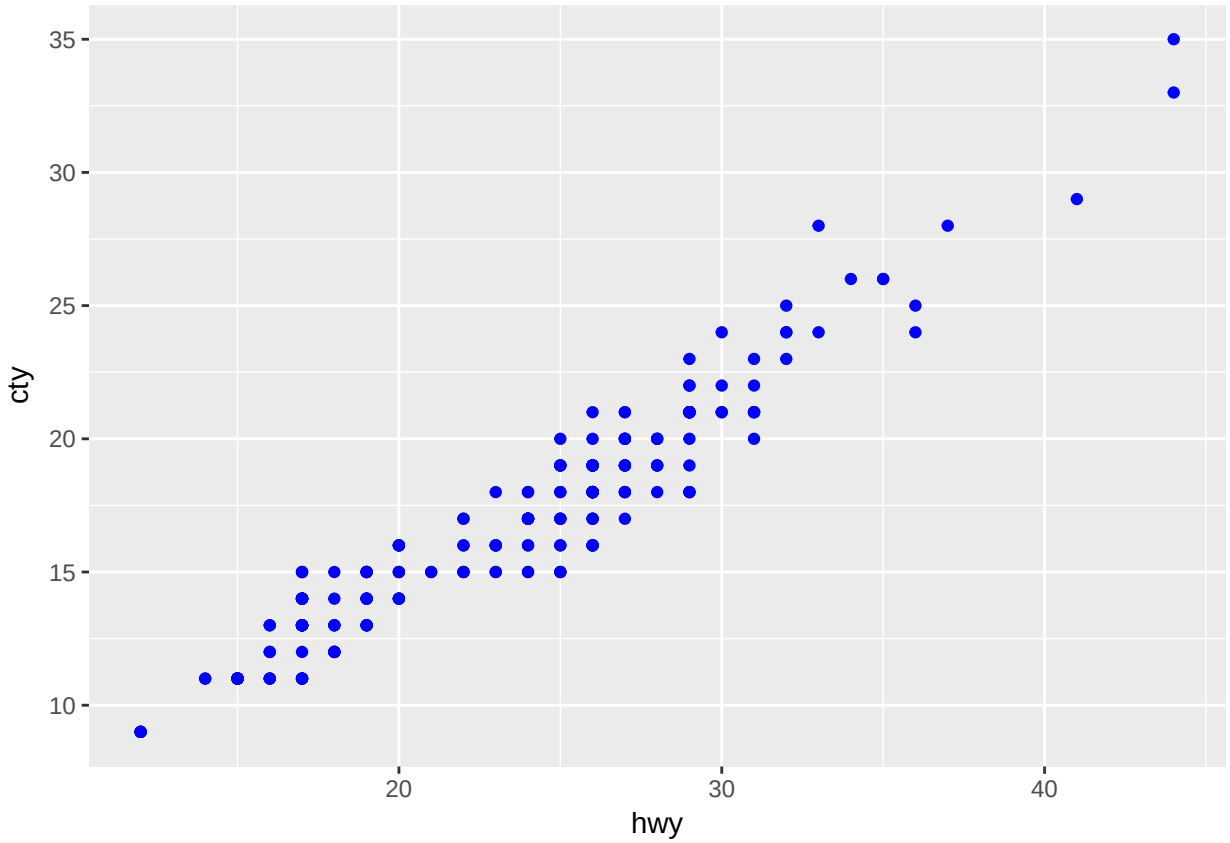


Also note that if you want to pick a colour directly, you shouldn't do it as an aesthetic, but directly in the geom:

```
ggplot(mpg, aes(x = hwy, y = cty,
                colour = 'blue')) +
  geom_point()
```



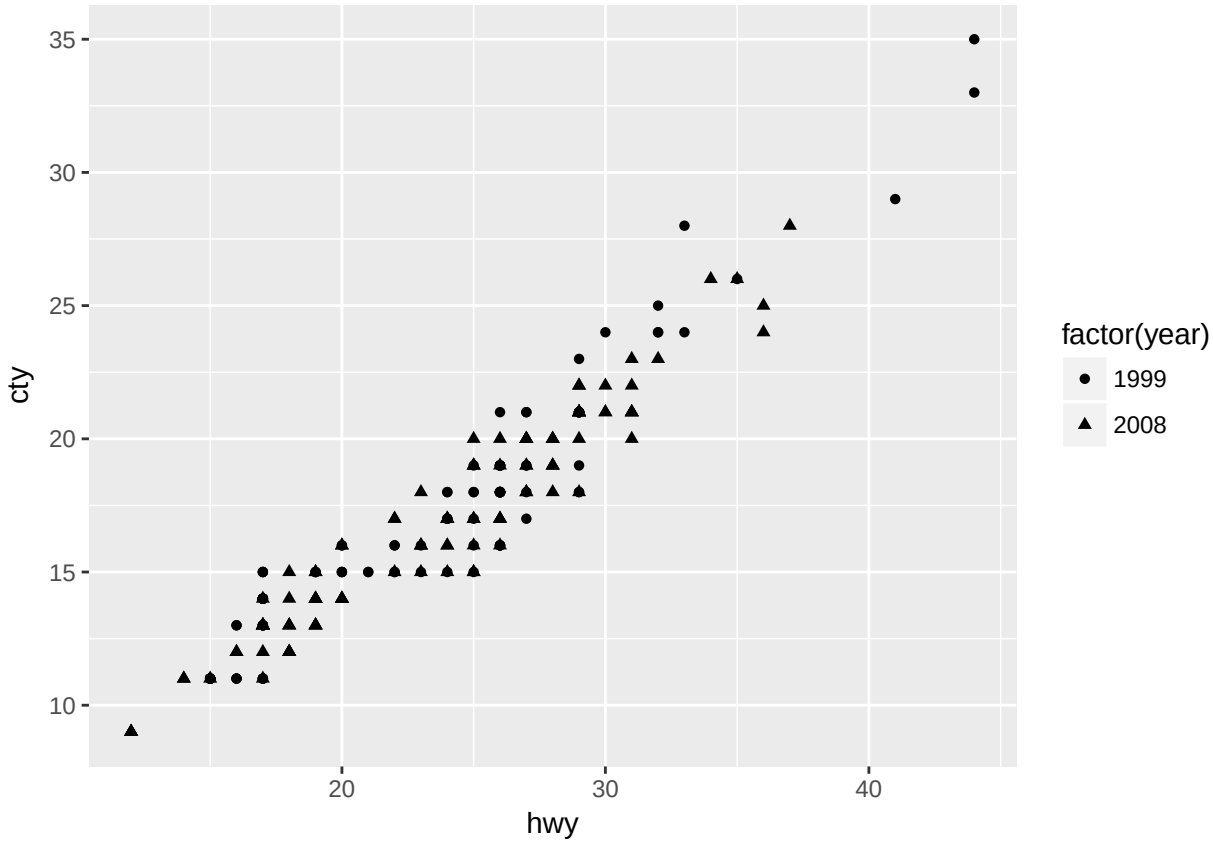

```
ggplot(mpg, aes(x = hwy, y = cty)) +
  geom_point(colour = 'blue')
```



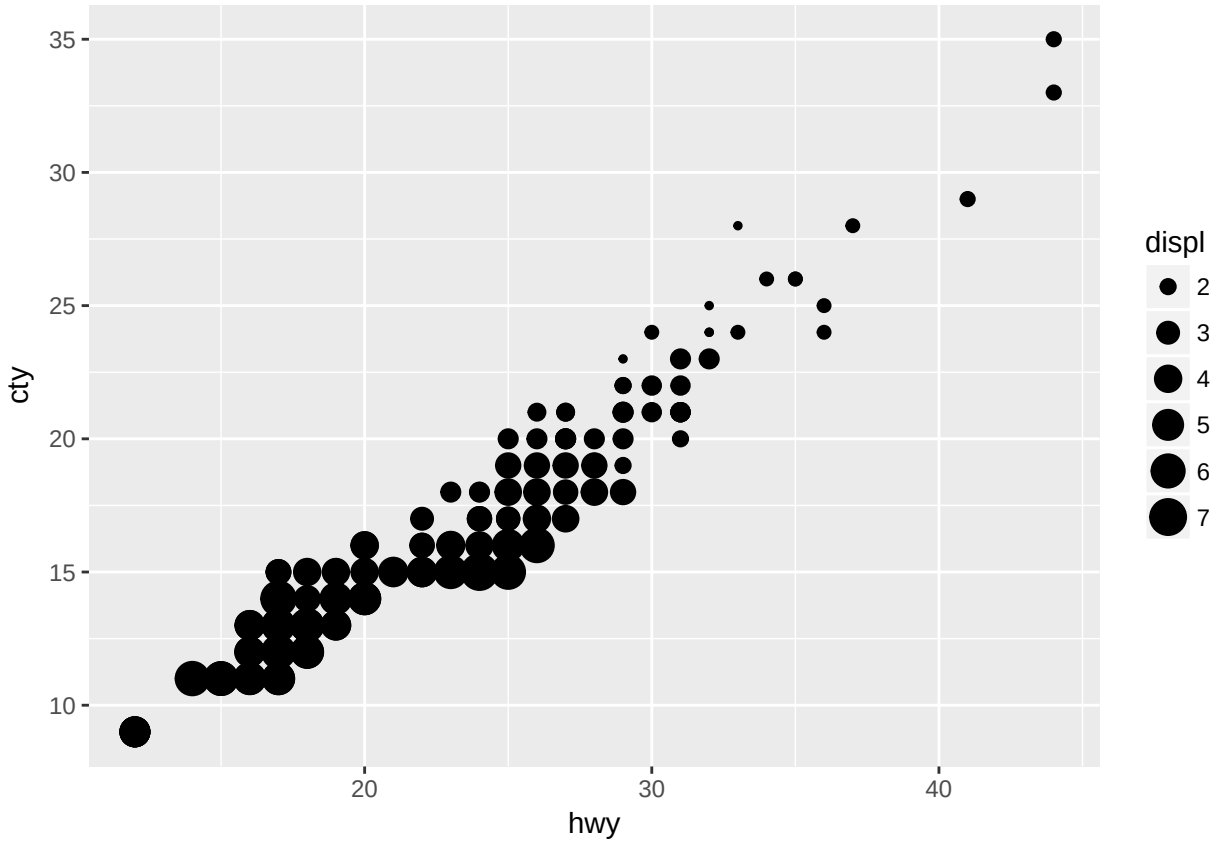
(But there are other ways of doing this with ggplot. We will see them later on.)

We can plot points using different shapes, or even the size of the points (a bubble chart) using other aesthetics:

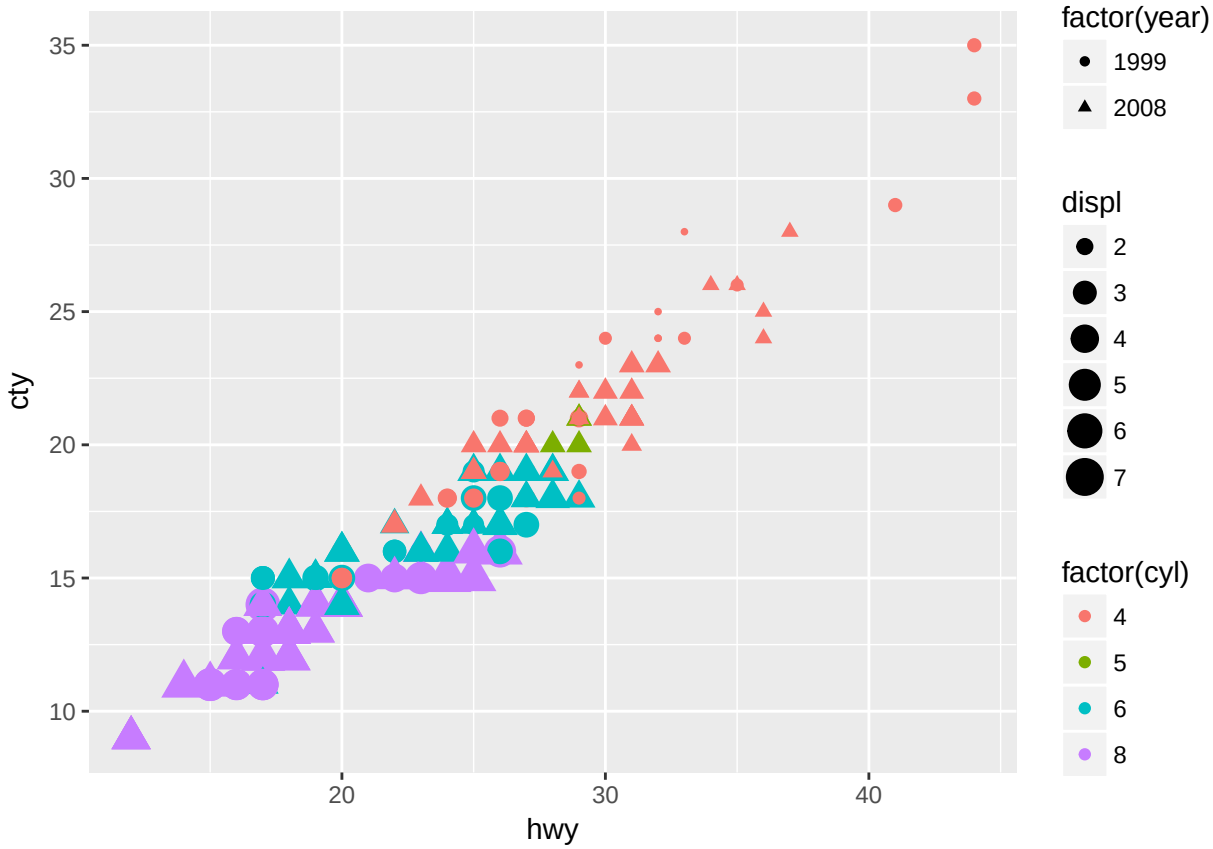
```
ggplot(mpg, aes(x = hwy, y = cty,
                shape = factor(year))) +
  geom_point()
```




```
ggplot(mpg, aes(x = hwy, y = cty,
                size = displ)) +
  geom_point()
```




```
ggplot(mpg, aes(x = hwy, y = cty,
                colour = factor(cyl),
                shape=factor(year),
                size=displ)) +
  geom_point()
```



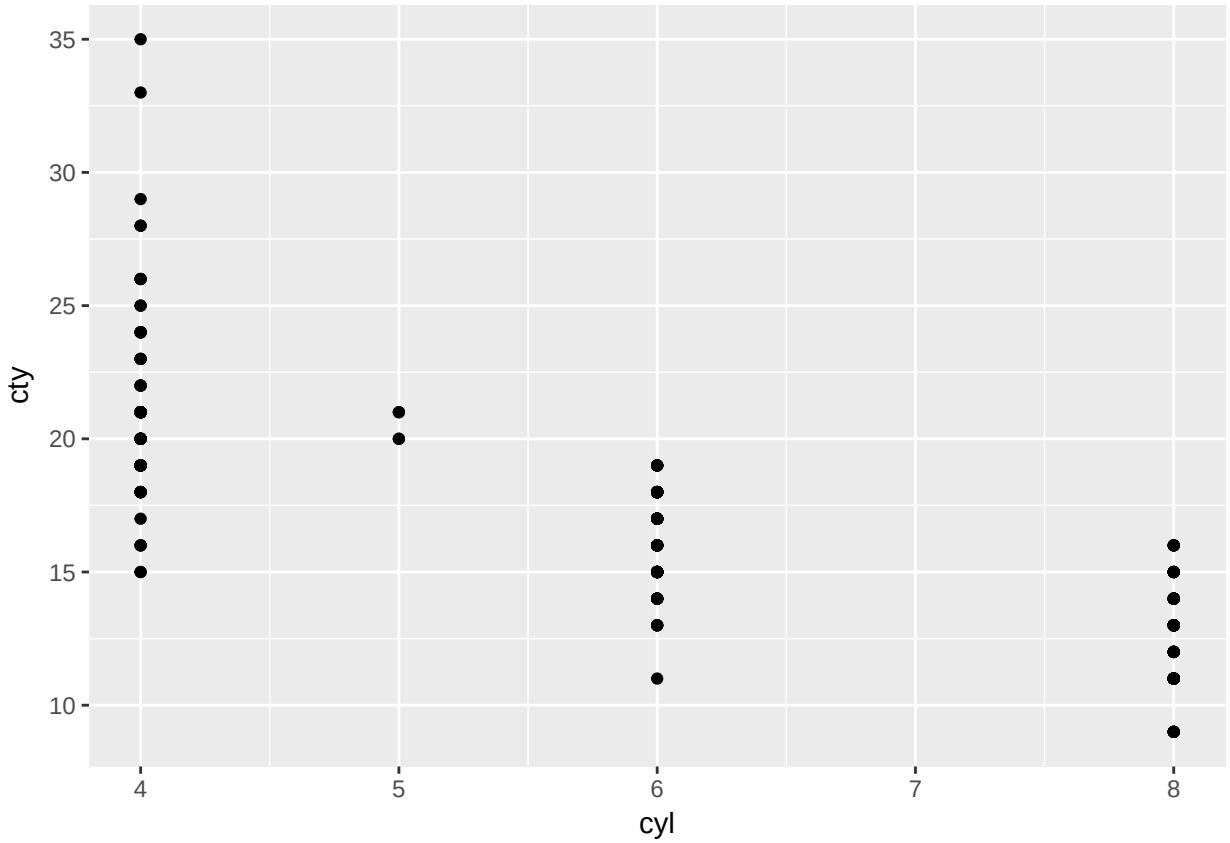
Exercise: How is the drive train related to engine size and vehicle class?

Other geoms

As you might have guessed, using a different `geom_` function would produce a completely different plot.

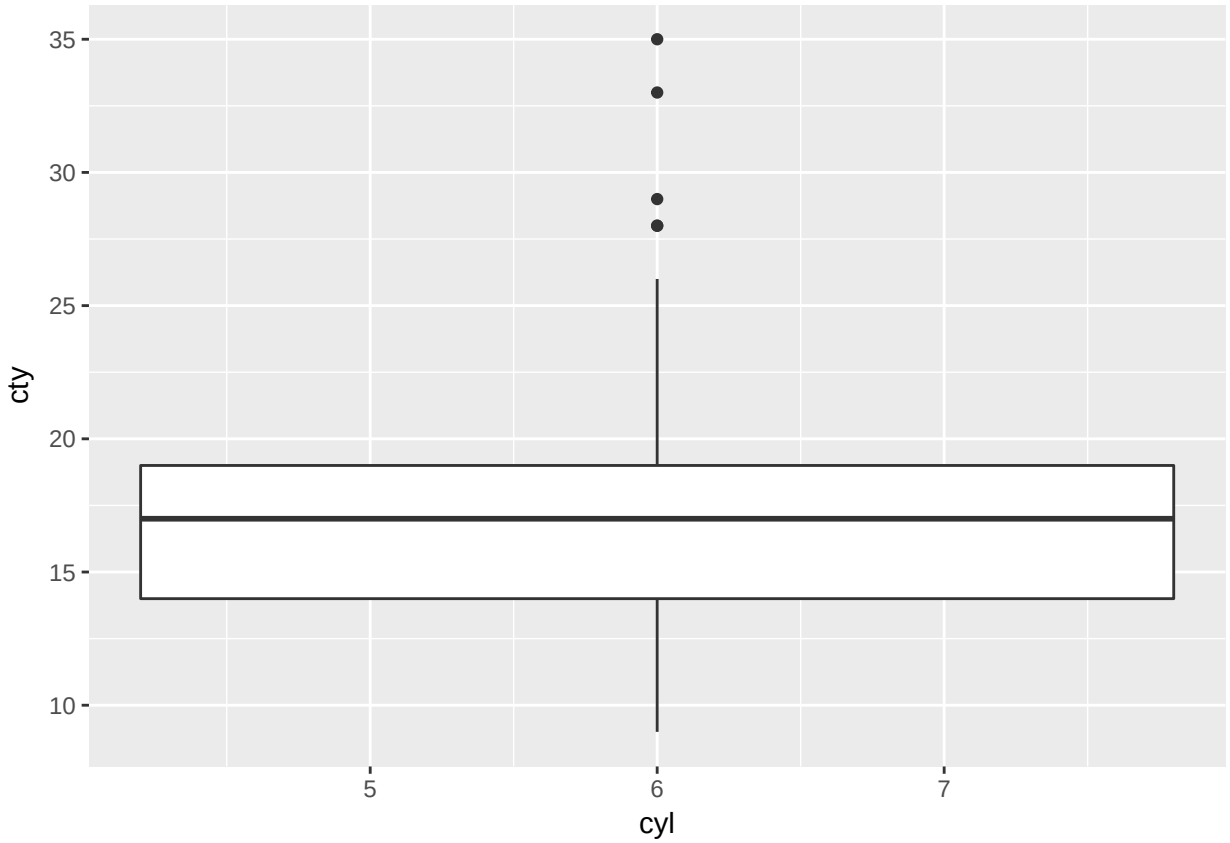
Box plots and related plots let us see how a continuous variable varies with the levels of a categorical variable:

```
ggplot(mpg, aes(cyl, cty)) + geom_point()
```



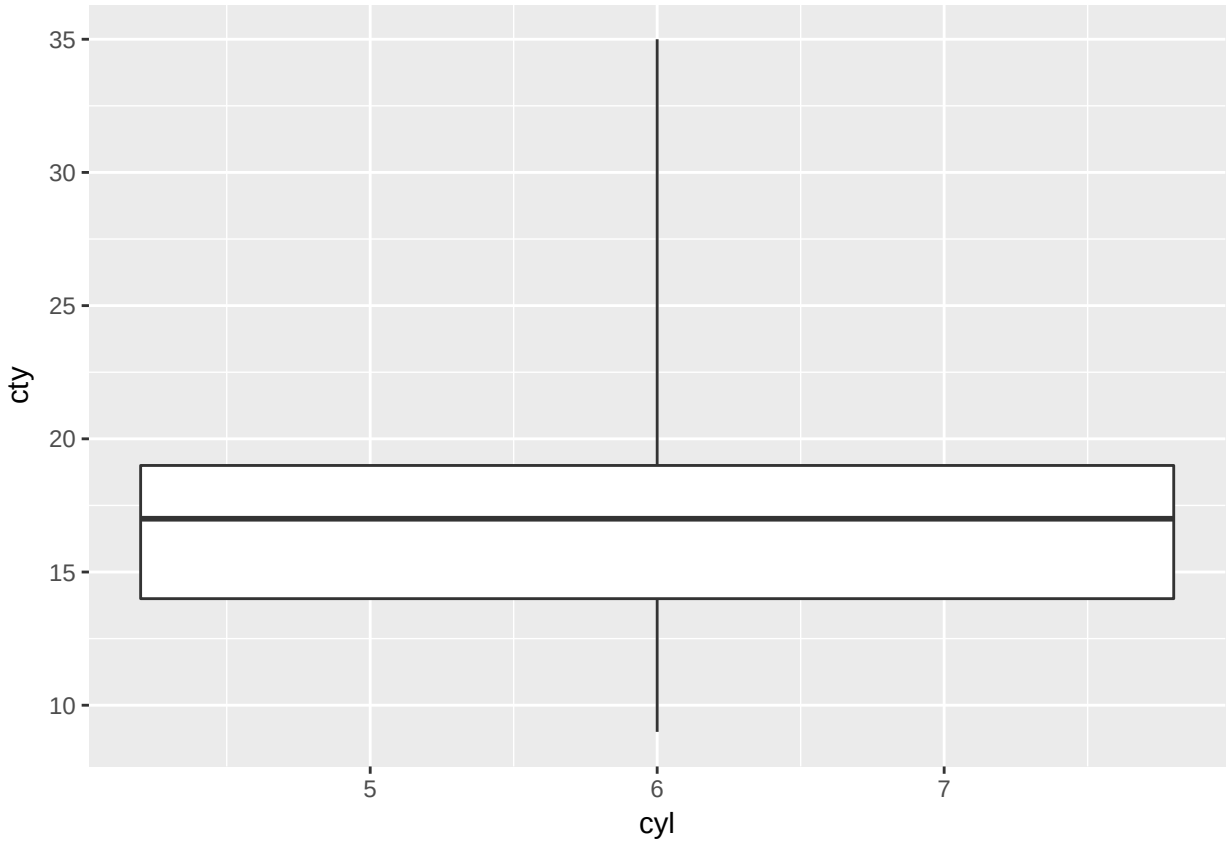

```
ggplot(mpg, aes(cyl, cty)) + geom_boxplot()
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```

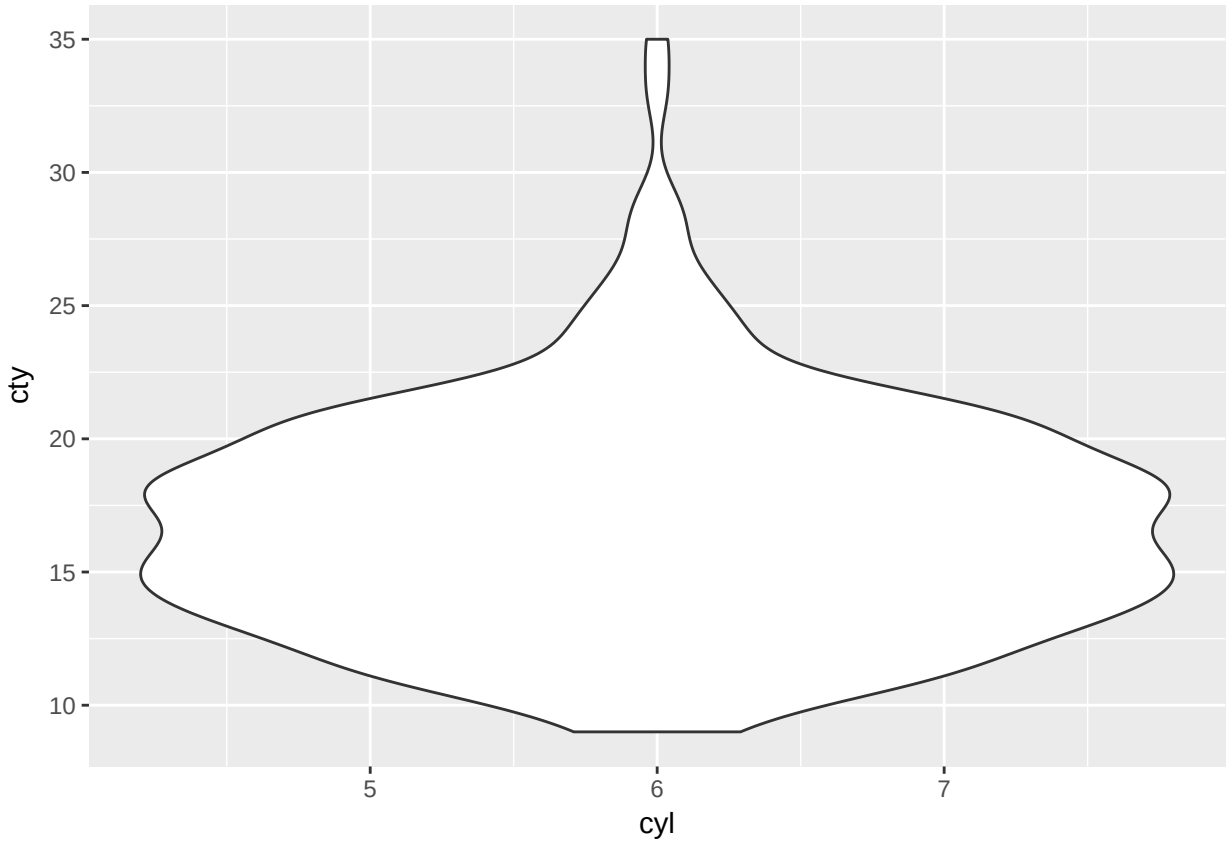



```
ggplot(mpg, aes(cyl, cty)) + geom_boxplot(coef = NULL)
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```

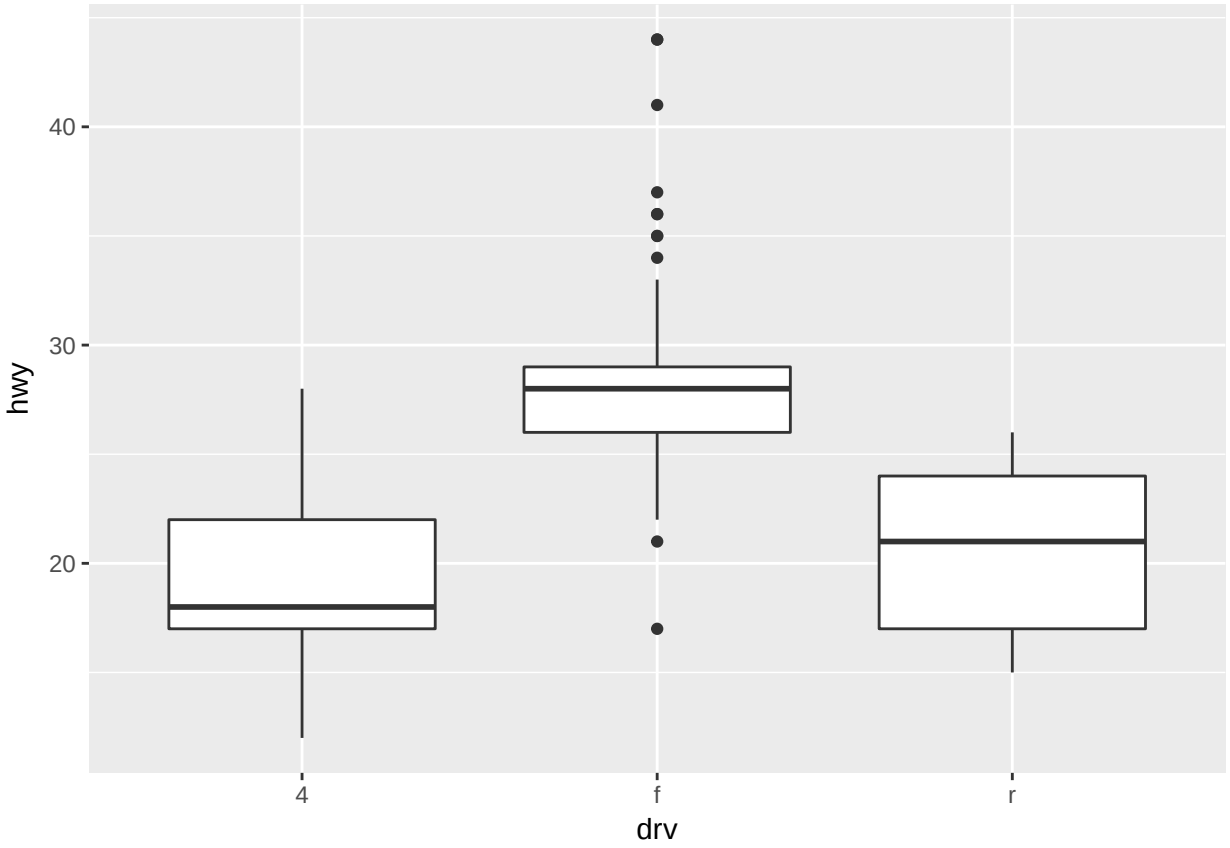



```
ggplot(mpg, aes(cyl, cty)) + geom_violin()
```



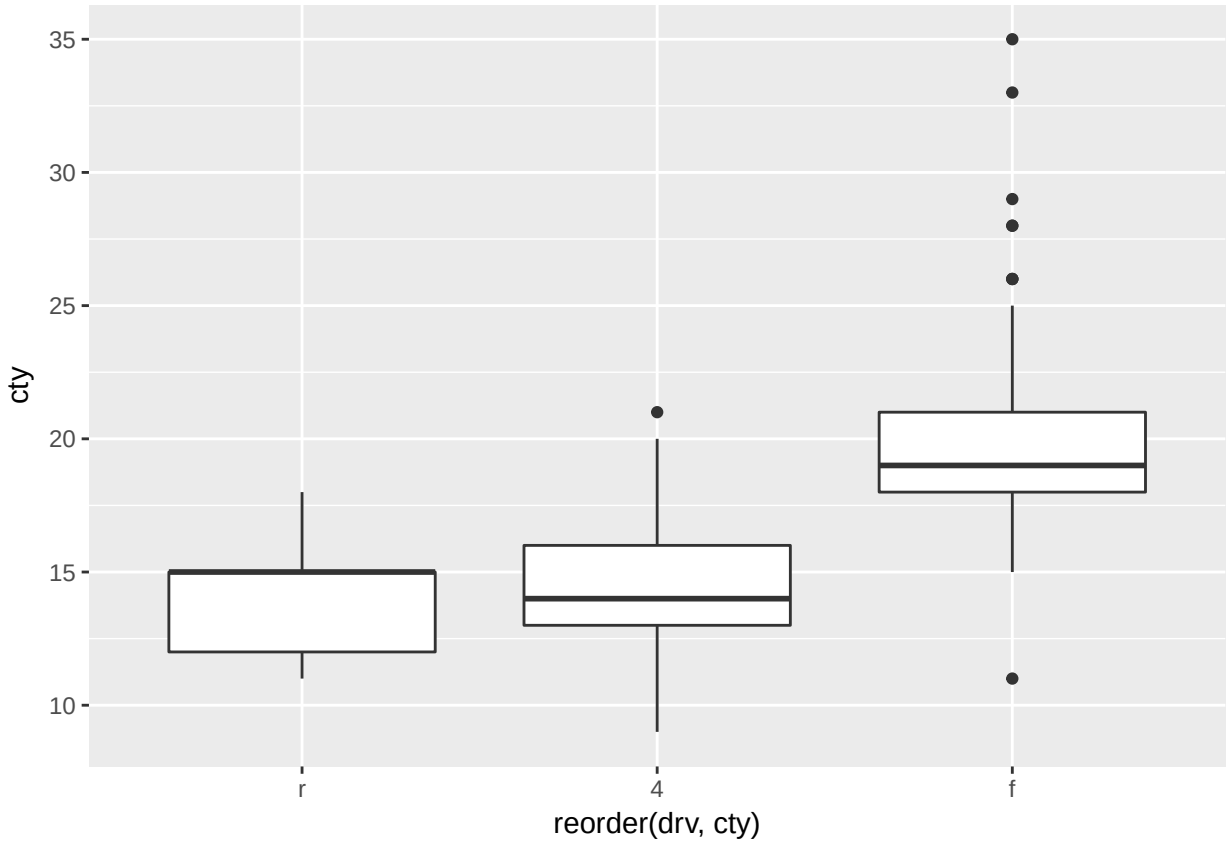
The default ordering on the X-axis will be alphabetical:

```
ggplot(mpg, aes(drv, hwy)) + geom_boxplot()
```



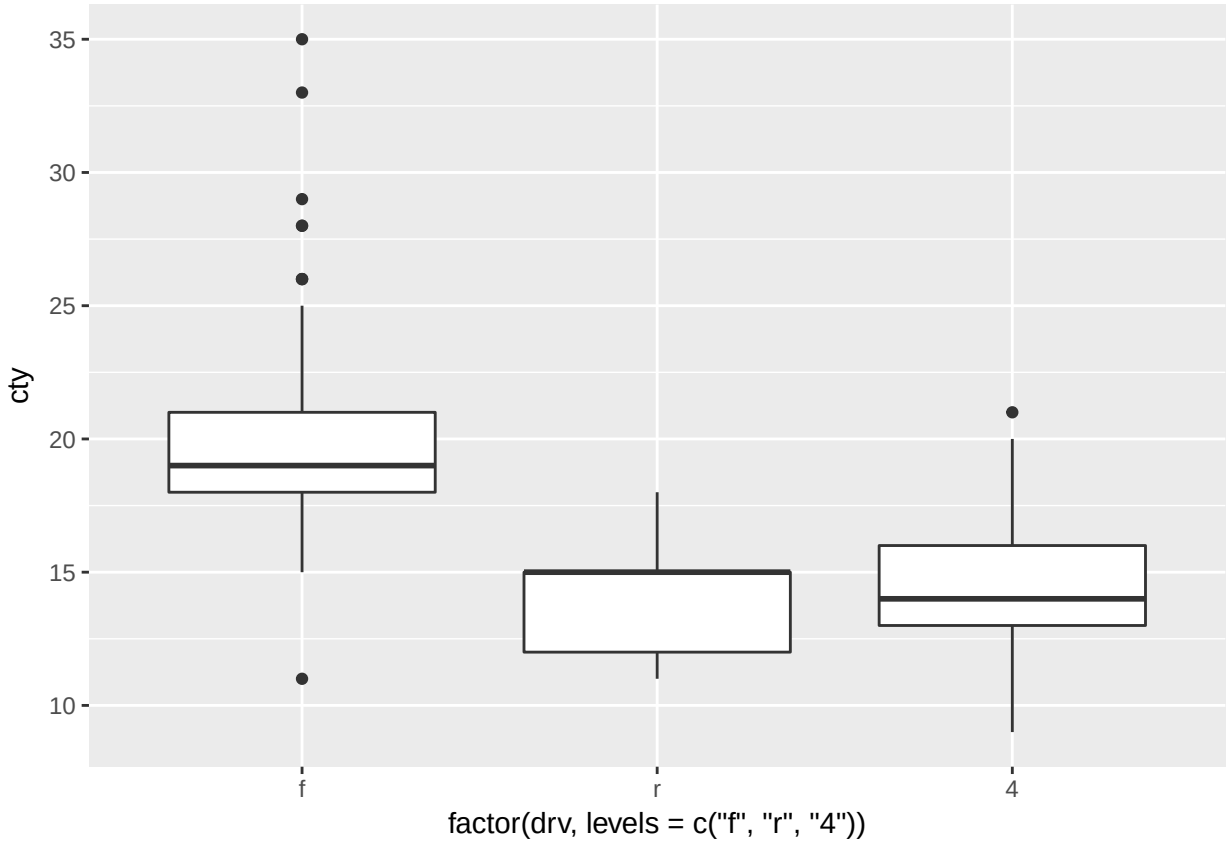
It can be convenient to order so that the values on the Y-axis increase from left to right. You can do this with `reorder`:

```
ggplot(mpg, aes(reorder(drv, cty), cty)) + geom_boxplot()
```



To choose the order explicitly is a little more awkward:

```
ggplot(mpg, aes(factor(drv, levels=c('f', 'r', '4')),
                cty)) +
  geom_boxplot()
```

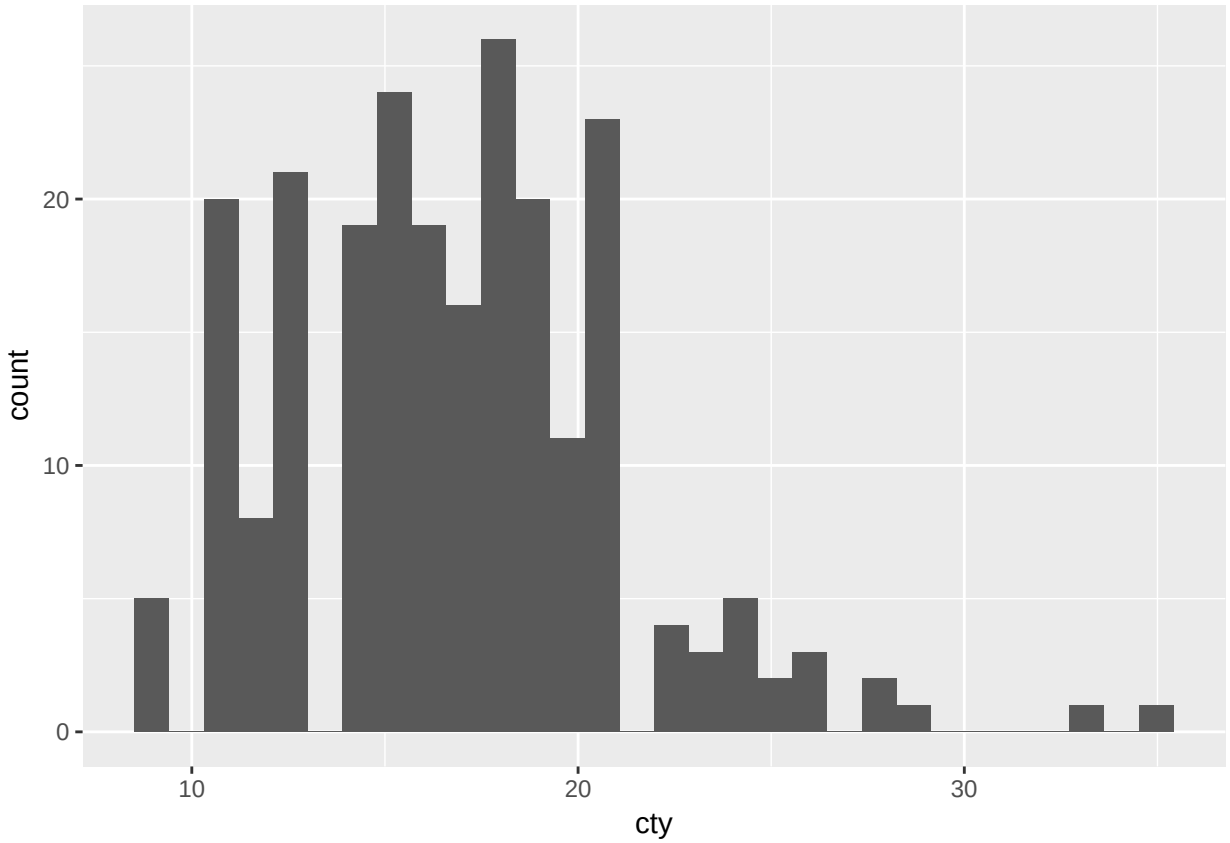


There are better ways to do this with ggplot, as we will see in the next session.

Histograms show the distribution of a single numeric variable:

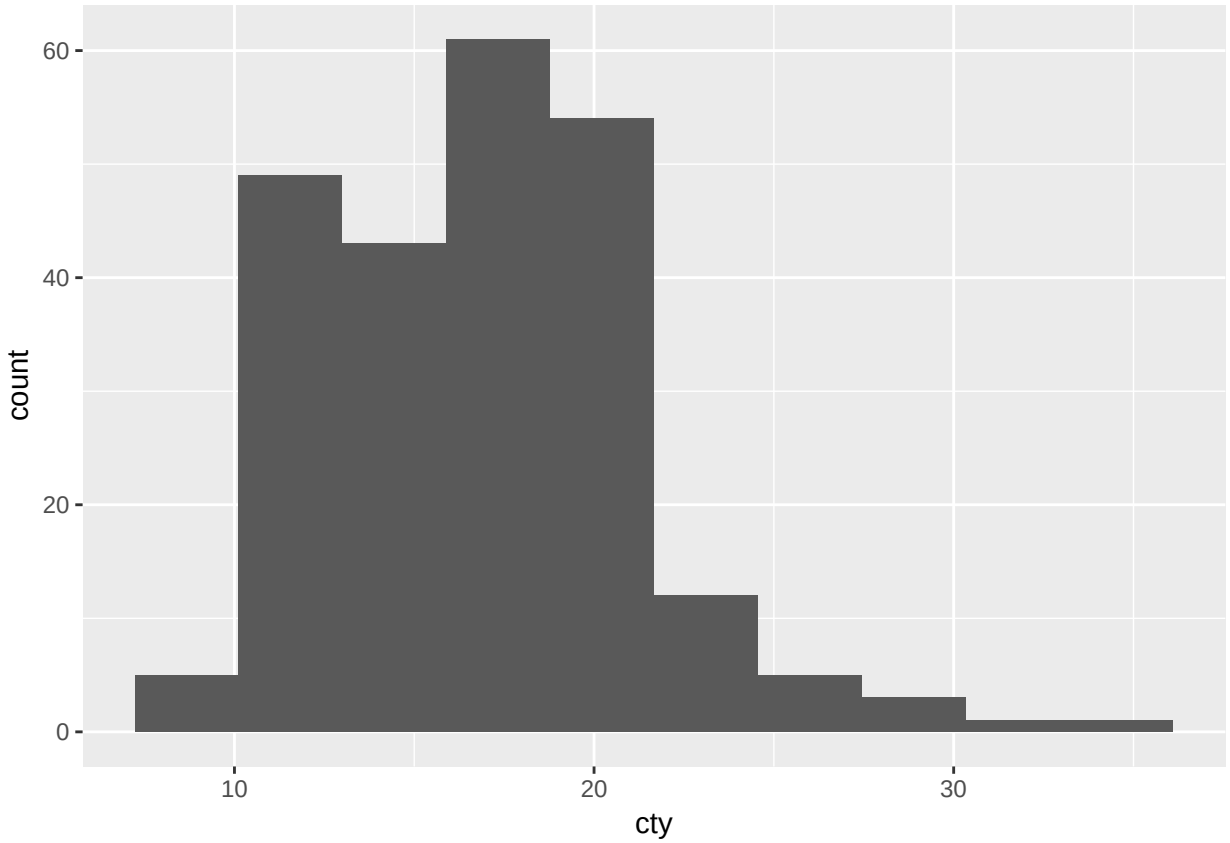
```
ggplot(mpg, aes(cty)) + geom_histogram()
```

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

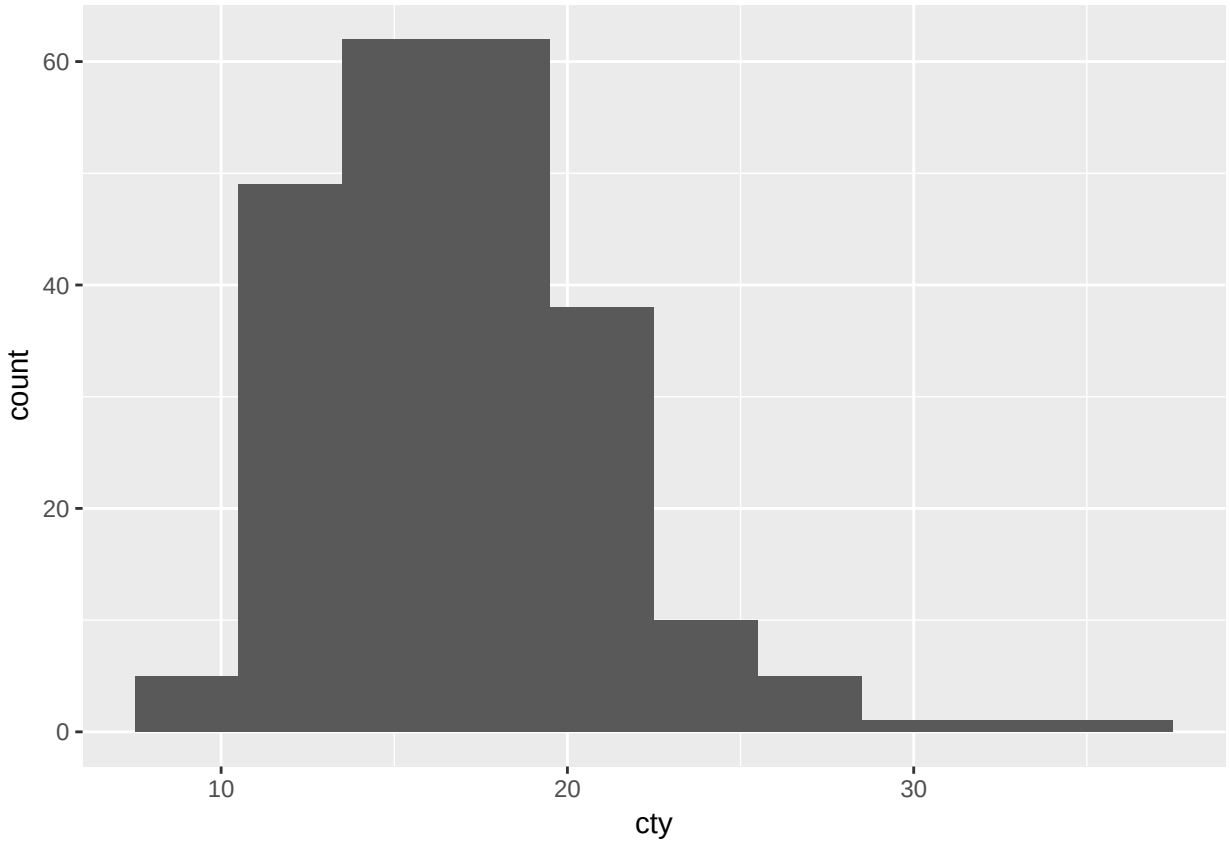


Histogram "bins" the data, but to really explore the distribution, you will want to try out different numbers of bins (or, equivalently, their widths):

```
ggplot(mpg, aes(cty)) + geom_histogram(bins=10)
```



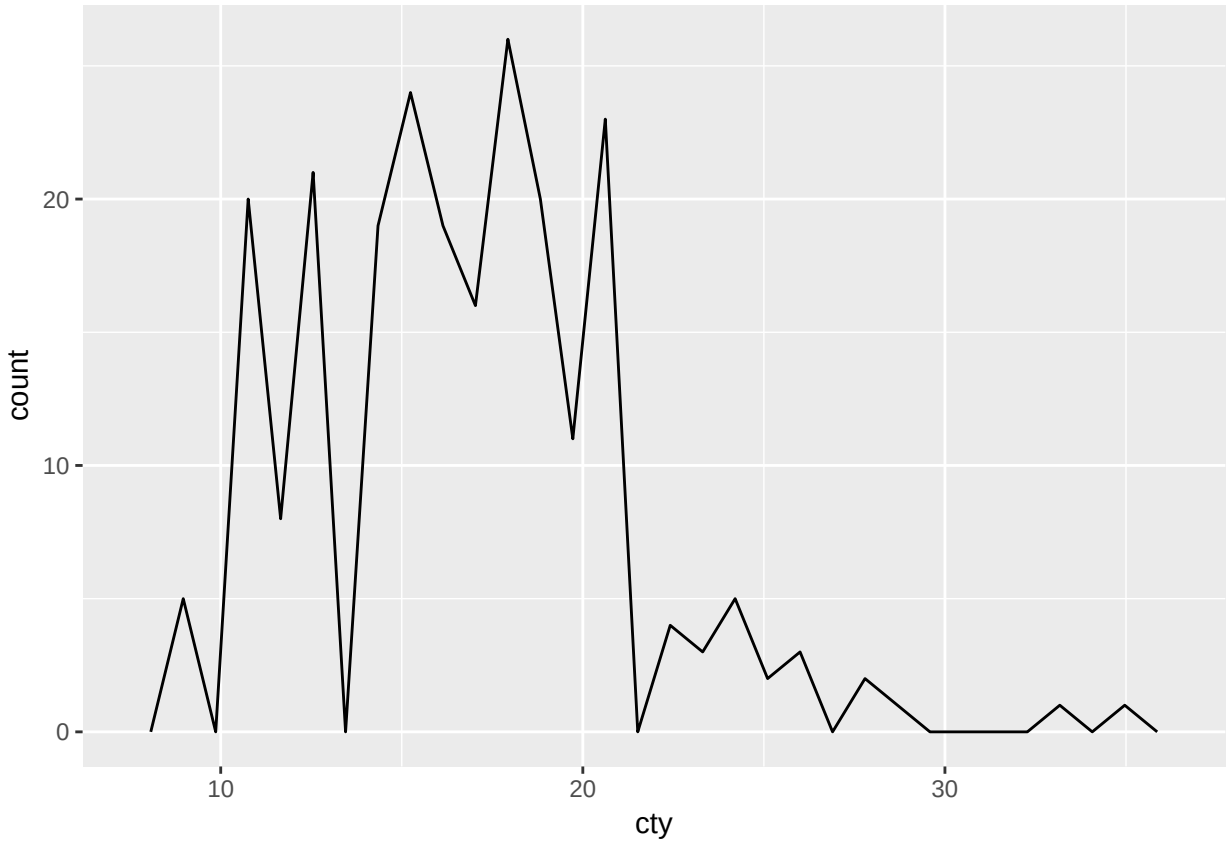

```
ggplot(mpg, aes(cty)) + geom_histogram(binwidth=3)
```



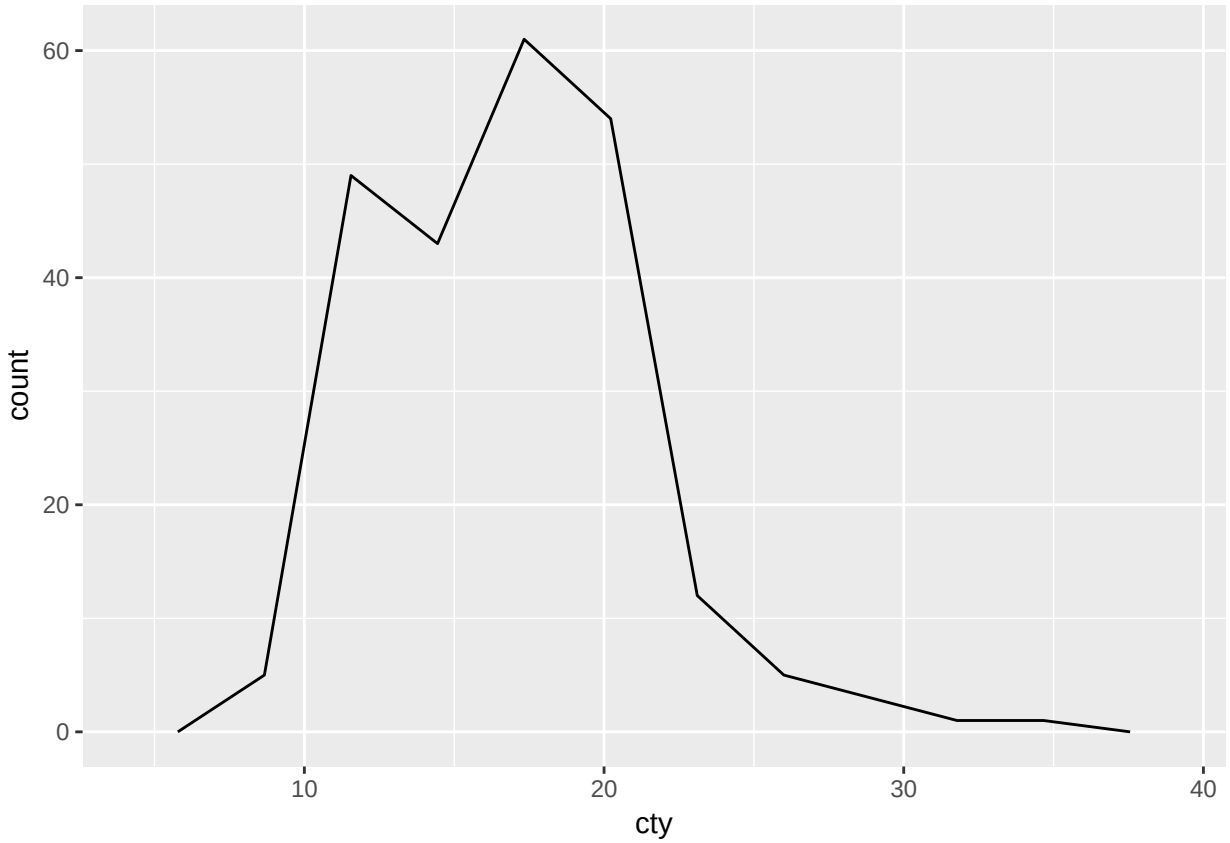
Frequency poligons work the same way except that they draw lines instead of bars:

```
ggplot(mpg, aes(cty)) + geom_freqpoly()
```

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

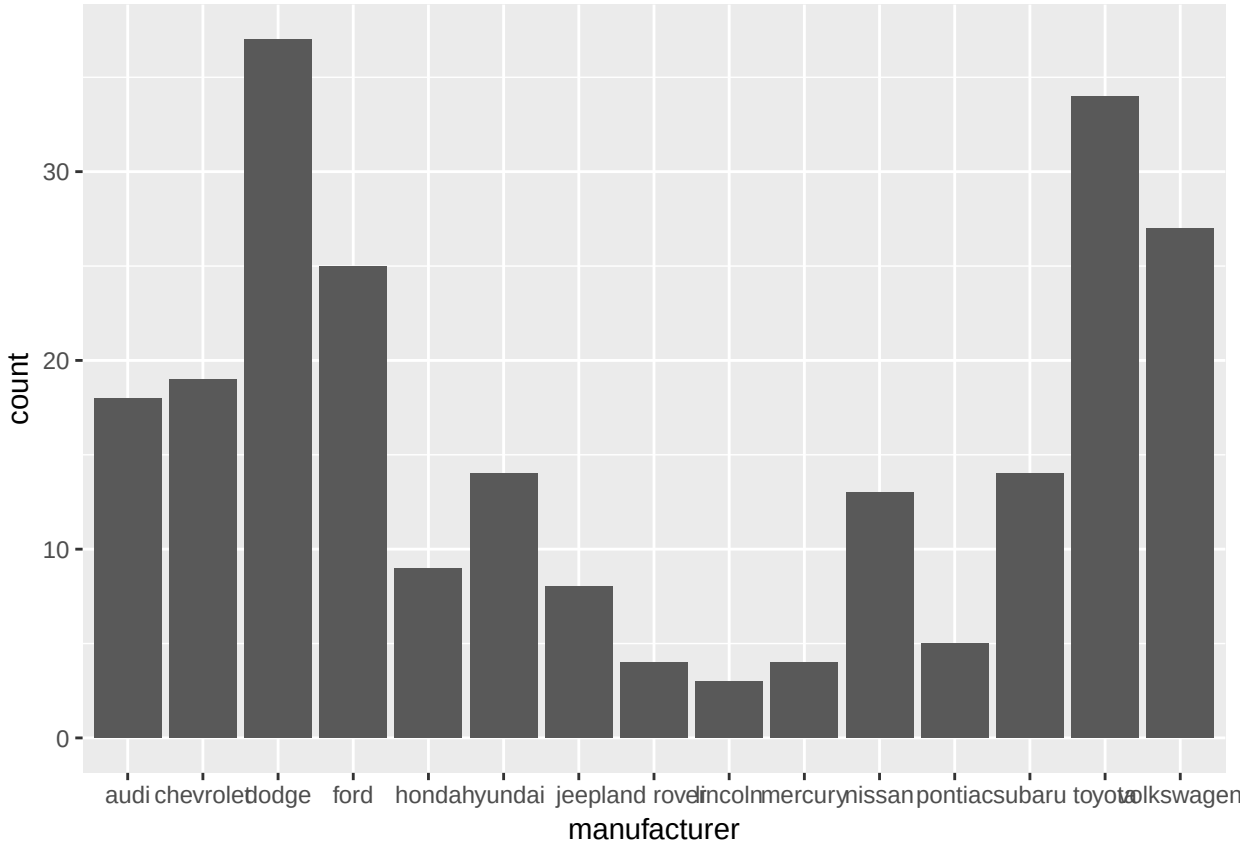



```
ggplot(mpg, aes(cty)) + geom_freqpoly(bins = 10)
```



Bar chart counts the occurrencies of a discrete variable:

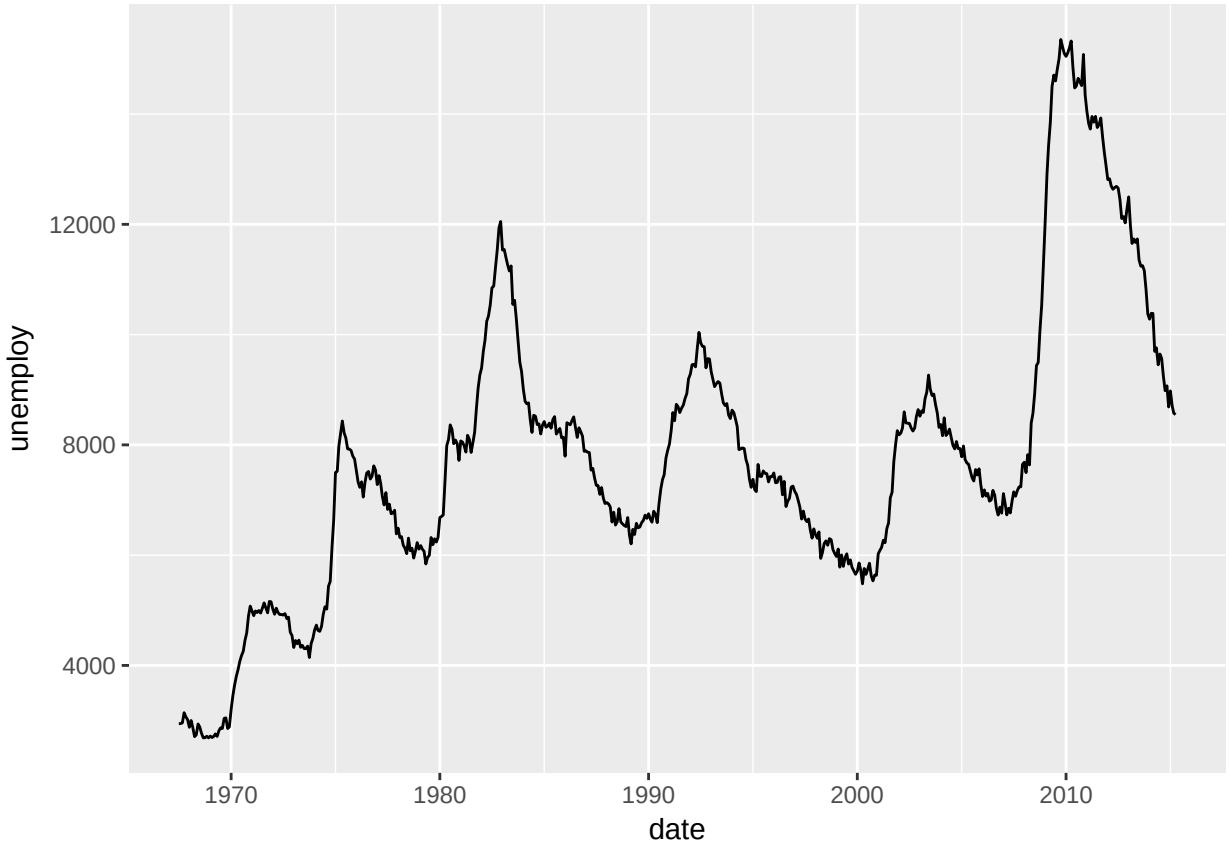
```
ggplot(mpg, aes(manufacturer)) + geom_bar()
```



Exercise: How does the price distribution in `diamonds` vary by cut?

Time series data is usually plotted using line plots to join the consecutive observations. Usually, we plot the time on the X axis:

```
ggplot(economics, aes(date, unemploy)) + geom_line()
```

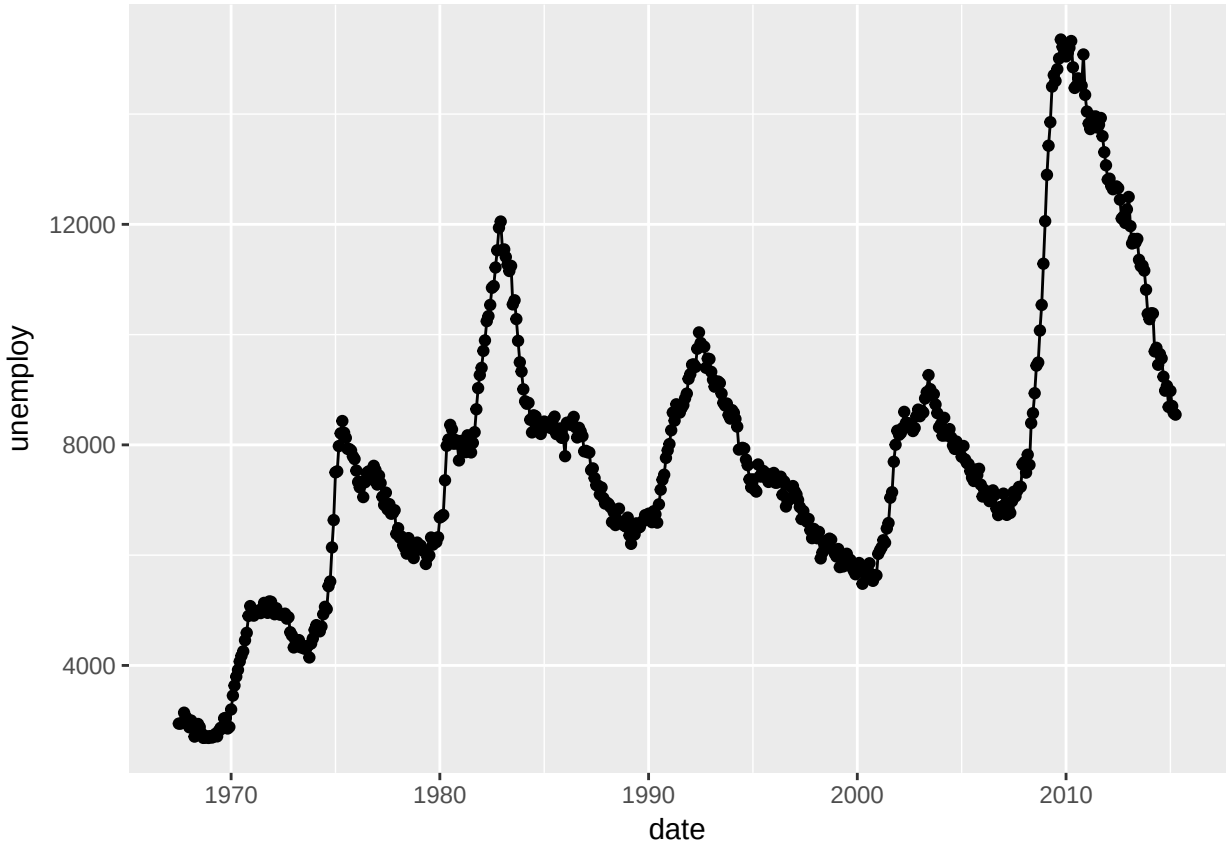


(See the help page for the `economics` dataset for the description of variables.)

Adding more information to the plot +

Notice how we add more information onto the plot using the + operator. Information is added in the order presented. This means we can layer multiple geoms one on top of the other. For instance:

```
ggplot(economics, aes(date, unemploy)) +
 geom_line() +
 geom_point()
```

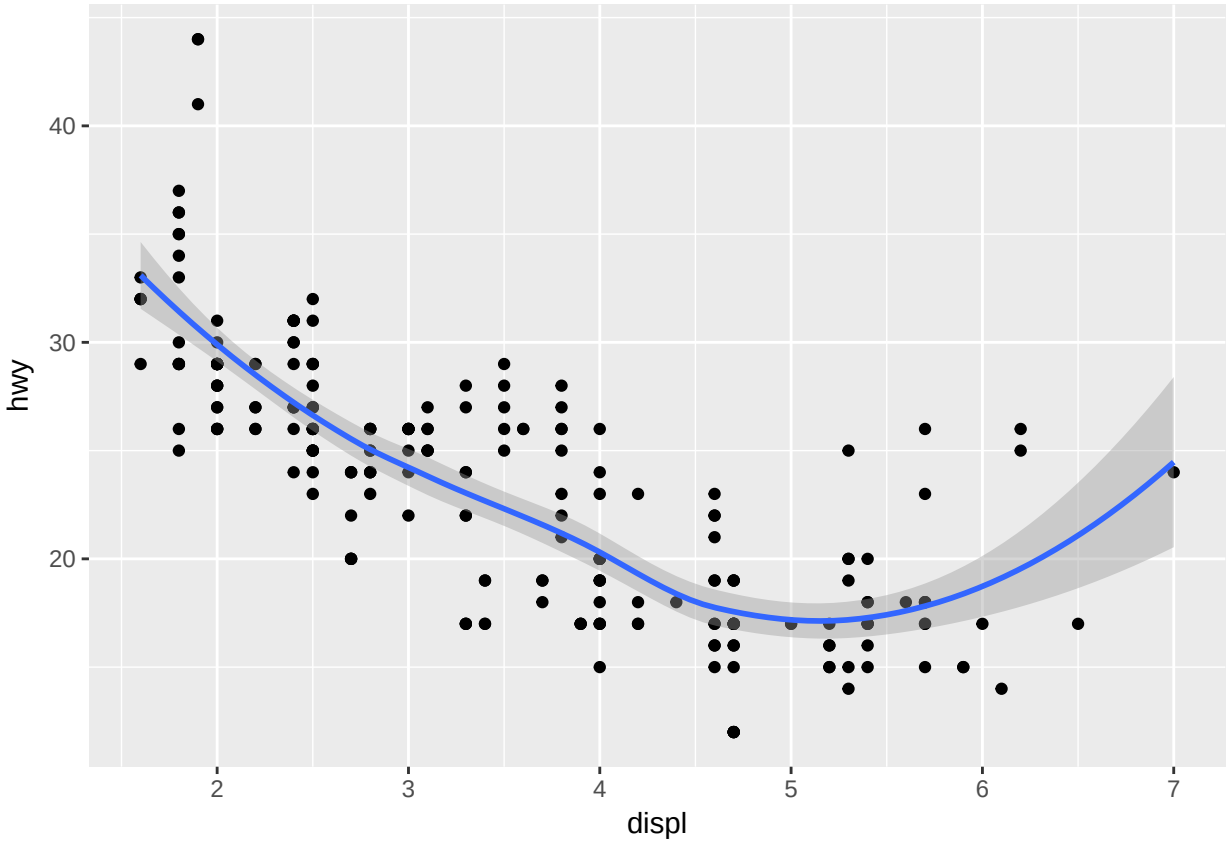


Adding a smoother

A "smoother" is a line overlaid on the plot that can help bring out the dominant pattern by leaving out some of the variability ("wigglyness") in the data.

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess'
```



The default method used to produce the smoothing curve is "smooth local regression" produced with the `loess` function. The confidence intervals at each point are displayed in gray shade. (You can turn those off with "`se = FALSE`".) The "wiggliness" of the smoothing line is controlled with the "`span`" parameter, which ranges from 0 to 1 (more to less wiggly).

There are different smoothing methods available. The default will switch from `loess` to `gam` when there are many data points, but there is also `lm`, which fits a least-squares linear regression, and `rlm`, which uses a robust fitting algorithm. (Load the "MASS" library first to use `rlm`.)

Exercise: Experiment with different smoothing methods and parameters

More about the aesthetic

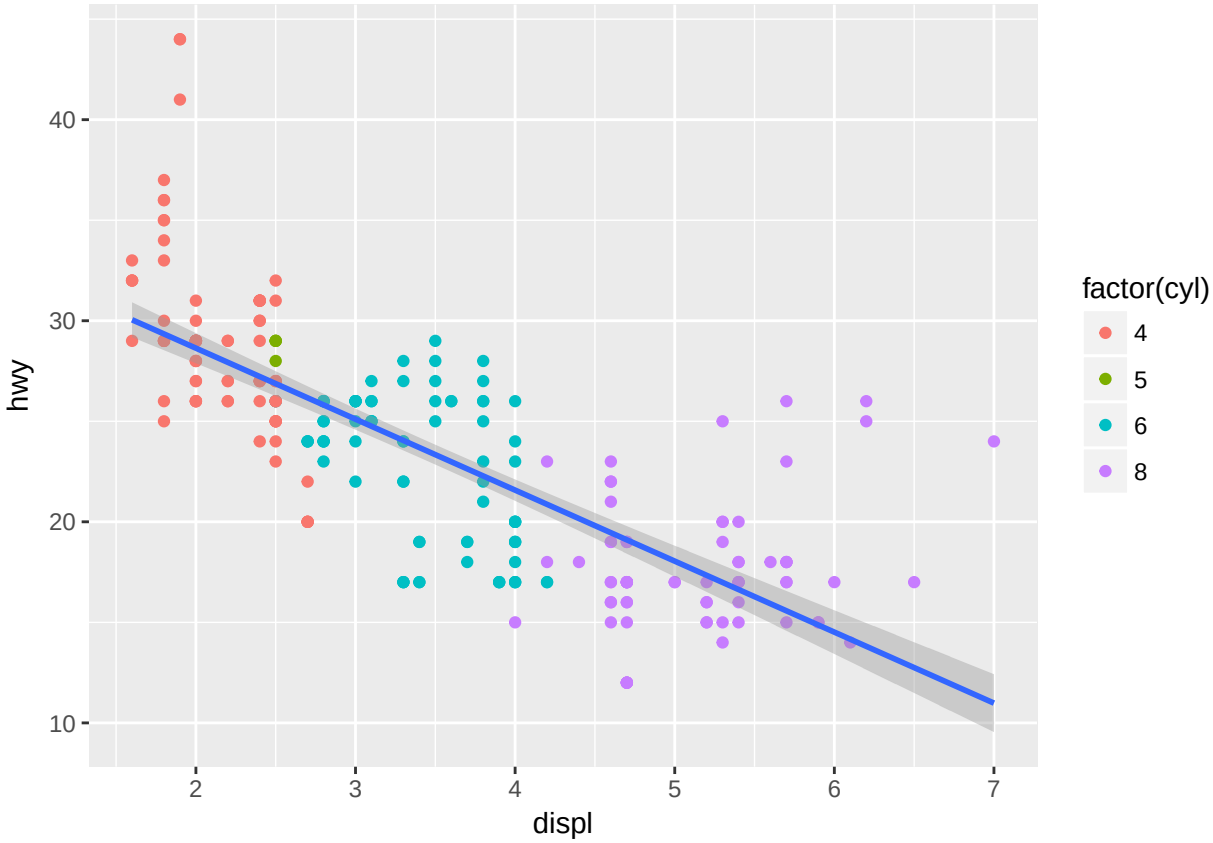
Usually we will specify the data and at least part of the aesthetic in the ggplot command. However we can still change the values in other layers.

```
p2 <- ggplot(mpg,aes(displ,hwy)) +
  geom_point(aes(colour=factor(cyl))) +
  geom_smooth(aes(y = cty), se=FALSE)
```

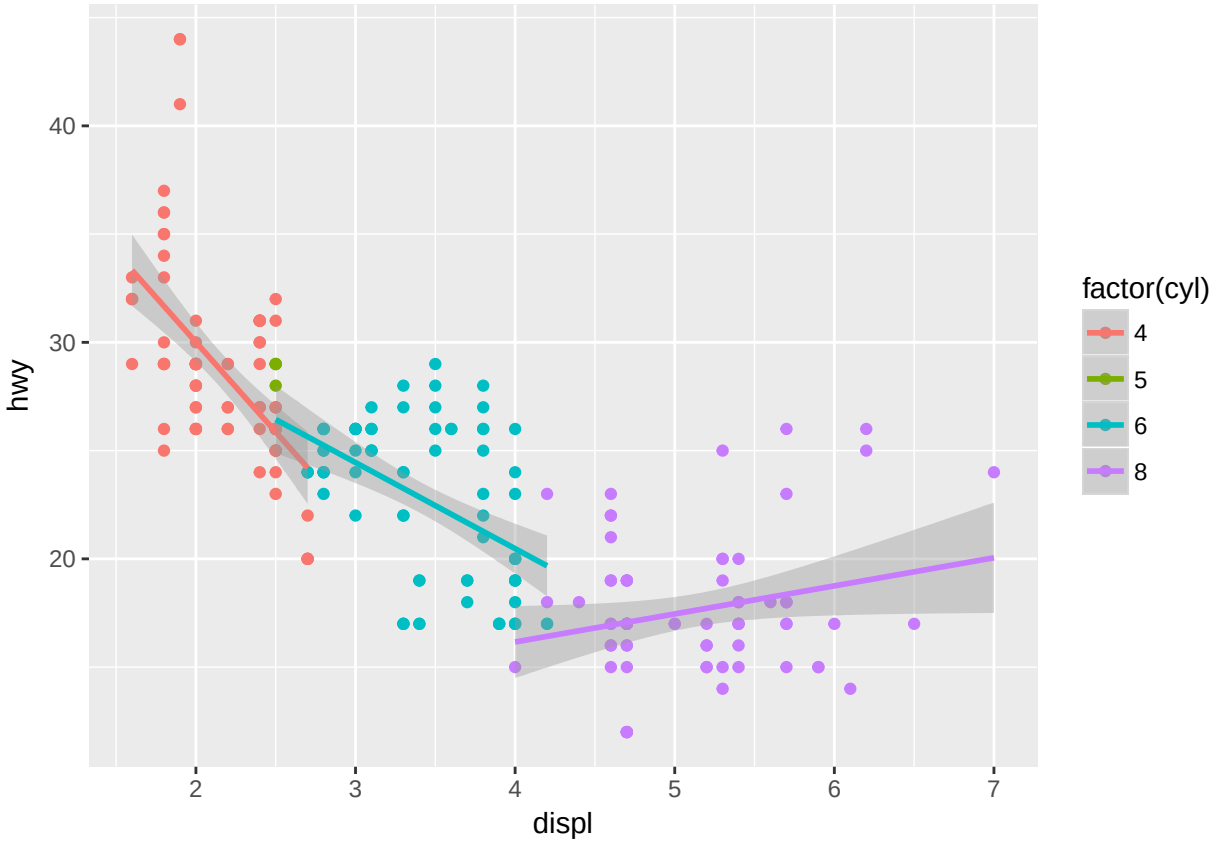
Notice the plot automatically adjusted the axes of the plot to account for the differences in the aesthetic in each layer. However be careful because the labels and scales may not make sense.

If I'm going to split my plot by cyl for several layers it's better to include it in the default aesthetic. That way it is passed to each new layer.

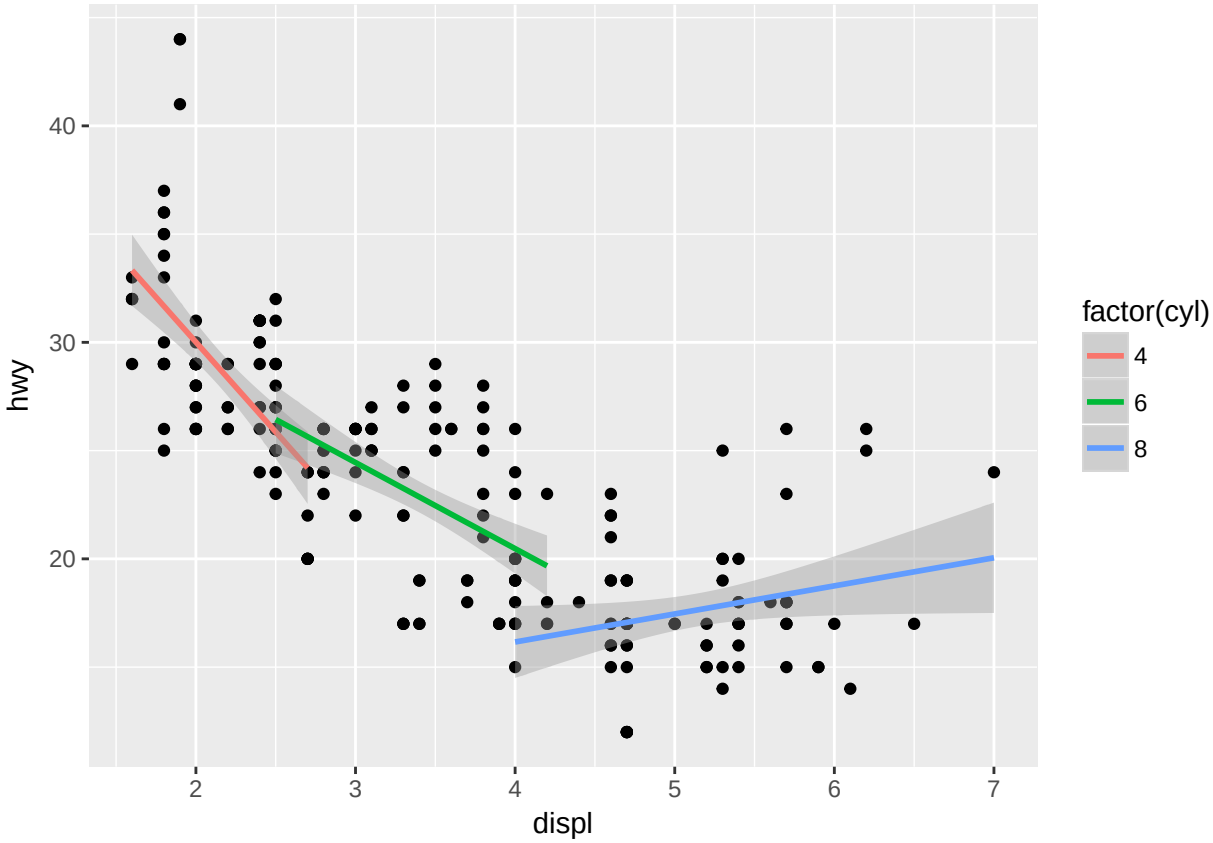
```
p3 <- ggplot(mpg,aes(displ,hwy))
p3 + geom_point(aes(colour=factor(cyl))) + geom_smooth(method="lm")
```




```
p3 + geom_point(aes(colour=factor(cyl))) + geom_smooth(aes(colour=factor(cyl)),method="lm")
```

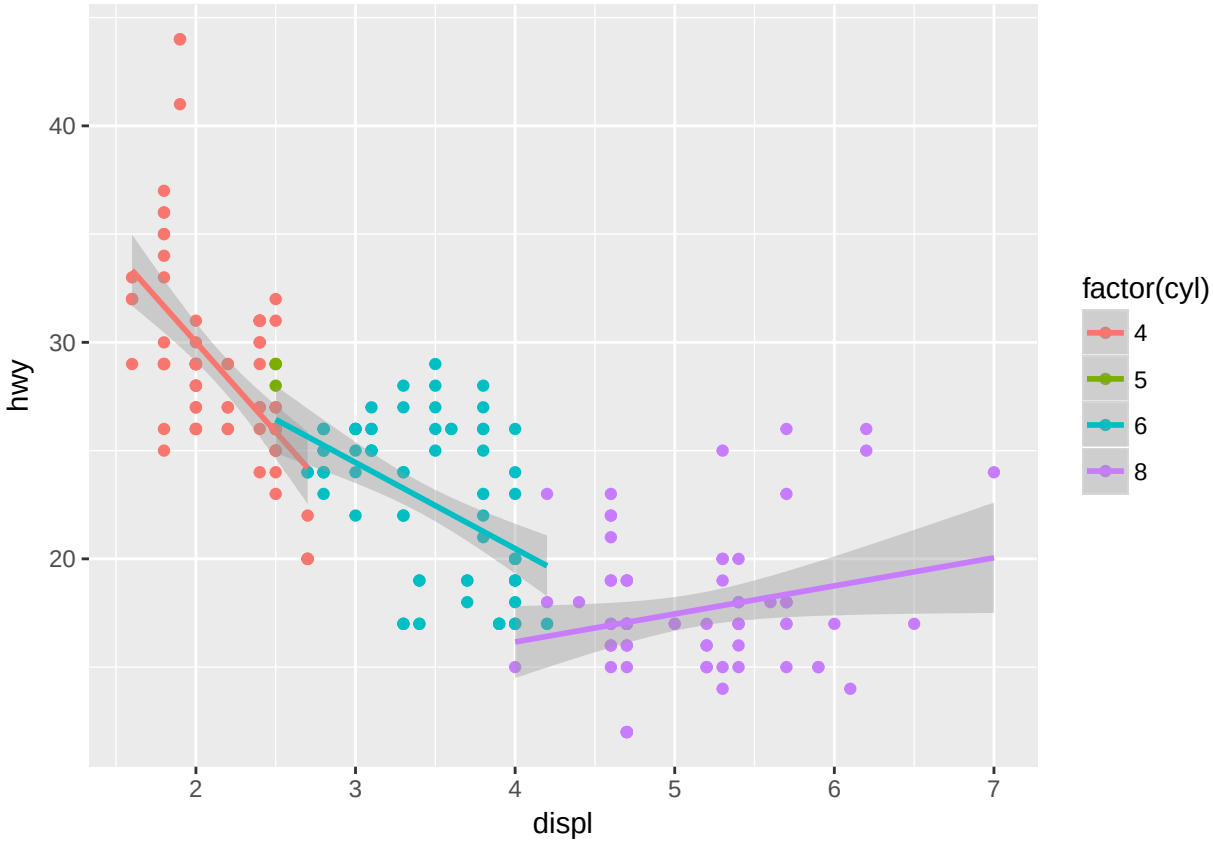



```
p3 + geom_point() + geom_smooth(aes(colour=factor(cyl)),method="lm")
```



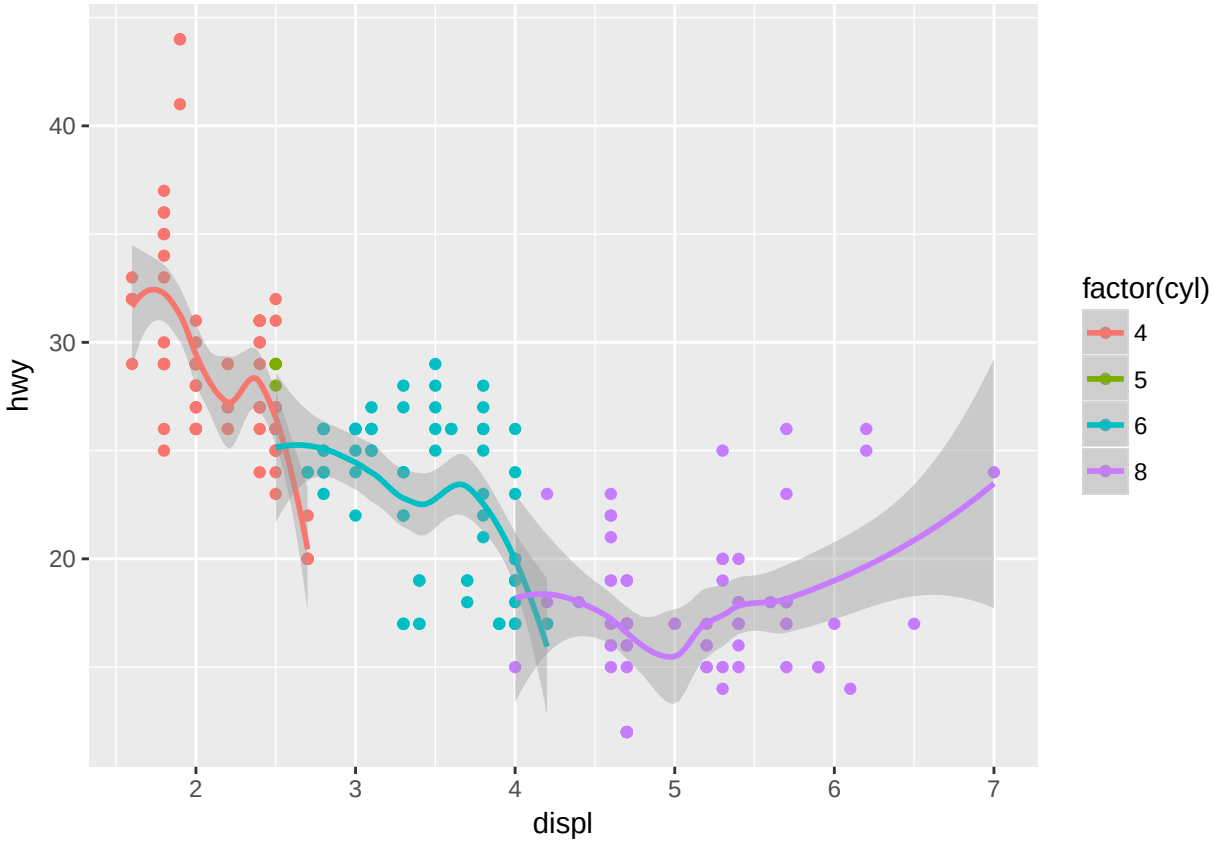
Compared to

```
p3 <- ggplot(mpg,aes(displ,hwy,color=factor(cyl)))
p3 + geom_point() + geom_smooth(method="lm")
```



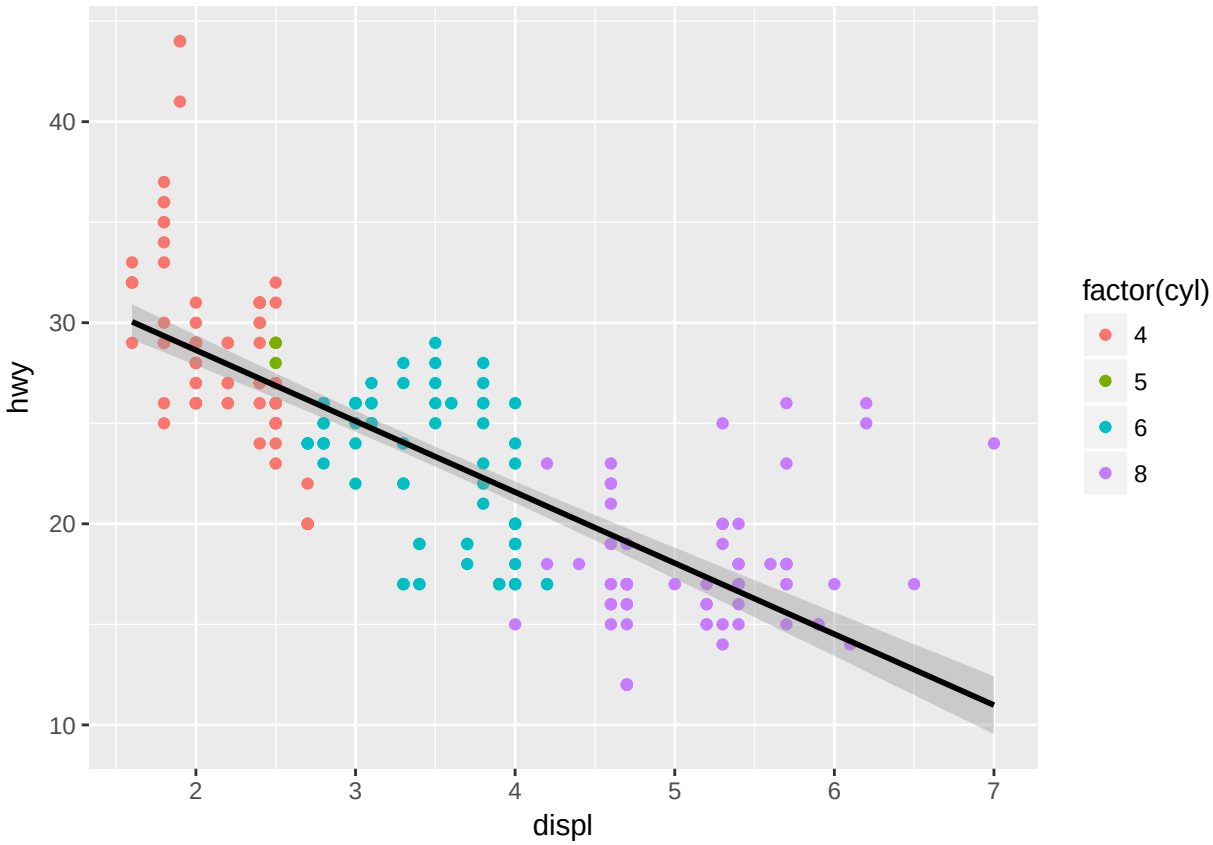

```
p4 <- ggplot(mpg,aes(displ,hwy,color=factor(cyl)))
p4 + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess'
```



You can override the default aesthetic for any layer.

```
p3 + geom_point() + geom_smooth(method="lm",color="black")
```


hwy
40
30
20
10
2
3
4
5
6
7
displ
factor(cyl)
4
5
6
8