ggplot2 basics

Hadley Wickham
Assistant Professor / Dobelman Family Junior Chair
Department of Statistics / Rice University

September 2011

1. Diving in: scatterplots & aesthetics
2. Facetting
3. Geoms
4. Histograms and barcharts
5. Scatterplots for large data

Diving in

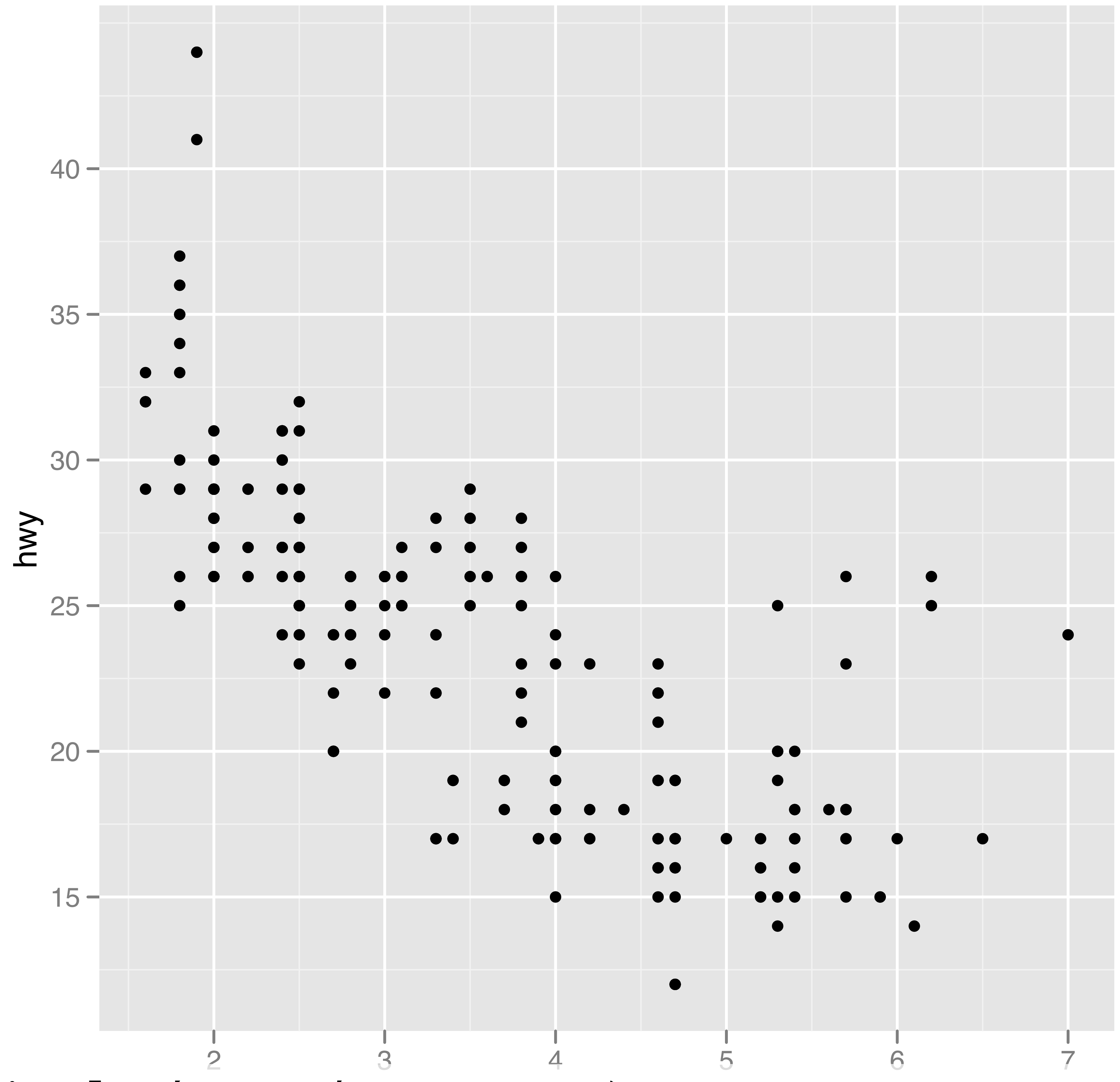
Scatterplot basics

```
install.packages(”ggplot2”)
library(ggplot2)

?mpg
head(mpg)
str(mpg)
summary(mpg)

qplot(displ, hwy, data = mpg)
```

Scatterplot basics

```
install.packages("ggplot2")
library(ggplot2)

?mpg
head(mpg)
str(mpg)
summary(mpg)

qplot(displ, hwy, data = mpg)
```

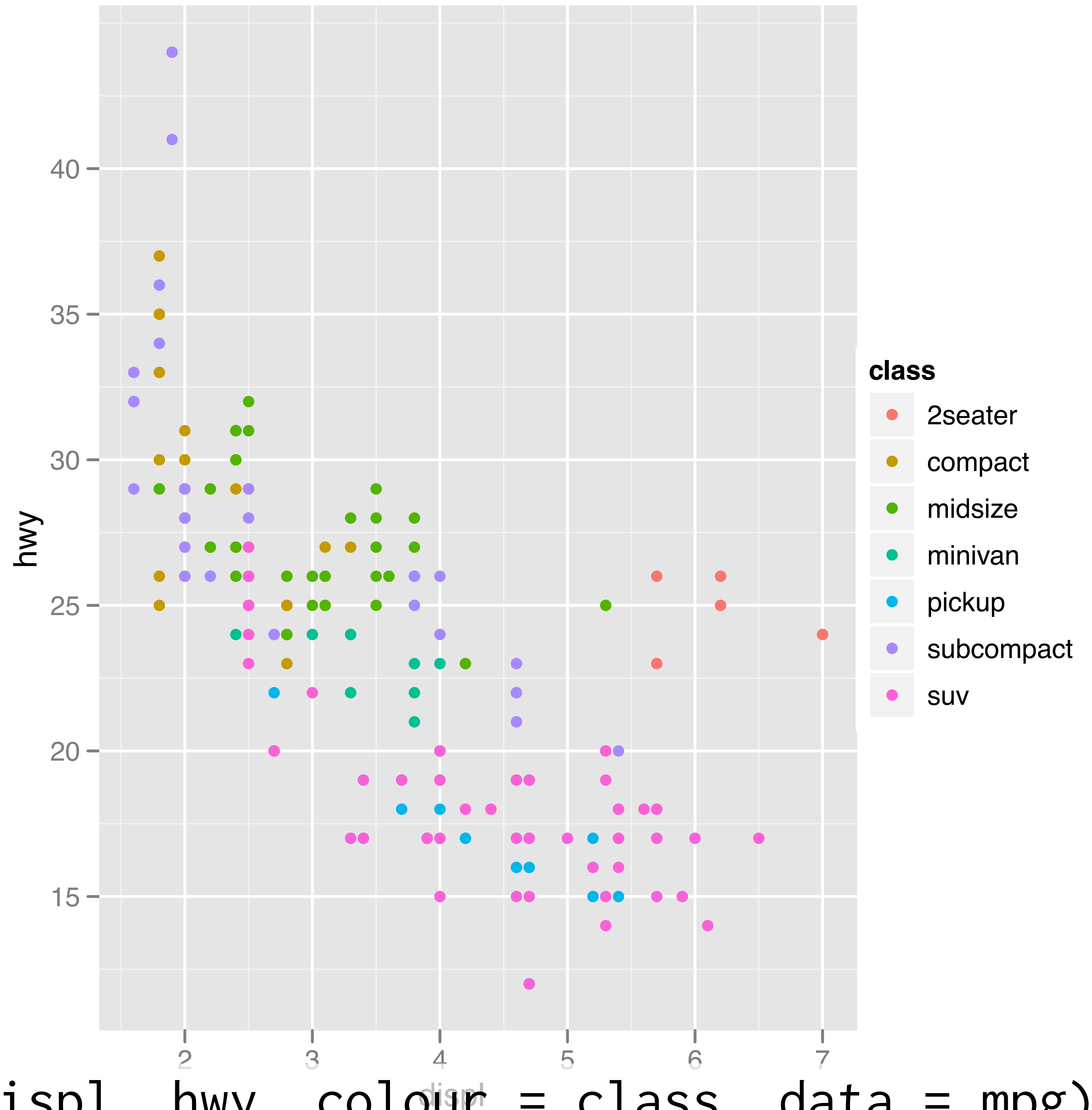
Always explicitly specify the data



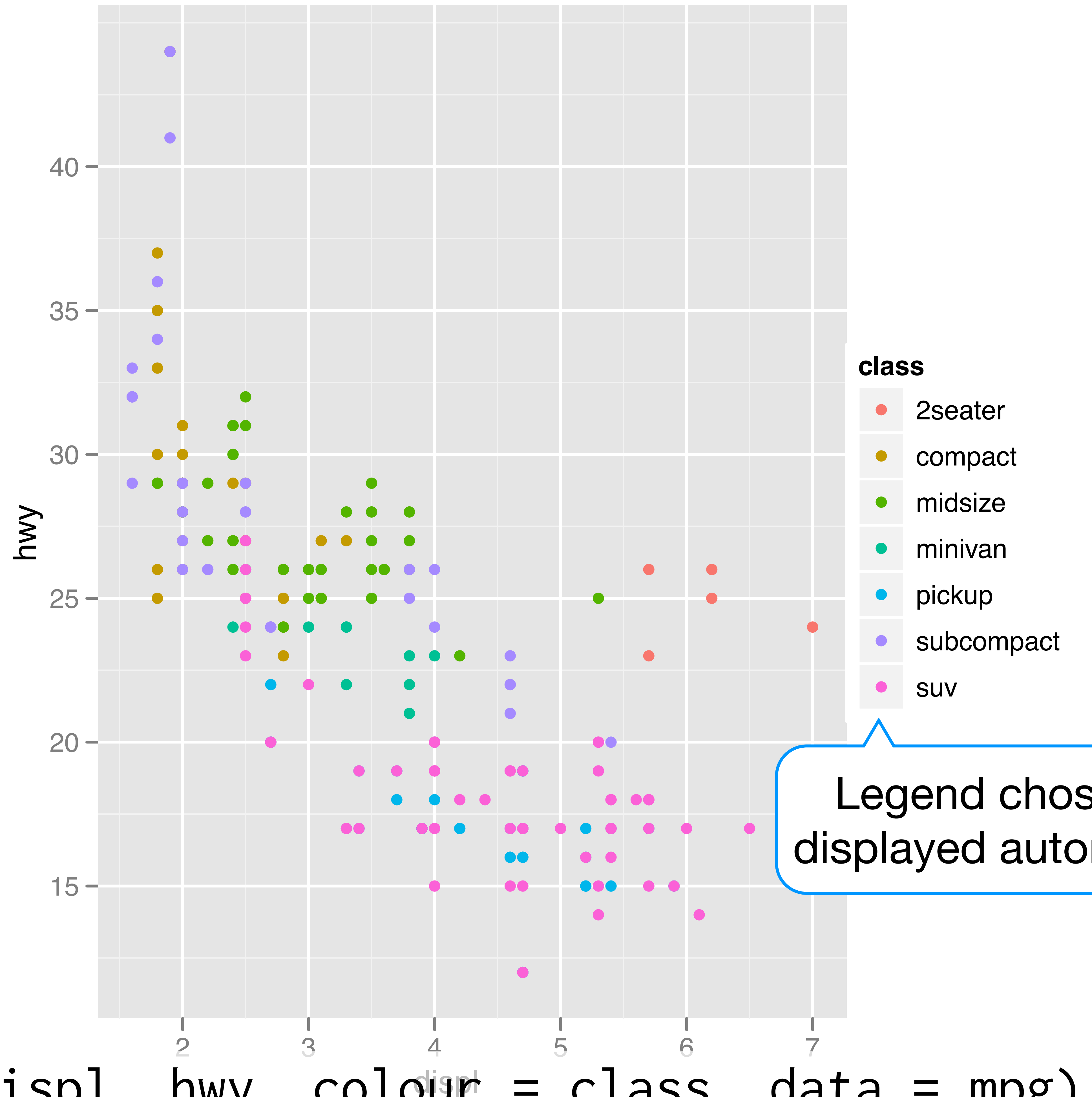
```
qplot(displ, hwy, data = mpg)
```

Additional variables

Can display additional variables with **aesthetics** (like shape, colour, size) or **facetting** (small multiples displaying different subsets)



```
qplot(displ, hwy, colour = class, data = mpg)
```



Legend chosen and displayed automatically.

```
qplot(displ, hwy, colour = class, data = mpg)
```

Your turn

Experiment with colour, size, and shape aesthetics.

What's the difference between discrete or continuous variables?

What happens when you combine multiple aesthetics?

	Discrete	Continuous
Colour	Rainbow of colours	Gradient from red to blue
Size	Discrete size steps	Linear mapping between radius and value
Shape	Different shape for each	Shouldn’t work

Facetting

Faceting

Small multiples displaying different subsets of the data.

Useful for exploring conditional relationships. Useful for large data.

Your turn

```
qplot(displ, hwy, data = mpg) +
facet_grid(. ~ cyl)

qplot(displ, hwy, data = mpg) +
facet_grid(drv ~ .)

qplot(displ, hwy, data = mpg) +
facet_grid(drv ~ cyl)

qplot(displ, hwy, data = mpg) +
facet_wrap(~ class)
```

Summary

`facet_grid()`: 2d grid, rows ~ cols, . for no split

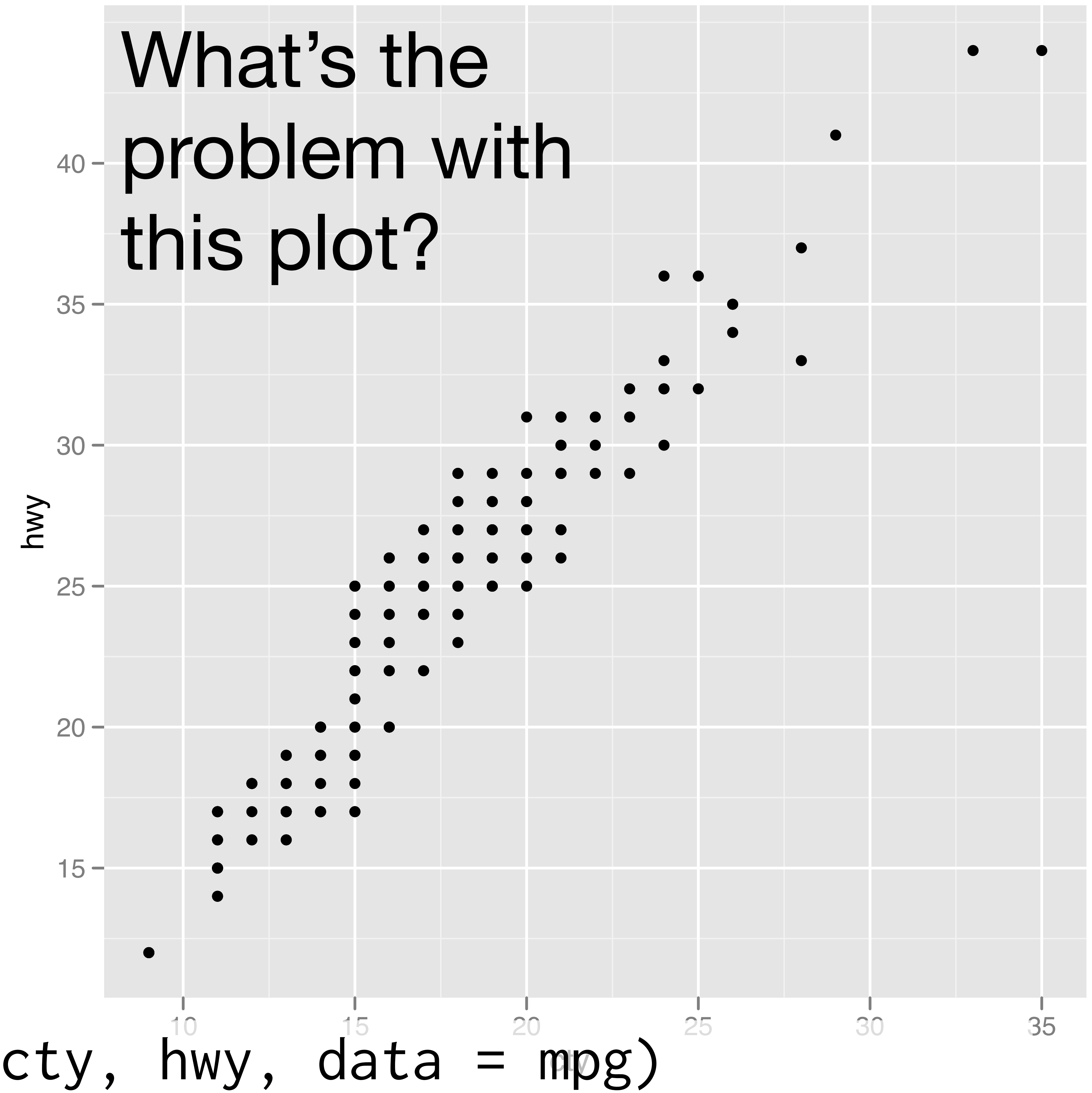
`facet_wrap()`: 1d ribbon wrapped into 2d

Aside: workflow

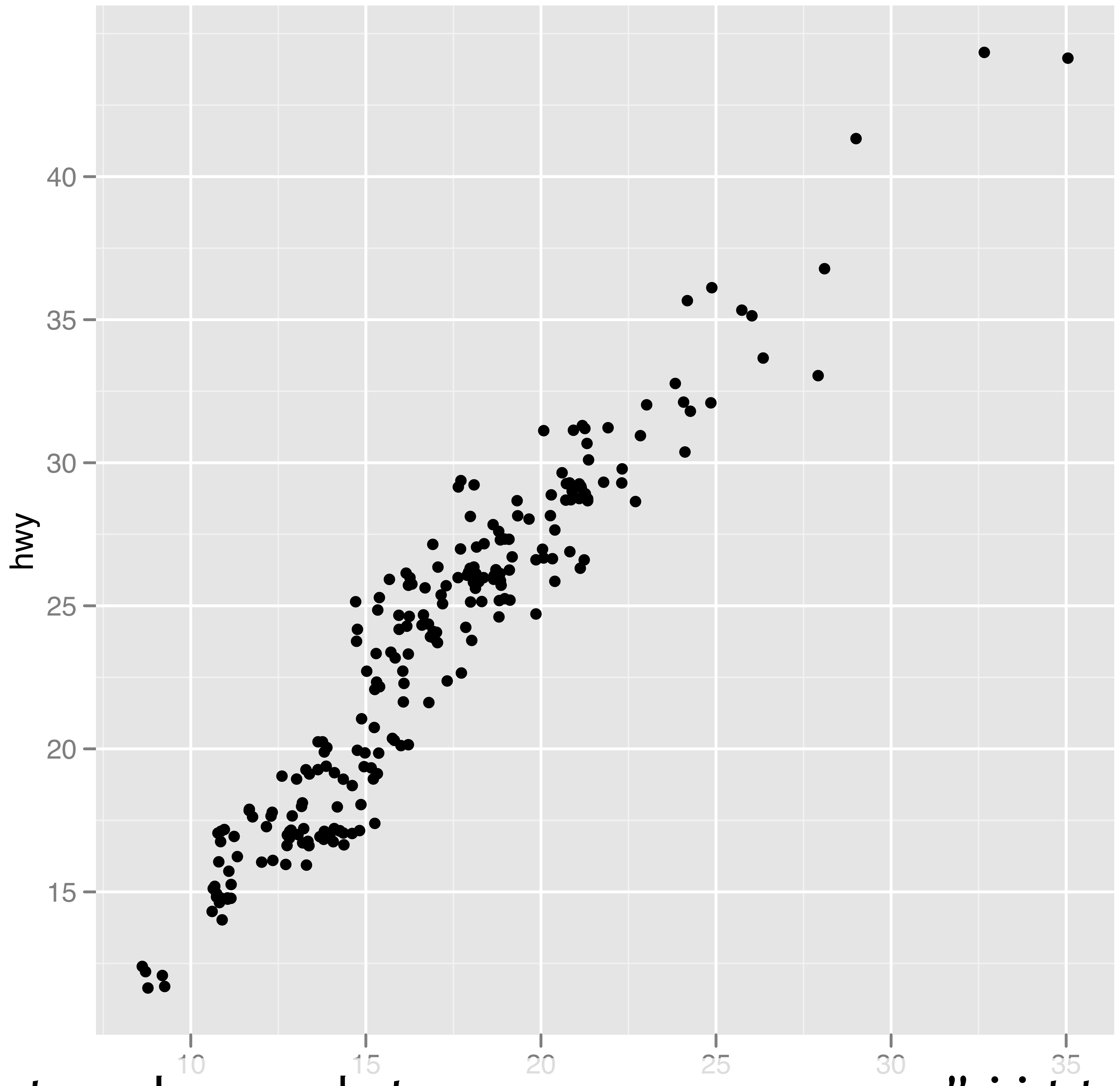
Keep a copy of the slides open so that you can copy and paste the code.

Geoms

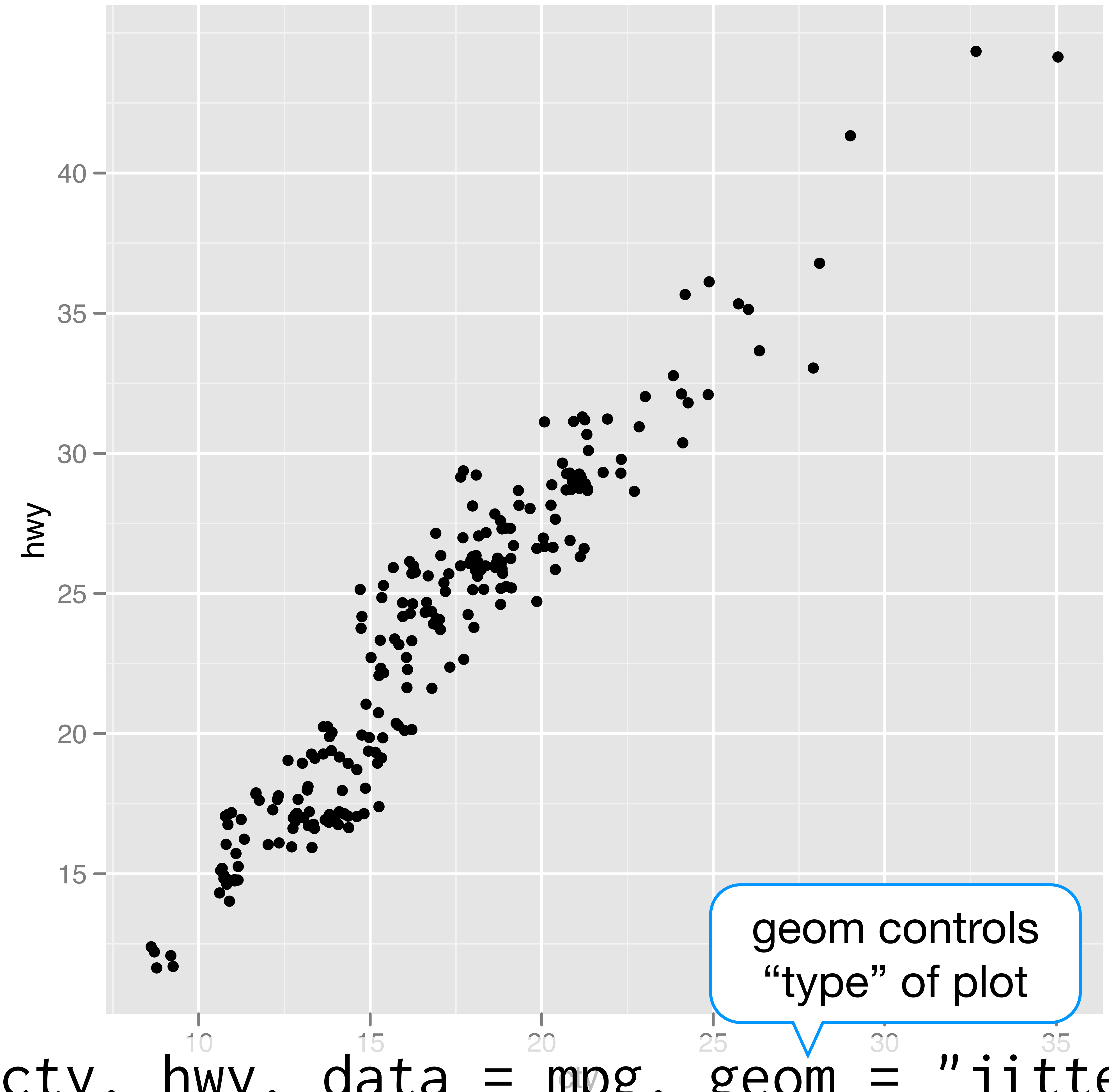
What's the problem with this plot?



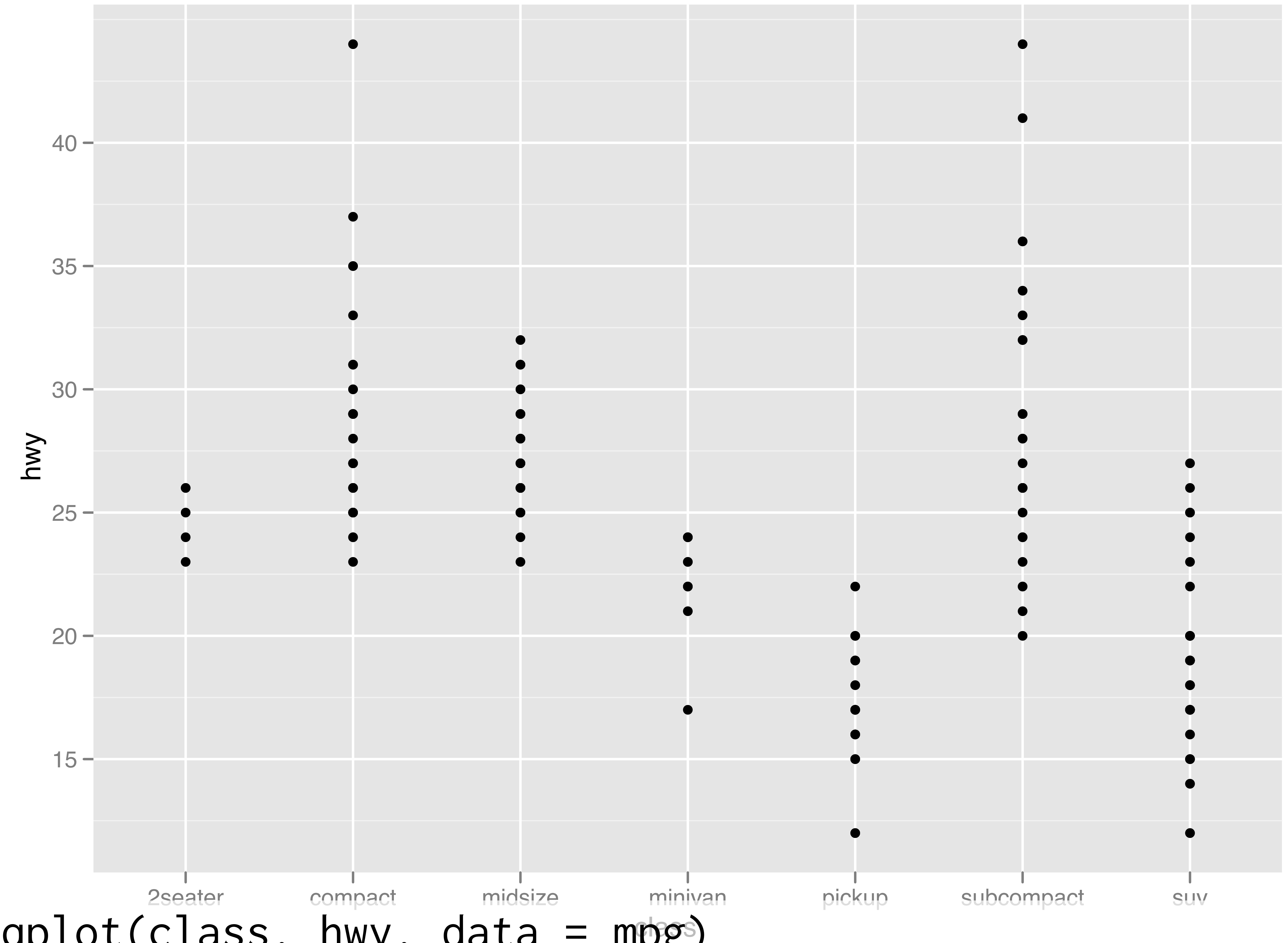
```
qplot(cty, hwy, data = mpg)
```



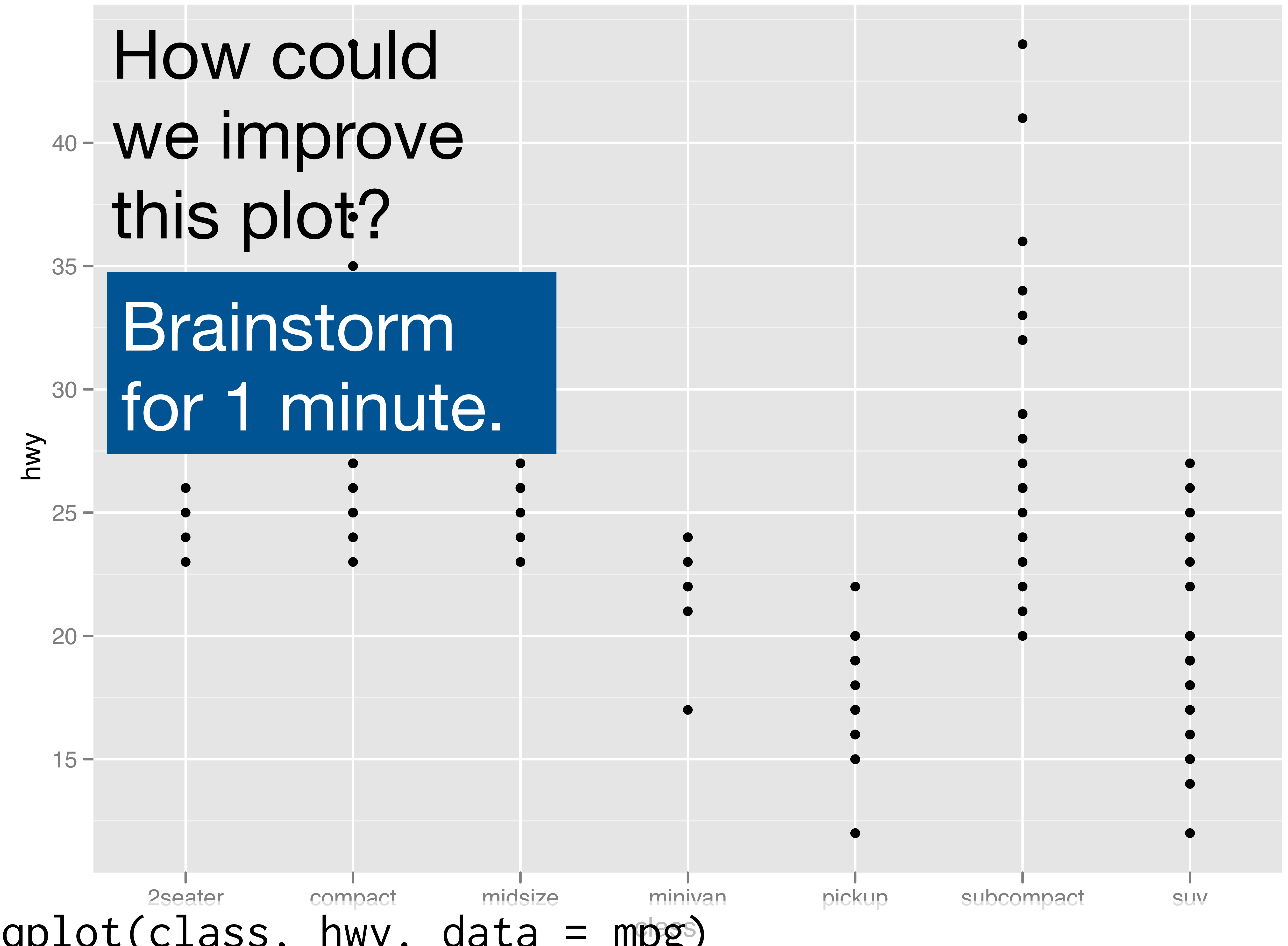
```
qplot(cty, hwy, data = mpg, geom = "jitter")
```



```
qplot(cty, hwy, data = mpg, geom = "jitter")
```



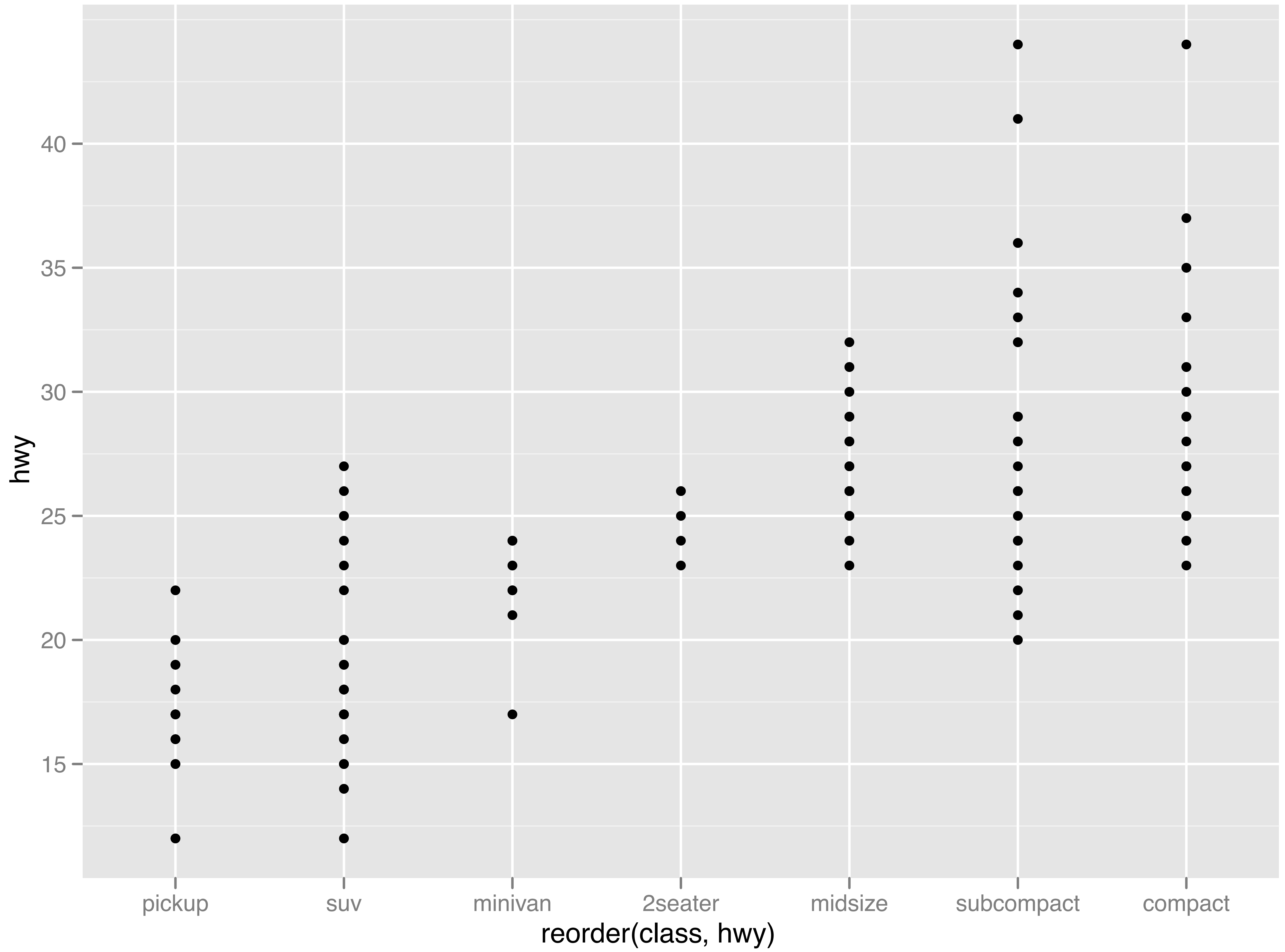
```
qplot(class, hwy, data = mpg)
```



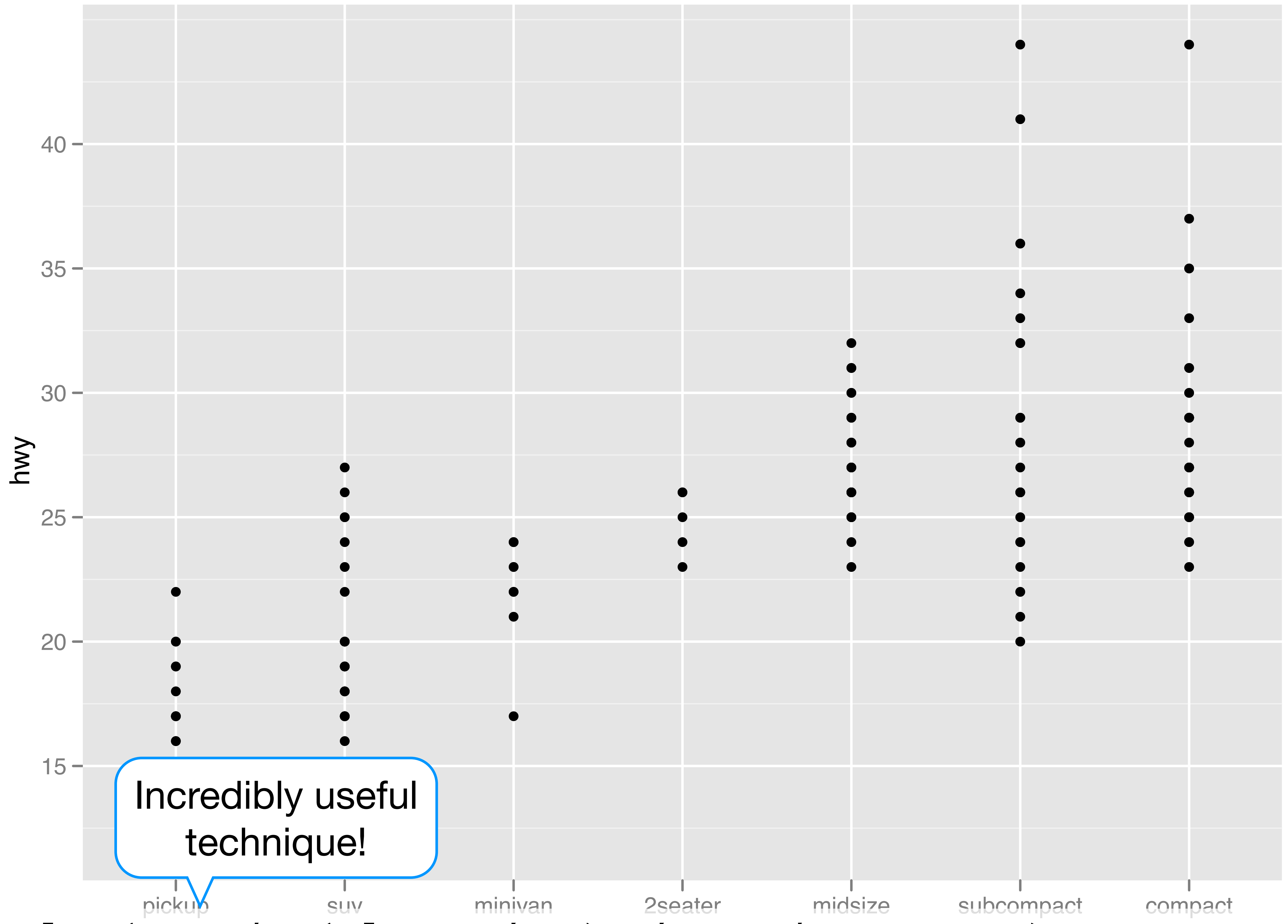
How could we improve this plot?

Brainstorm for 1 minute.

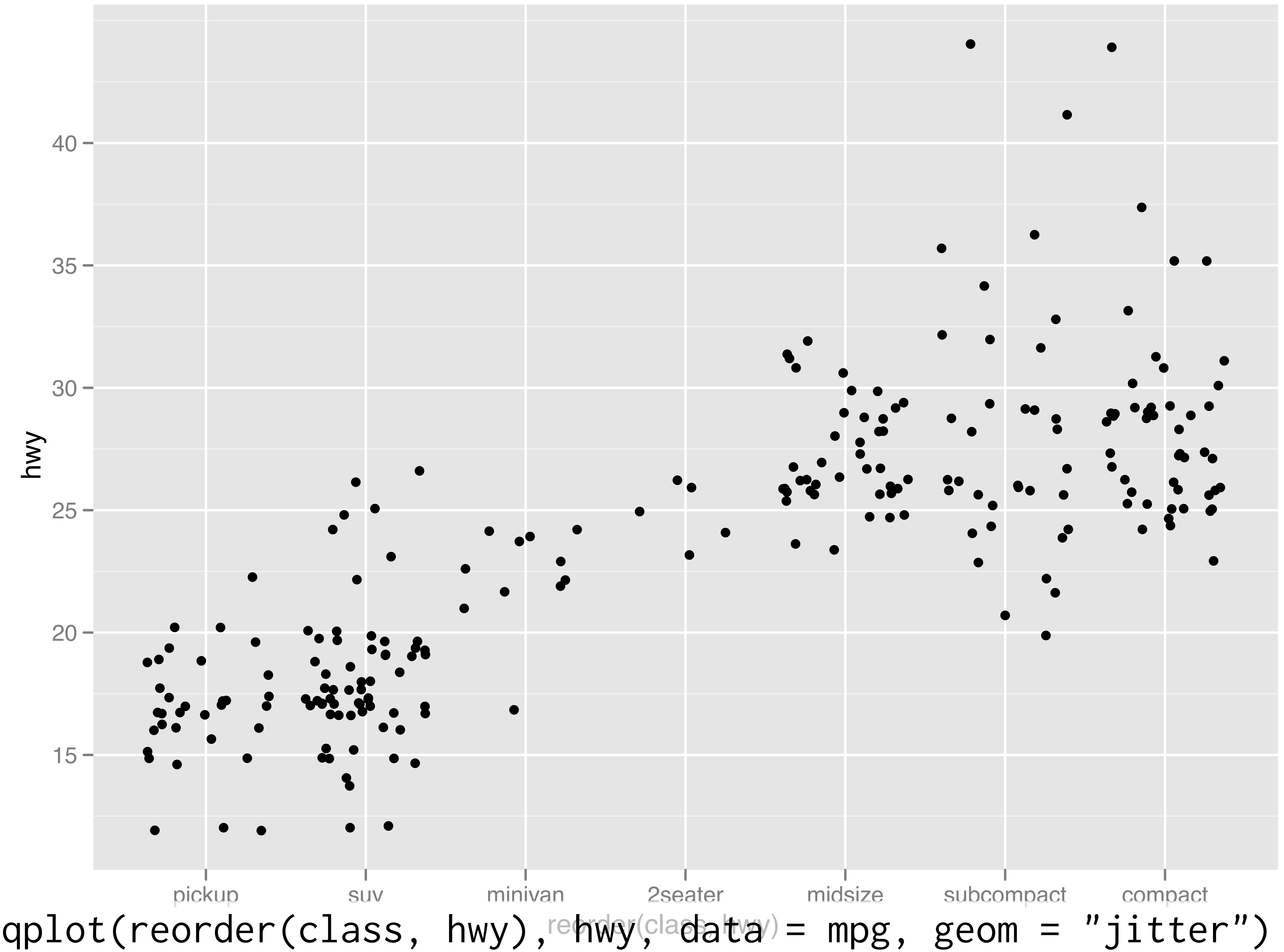
```
qplot(class, hwy, data = mpg)
```



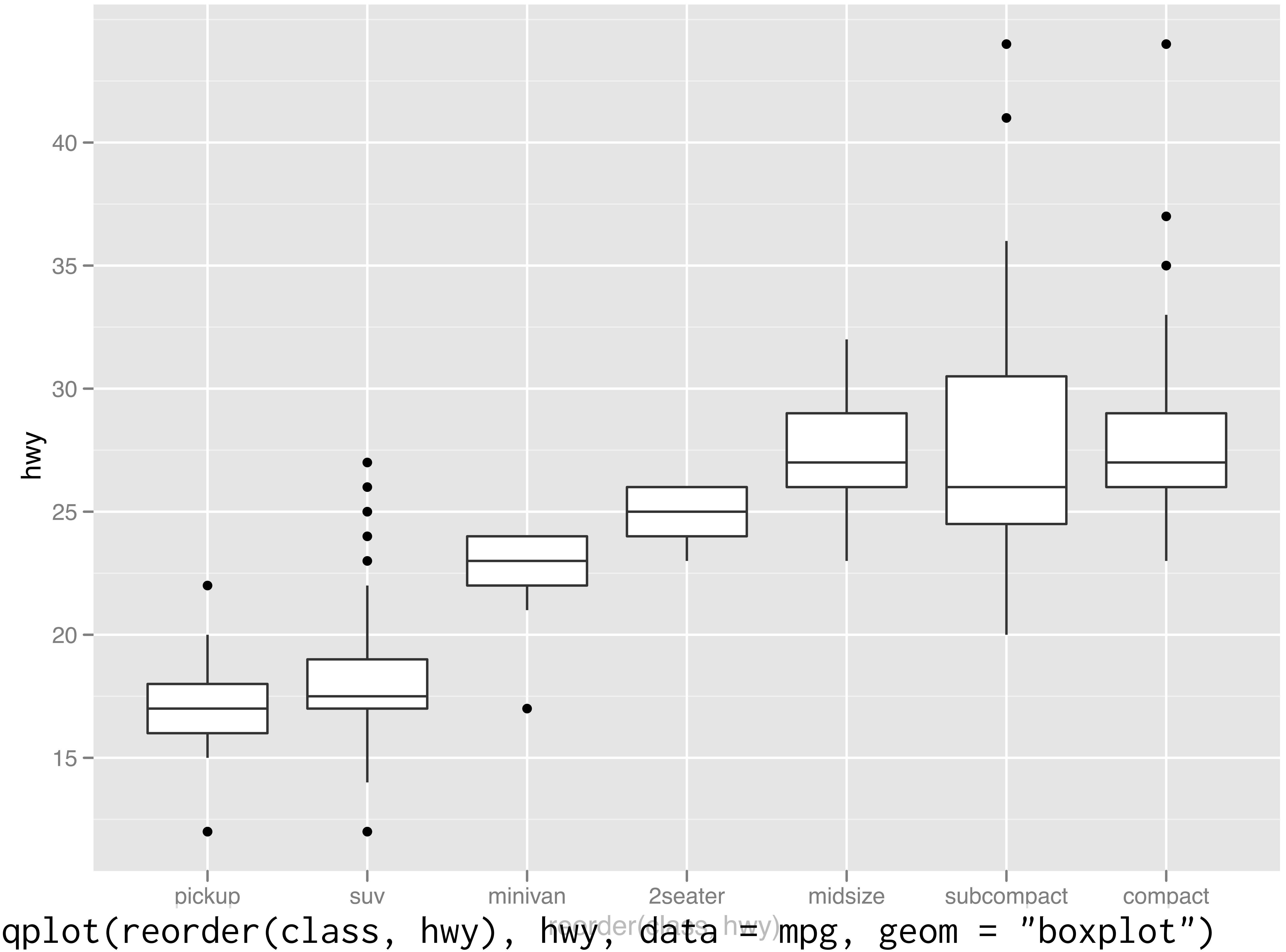
hwy
40
35
30
25
20
15
pickup
suv
minivan
2seater
midsize
subcompact
compact
reorder(class, hwy)



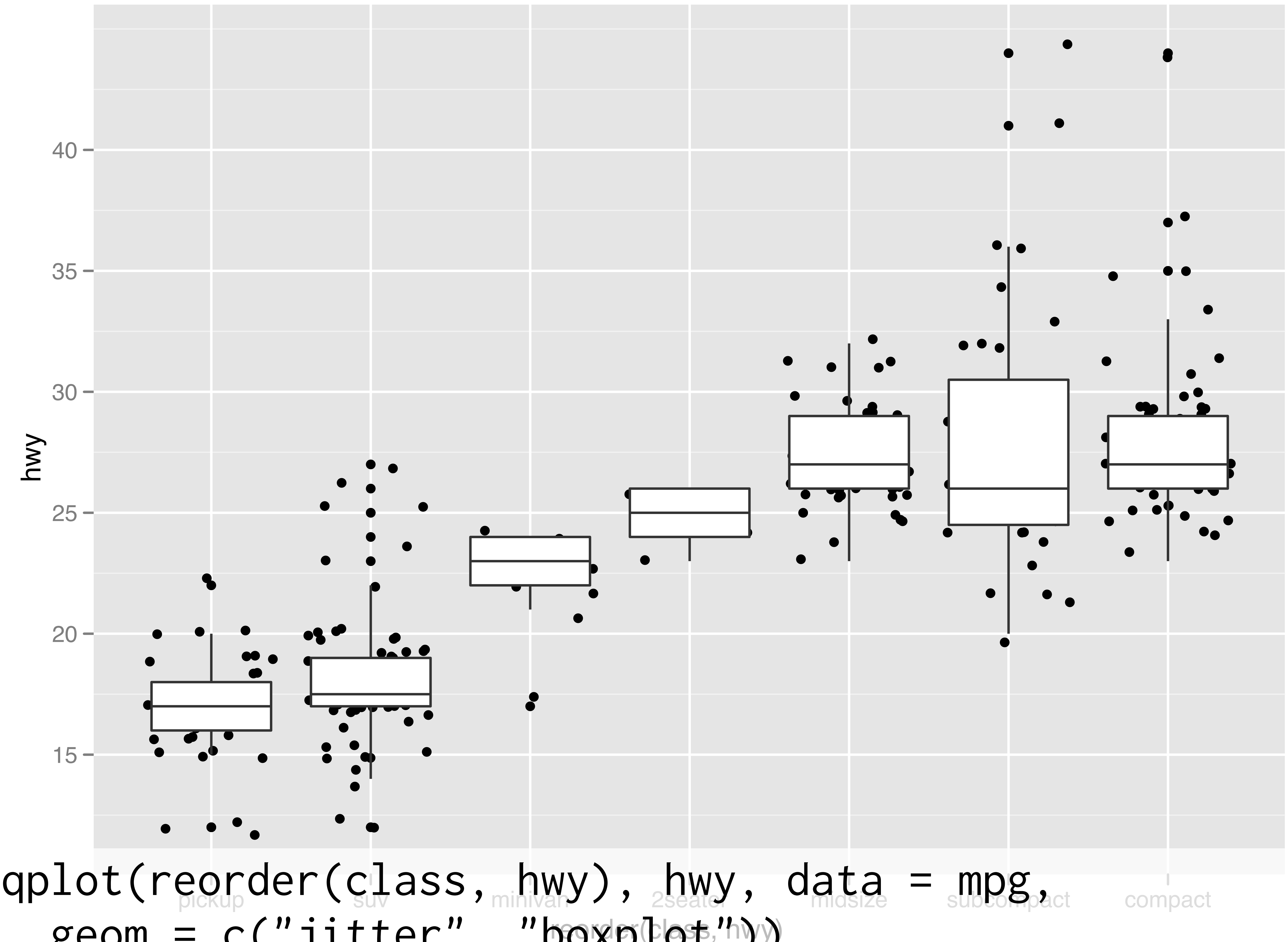
```
qplot(reorder(class, hwy), hwy, data = mpg)
```



```
qplot(reorder(class, hwy), hwy, data = mpg, geom = "jitter")
```



```
qplot(reorder(class, hwy), hwy, data = mpg, geom = "boxplot")
```



```
qplot(reorder(class, hwy), hwy, data = mpg,
  geom = c("jitter", "boxplot"))
```

Your turn

Read the help for reorder. Redraw the previous plots with class ordered by median hwy.

How would you put the jittered points on top of the boxplots?

Diamonds

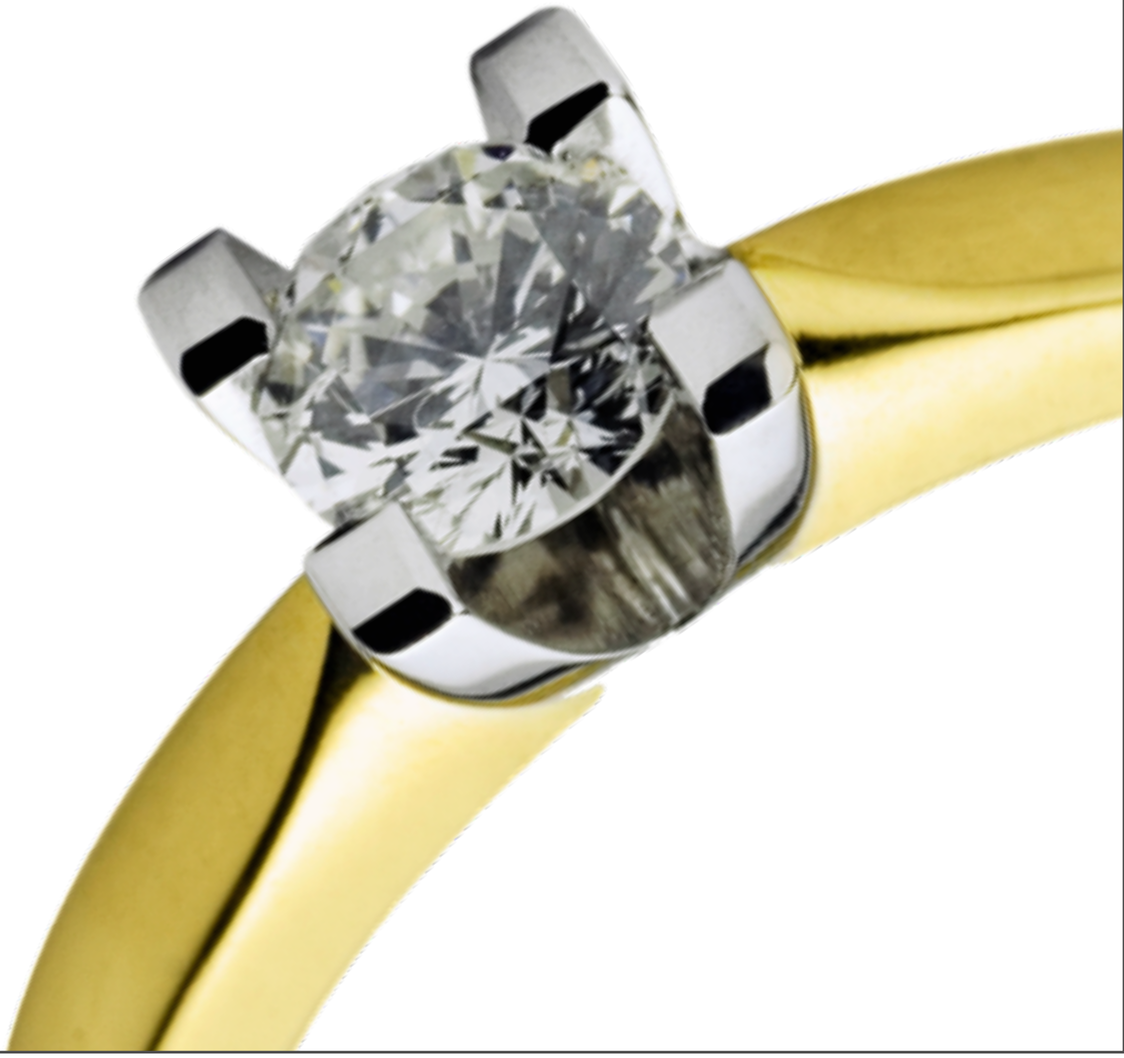
Diamonds data

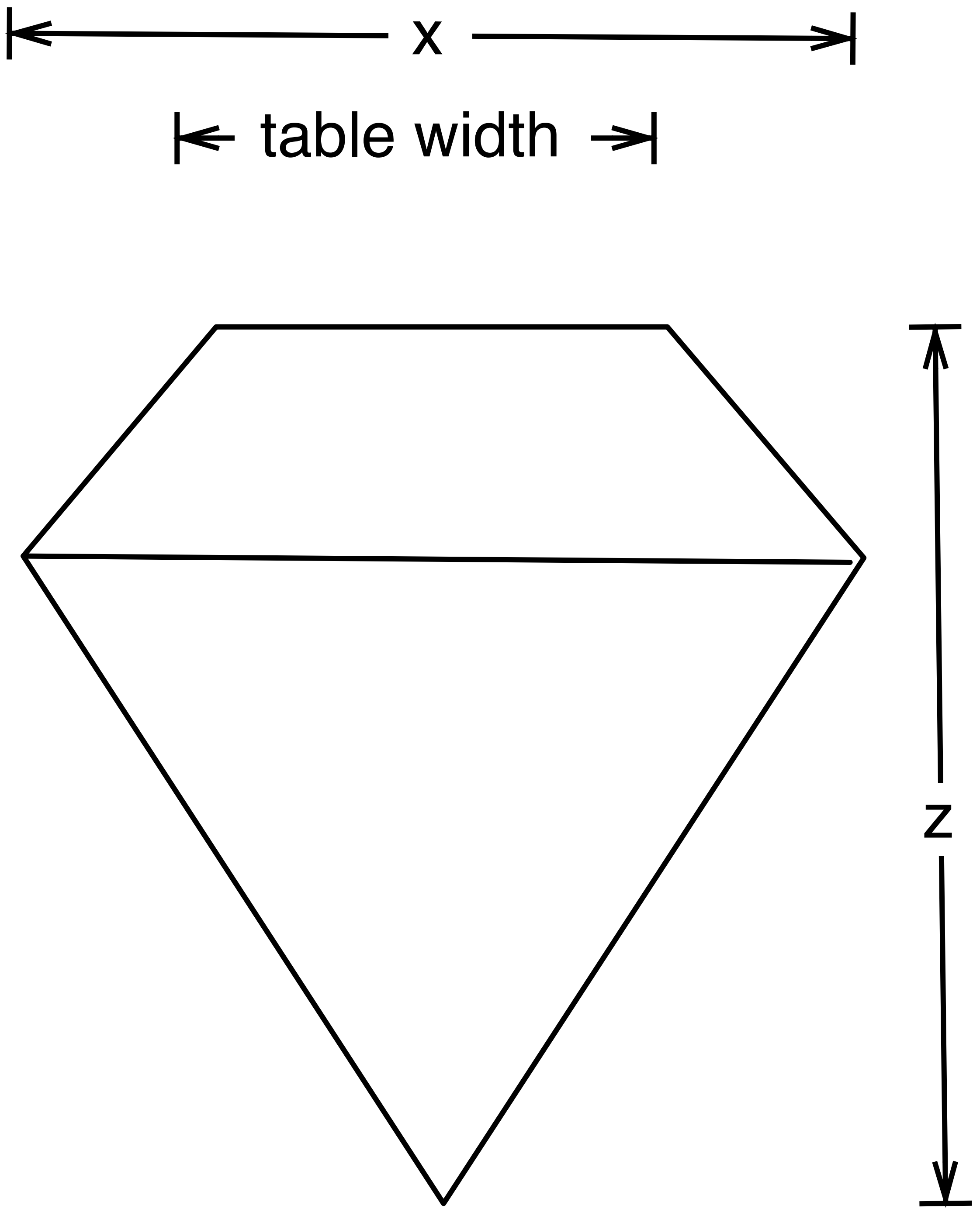
~**54,000** round diamonds from http://www.diamondse.info/

Carat, colour, clarity, cut

Total depth, table, depth, width, height

Price





depth = z / diameter
table = table width / x * 100

Histogram & bar charts

Histograms and barcharts

Used to display the **distribution** of a variable

Categorical variable → bar chart

Continuous variable → histogram

Examples

```
# With only one variable, qplot guesses that
# you want a bar chart or histogram
qplot(cut, data = diamonds)

qplot(carat, data = diamonds)
qplot(carat, data = diamonds, binwidth = 1)
qplot(carat, data = diamonds, binwidth = 0.1)
qplot(carat, data = diamonds, binwidth = 0.01)
resolution(diamonds$carat)

last_plot() + xlim(0, 3)
```

Examples

```
# With only one variable, qplot guesses that
# you want a bar chart or histogram
qplot(cut, data = diamonds)

qplot(carat, data = diamonds)
qplot(carat, data = diamonds, binwidth = 1)
qplot                     amonds, binwidth = 0.1)
qplot                     amonds, binwidth = 0.01)
resol                     rat)

last_plot() + xlim(0, 3)
```

Common ggplot2 technique: adding together plot components

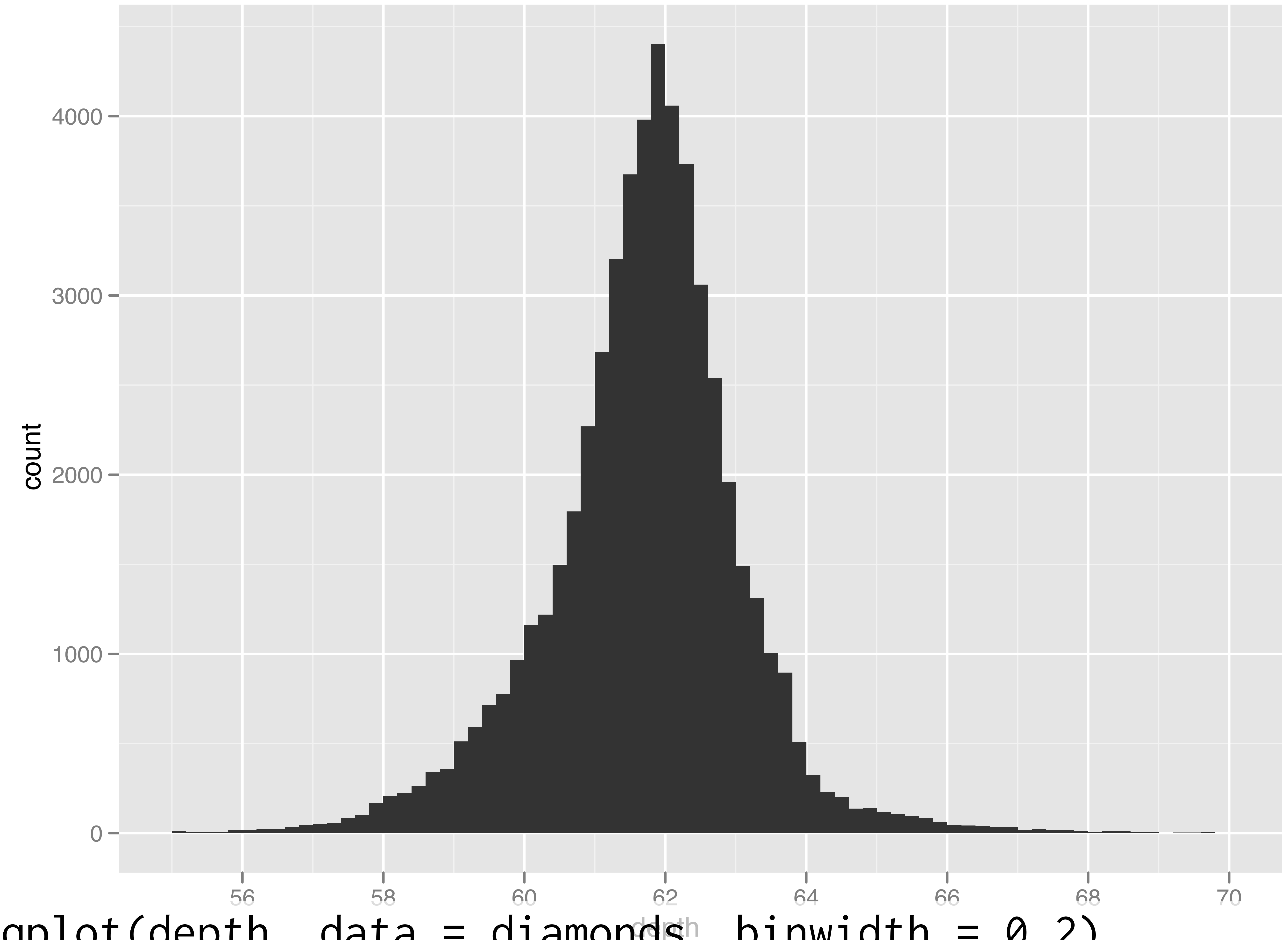
Always experiment with the bin width!

```
qplot(table, data = diamonds, binwidth = 1)

# To zoom in on a plot region use xlim() and ylim()
qplot(table, data = diamonds, binwidth = 1) +
   xlim(50, 70)
qplot(table, data = diamonds, binwidth = 0.1) +
  xlim(50, 70)
qplot(table, data = diamonds, binwidth = 0.1) +
  xlim(50, 70) + ylim(0, 50)

# Note that this type of zooming discards data
outside of the plot regions
# See coord_cartesian() for an alternative
```

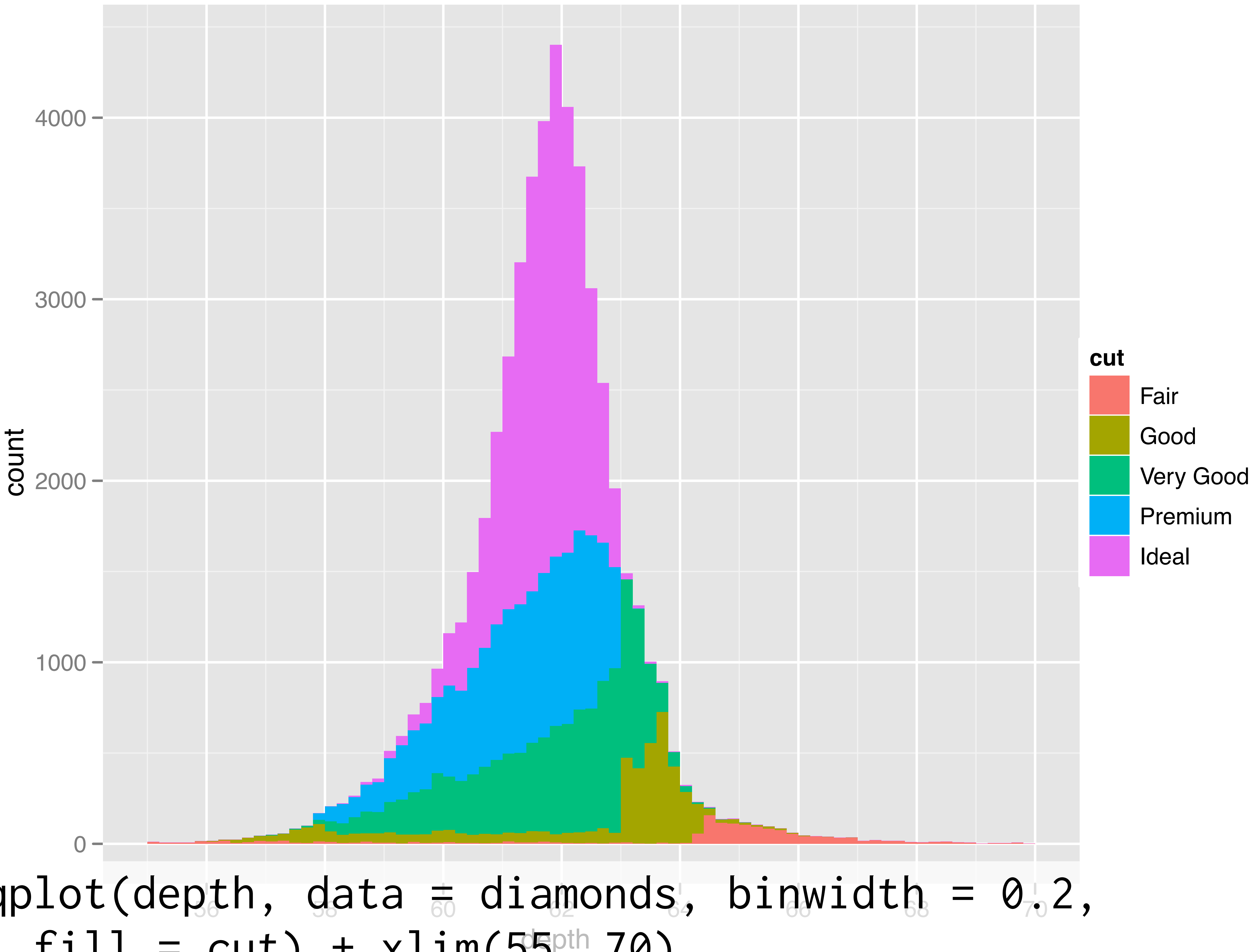
Additional variables

As with scatterplots can use **aesthetics** or **faceting**. Using aesthetics creates pretty, but ineffective, plots.

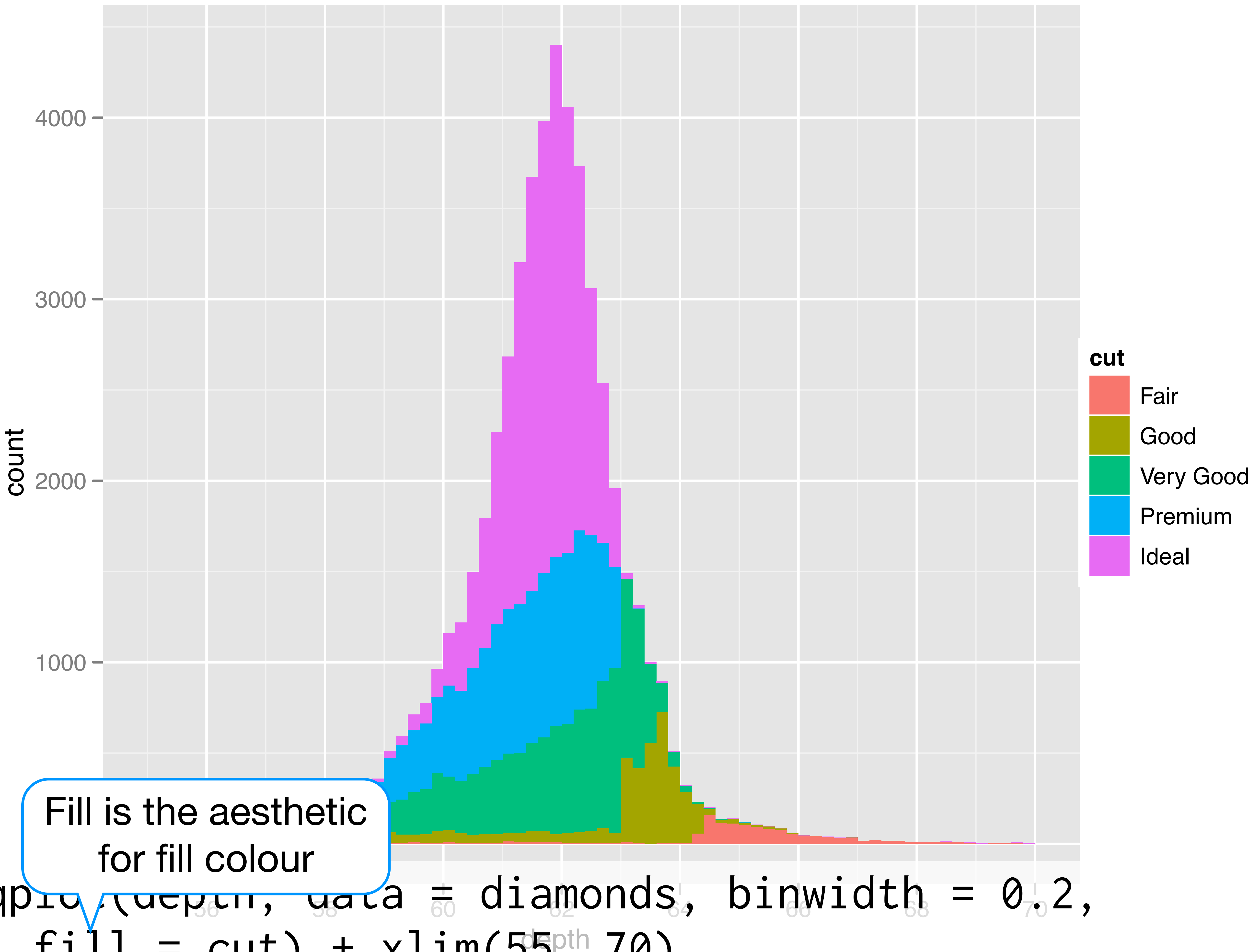
The following examples show the difference, when investigation the relationship between cut and depth.



```
qplot(depth, data = diamonds, binwidth = 0.2)
```

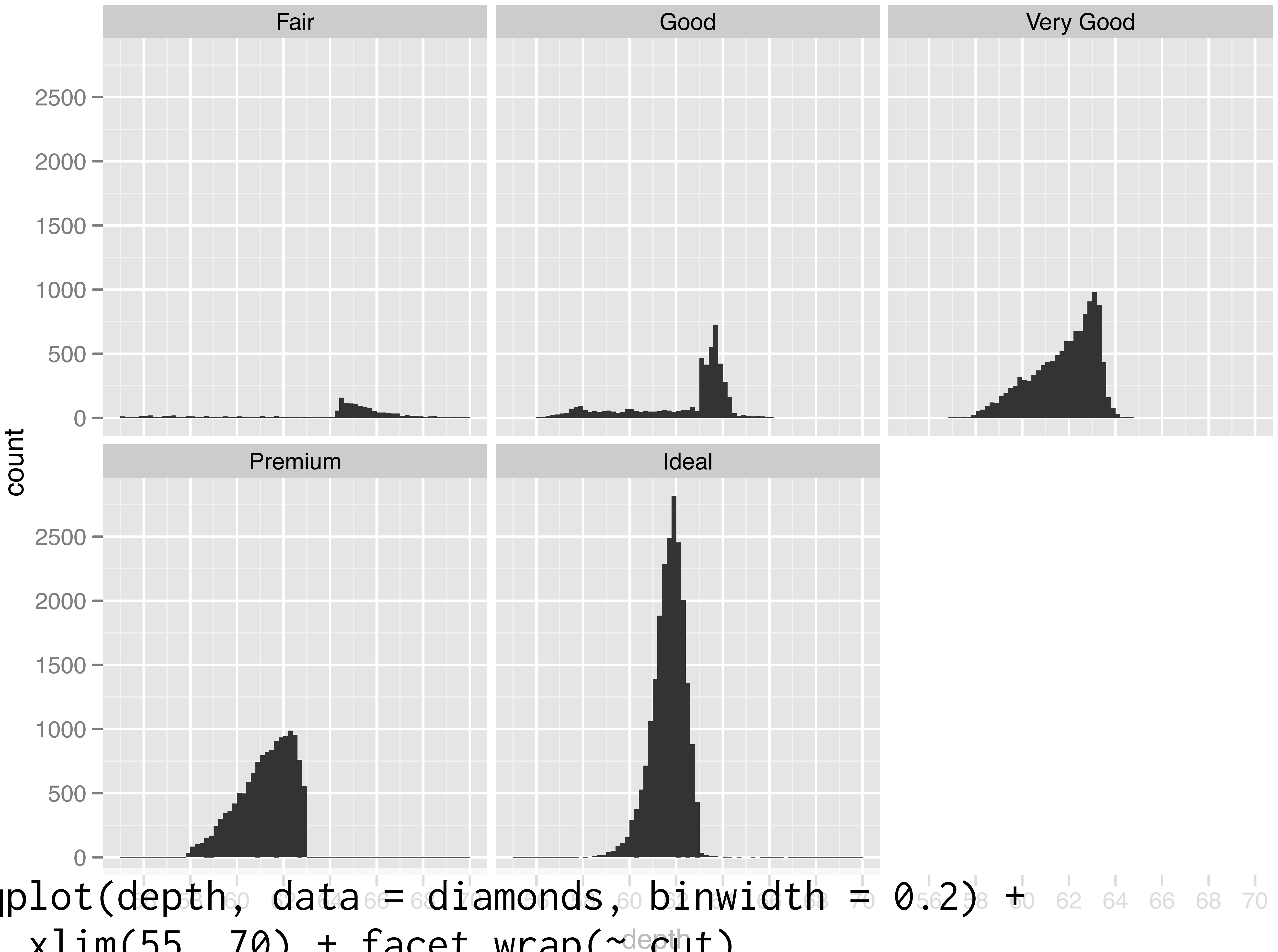


```
qplot(depth, data = diamonds, binwidth = 0.2,
  fill = cut) + xlim(55, 70)
```



Fill is the aesthetic for fill colour

```
qplot(depth, data = diamonds, binwidth = 0.2,
  fill = cut) + xlim(55, 70)
```



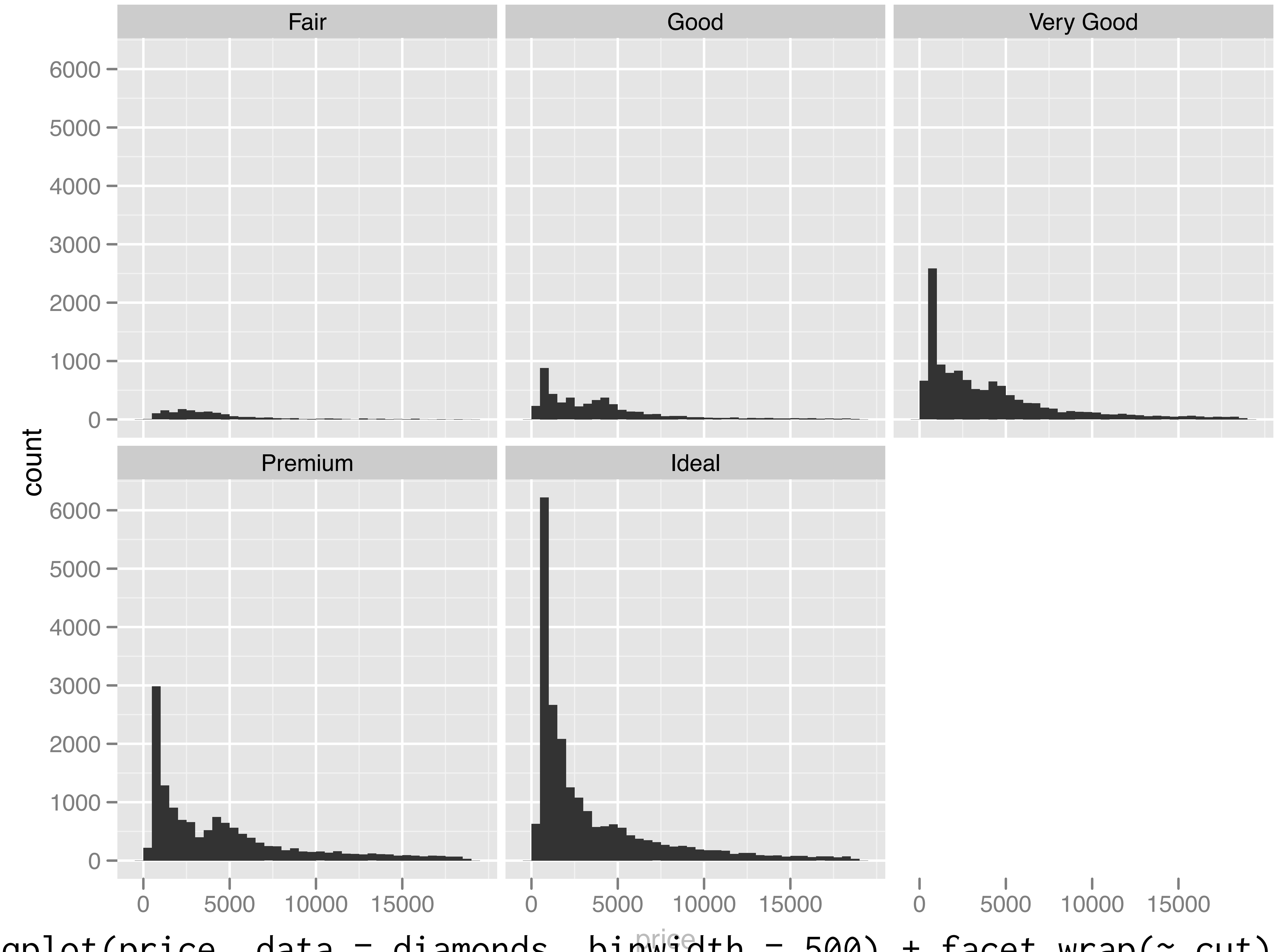
```
qplot(depth, data = diamonds, binwidth = 0.2) +
  xlim(55, 70) + facet_wrap(~ cut)
```

Your turn

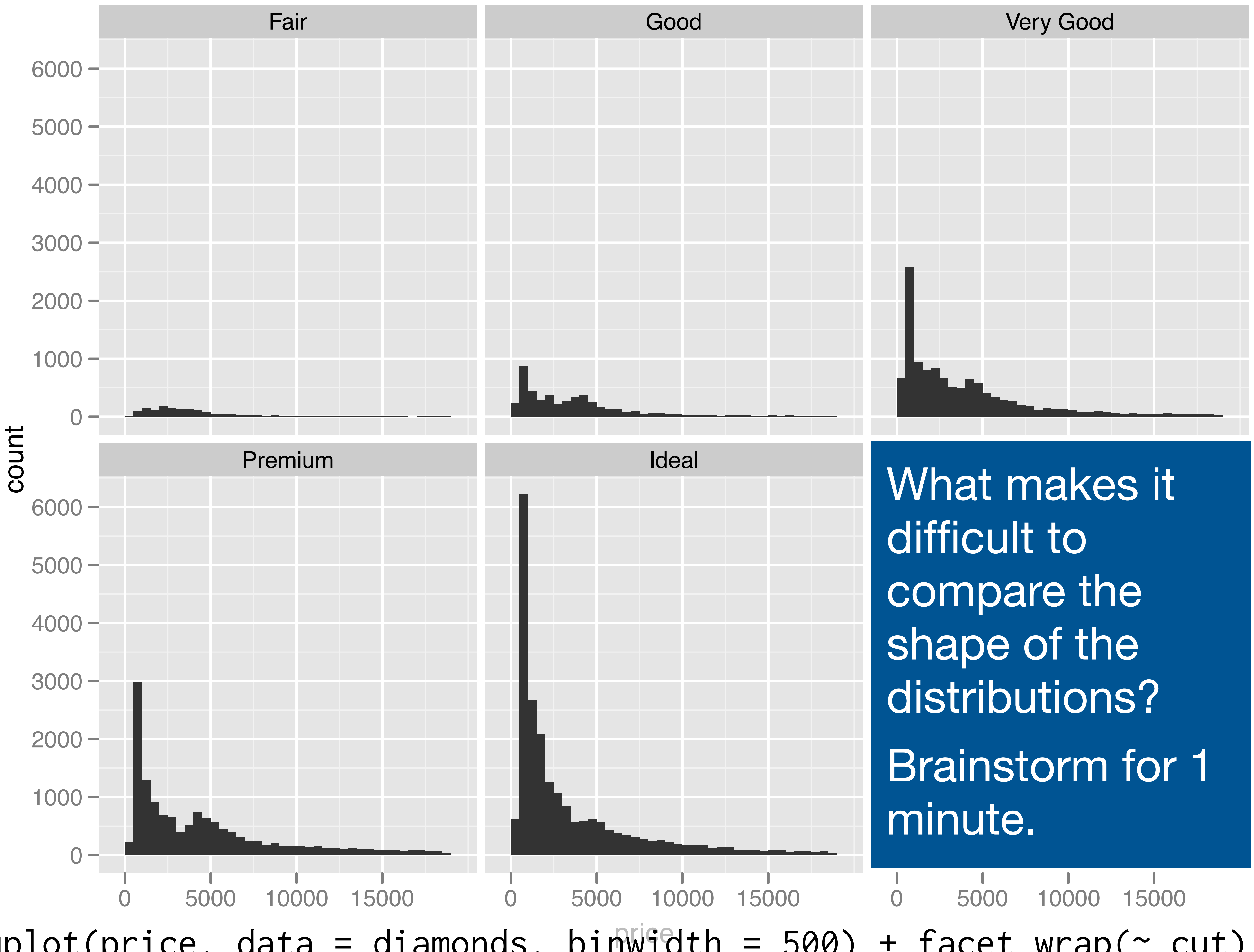
Explore the distribution of price.

How does it vary with colour, or cut, and clarity?

Practice zooming in on regions of interest.



```
qplot(price, data = diamonds, binwidth = 500) + facet_wrap(~ cut)
```



What makes it difficult to compare the shape of the distributions?

Brainstorm for 1 minute.

```
qplot(price, data = diamonds, binwidth = 500) + facet_wrap(~ cut)
```

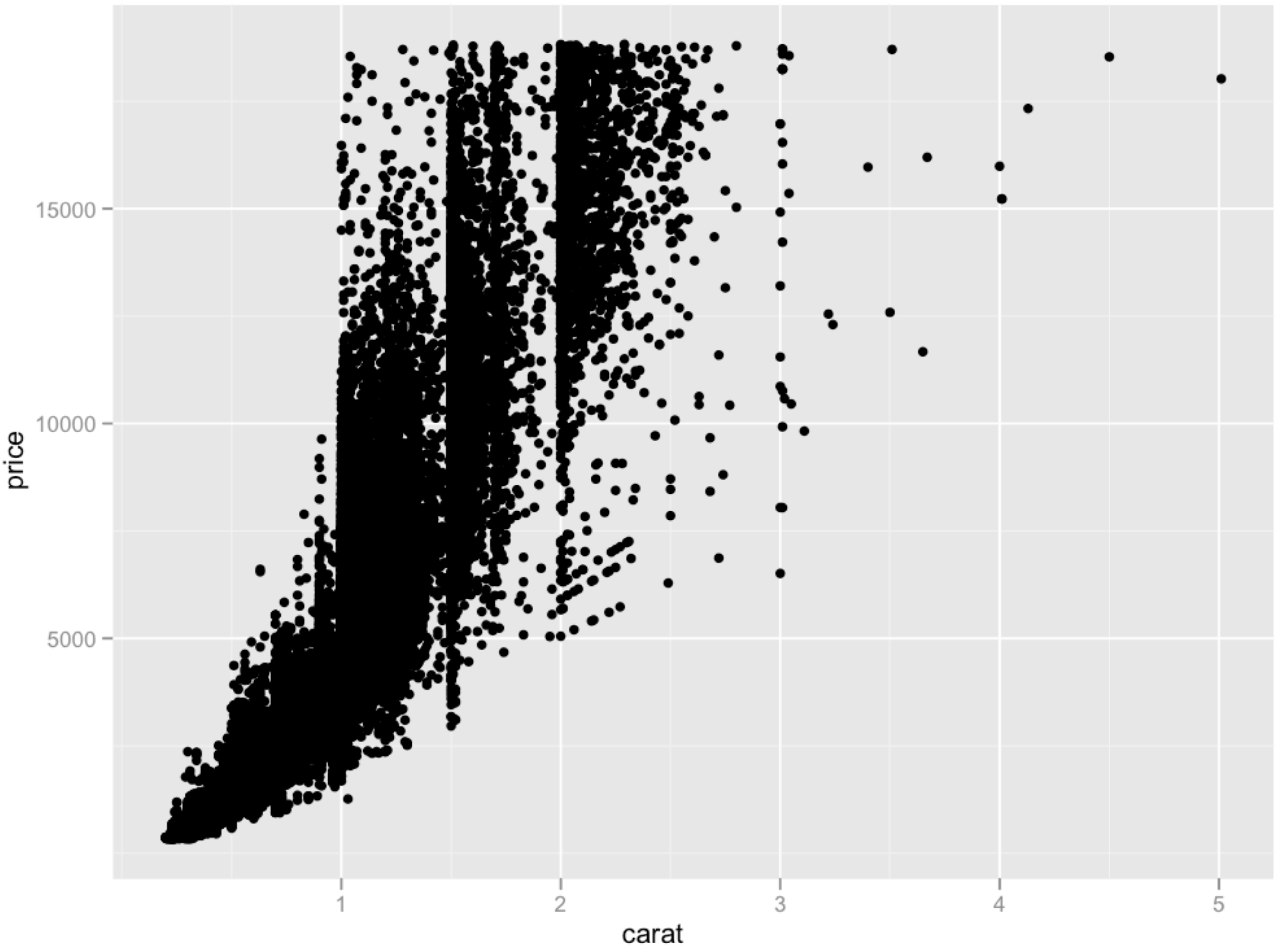
Problems

Each histogram far away from the others, but we know stacking is hard to read → *use another way of displaying densities*

Varying relative abundance makes comparisons difficult → *rescale to ensure constant area*

```
# Large distances make comparisons hard
qplot(price, data = diamonds, binwidth = 500) +
  facet_wrap(~ cut)

# Stacked heights hard to compare
qplot(price, data = diamonds, binwidth = 500, fill = cut)

# Much better - but still have differing relative abundance
qplot(price, data = diamonds, binwidth = 500,
  geom = "freqpoly", colour = cut)

# Instead of displaying count on y-axis, display density
# .. indicates that variable isn't in original data
qplot(price, ..density.., data = diamonds, binwidth = 500,
  geom = "freqpoly", colour = cut)

# To use with histogram, you need to be explicit
qplot(price, ..density.., data = diamonds, binwidth = 500,
  geom = "histogram") + facet_wrap(~ cut)
```

Big scatterplots


15000
10000
5000
price
1
2
3
4
5
carat

Your turn

Take two minutes to brainstorm possible solutions to the overplotting problem.

Idea	ggplot
Small points	`shape = I(".")`
Transparency	`alpha = I(1/50)`
Jittering	`geom = "jitter"`
Smooth curve	`geom = "smooth"`
2d bins	`geom = "bin2d"` or `geom = "hex"`
Density contours	`geom = "density2d"`

```
# There are two ways to add additional geoms
# 1) A vector of geom names:
qplot(price, carat, data = diamonds,
  geom = c("point", "smooth"))

# 2) Add on extra geoms
qplot(price, carat, data = diamonds) + geom_smooth()

# This how you get help about a specific geom:
# ?geom_smooth
# To set aesthetics to a particular value, you need
# to wrap that value in I()

qplot(price, carat, data = diamonds, colour = "blue")
qplot(price, carat, data = diamonds, colour = I("blue"))

# Practical application: varying alpha
qplot(price, carat, data = diamonds, alpha = I(1/10))
qplot(price, carat, data = diamonds, alpha = I(1/50))
qplot(price, carat, data = diamonds, alpha = I(1/100))
qplot(price, carat, data = diamonds, alpha = I(1/250))
```

Your turn

Explore the relationship between carat, price and clarity, using these techniques.

(i.e. make this plot more informative:

`qplot(carat, price, data = diamonds, colour = clarity)`)

Which did you find most useful?

```r
qplot(carat, price, data = diamonds,
  colour = clarity)
qplot(log10(carat), log10(price),
  data = diamonds, colour = clarity)
qplot(log10(carat), log10(carat / price),
  data = diamonds, colour = clarity)
```

```r
qplot(log10(carat), log10(price), data = diamonds,
  geom = "hex", bins = 10) + facet_wrap(~ clarity)
qplot(log10(carat), log10(price), data = diamonds,
  colour = clarity, geom = "smooth")
```

Workflow

Coding strategy

At the end of each interactive session, you want a summary of everything you did. Two options:

1. Save everything you did with `savehistory()` then remove the unimportant bits.

2. Build up the important bits as you go. (this is how I work)

Working directory

Set your working directory to specify where files will be loaded from and saved to – all paths are relative to the working directory.

From the terminal (linux or mac): the working directory is the directory you're in when you start R

On windows: File | Change dir.

On the mac: ⌘-D

In one directory

Data (`.csv`)

+

Code (`.r`)

+

Graphics (`.png, .pdf`)

+

Written report (`.tex`)