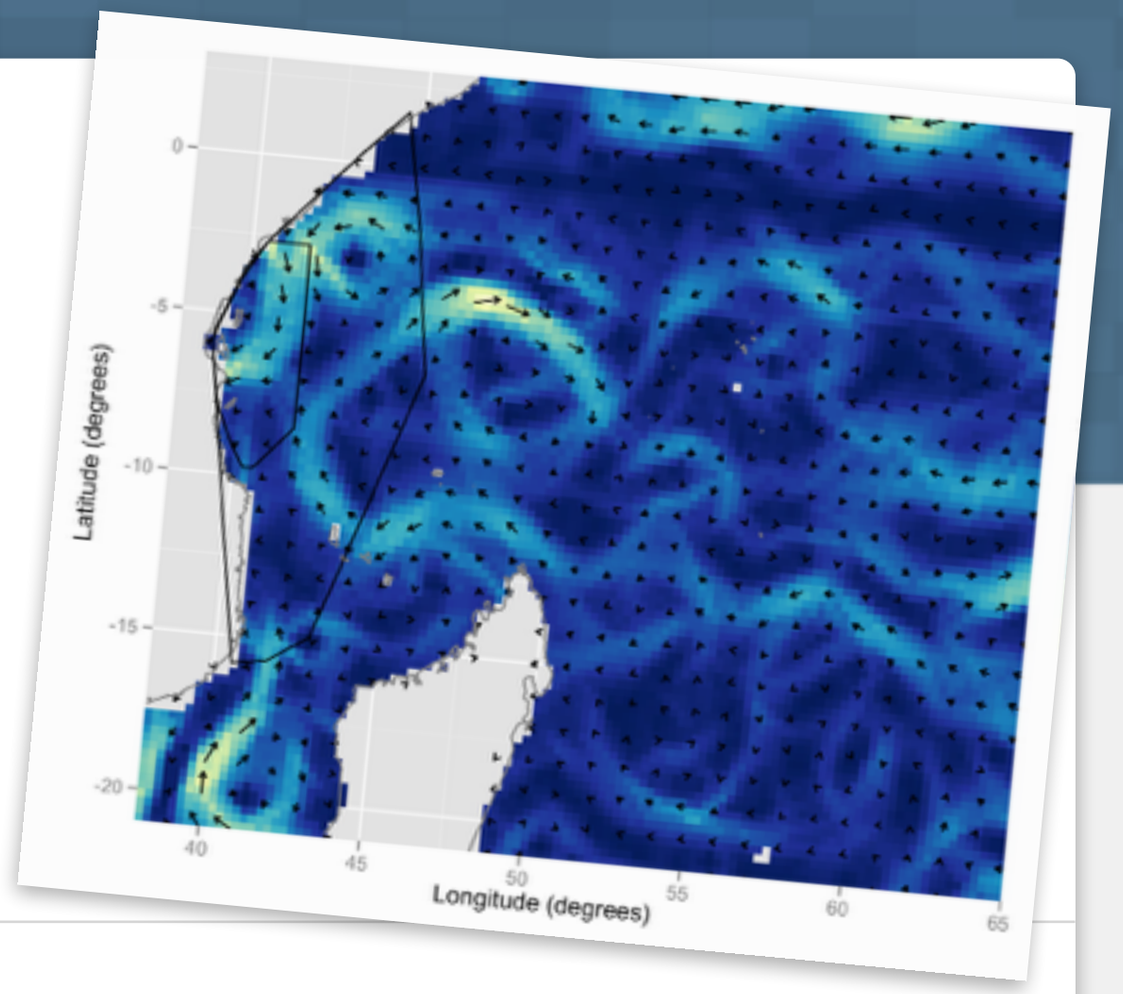
All Training materials are provided "as is" and without warranty and RStudio disclaims any and all express and implied warranties including without limitation the implied warranties of title, fitness for a particular purpose, merchantability and noninfringement.

RStudio

Visualizing Data

Discover the unexpected in your data with ggplot2



Garrett Grolemund

Master Instructor, RStudio

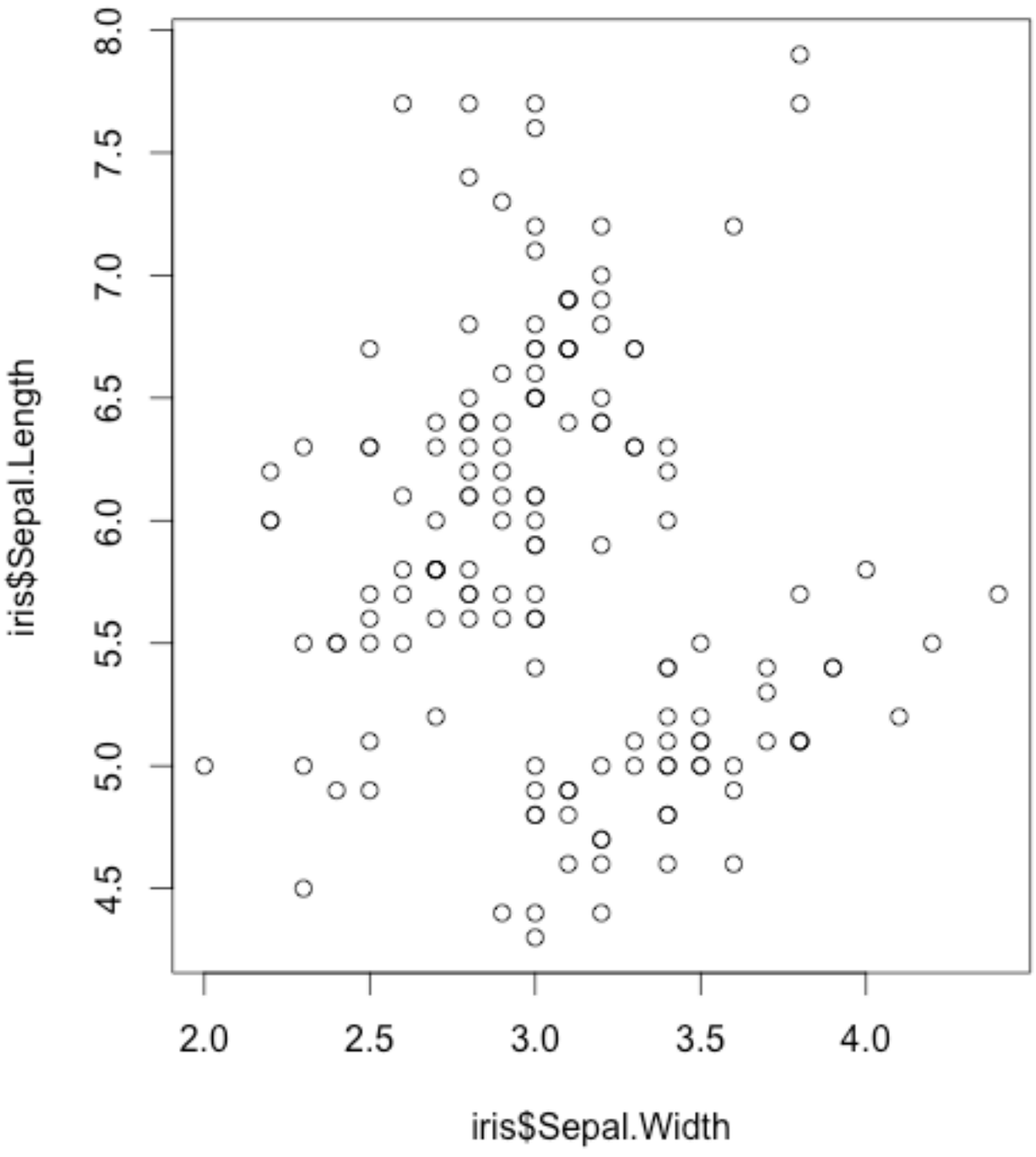
April 2014

1. Scatterplots
    a. Aesthetics, Facets, Geoms
2. Bar charts
    b. Positions
3. Histograms
    c. Parameters
4. Visualizing big data
5. Saving Graphs

The simple graph has brought more information to the data analyst's mind than any other device.

– John Tukey

plot




```
plot(iris$Sepal.Width,
     iris$Sepal.Length)
```

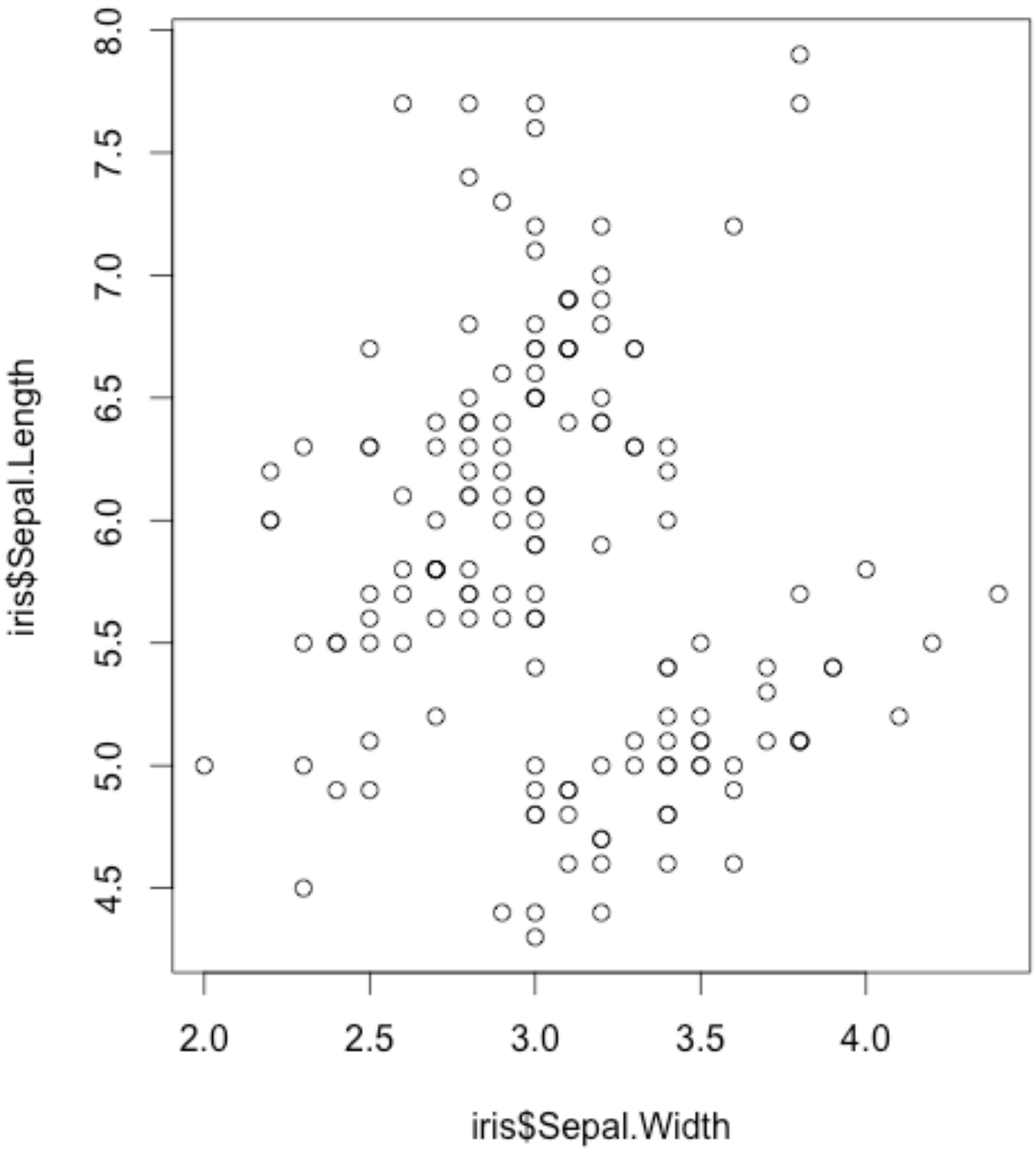
simple plots in R

x variable

y variable

```
plot(iris$Sepal.Width,iris$Sepal.Length)
```

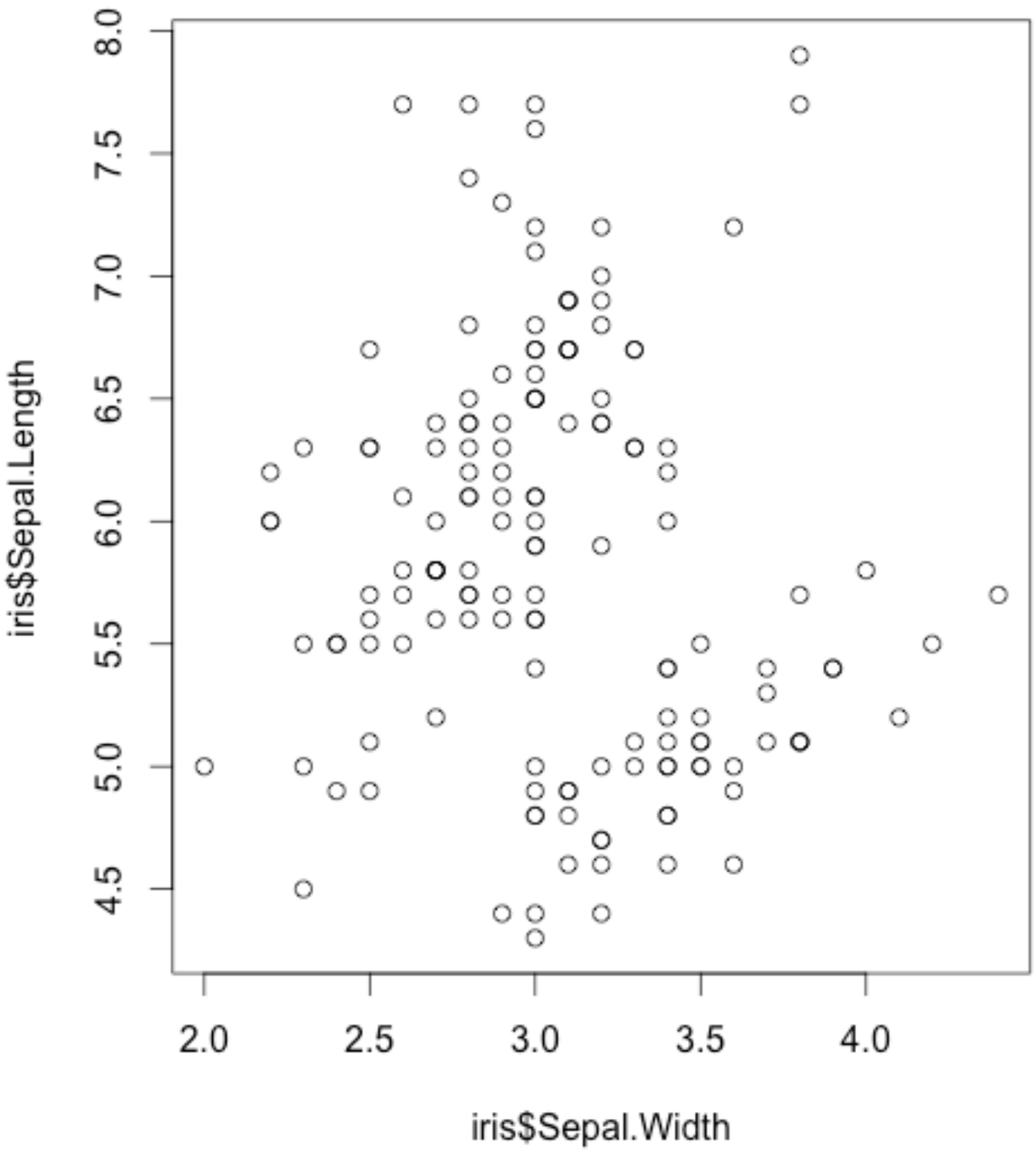
plot

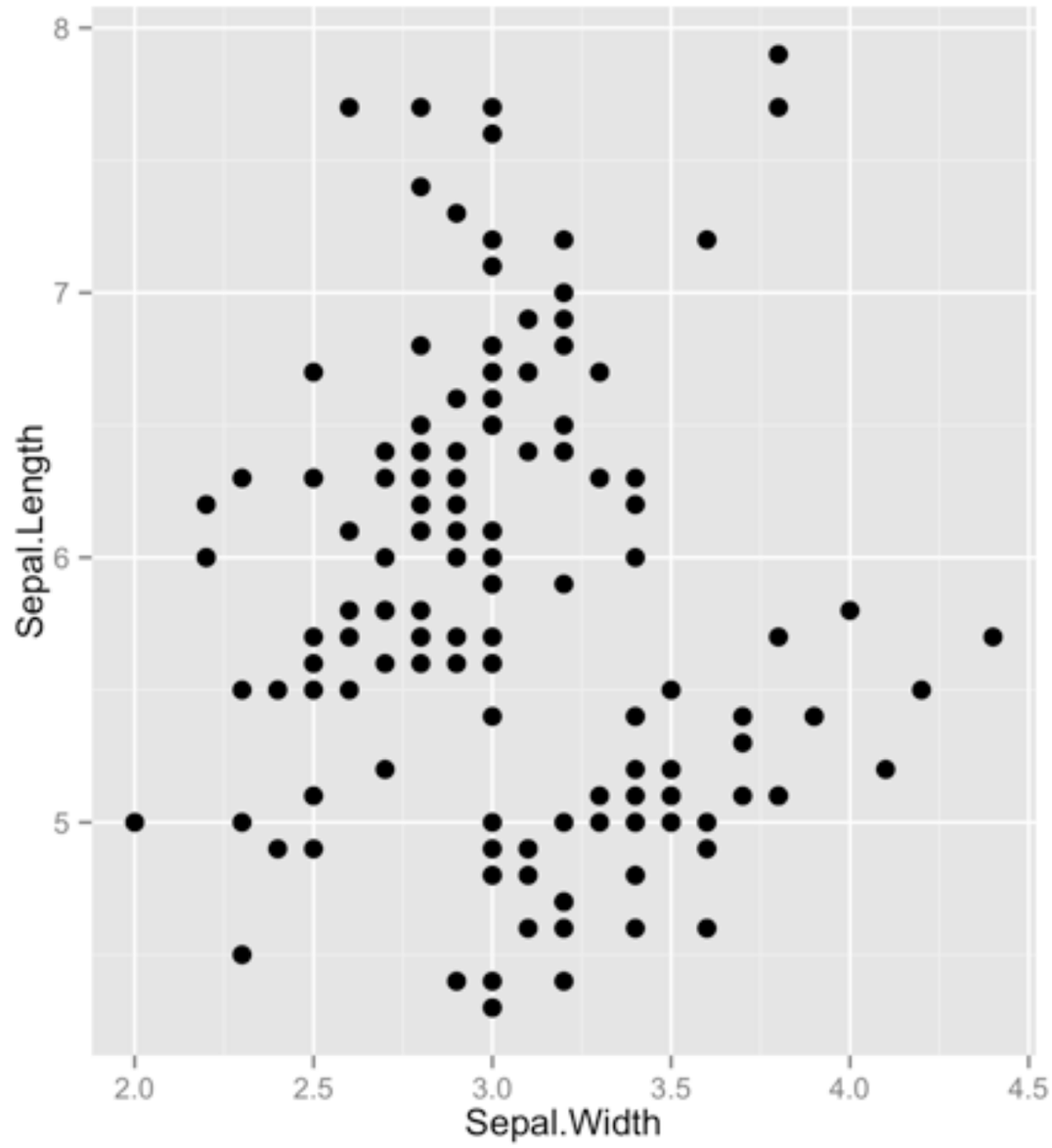



```
plot(iris$Sepal.Width,
     iris$Sepal.Length)
```

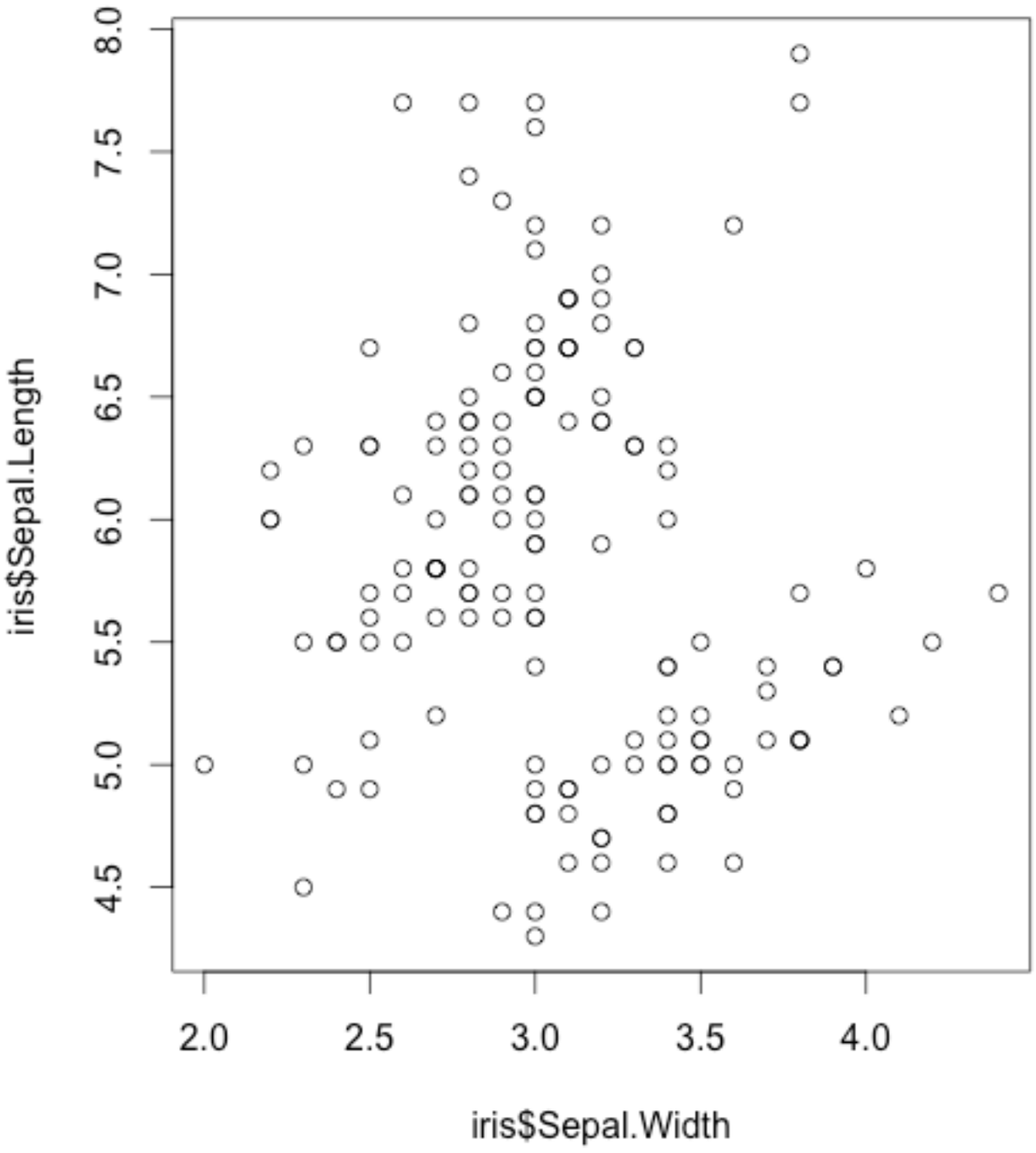
- R's basic plot method
- simple
- does different things in different contexts (usually in a helpful way)
- difficult to customize

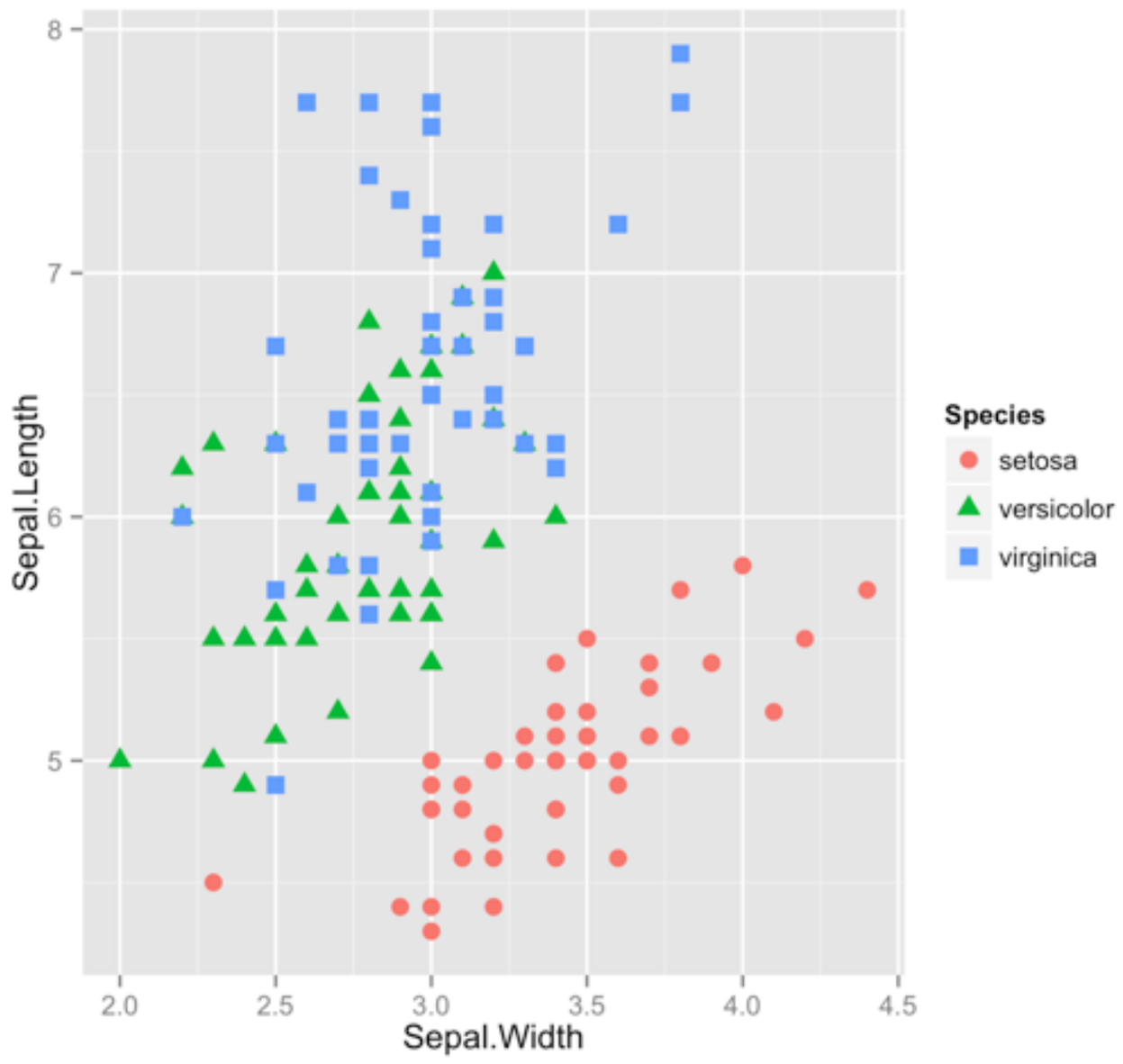
ggplot2



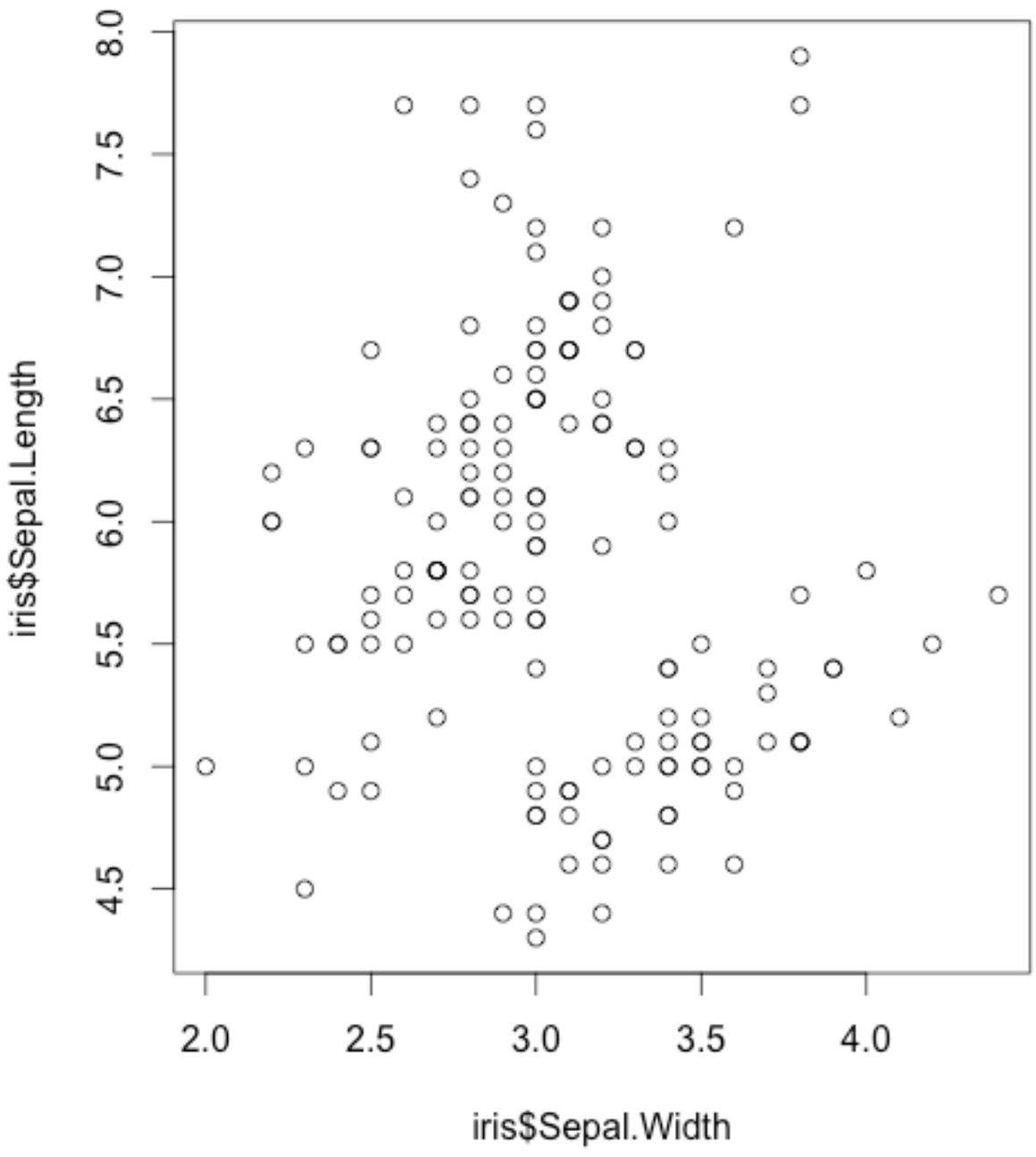


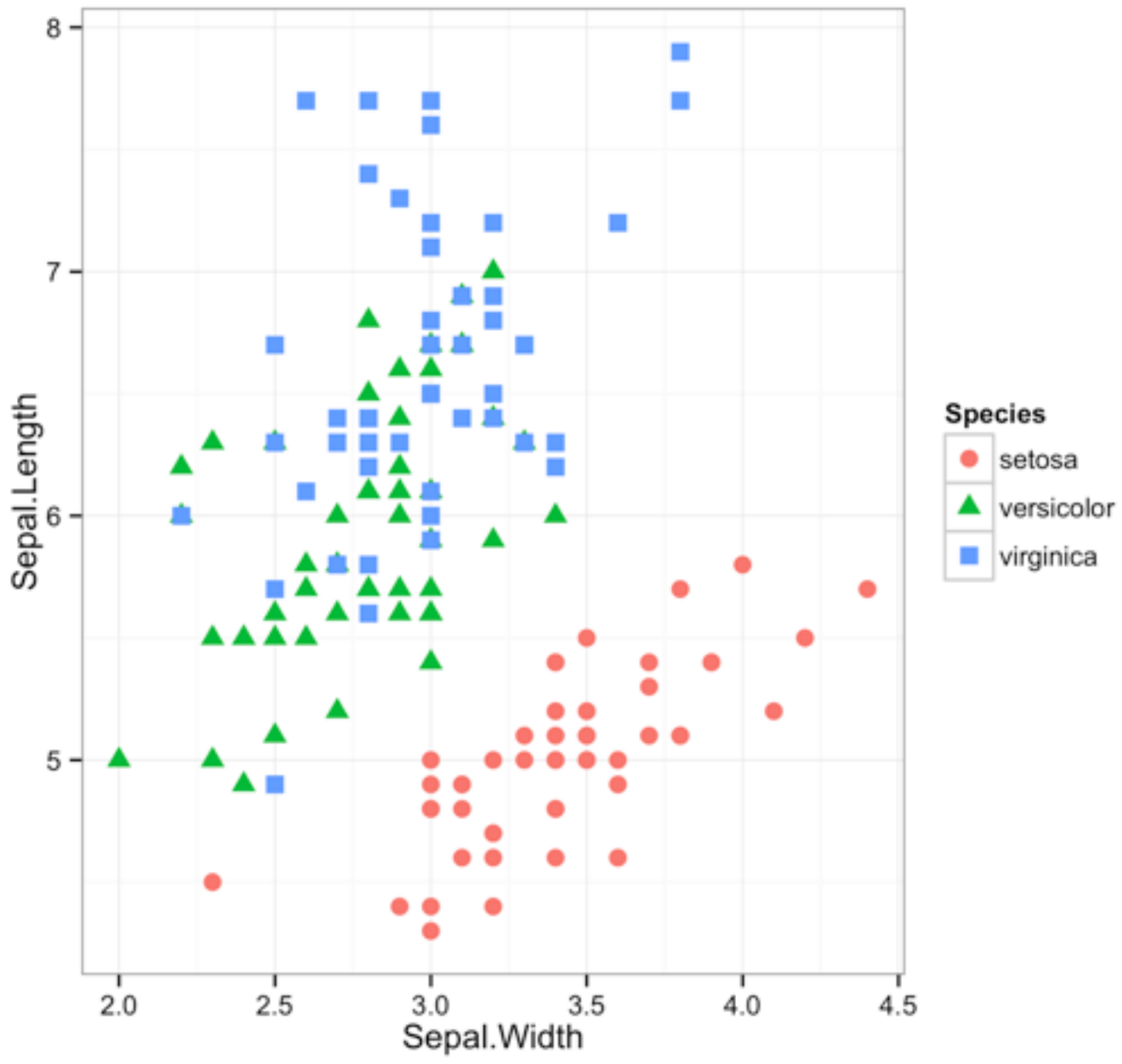
ggplot2



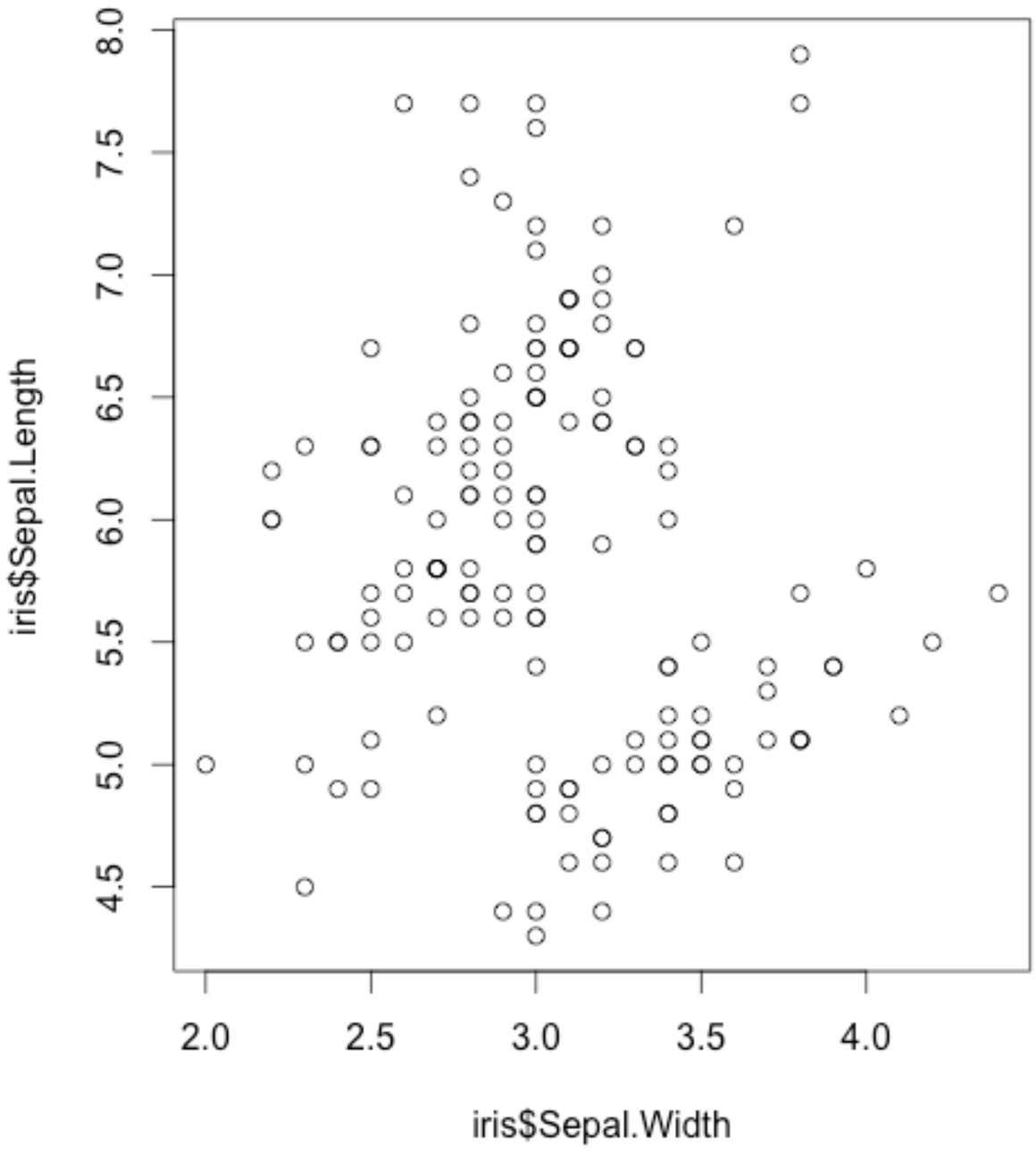


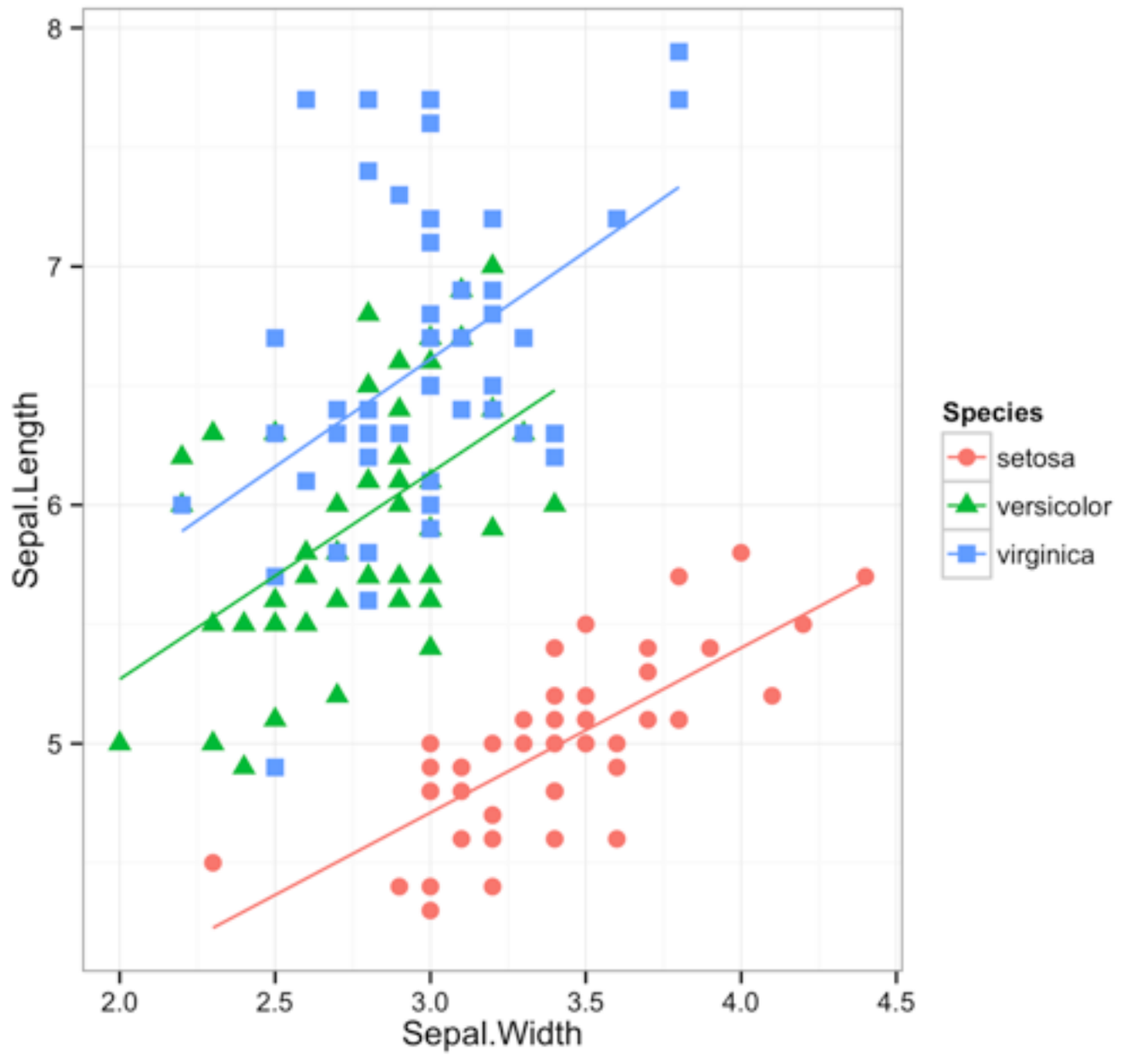
ggplot2



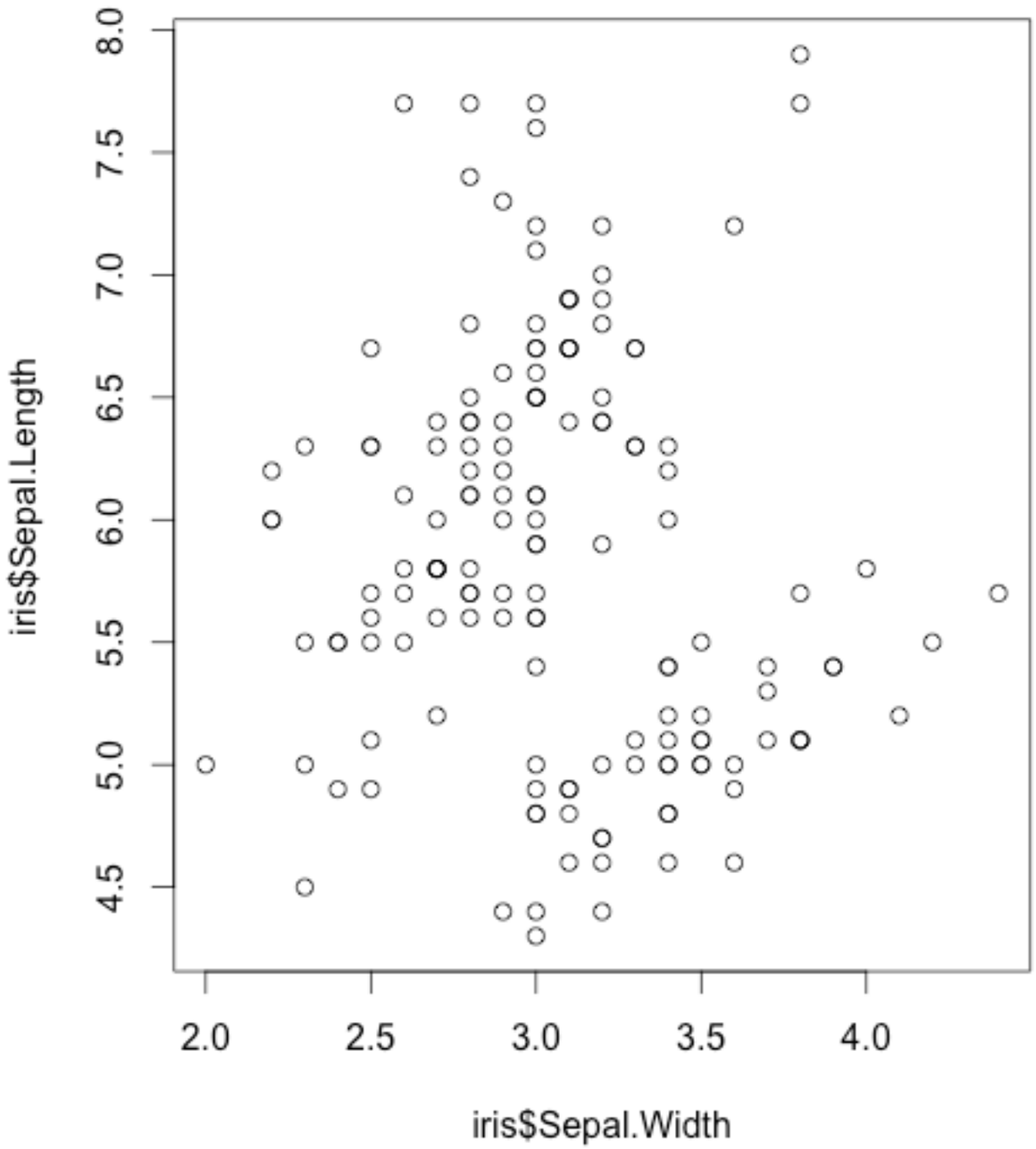


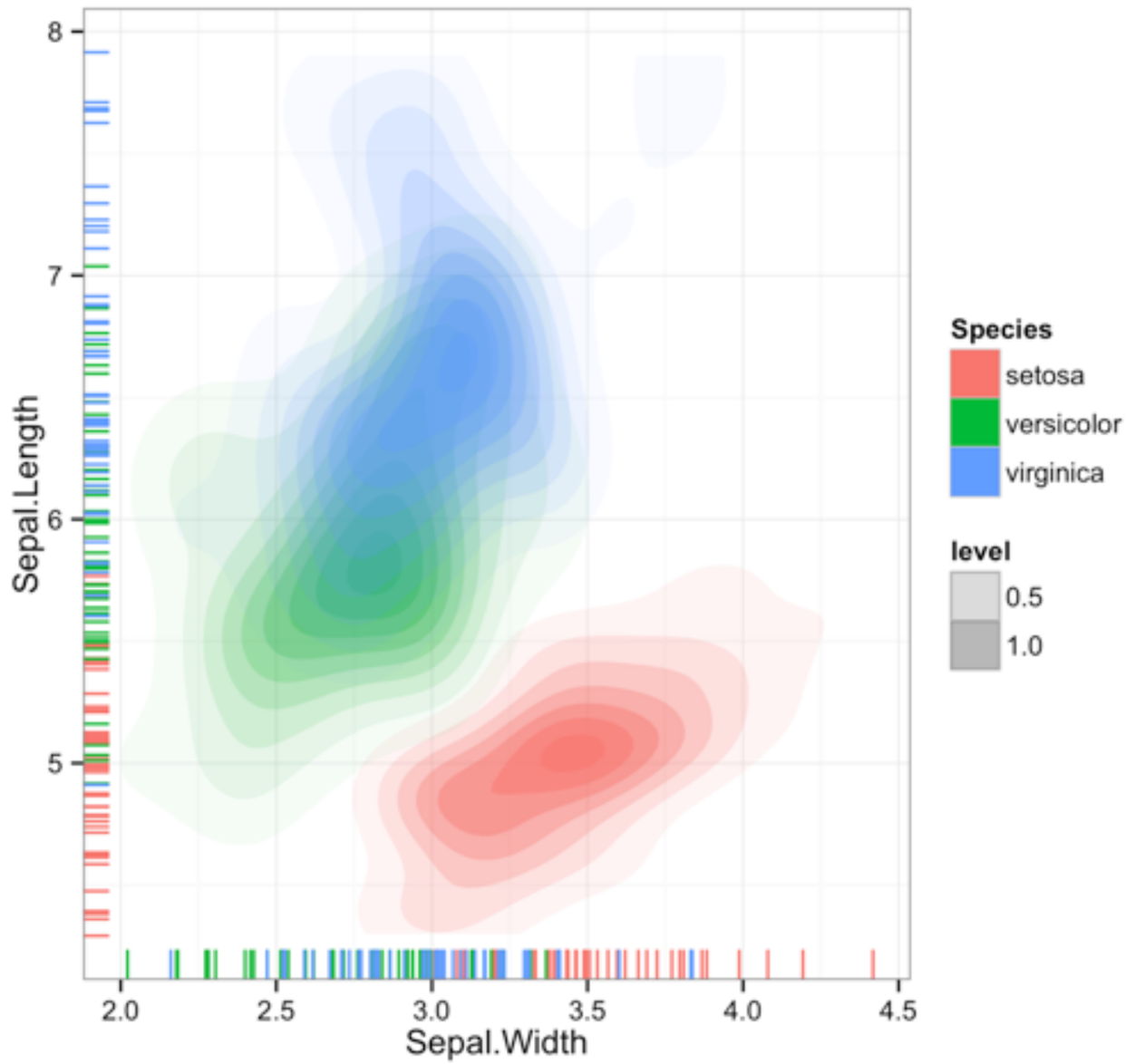
ggplot2



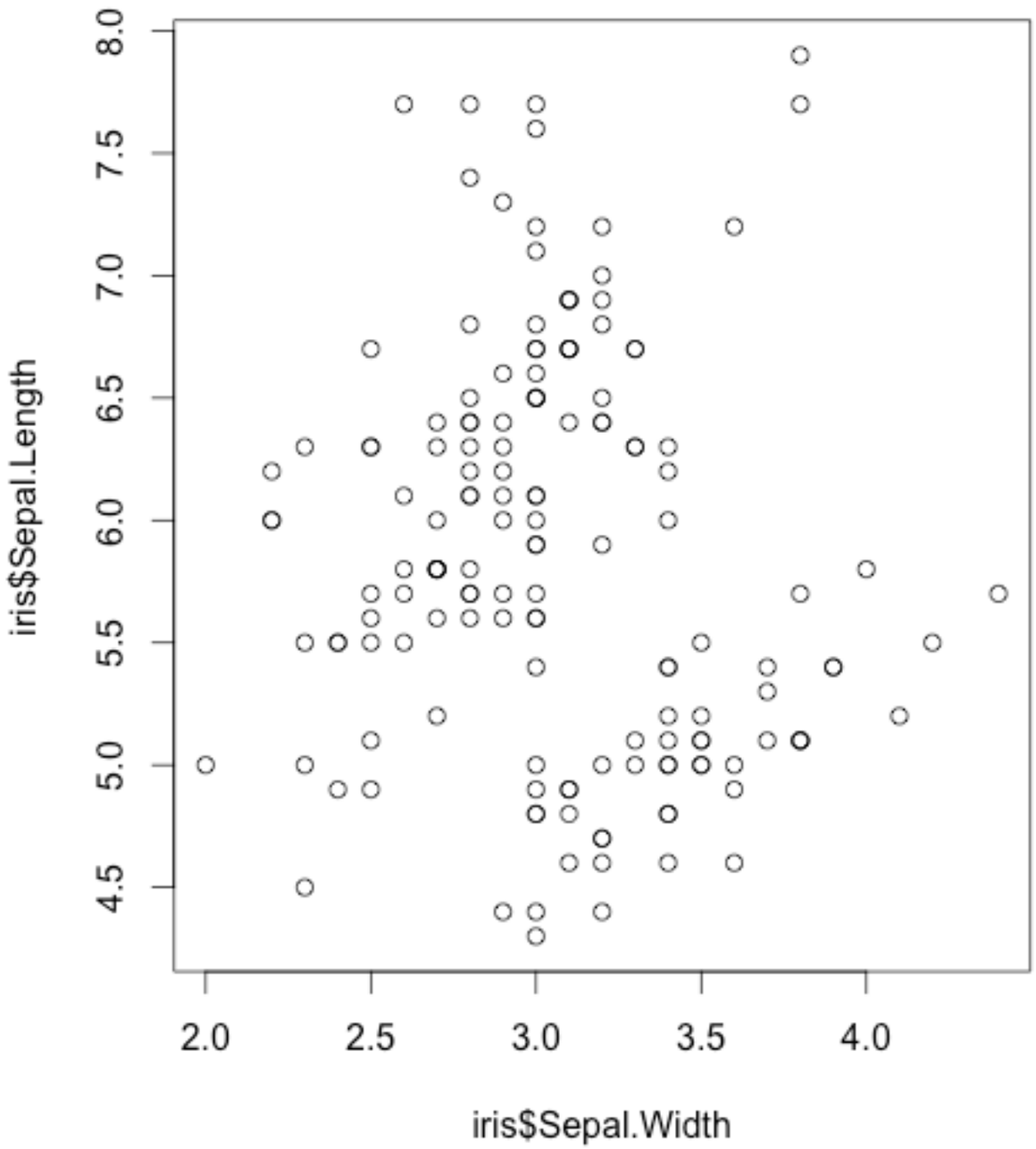


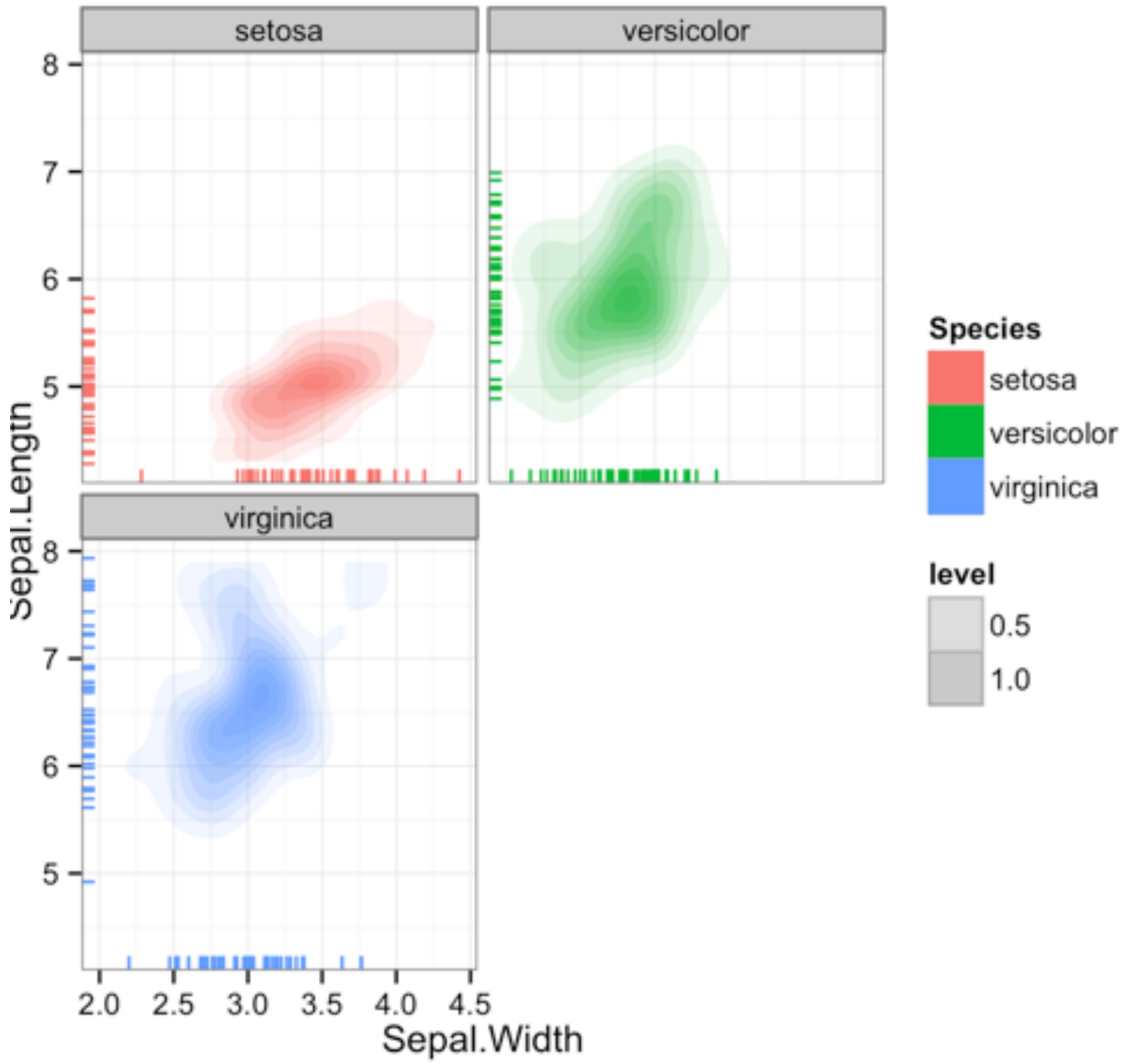
ggplot2

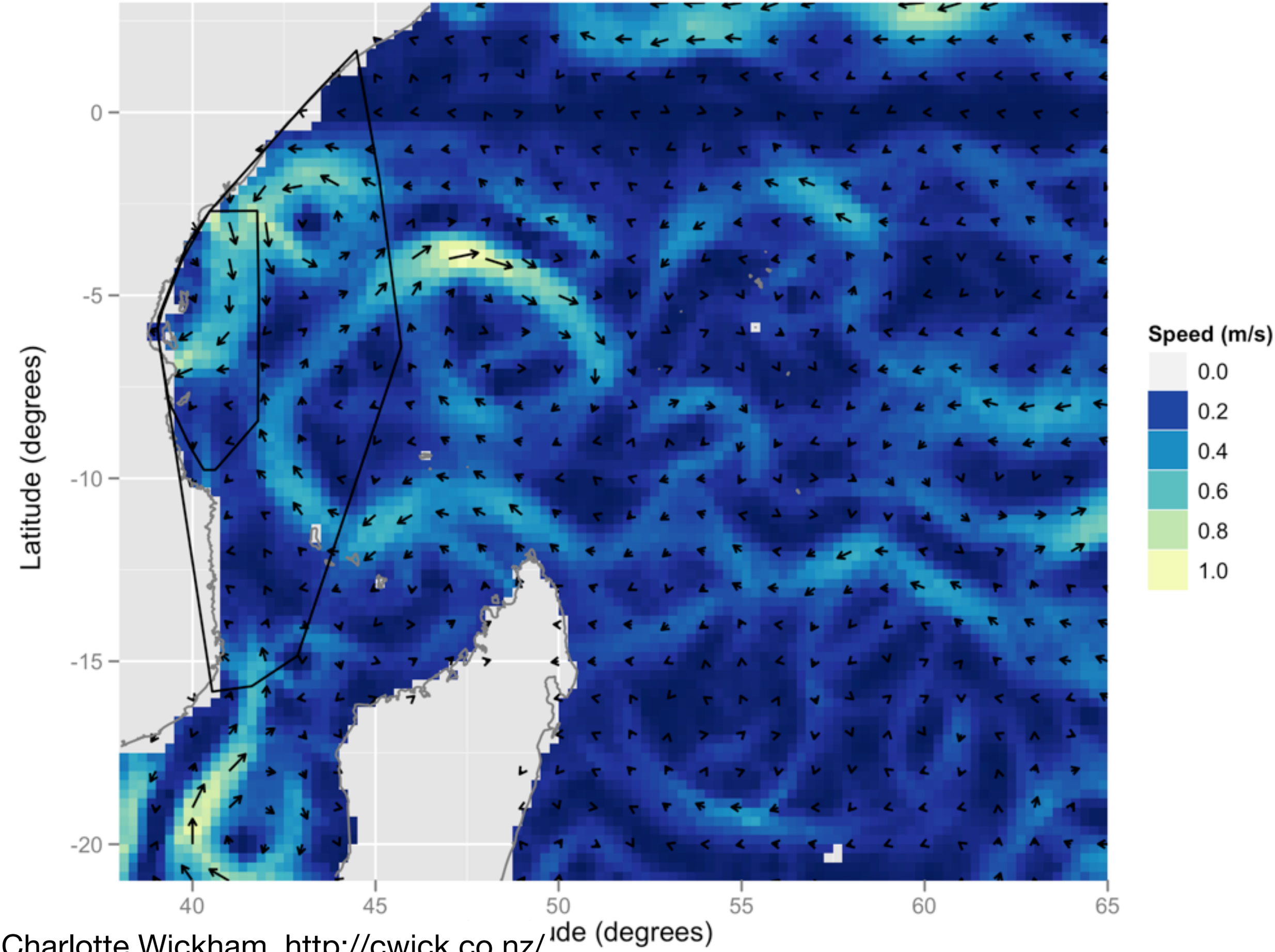




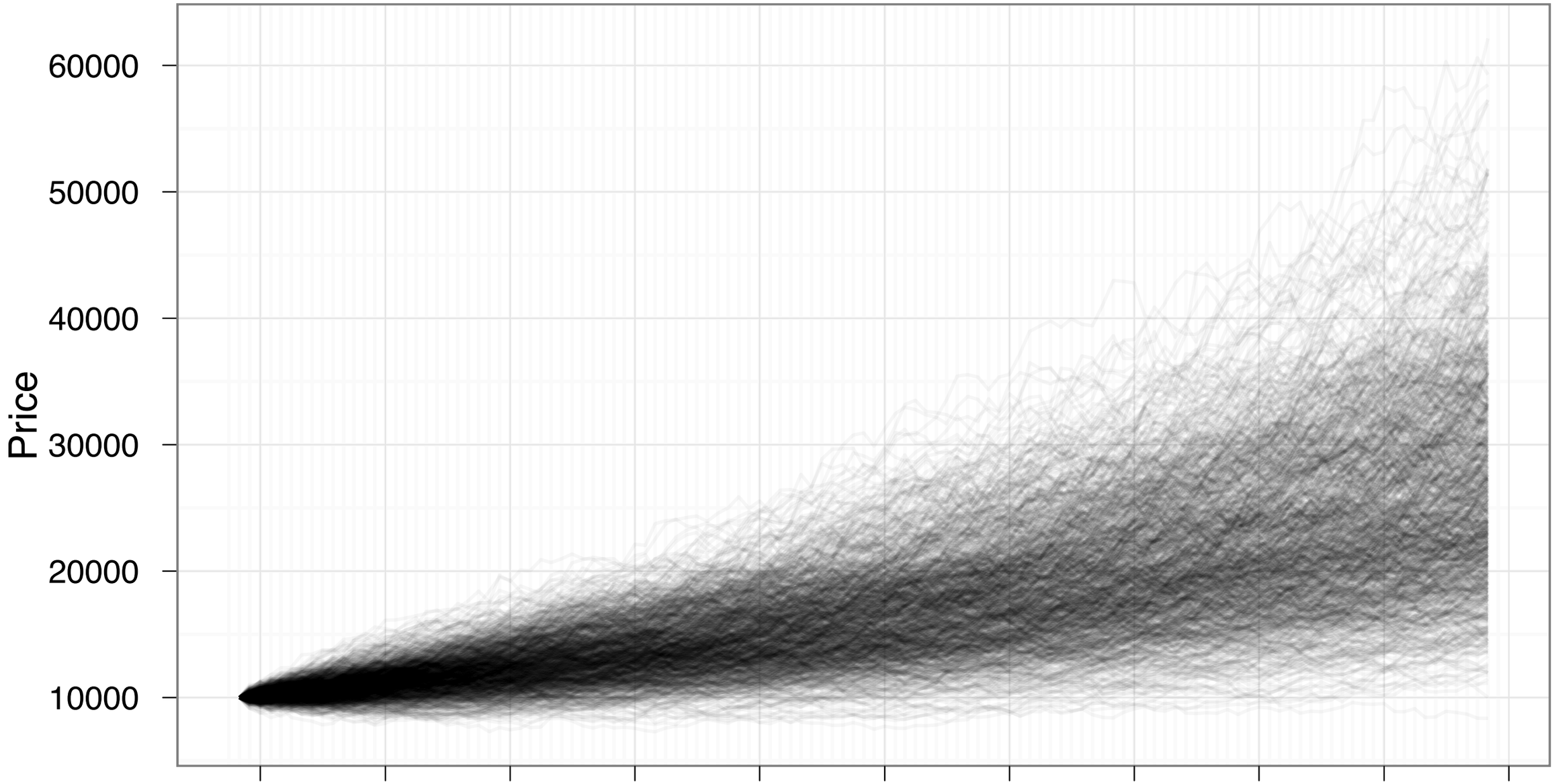
ggplot2

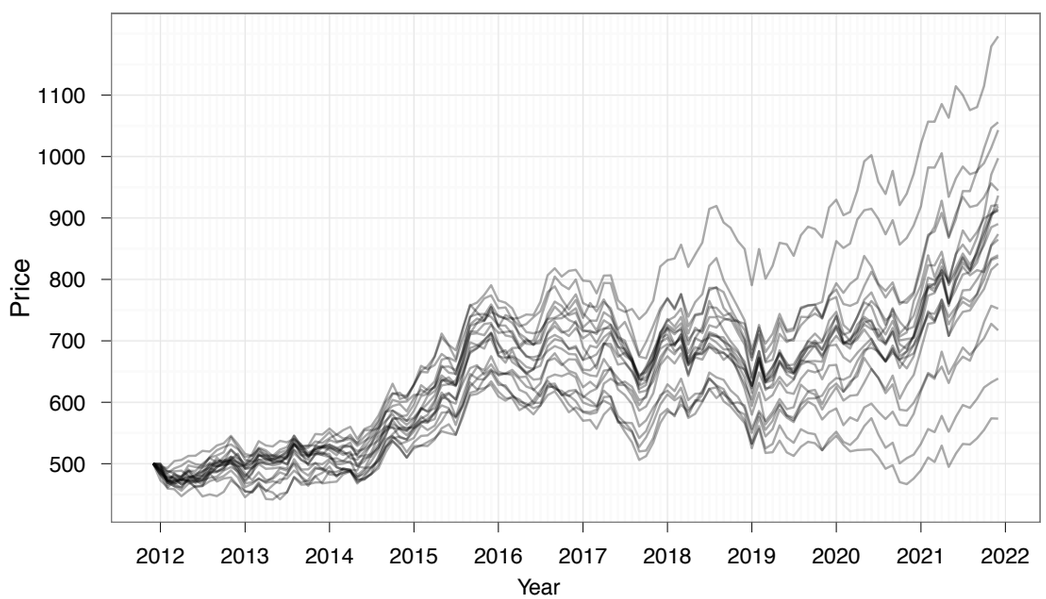


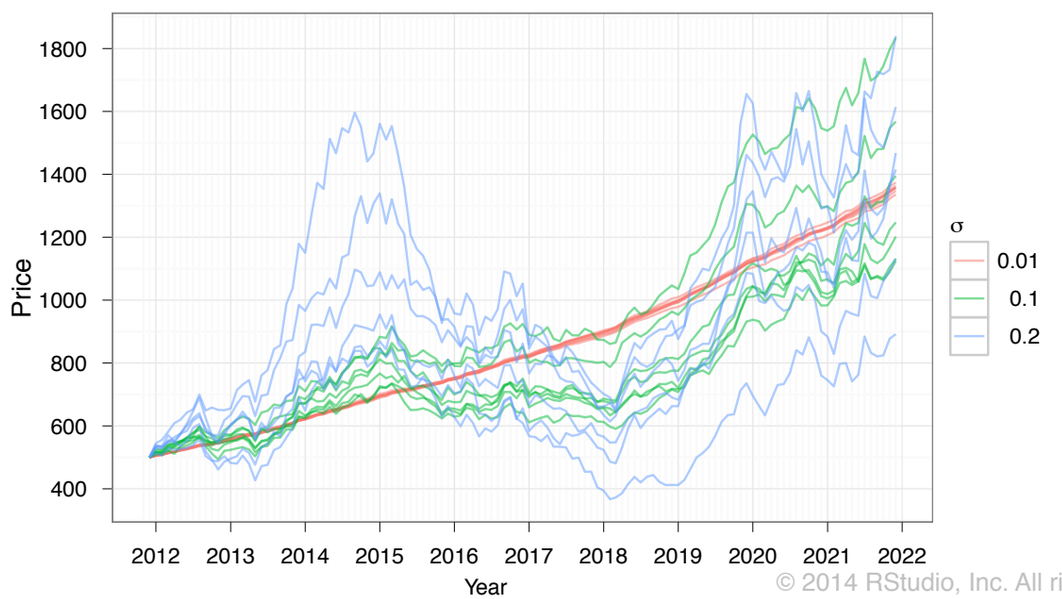


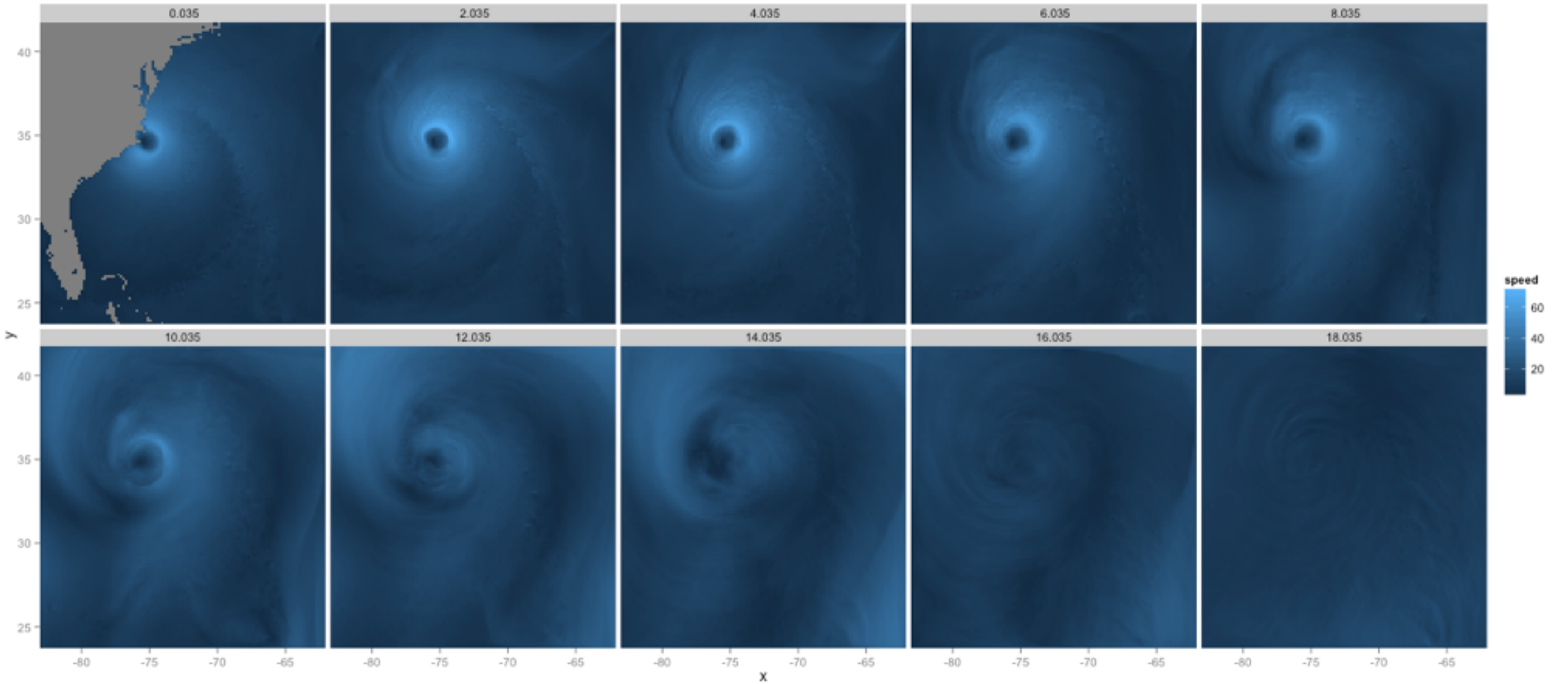


Charlotte Wickham, http://cwick.co.nz/

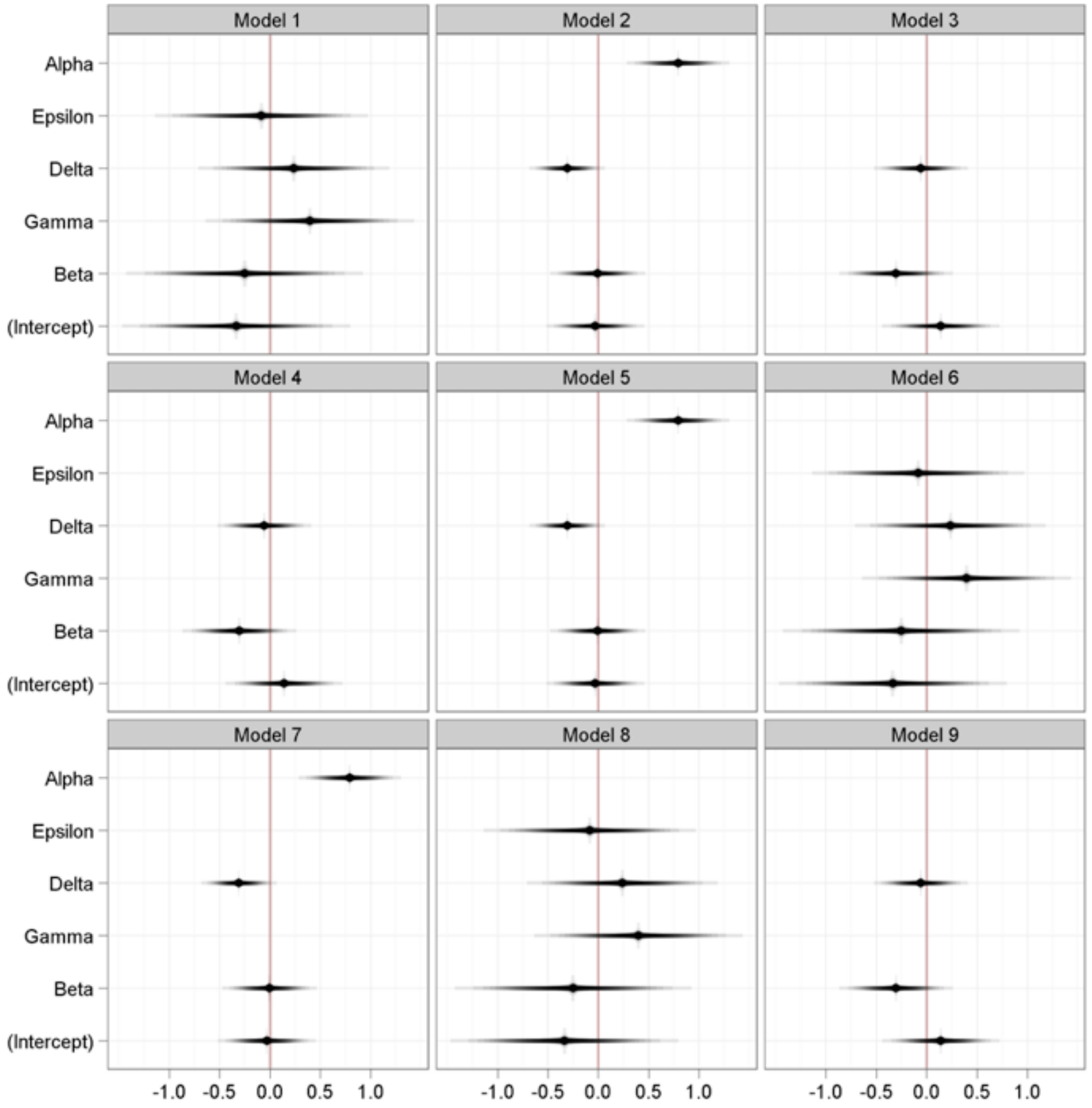




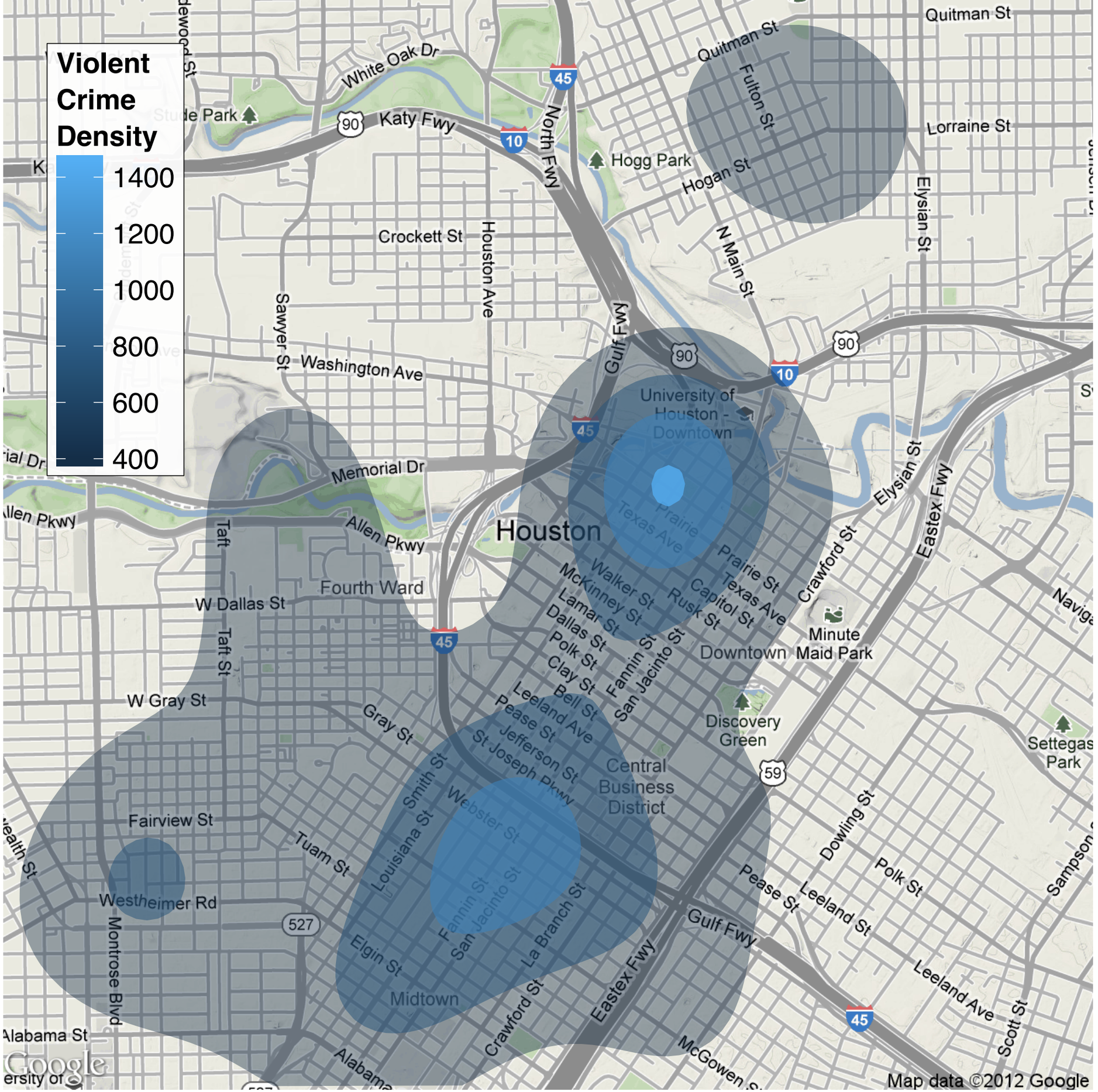




Winston Chang, http://shop.oreilly.com/product/0636920023135.do



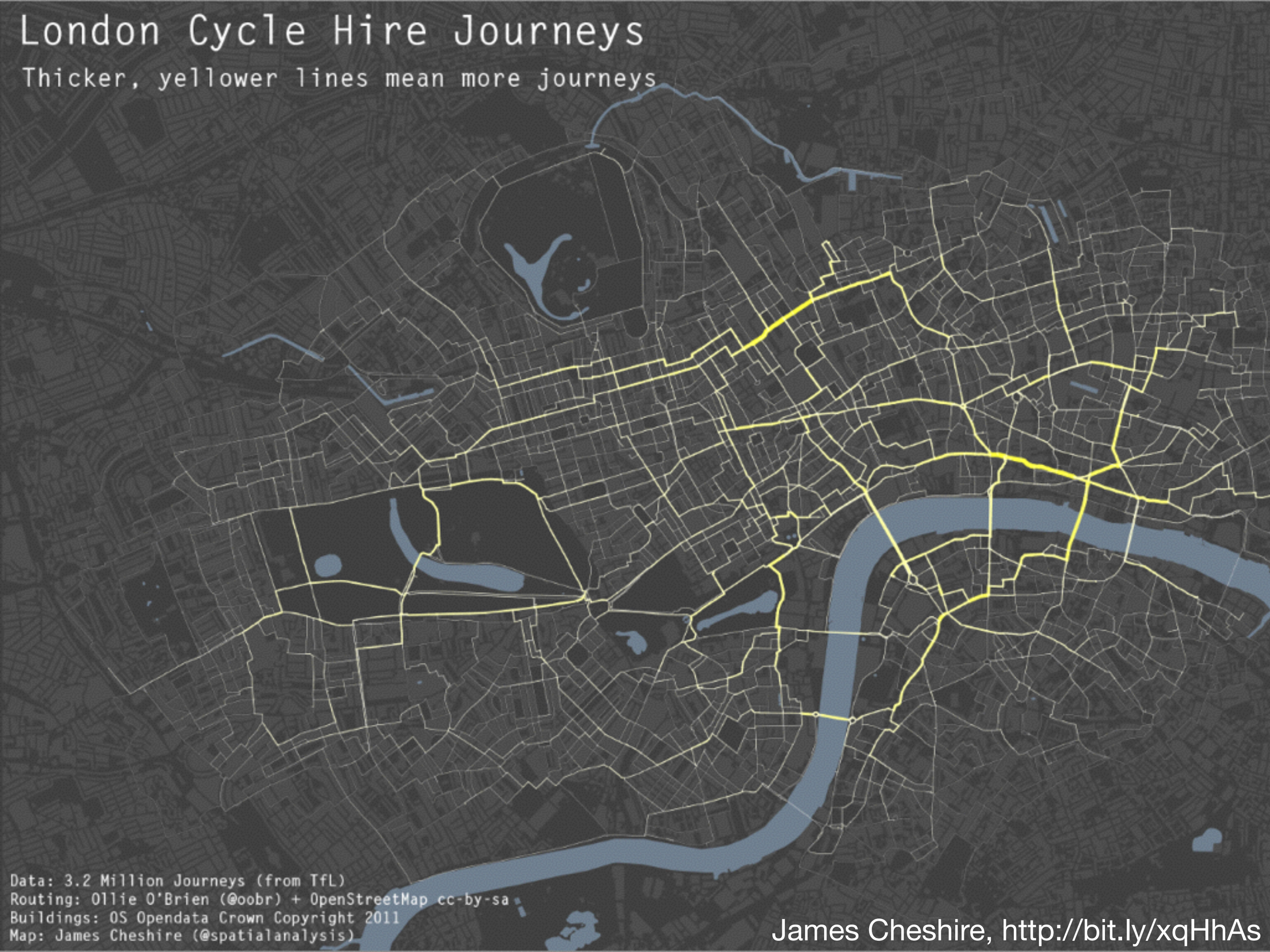
David B Sparks, http://bit.ly/hn54NW



David Kahle, https://dl.dropbox.com/u/24648660/ggmap%20useR%202012.pdf

London Cycle Hire Journeys

Thicker, yellower lines mean more journeys



Data: 3.2 Million Journeys (from TfL)
Routing: Ollie O'Brien (@oobr) + OpenStreetMap cc-by-sa
Buildings: OS Opendata Crown Copyright 2011
Map: James Cheshire (@spatialanalysis)

James Cheshire, http://bit.ly/xqHhAs

A picture is not merely worth a thousand words, it is much more likely to be scrutinized than words are to be read.

– John Tukey

Diving in:

Scatterplots

Looking at data with R

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

The mpg data set comes in the ggplot2 package

```
?mpg
View(mpg)
```

Always read the help page

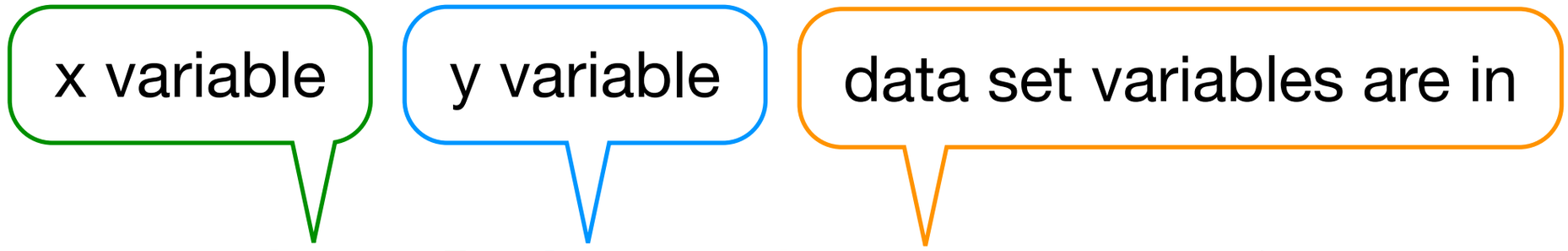
Your turn

Make a prediction. What relationship do you expect to see between engine size (displ) and mileage (hwy)?

No peeking ahead!

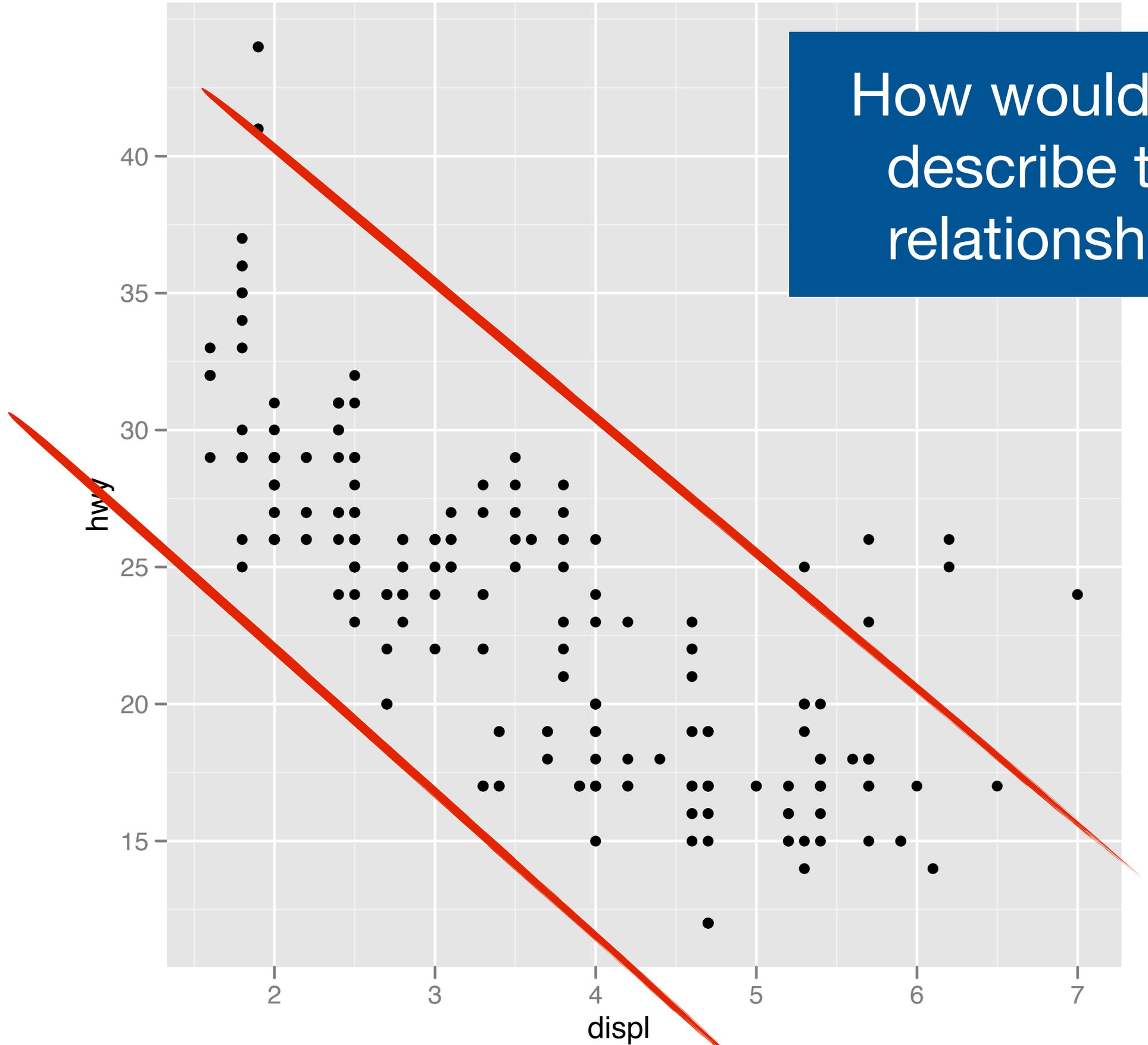
How can we explore this?

(quick) plots in R




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

How would you describe this relationship?



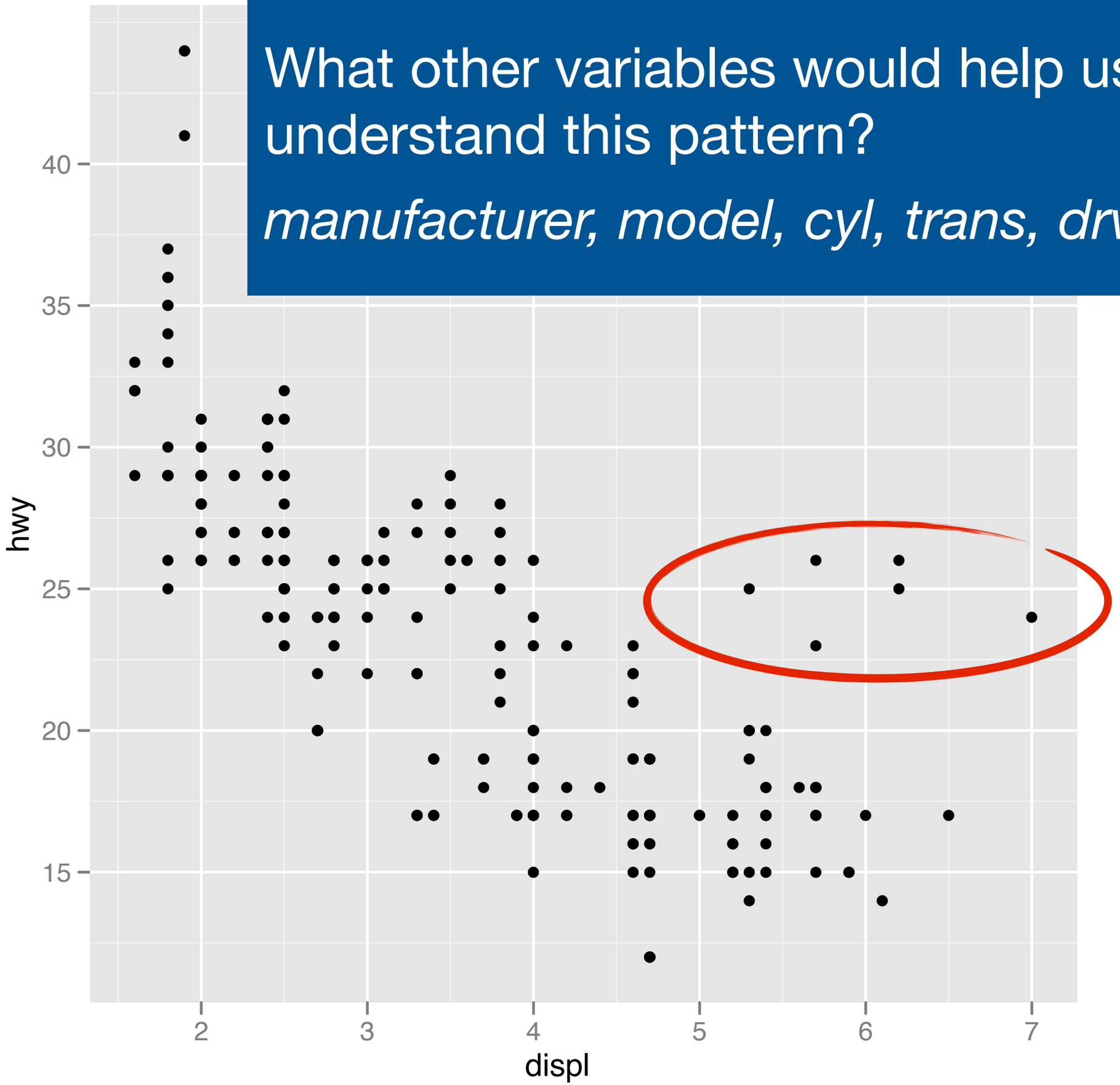

```
qplot(displ, hwy, data = mpg)
```

The greatest value of a picture is when it forces us to notice what we never expected to see.

– John Tukey

What other variables would help us understand this pattern?

manufacturer, model, cyl, trans, drv, class




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

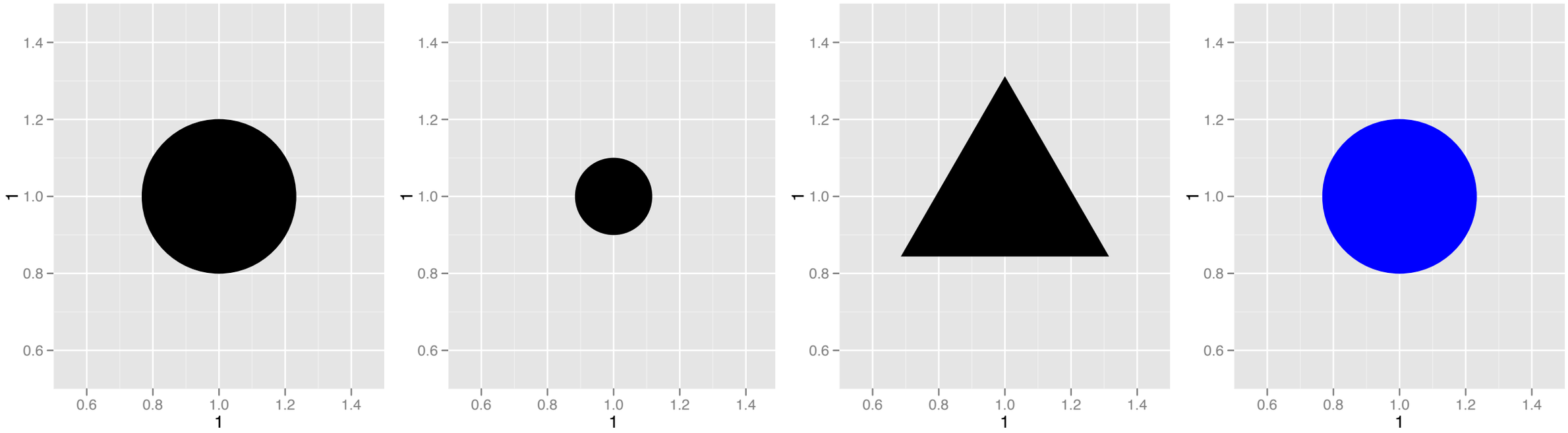
Additional variables

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

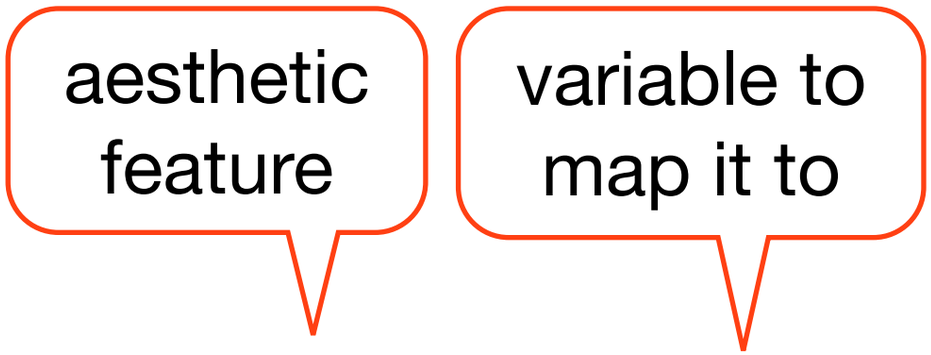
Aesthetics

Aesthetics

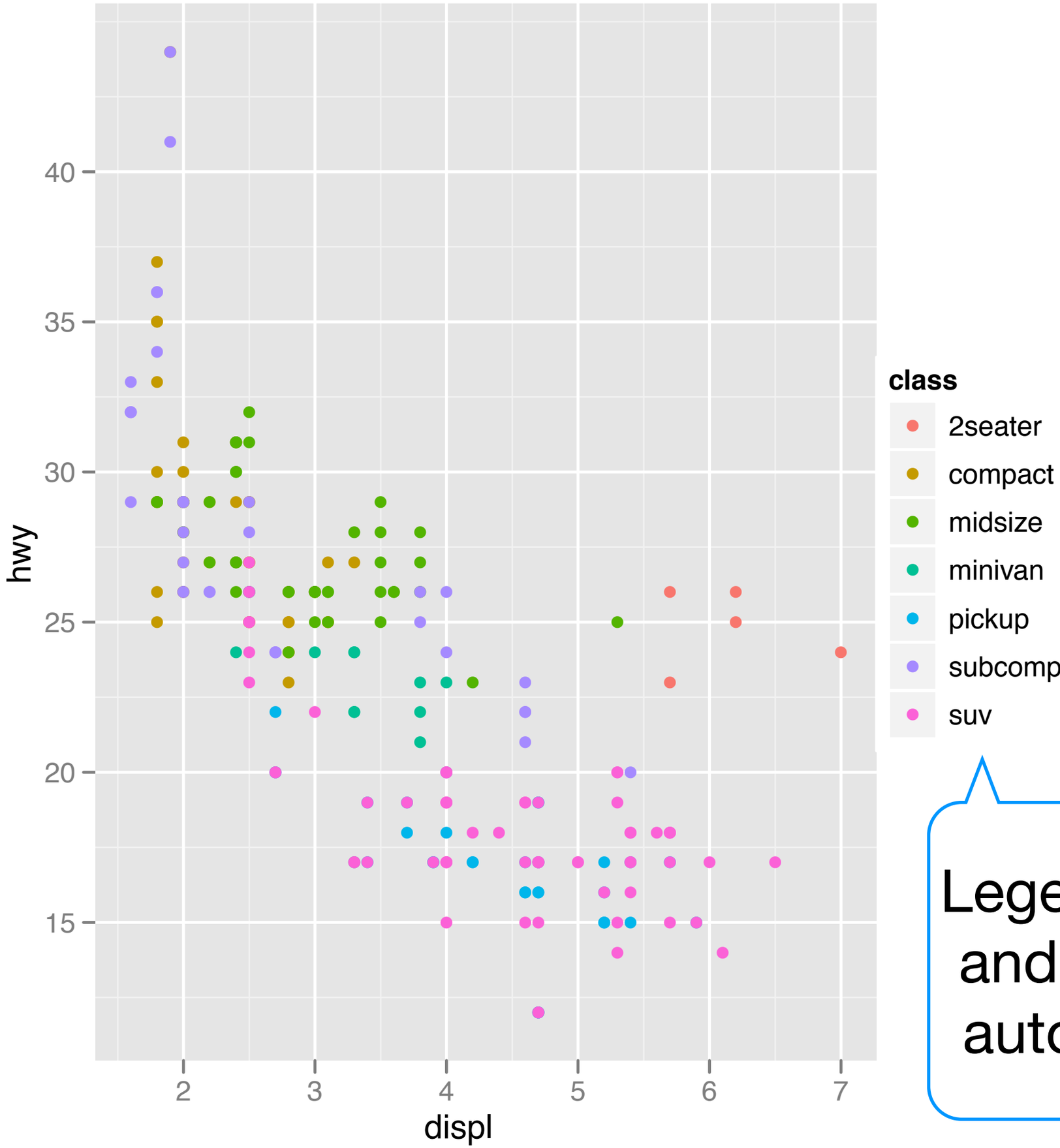
Visual characteristics that can be mapped to data



Aesthetics




```
qplot(displ, hwy, data = mpg, color = class)
qplot(displ, hwy, data = mpg, size = class)
qplot(displ, hwy, data = mpg, shape = class)
qplot(displ, hwy, data = mpg, alpha = class)
```



Legend chosen and displayed automatically.

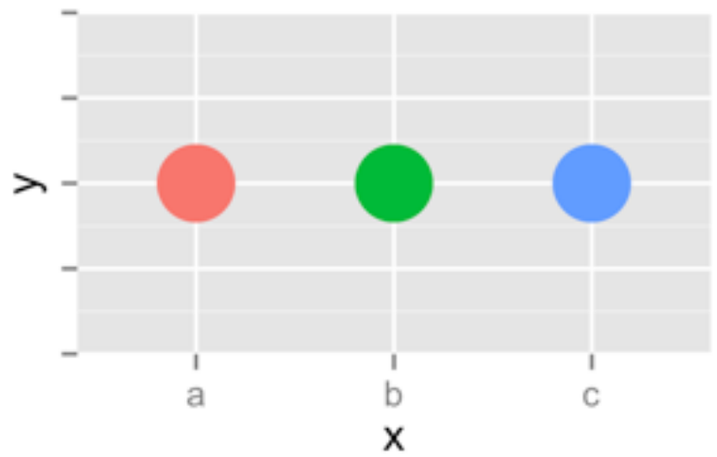
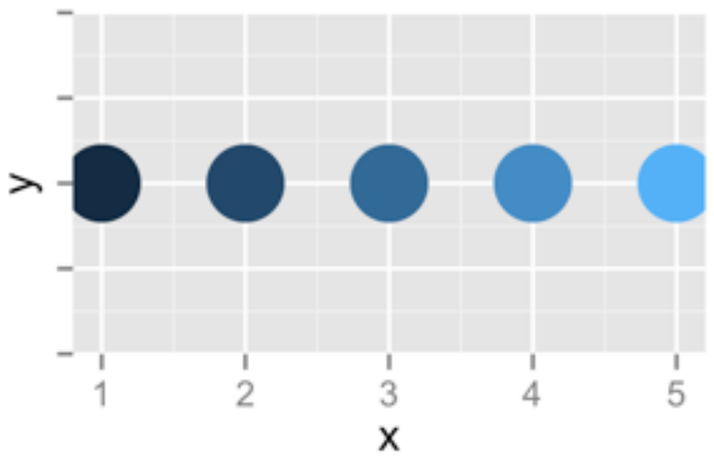
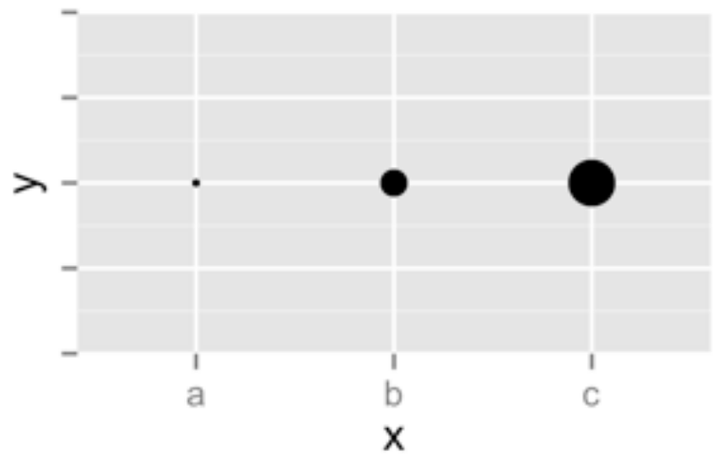
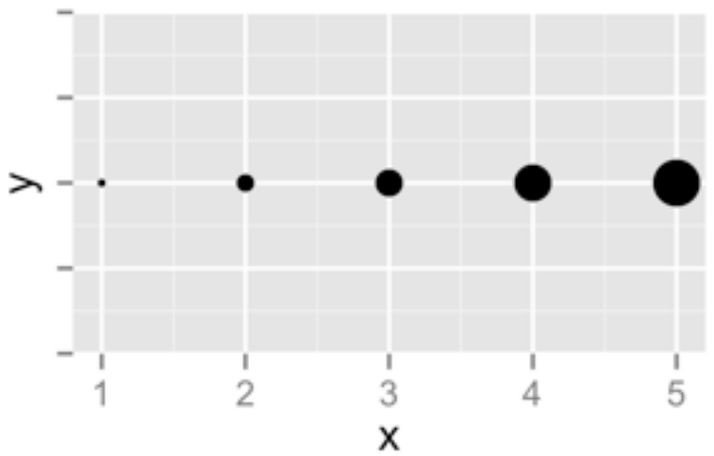
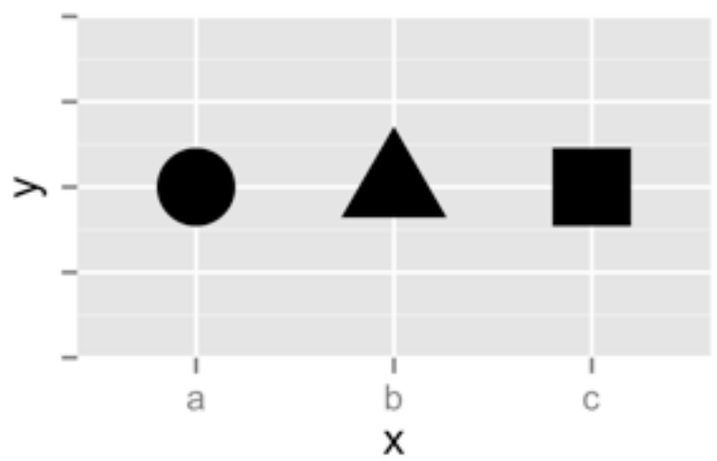
```
qplot(displ, hwy, data = mpg, color = class)
```

Your turn

Add color, size, and shape aesthetics to your graph. Experiment.

Do different things happen for discrete and continuous variables?

What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		

	Discrete	Continuous
Color	Rainbow of colors	Gradient from light blue to dark blue
Size	Discrete size steps	Linear mapping between radius and value
Shape	Different shape for each	Shouldn’t (and doesn’t) work

Faceting

Faceting

Smaller plots that display different subsets of the data.

Also 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

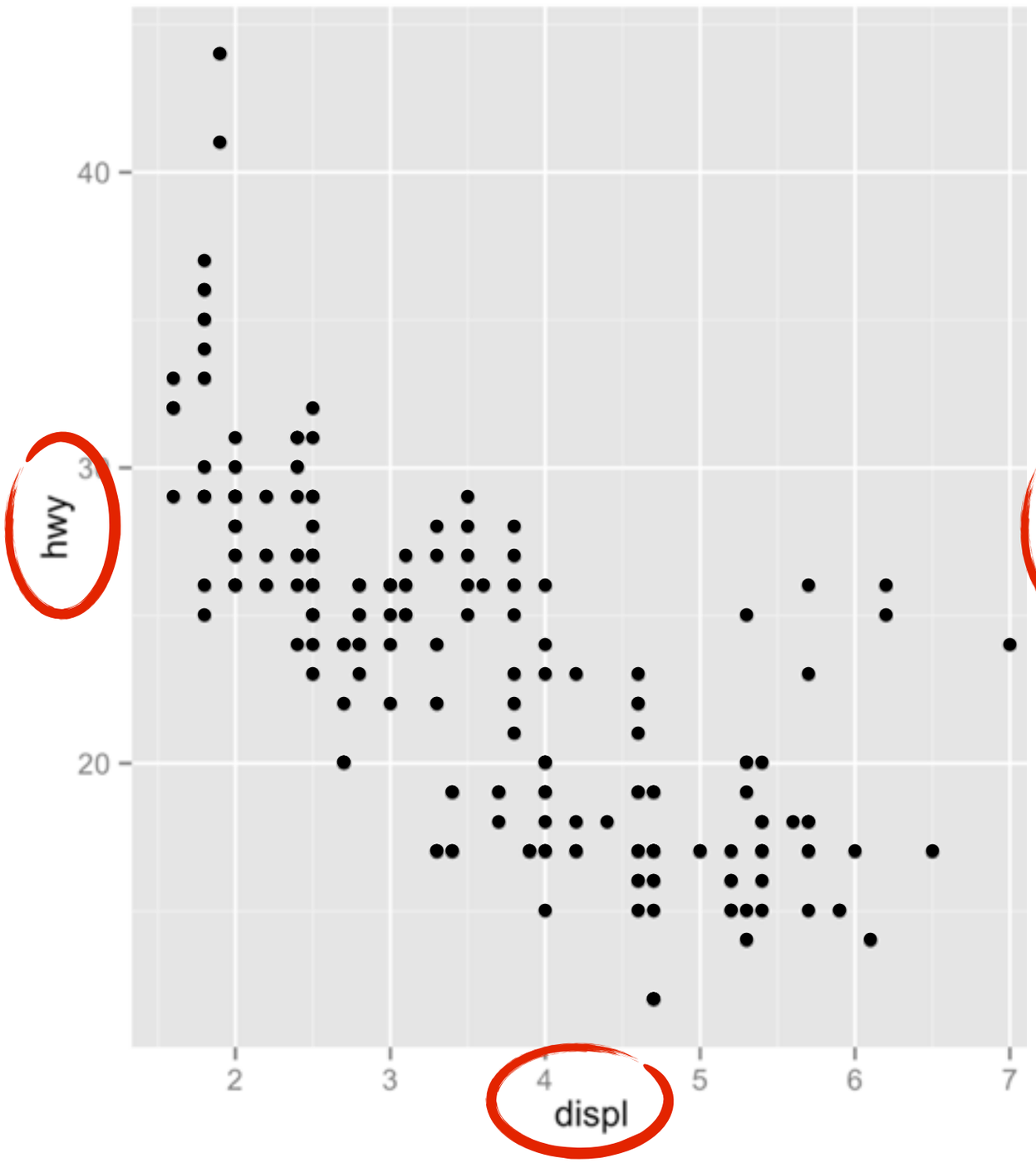
`facet_grid()`: 2d grid, rows ~ cols, . for no split

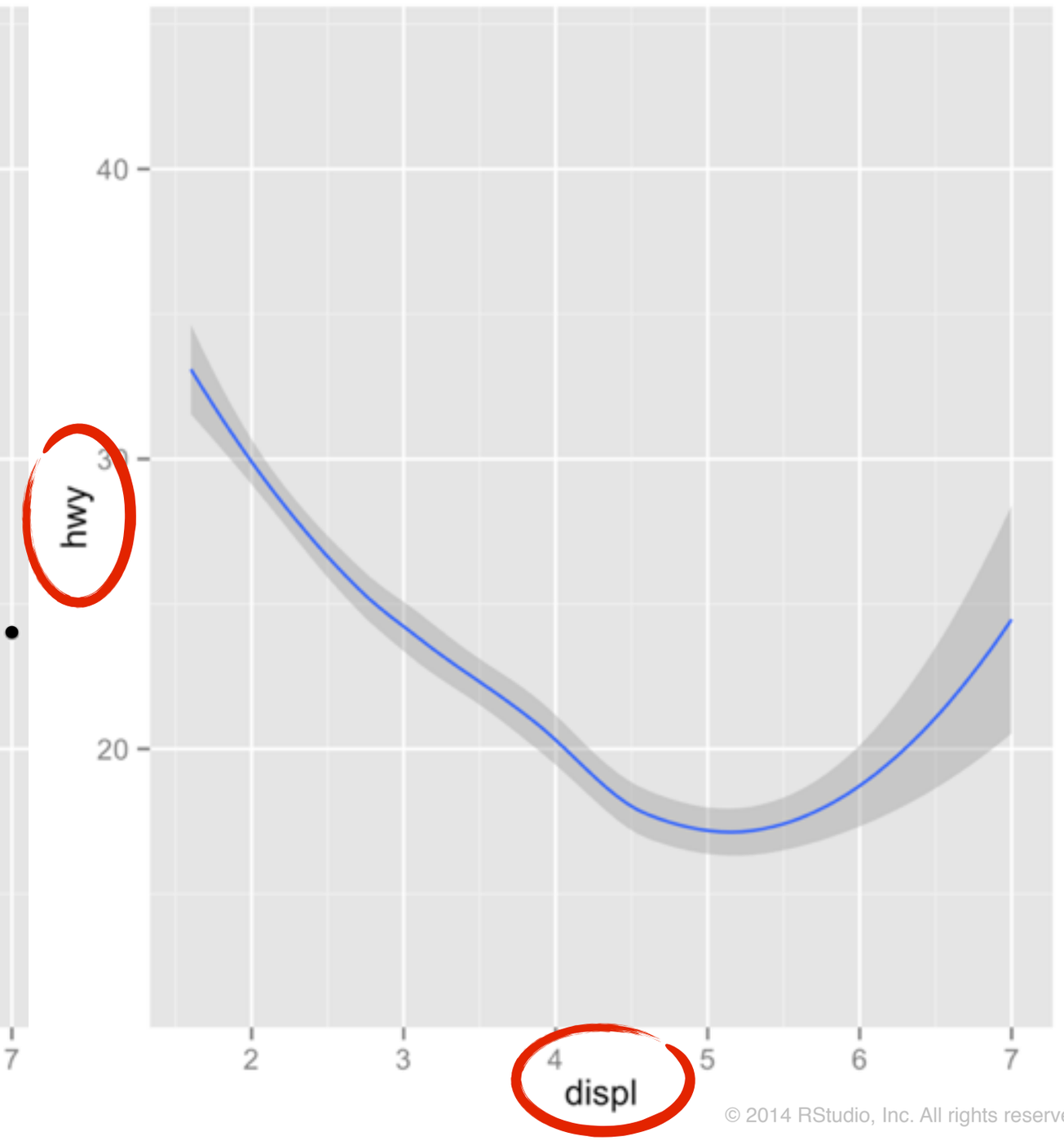
`facet_wrap()`: 1d ribbon wrapped into 2d

Geoms

How are these plots similar?

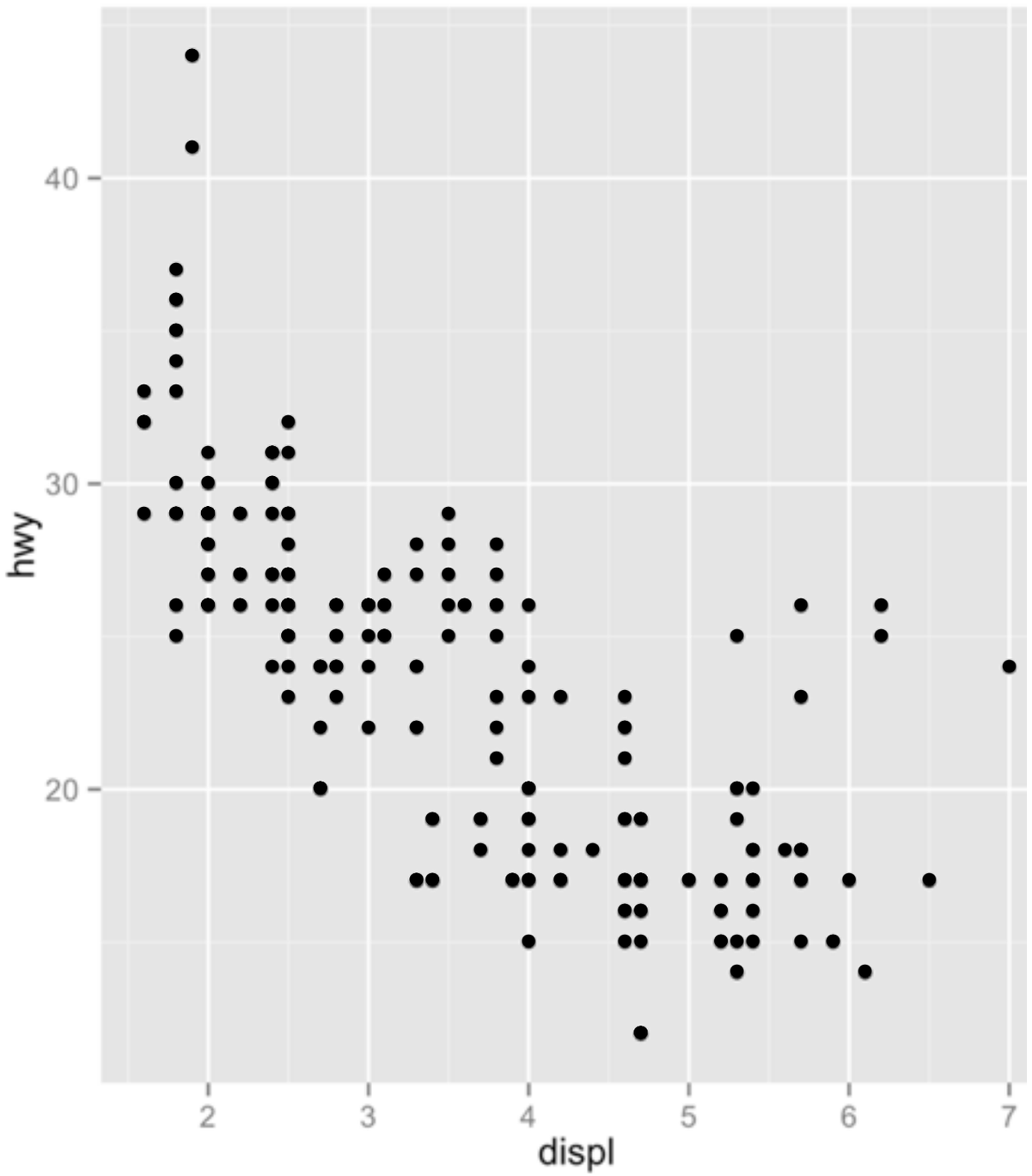
Same: x var, y var, data

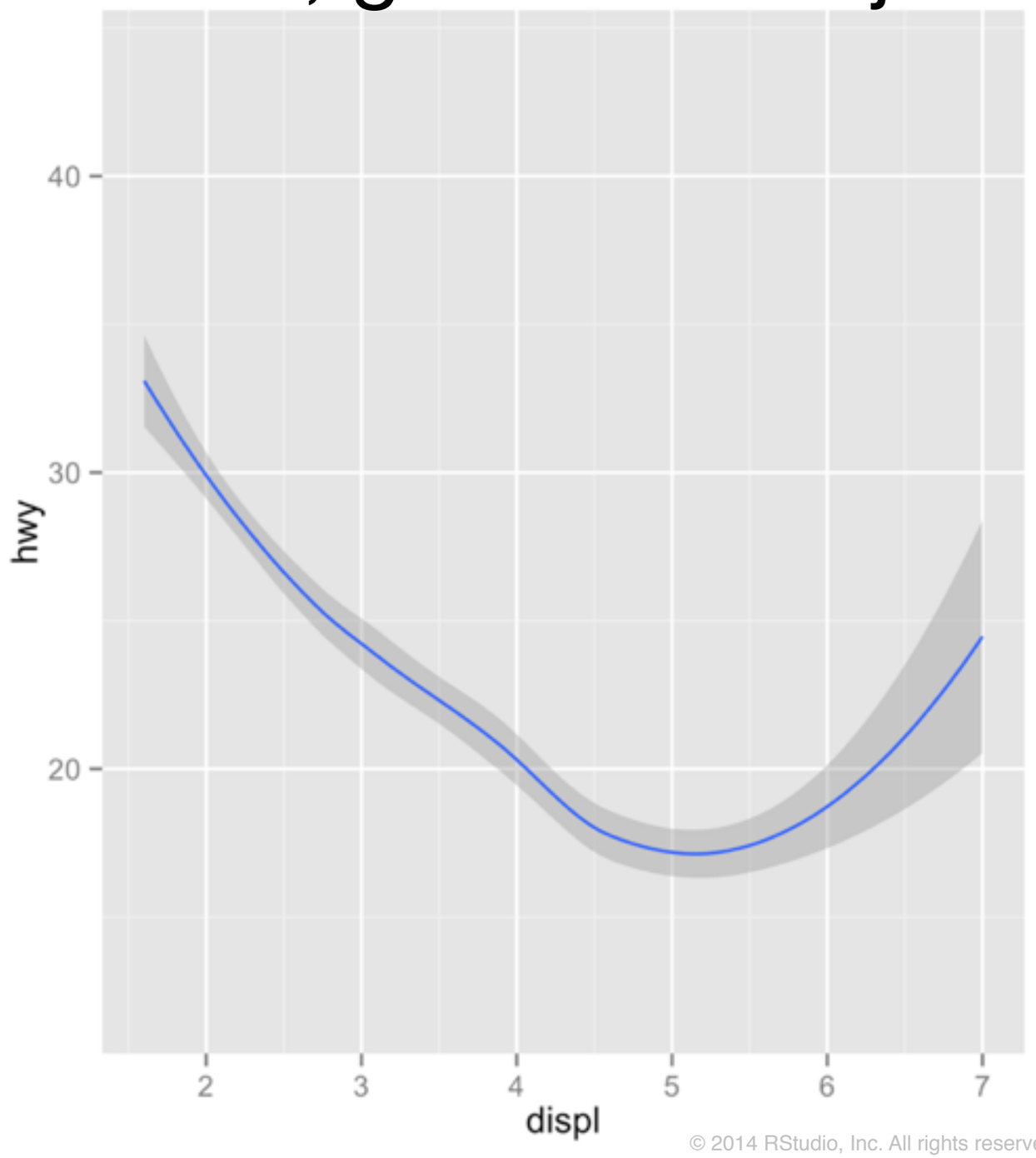




How are these plots different?

Different: "type" of plot
i.e, what plot draws
i.e, geometric object





Geometric object

the "type" of graph, or
what the graph draws

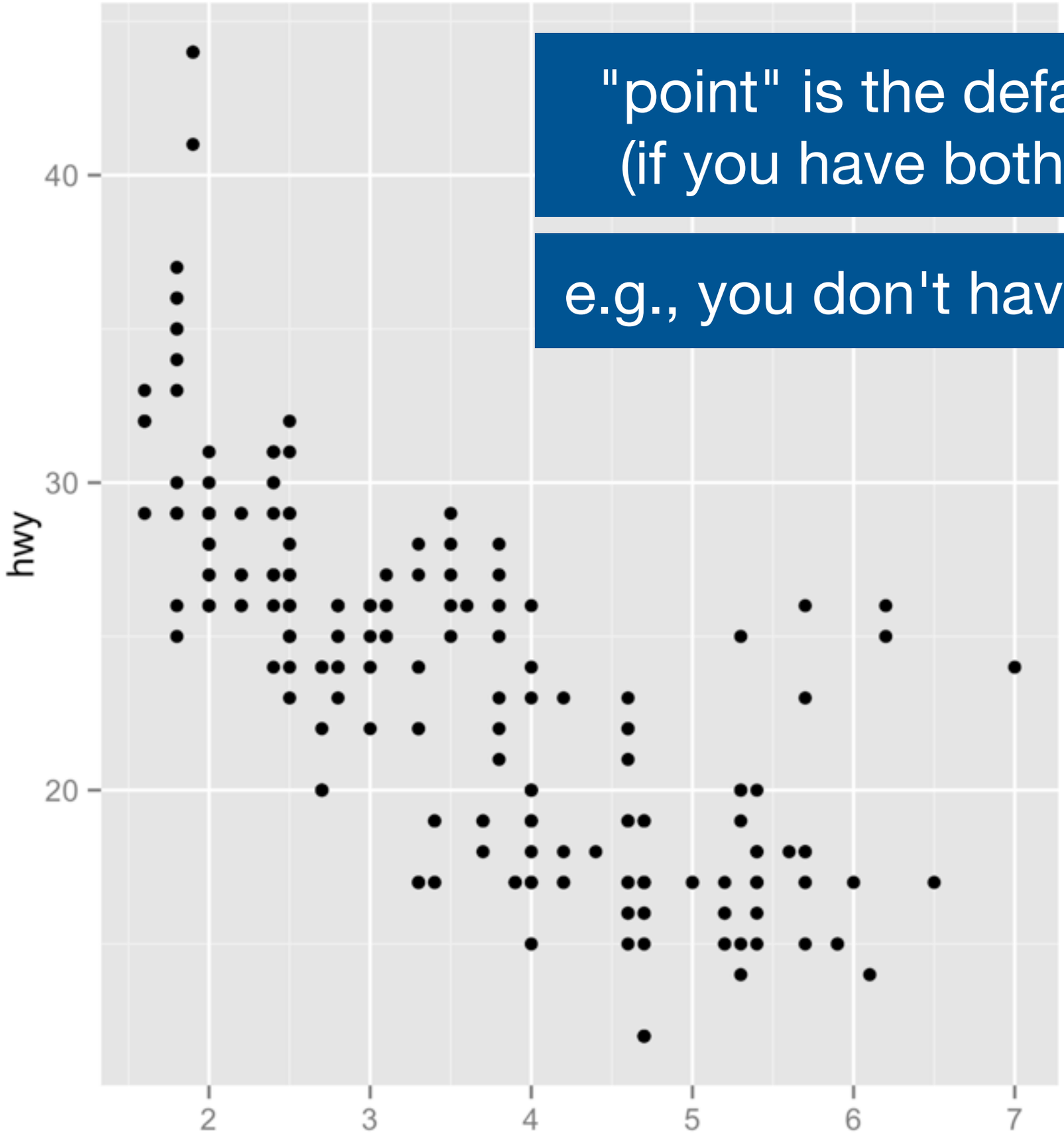
x variable

y variable

data set
variables are in

type of plot

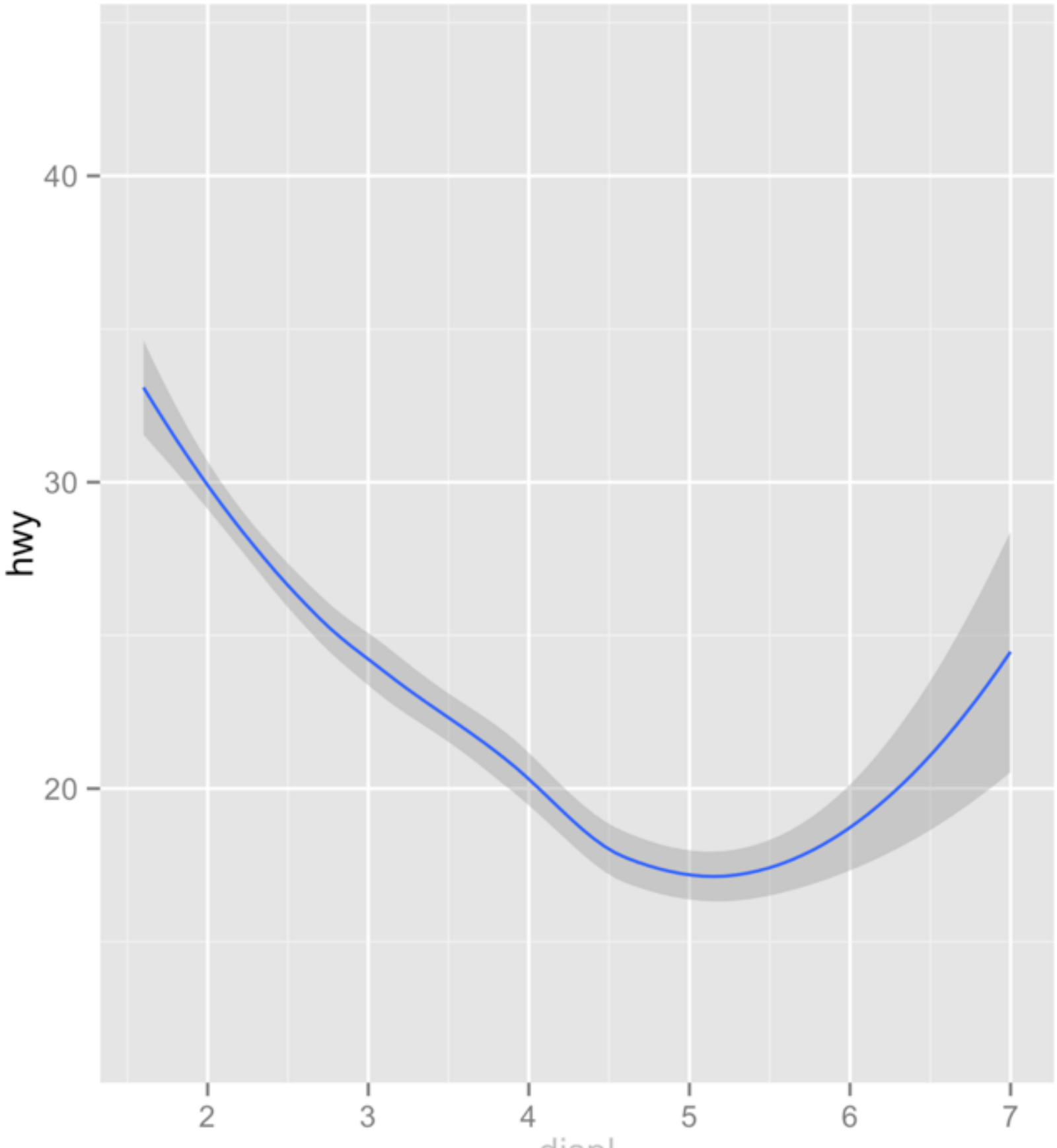
```
qplot(displ,hwy,data = mpg,geom = "smooth")
```



"point" is the default geom
(if you have both x and y)

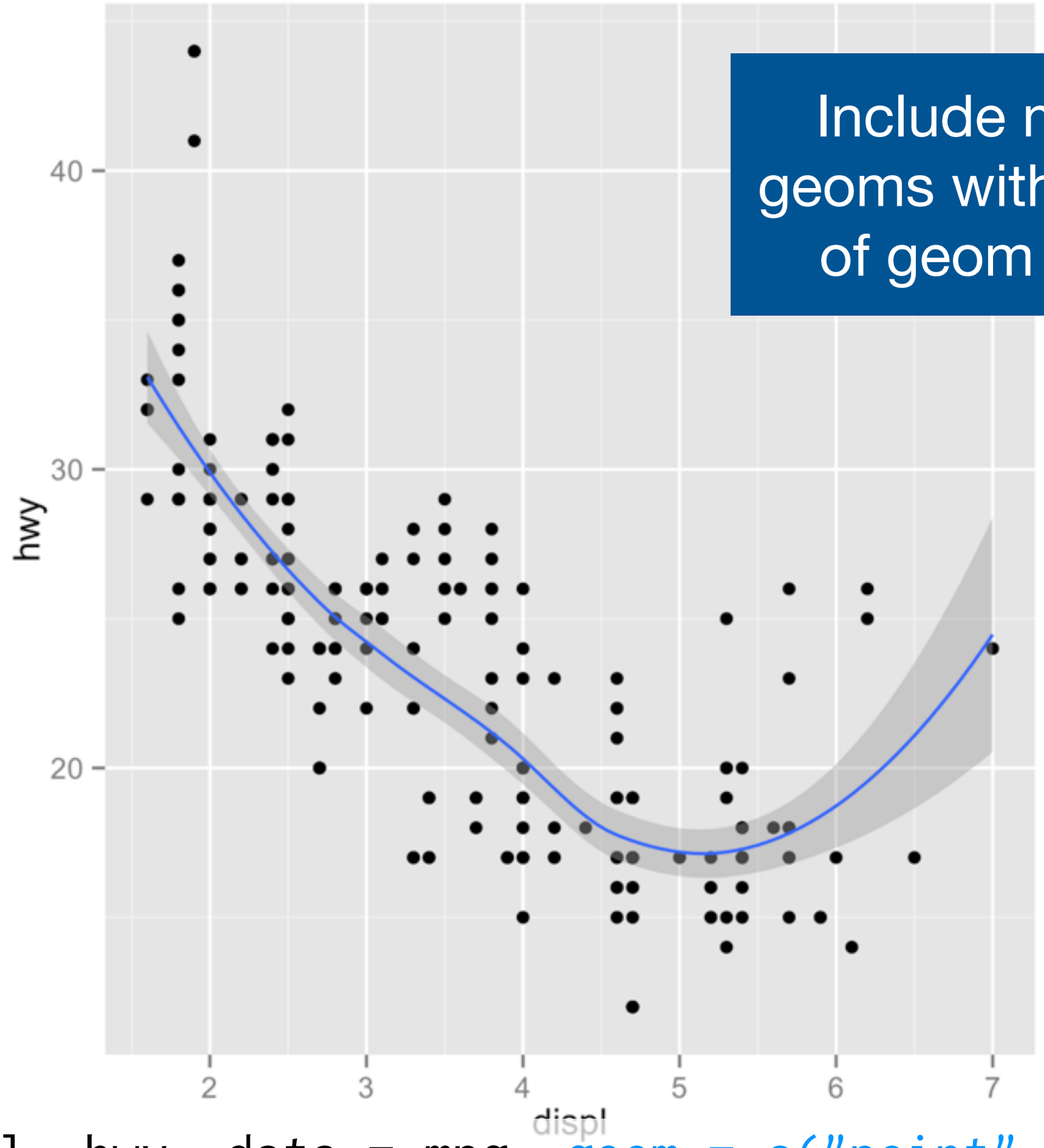
e.g., you don't have to type it

```
qplot(displ, hwy, data = mpg) geom = "point")
```




```
qplot(displ, hwy, data = mpg, geom = "smooth")
```

Include multiple geoms with a vector of geom names

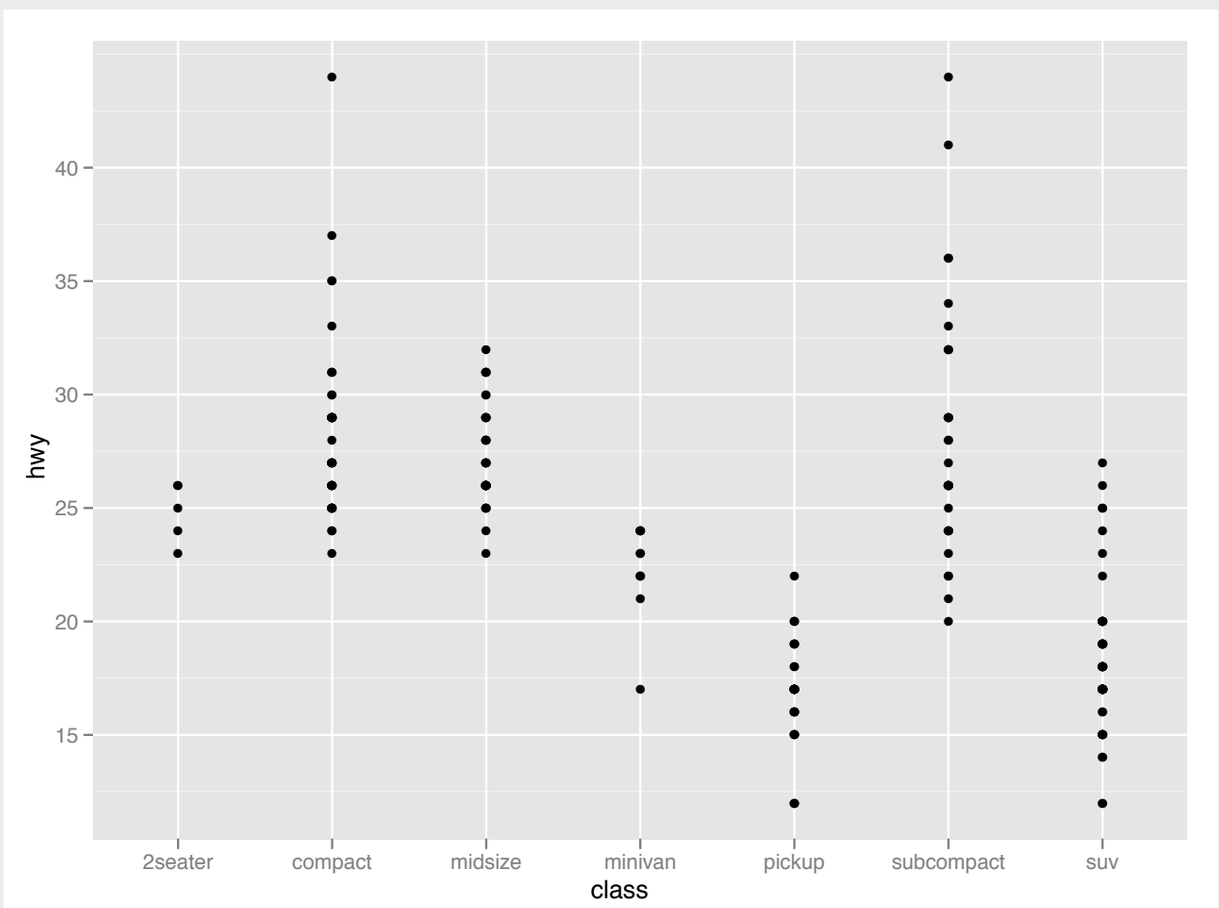


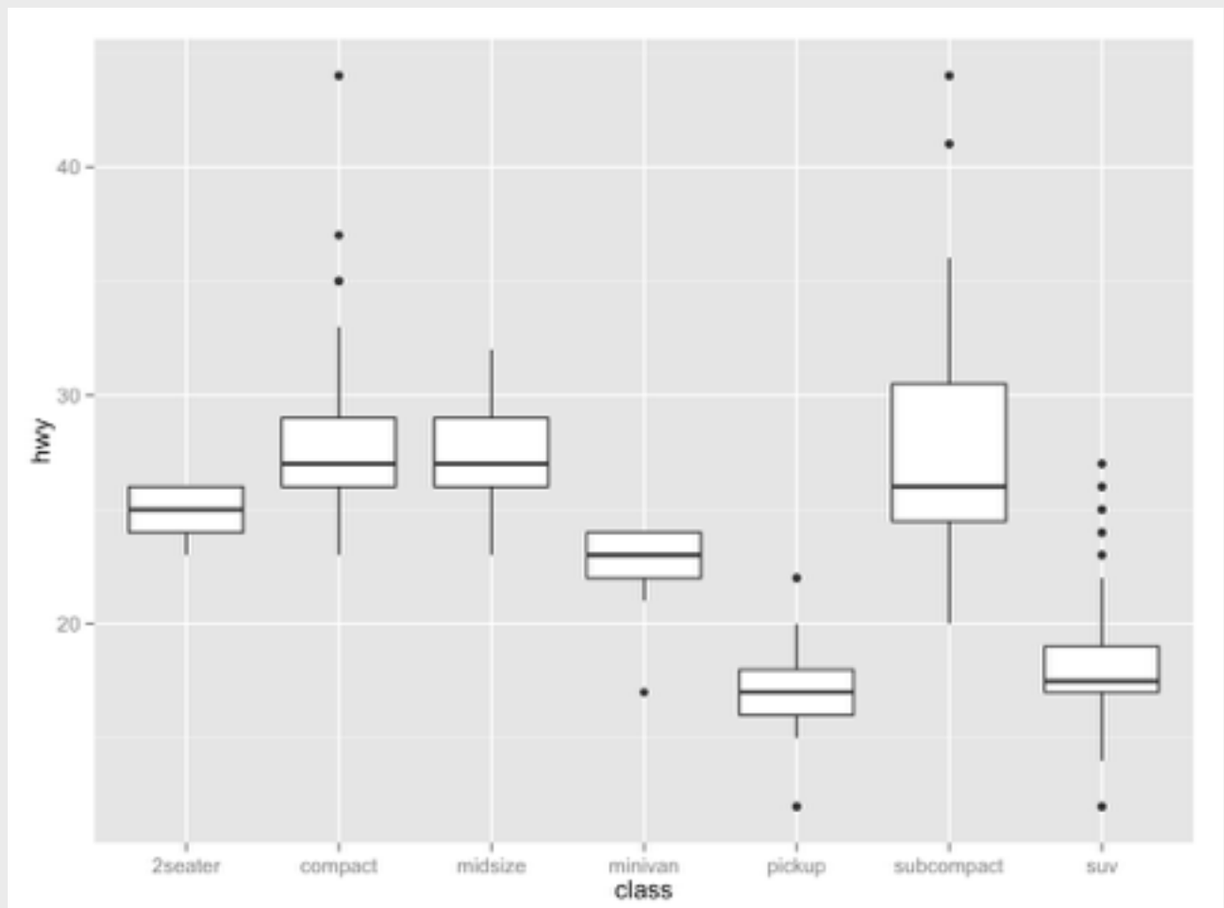

```
qplot(displ, hwy, data = mpg, geom = c("point", "smooth"))
```

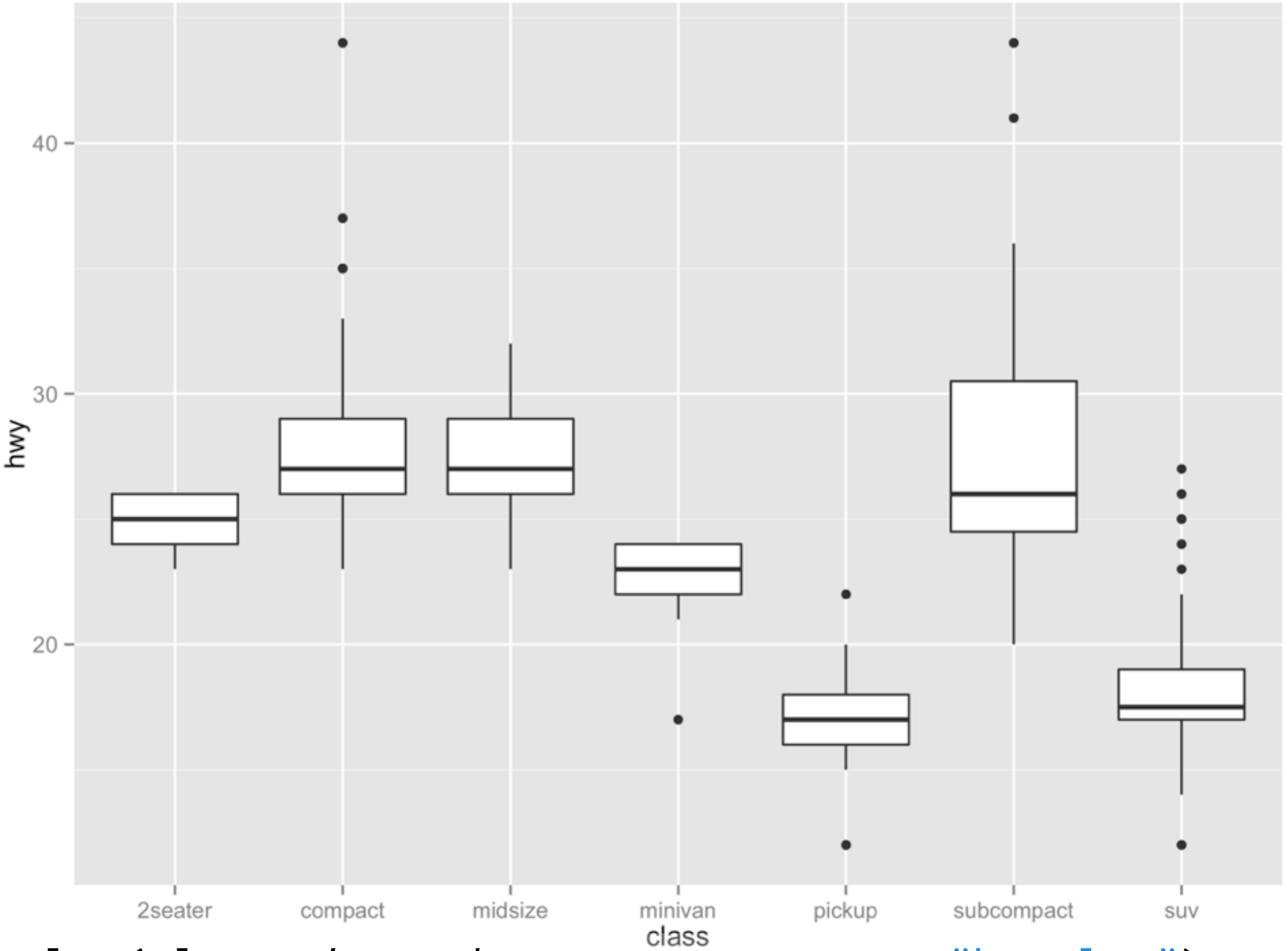
Your turn

How would you replace this scatterplot with one that draws boxplots? Try out your best guess.

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

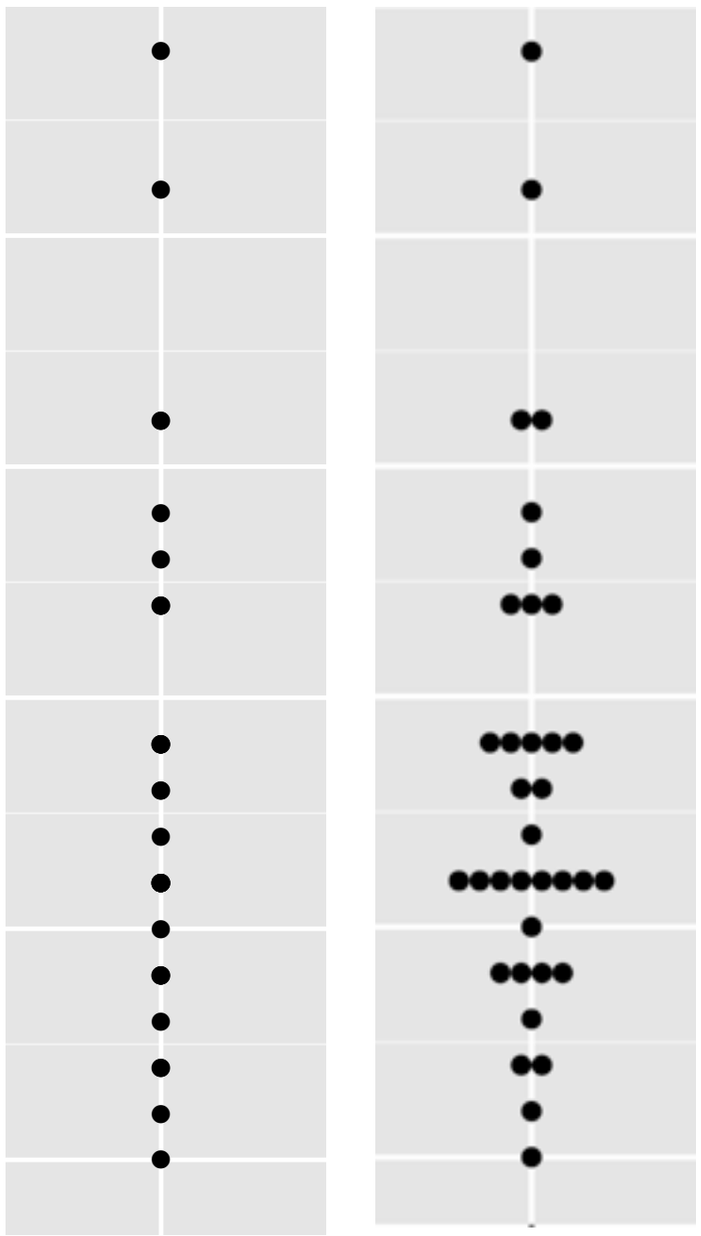




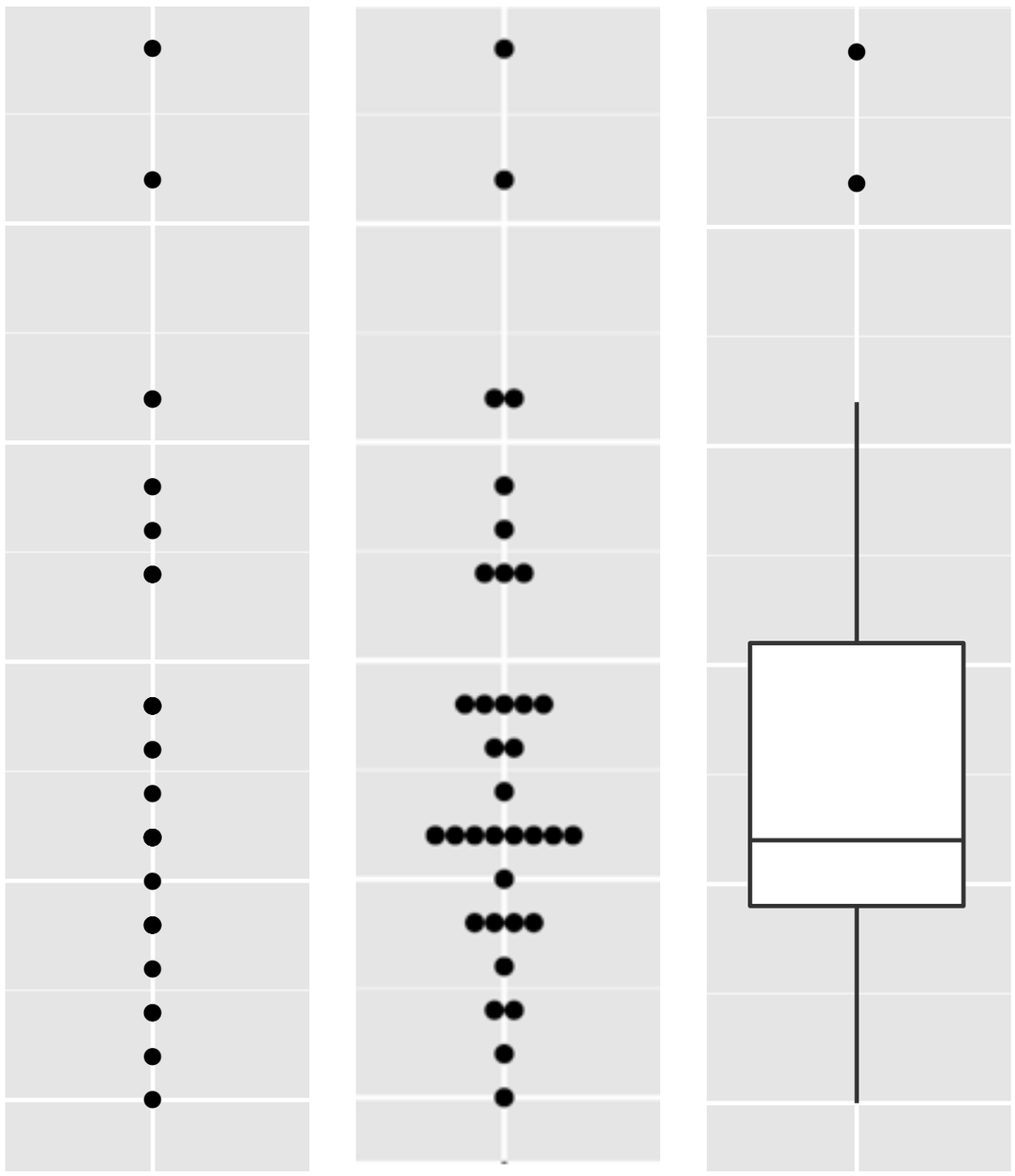



```
qplot(class, hwy, data = mpg, geom = "boxplot")
```

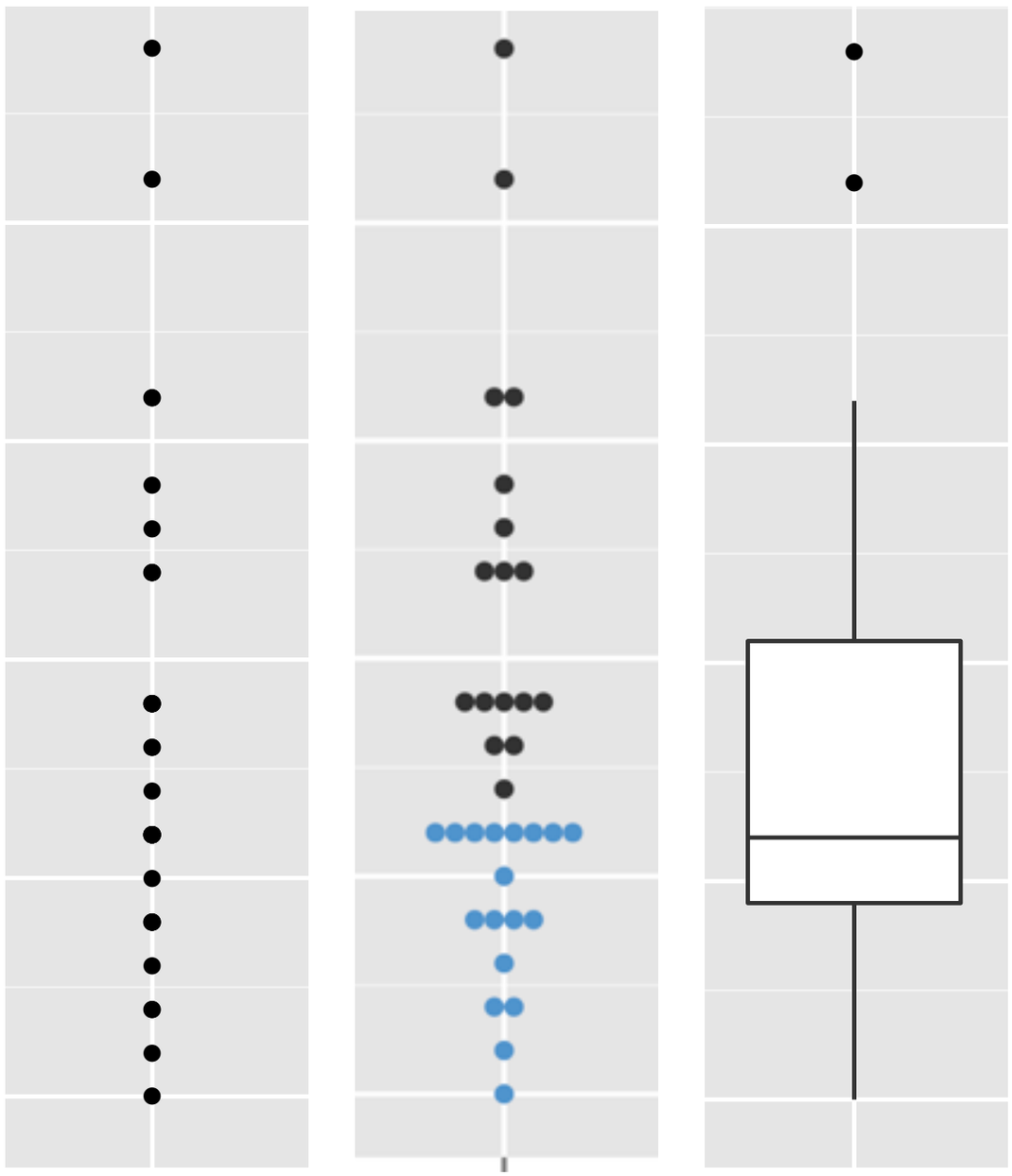
boxplots



boxplots

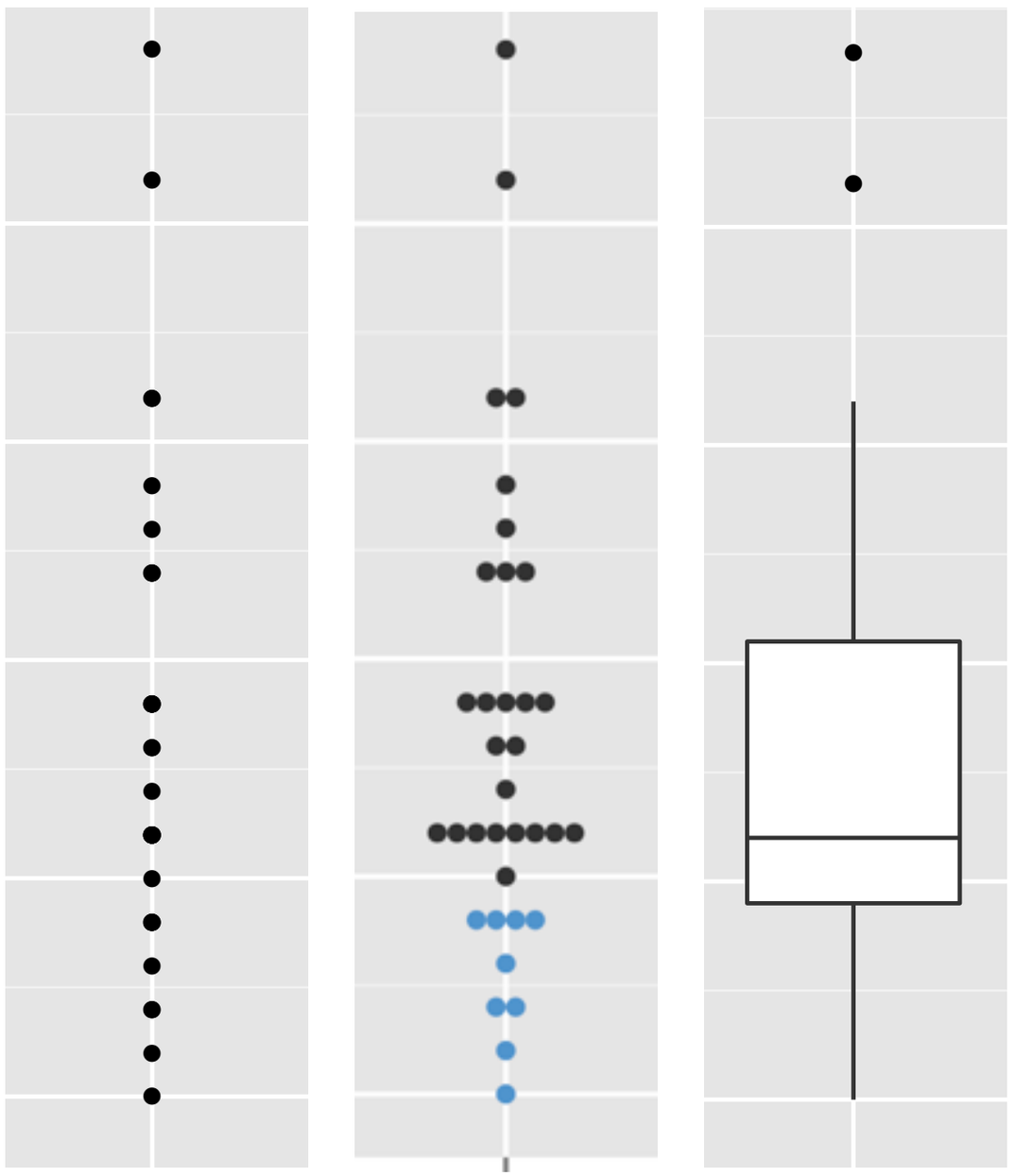


boxplots

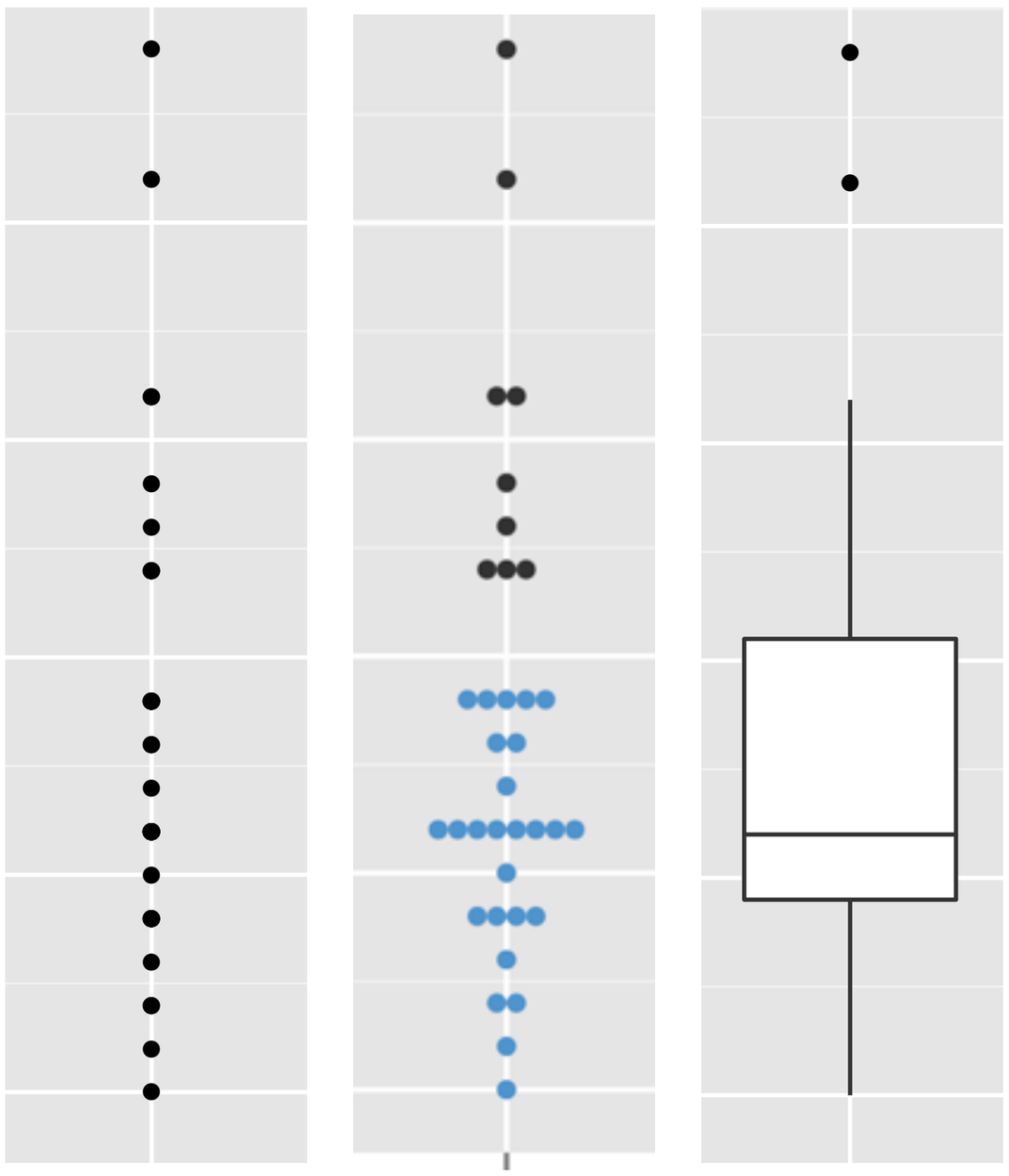


median
(50th percentile)

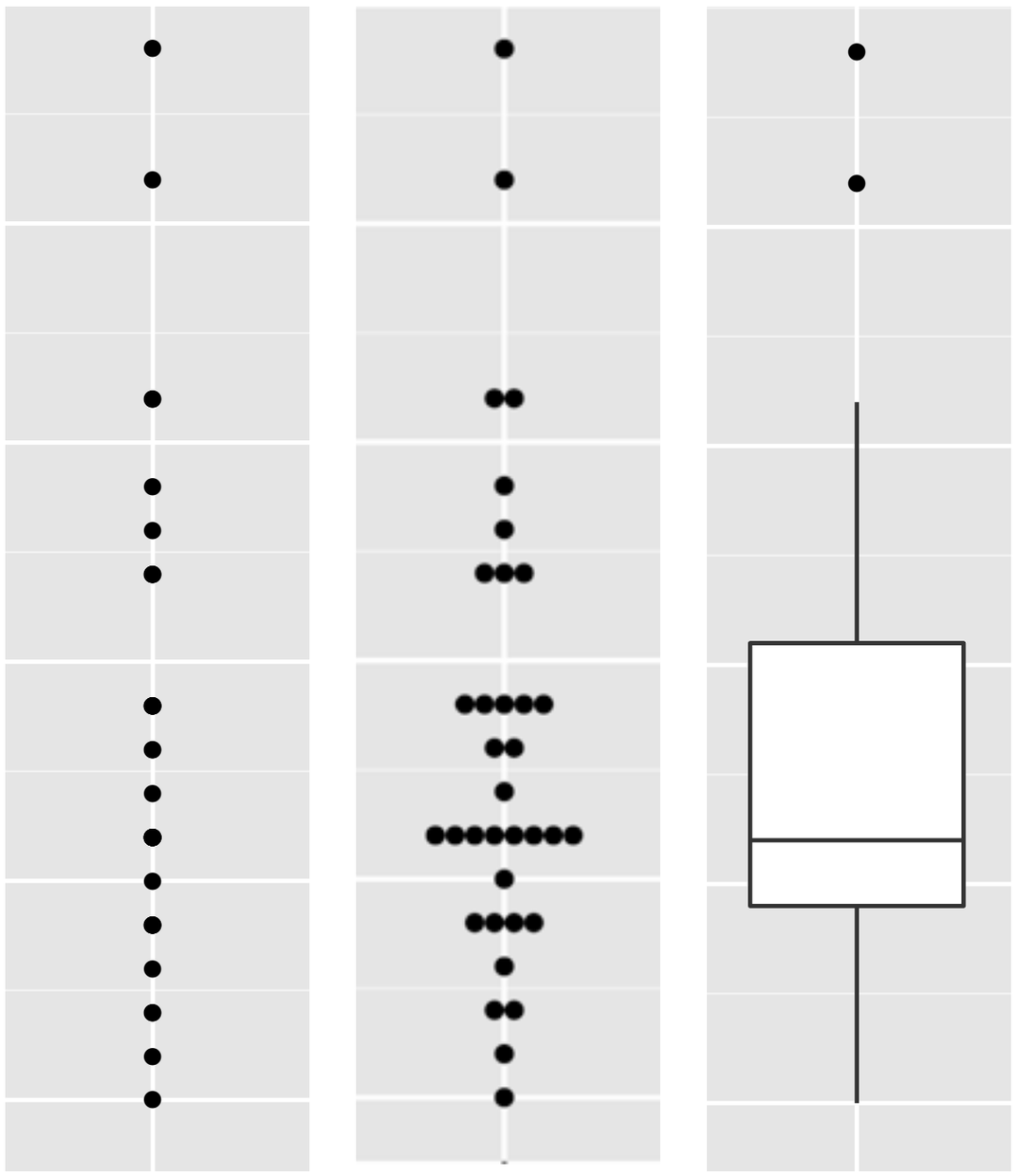
boxplots



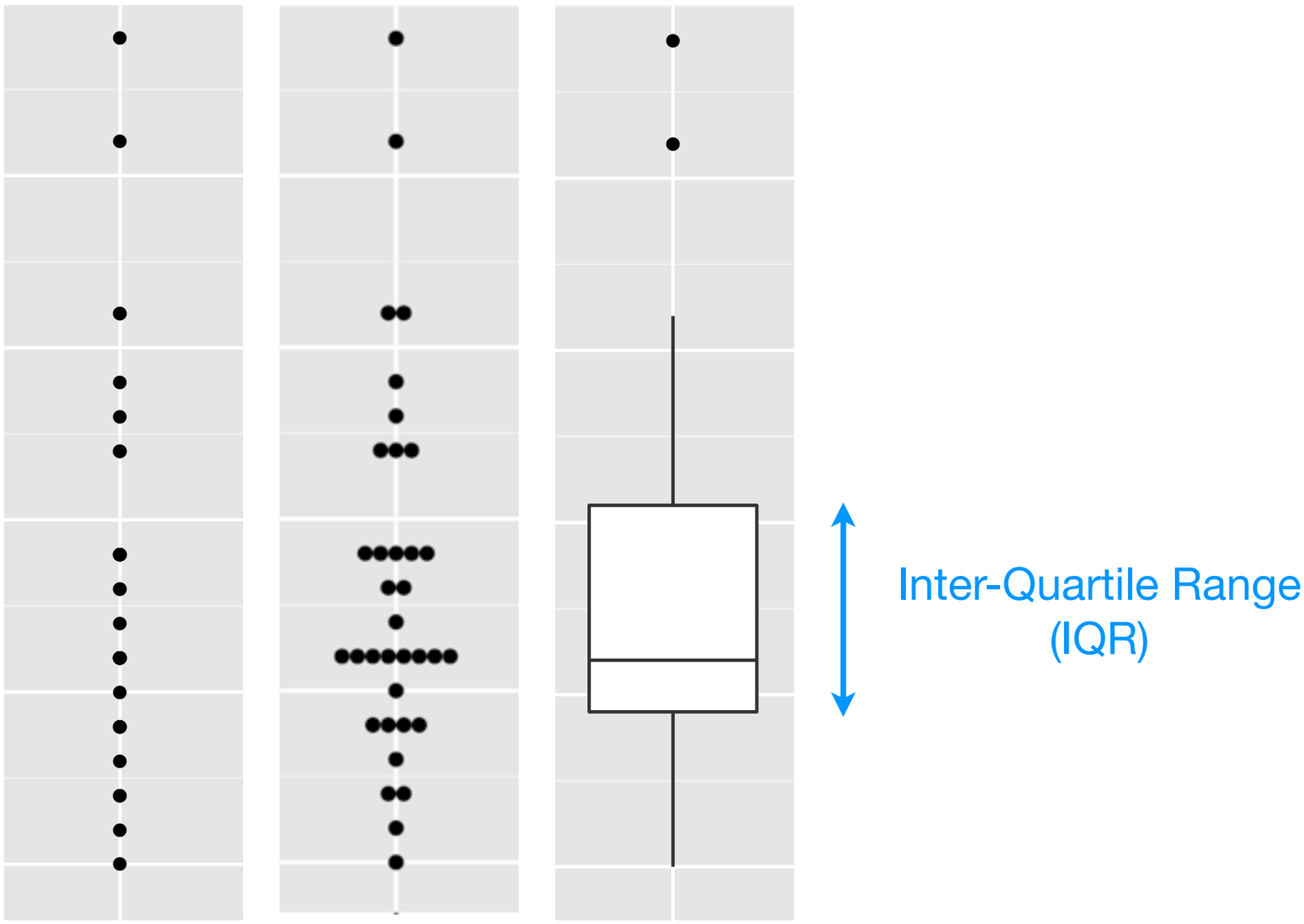
boxplots



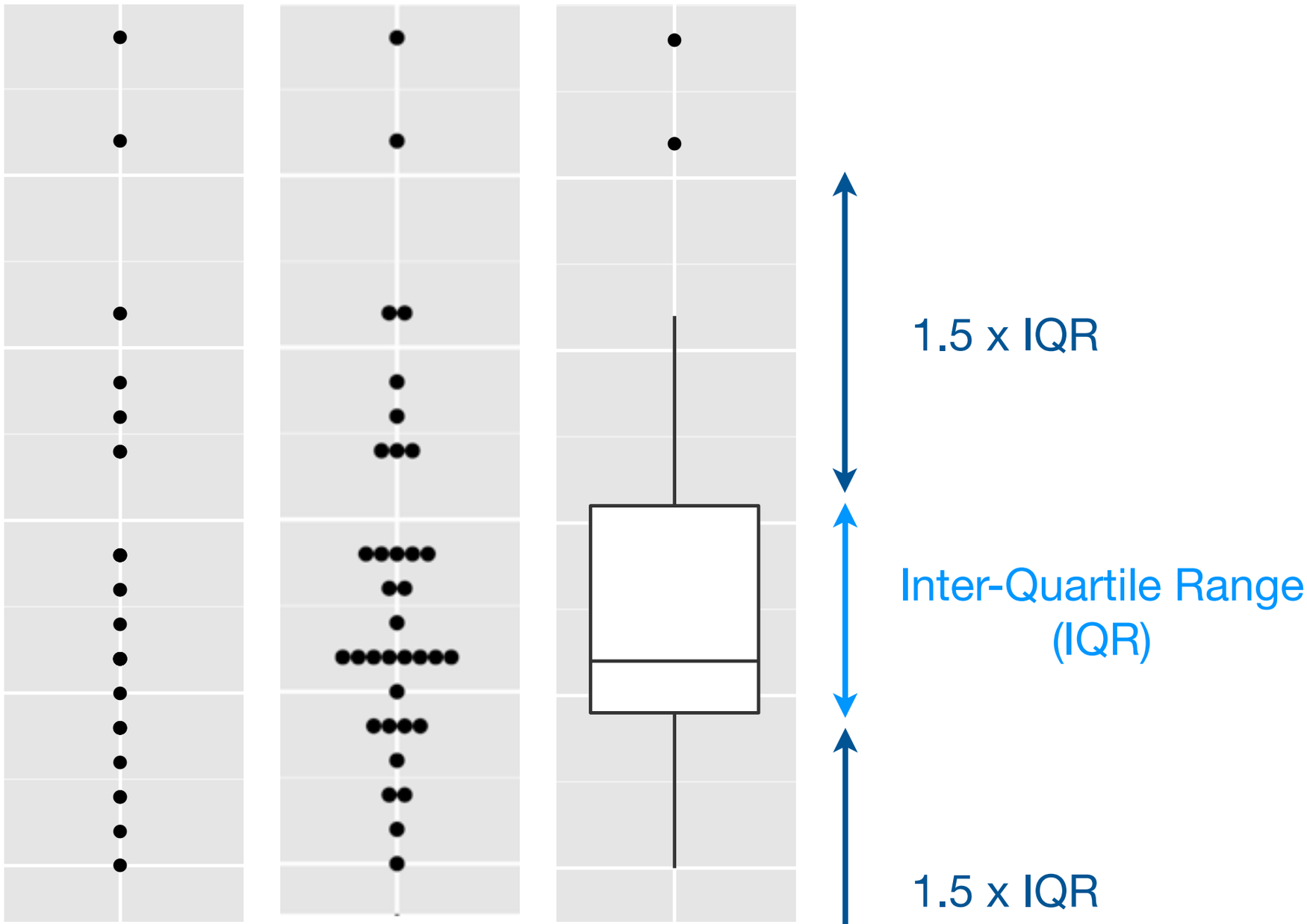
boxplots



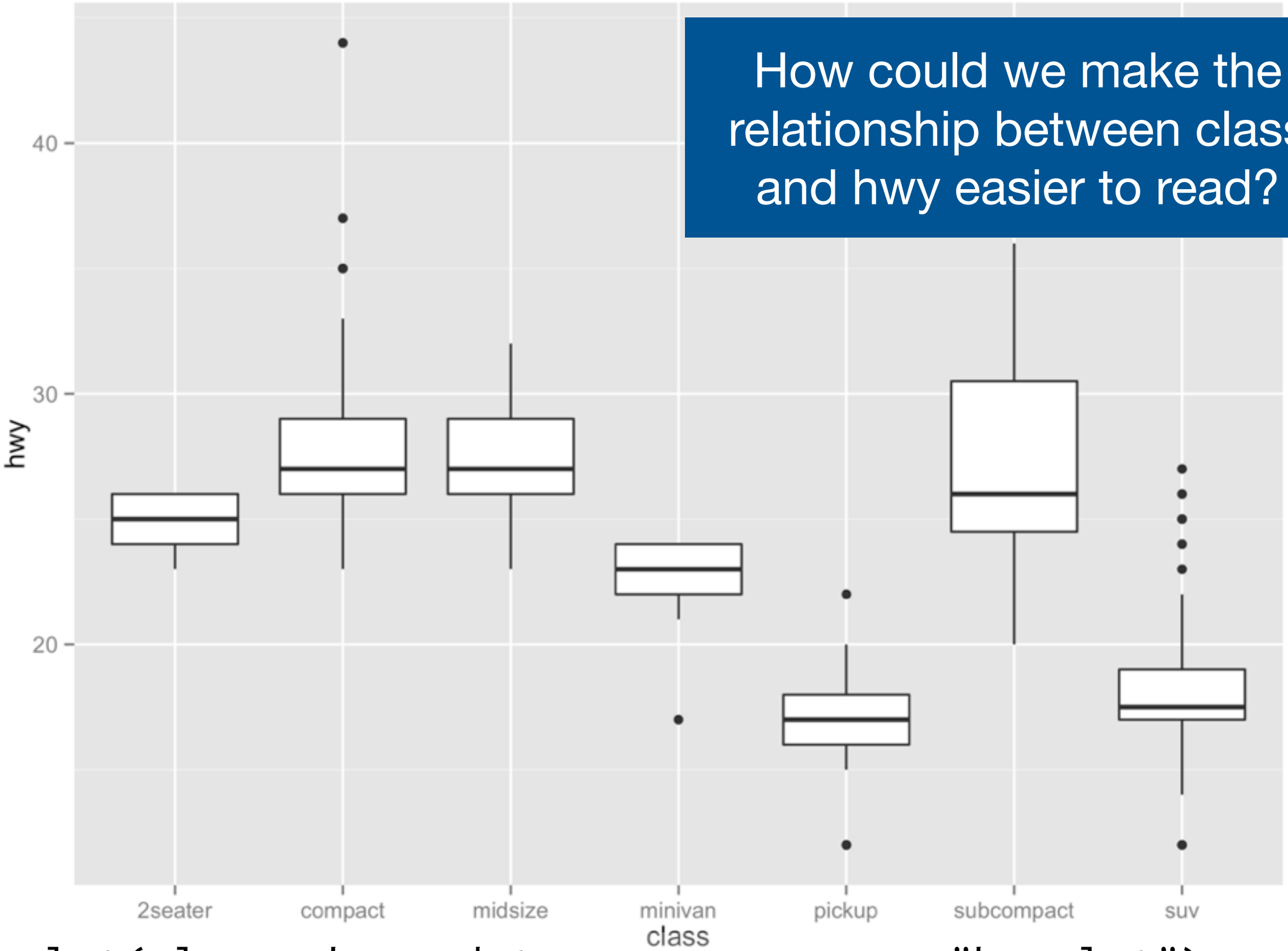
boxplots



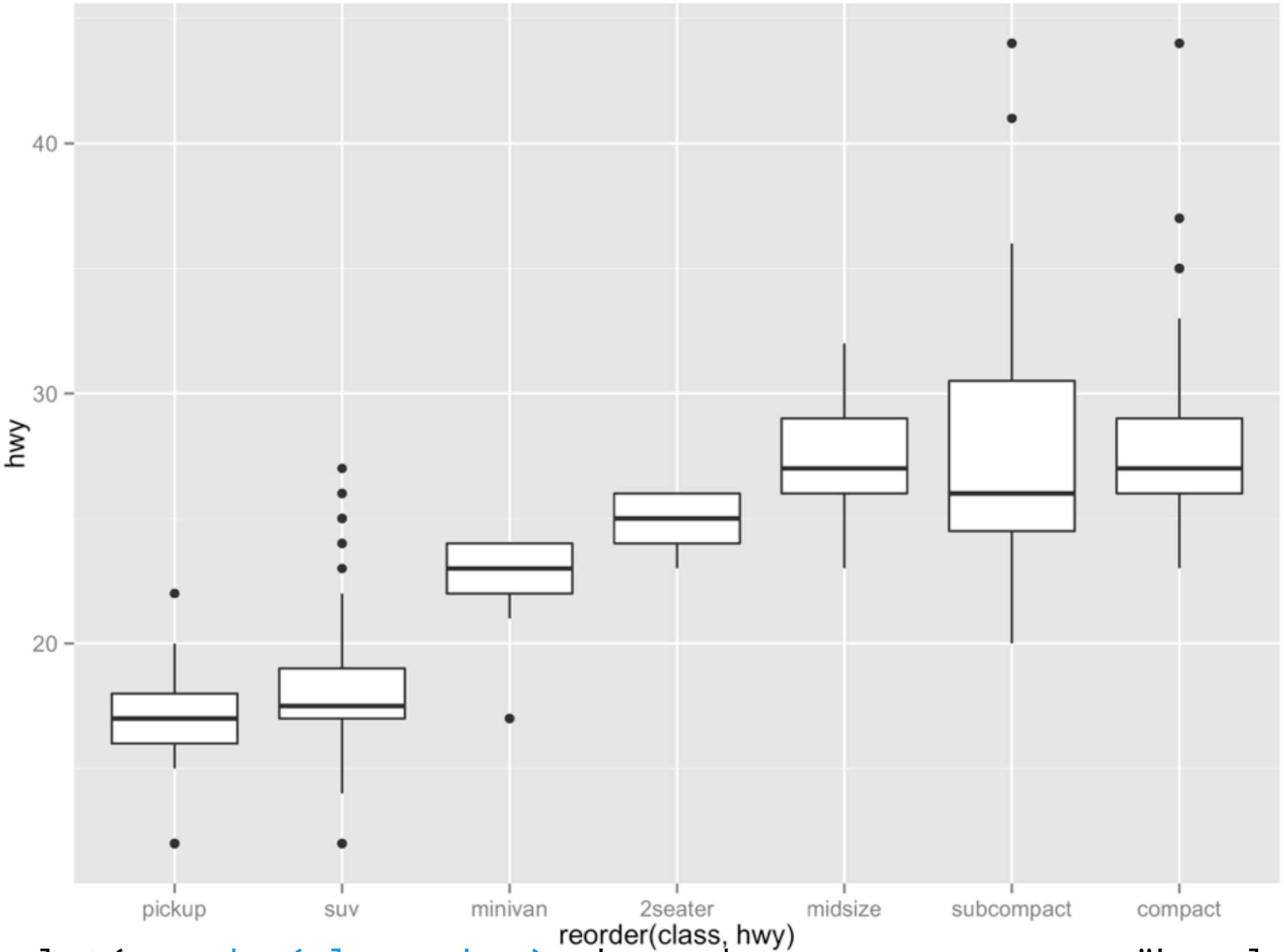
boxplots



How could we make the relationship between class and hwy easier to read?




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




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

Your turn

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

Help pages

Tips:

- scan page for relevant info
- ignore things that don't make sense
- try out the examples

Description

Useful overview

Usage

Good place to spot default values

Arguments

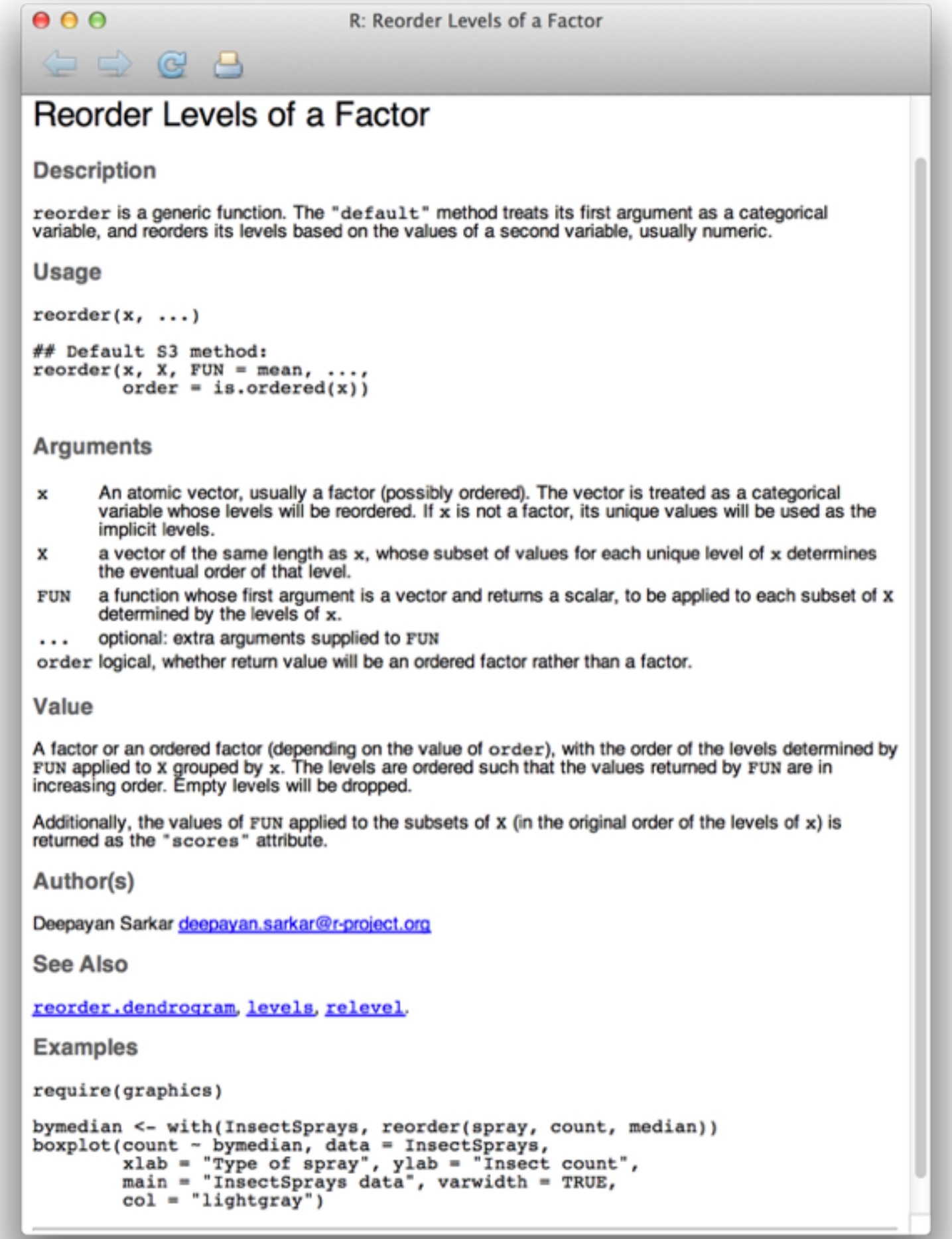
explanation of each argument

Value

what the function returns

Examples

Most helpful section!



R: Reorder Levels of a Factor

Reorder Levels of a Factor

Description

`reorder` is a generic function. The `"default"` method treats its first argument as a categorical variable, and reorders its levels based on the values of a second variable, usually numeric.

Usage

```
reorder(x, ...)

## Default S3 method:
reorder(x, X, FUN = mean, ...,
        order = is.ordered(x))
```

Arguments

`x`	An atomic vector, usually a factor (possibly ordered). The vector is treated as a categorical variable whose levels will be reordered. If `x` is not a factor, its unique values will be used as the implicit levels.
`X`	a vector of the same length as `x`, whose subset of values for each unique level of `x` determines the eventual order of that level.
`FUN`	a function whose first argument is a vector and returns a scalar, to be applied to each subset of `X` determined by the levels of `x`.
`...`	optional: extra arguments supplied to `FUN`
`order`	logical, whether return value will be an ordered factor rather than a factor.

Value

A factor or an ordered factor (depending on the value of `order`), with the order of the levels determined by `FUN` applied to `X` grouped by `x`. The levels are ordered such that the values returned by `FUN` are in increasing order. Empty levels will be dropped.

Additionally, the values of `FUN` applied to the subsets of `X` (in the original order of the levels of `x`) is returned as the `"scores"` attribute.

Author(s)

Deepayan Sarkar deepayan.sarkar@r-project.org

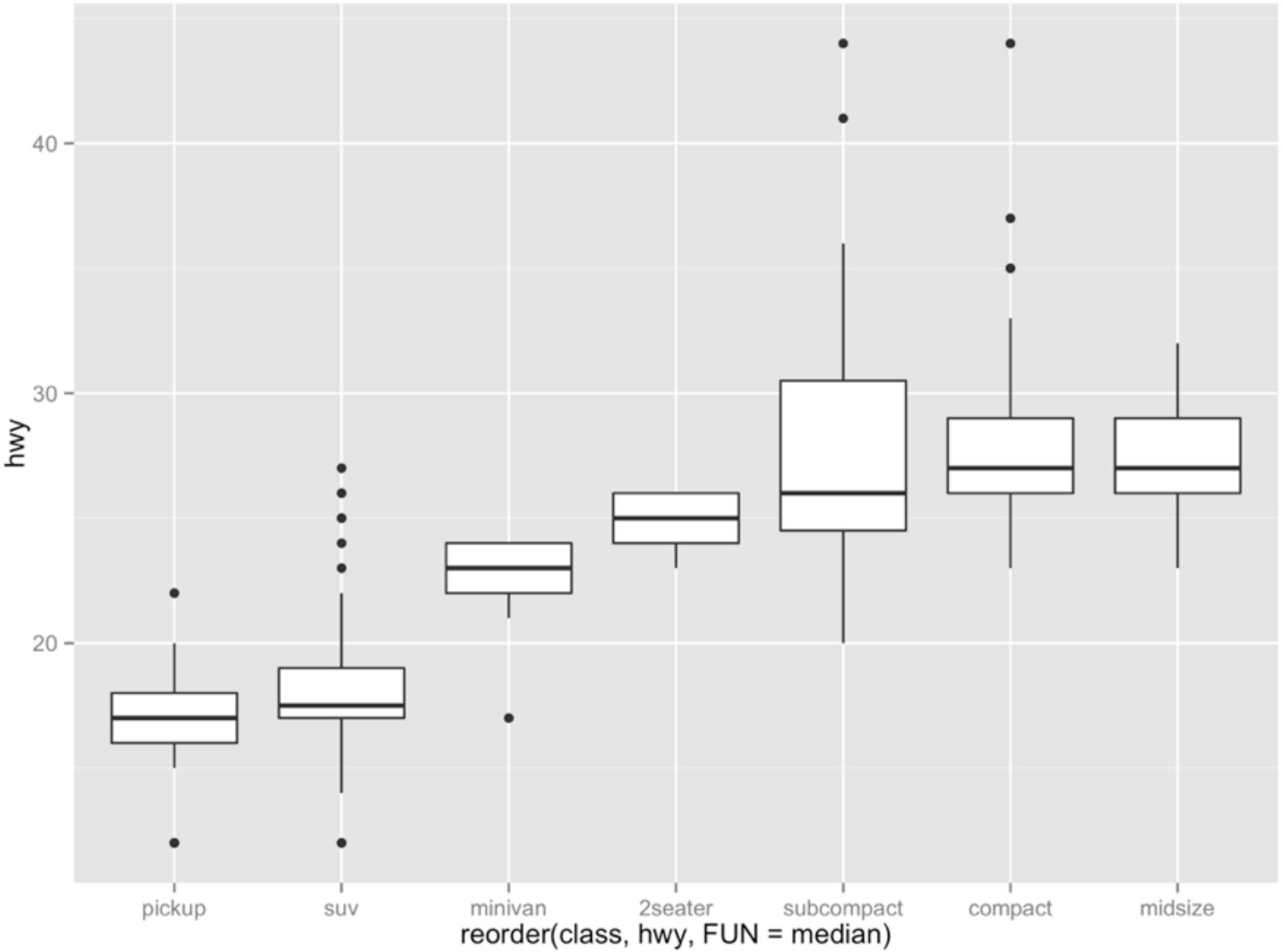
See Also

`reorder.dendrogram`, `levels`, `relevel`.

Examples

```
require(graphics)

bymedian <- with(InsectSprays, reorder(spray, count, median))
boxplot(count ~ bymedian, data = InsectSprays,
        xlab = "Type of spray", ylab = "Insect count",
        main = "InsectSprays data", varwidth = TRUE,
        col = "lightgray")
```




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

Index. ggplot2 0.9.2.1

docs.ggplot2.org/current/

ggplot2 0.9.2.1 Index

Help topics

Geoms

Geoms, short for geometric objects, describe the type of plot you will produce.

- `geom_abline`
  Line specified by slope and intercept.
- `geom_area`
  Area plot.
- `geom_bar`
  Bars, rectangles with bases on x-axis
- `geom_bin2d`
  Add heatmap of 2d bin counts.
- `geom_blank`
  Blank, draws nothing.
- `geom_boxplot`
  Box and whiskers plot.
- `geom_contour`
  Display contours of a 3d surface in 2d.
- `geom_crossbar`
  Hollow bar with middle indicated by horizontal line.
- `geom_density`
  Display a smooth density estimate.
- `geom_density2d`
  Contours from a 2d density estima
- `geom_dotplot`
  Dot plot

Dependencies

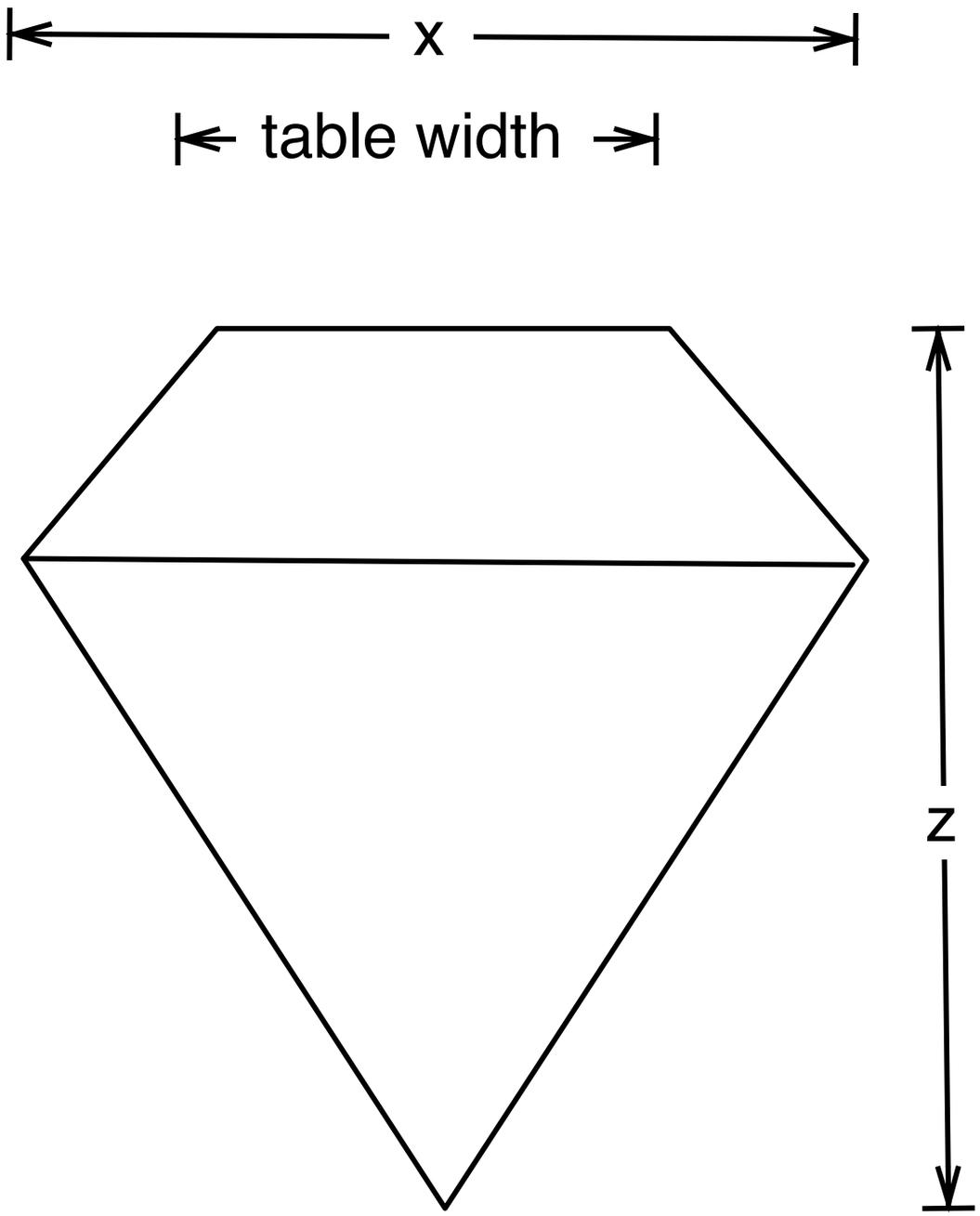
- **Depends**: stats, methods
- **Imports**: plyr, digest, grid, gtable, reshape2, scales, memoise, proto, MASS
- **Suggests**: quantreg, Hmisc, mapproj, maps, hexbin, maptools, multcomp, nlme, testthat
- **Extends**: sp

http://docs.ggplot2.org/current/

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

COLOR GRADING SCALE

D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Colorless			Near Colorless				Faint Yellow			Very Light Yellow					Light Yellow							

http://www.mlcld.com/colour.aspx

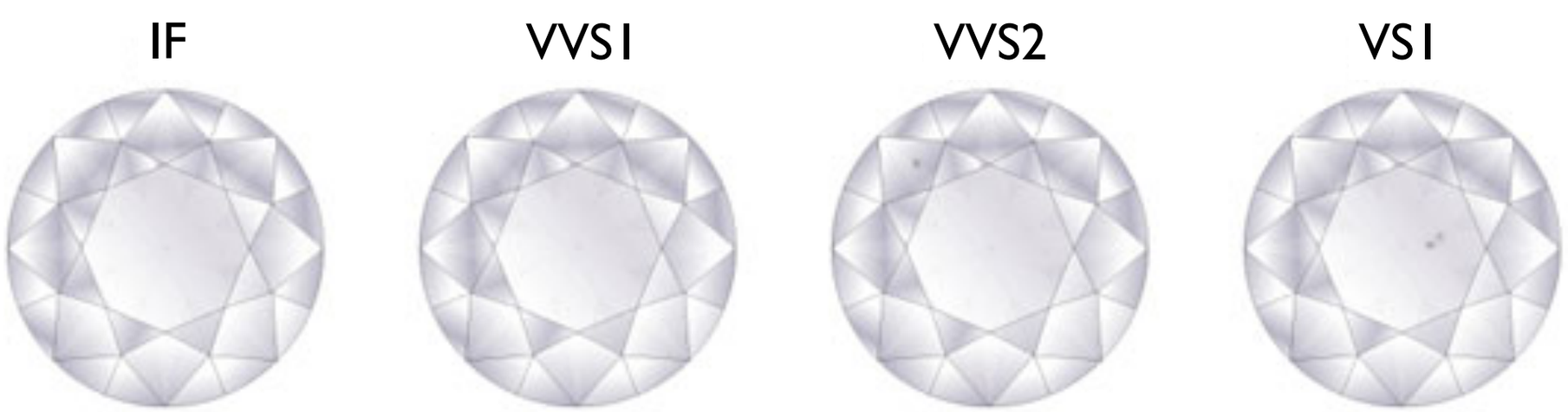


Illustration of inclusions as seen under X10 magnification

http://www.thediamondsexperts.com/diamonds-guide

Bar charts

Your turn

What types of plots do the following lines of code return?

```
qplot(x, z, data = diamonds)

qplot(x, data  = diamonds)

qplot(cut, data = diamonds)
```

Default geoms for qplot

Two variables → scatterplot (point)

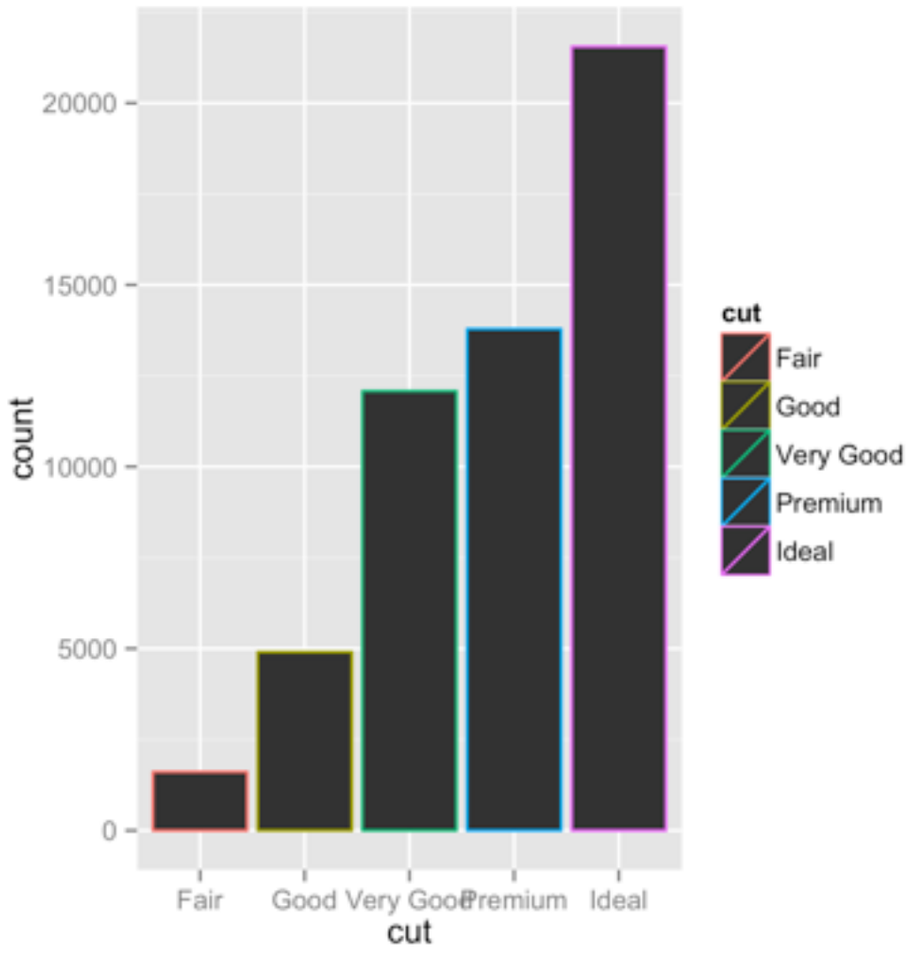
One continuous variable → histogram

One categorical variable → bar chart

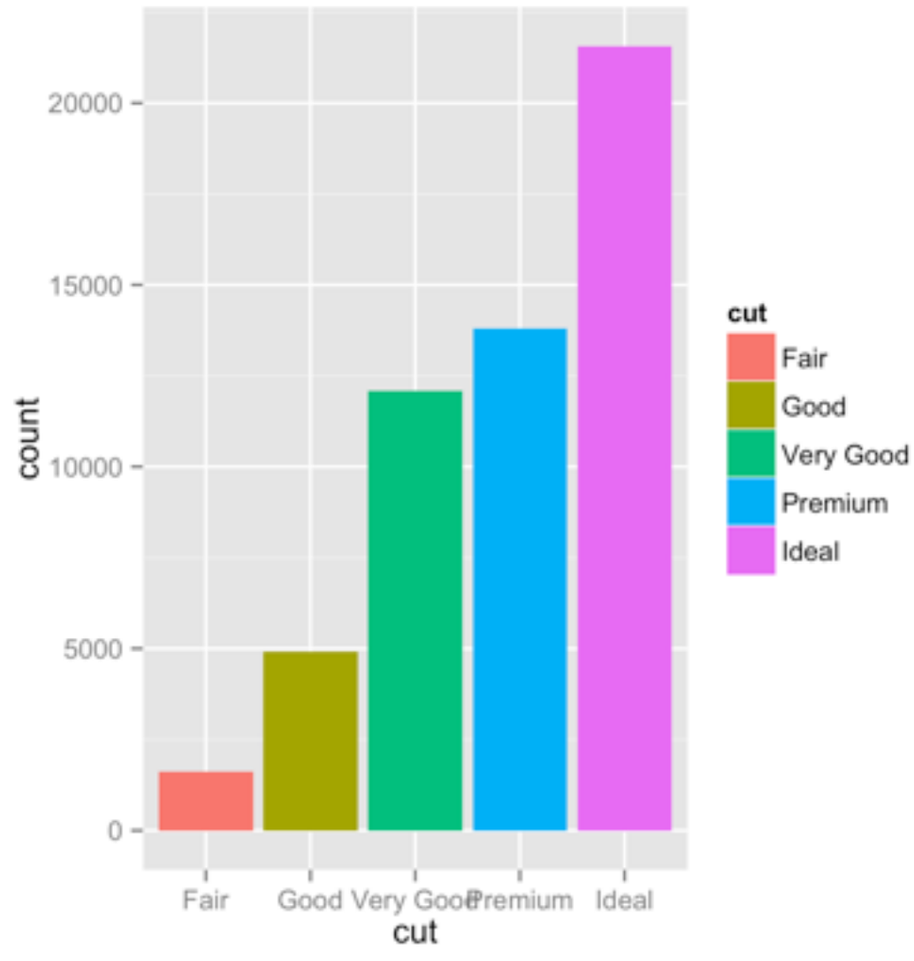
Fill

geoms that span an area have both a color aesthetic and a fill aesthetic.

```
qplot(cut, data = diamonds, geom = "bar", * = cut)
```



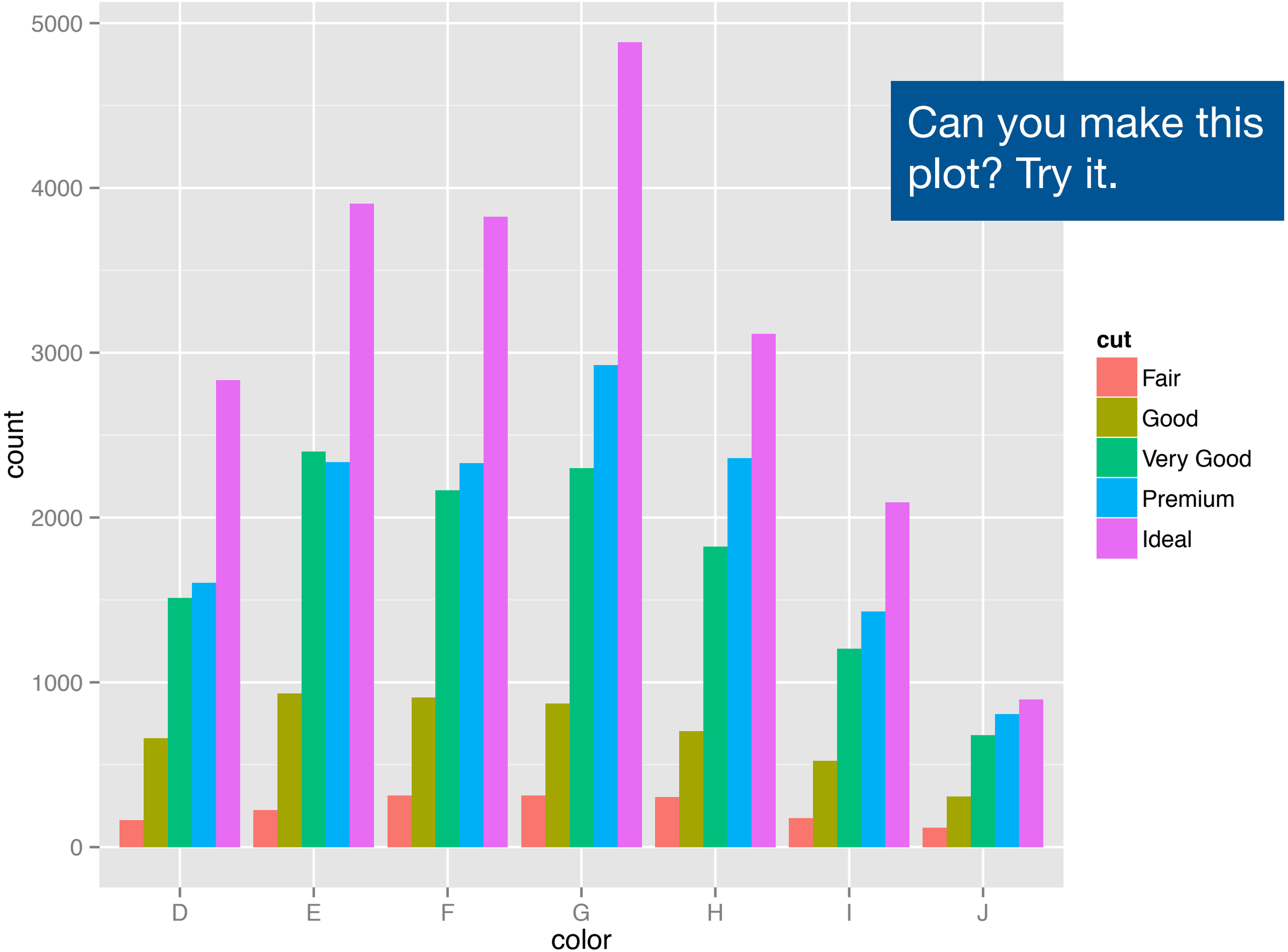
* = color



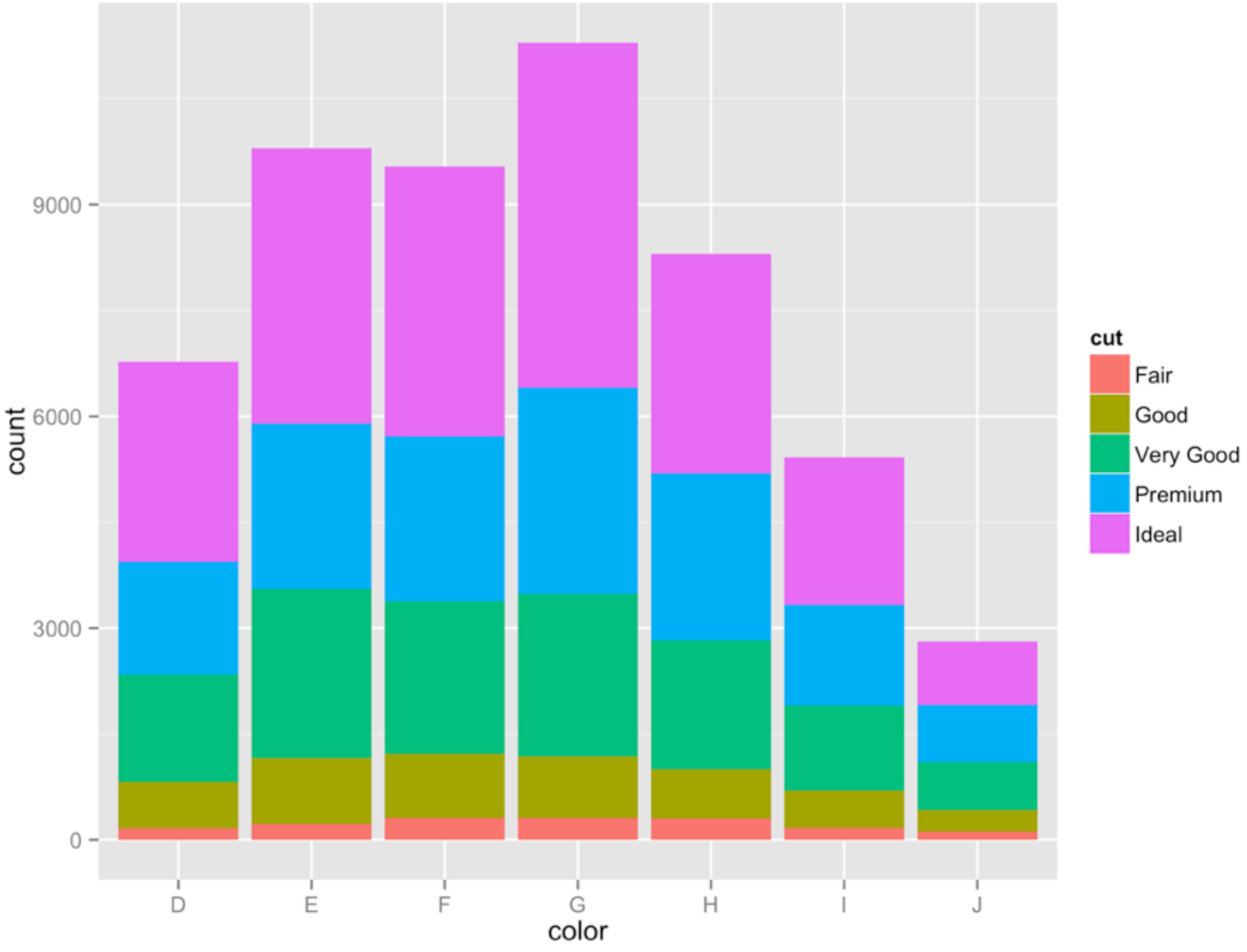
* = fill

Position adjustments

Can you make this plot? Try it.




```
qplot(color, data = diamonds, geom = "bar", fill = cut)
```




```
qplot(color, data = diamonds, geom = "bar", fill = cut)
```

position adjustment

How your graph arranges geoms that overlap with each other.

```
qplot(color, data = diamonds, fill = cut, position = "stack")
```

Set the adjustment method with the position argument

Your turn

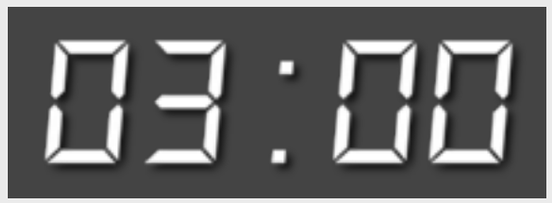
What do each of the position adjustments below do?

```
qplot(color, data = diamonds, fill = cut,
position = "stack")
```

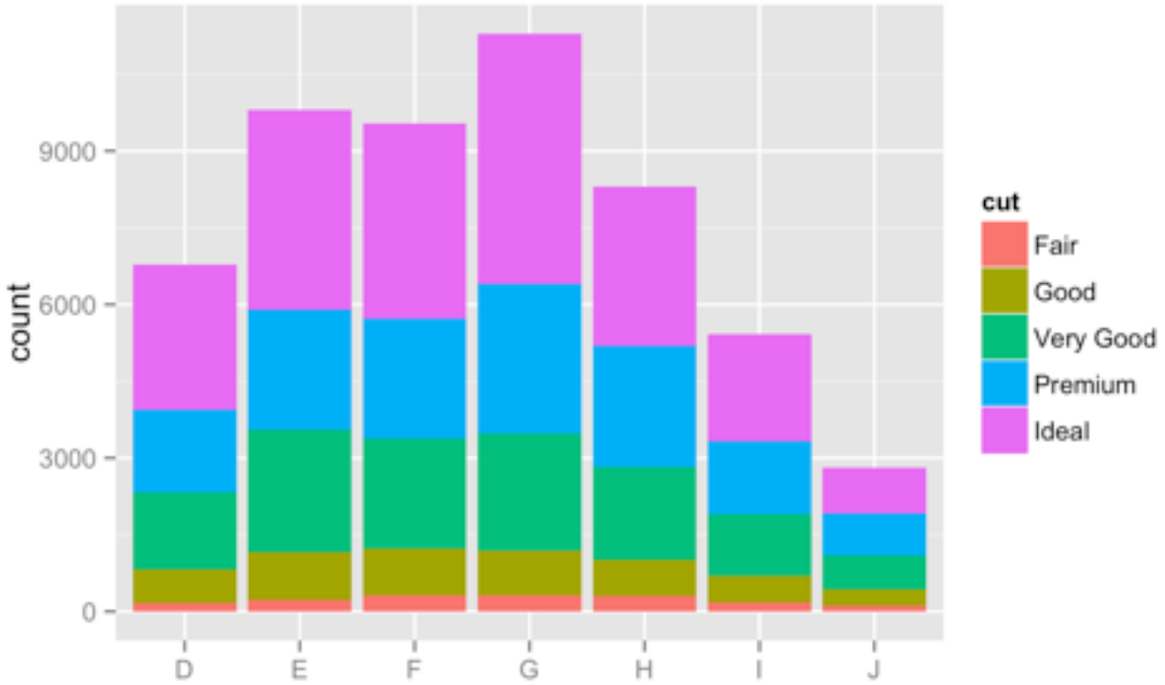
```
qplot(color, data = diamonds, fill = cut,
position = "dodge")
```

```
qplot(color, data = diamonds, fill = cut,
position = "identity")
```

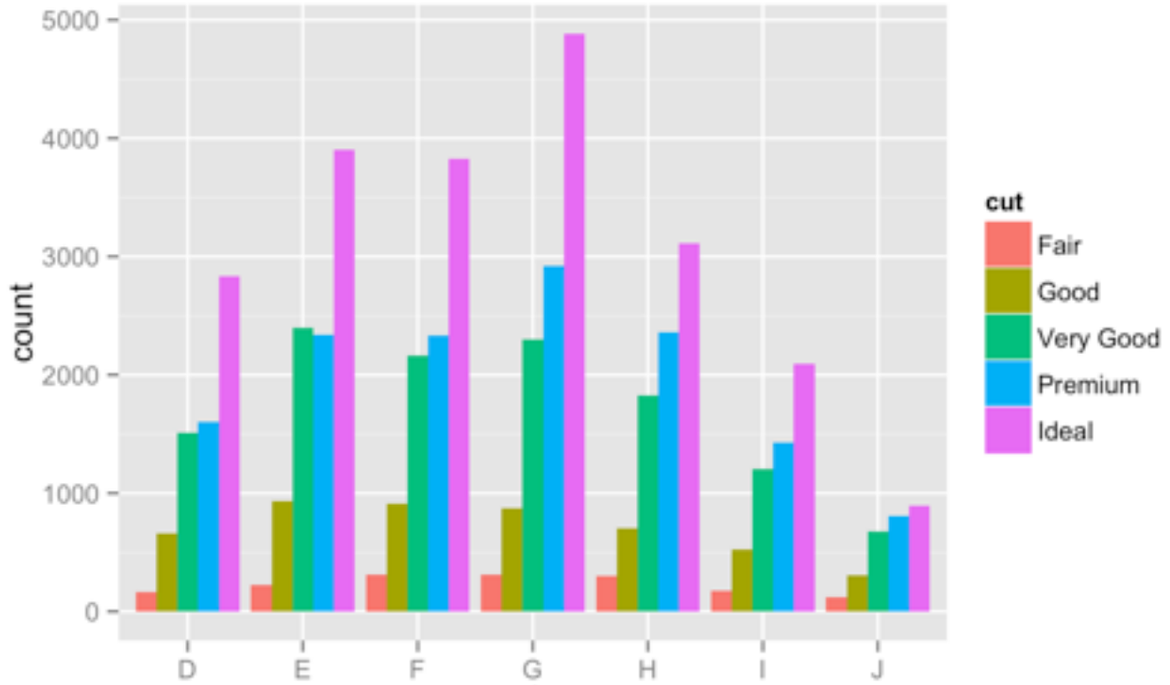
```
qplot(color, data = diamonds, fill = cut,
position = "fill")
```



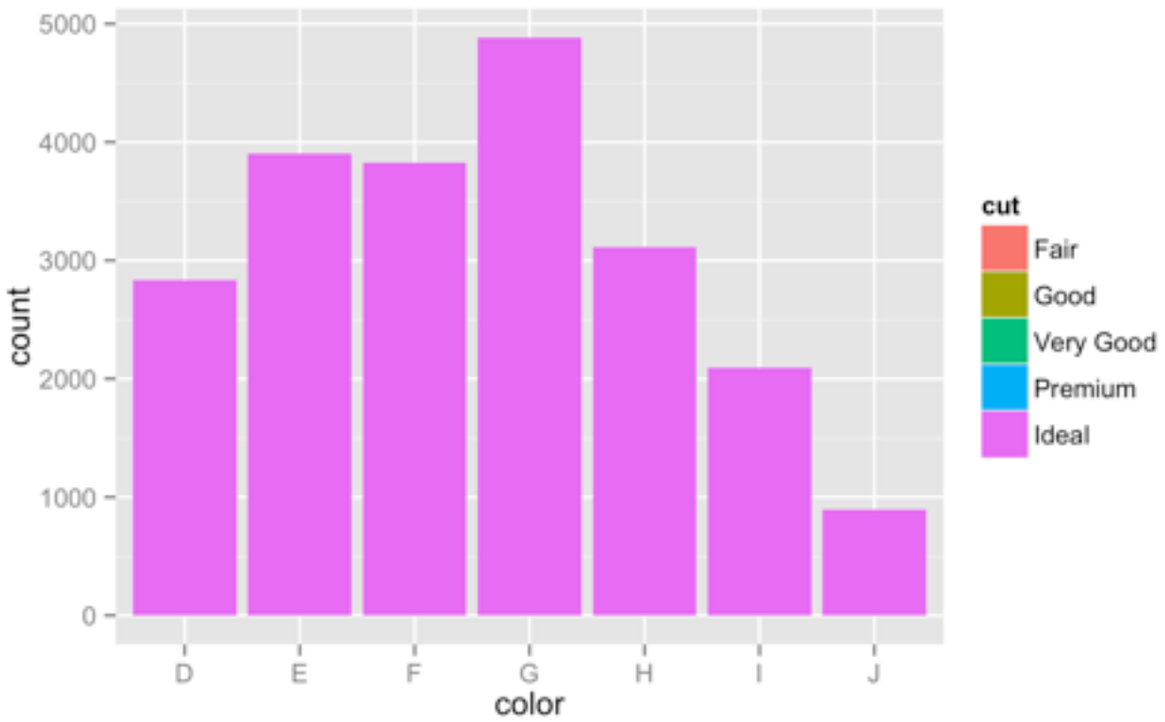
stack



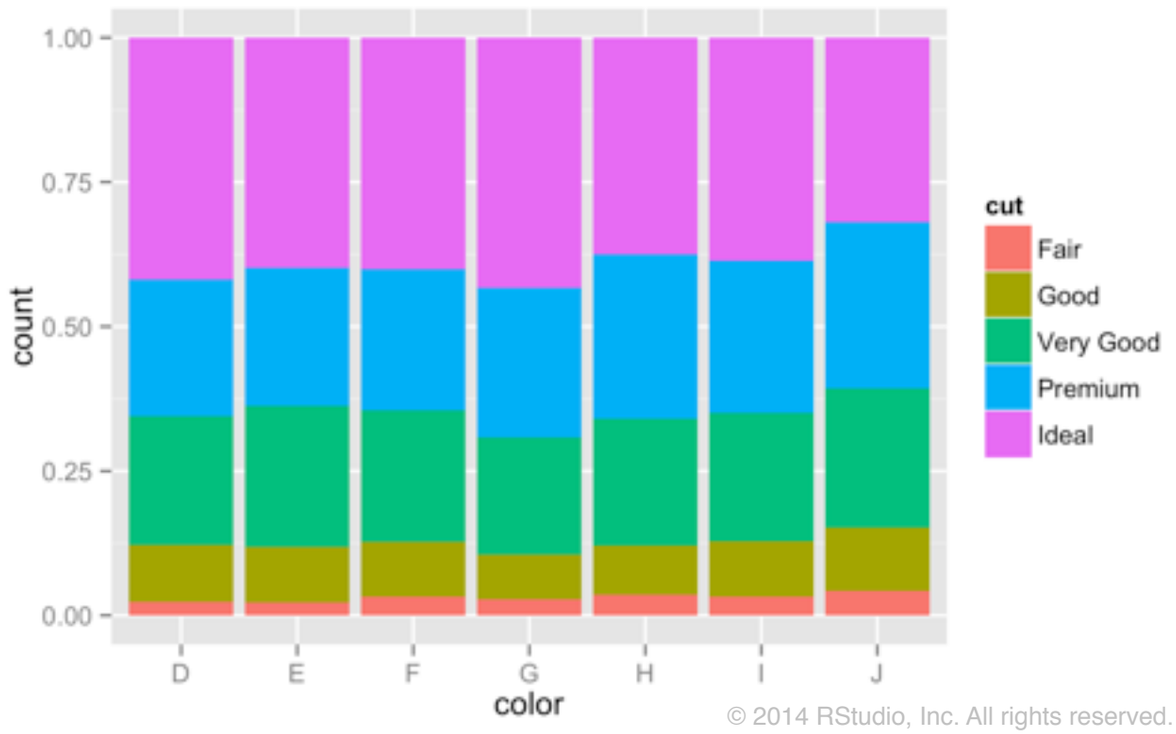
dodge



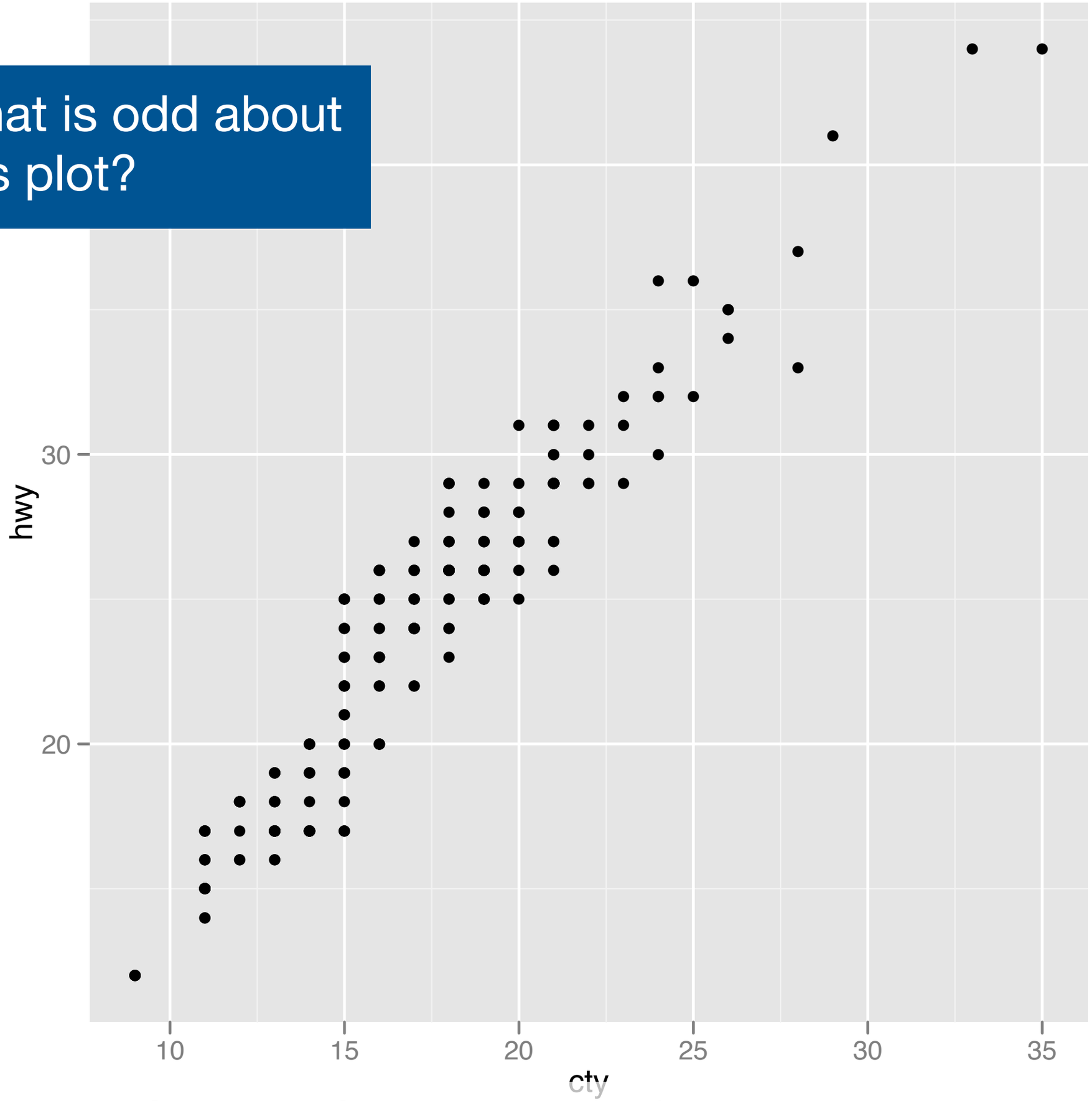
identity
(overlaps, last on top)



fill
(displays proportions)

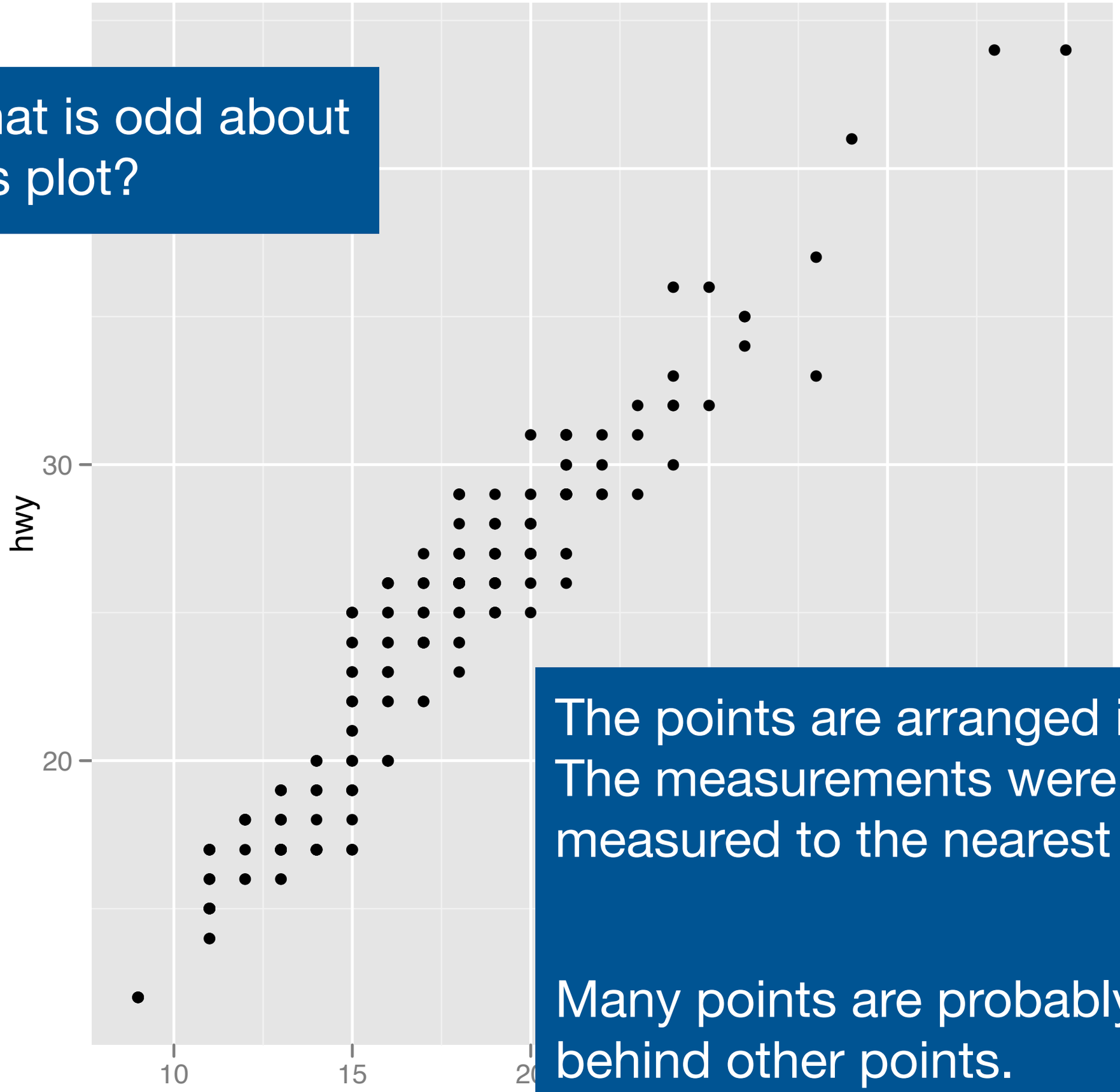


What is odd about this plot?




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

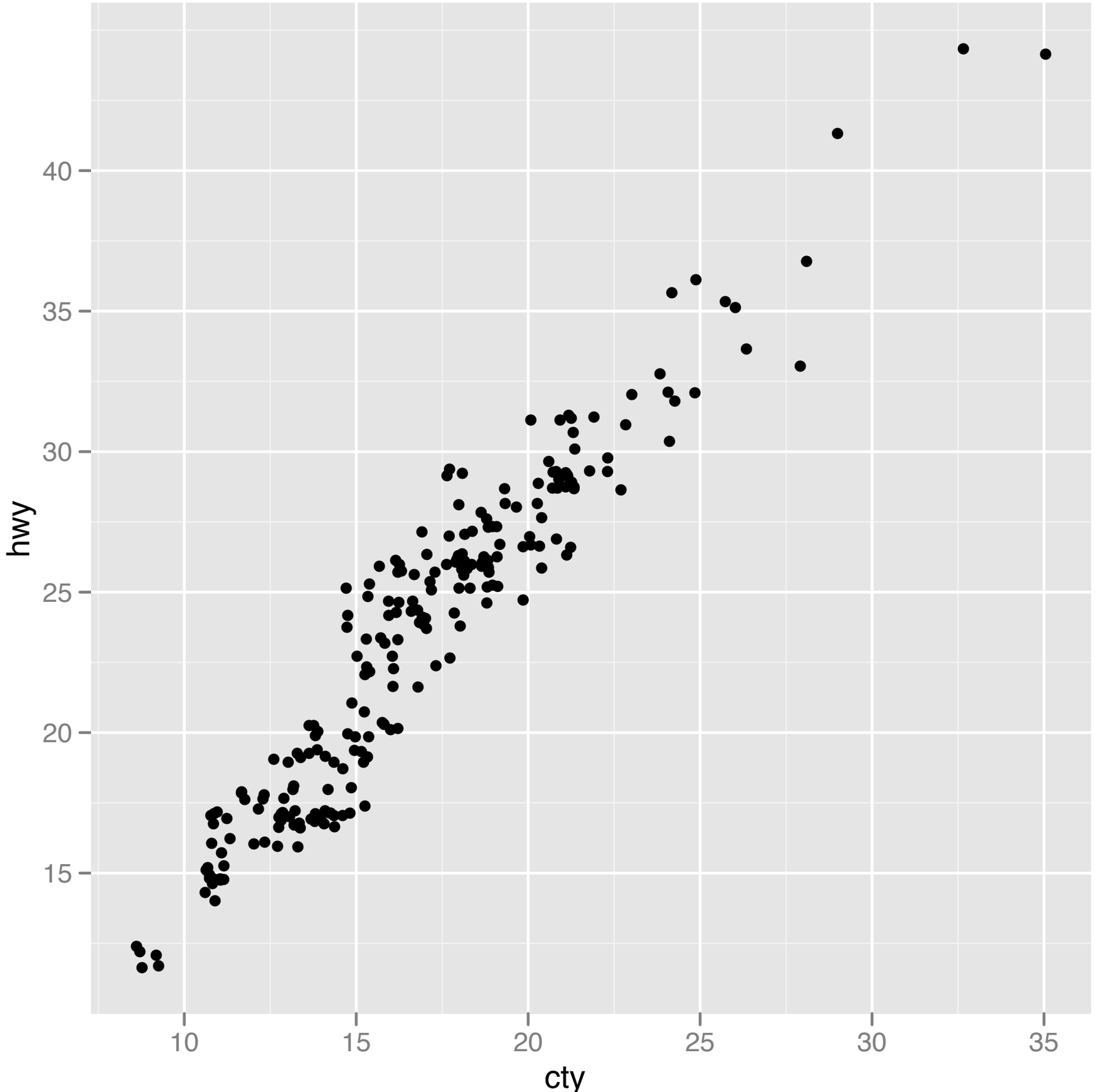
What is odd about this plot?



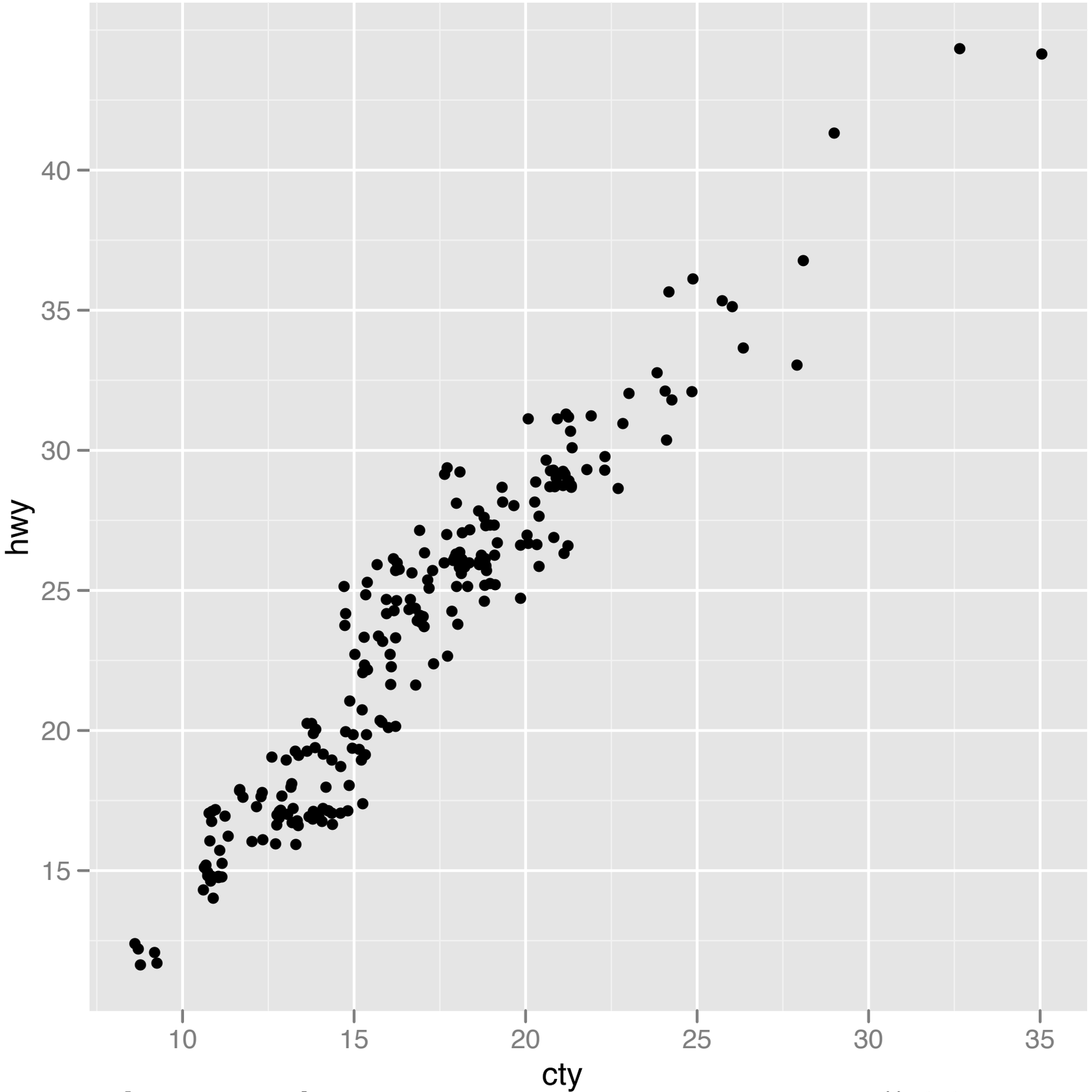
The points are arranged in a grid. The measurements were probably measured to the nearest integer.

Many points are probably hidden behind other points.

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




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




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

jittering

The jittering adjustment adds random noise to each point. As a result they are unlikely to overlap.

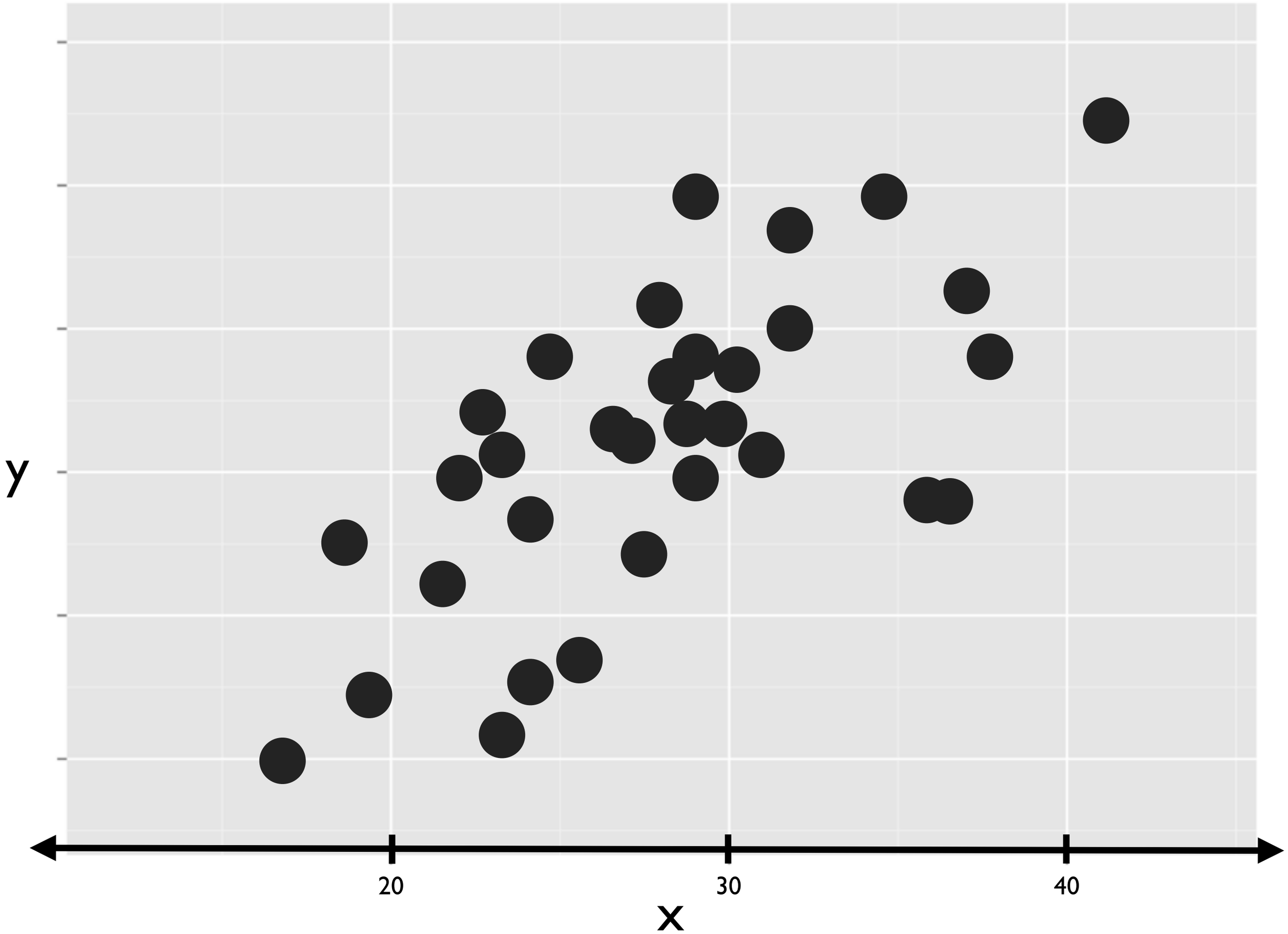
Jittering is so common that ggplot2 comes with a jitter geom. Its just a short cut for a point geom with a jitter position adjustment. e.g., these are the same

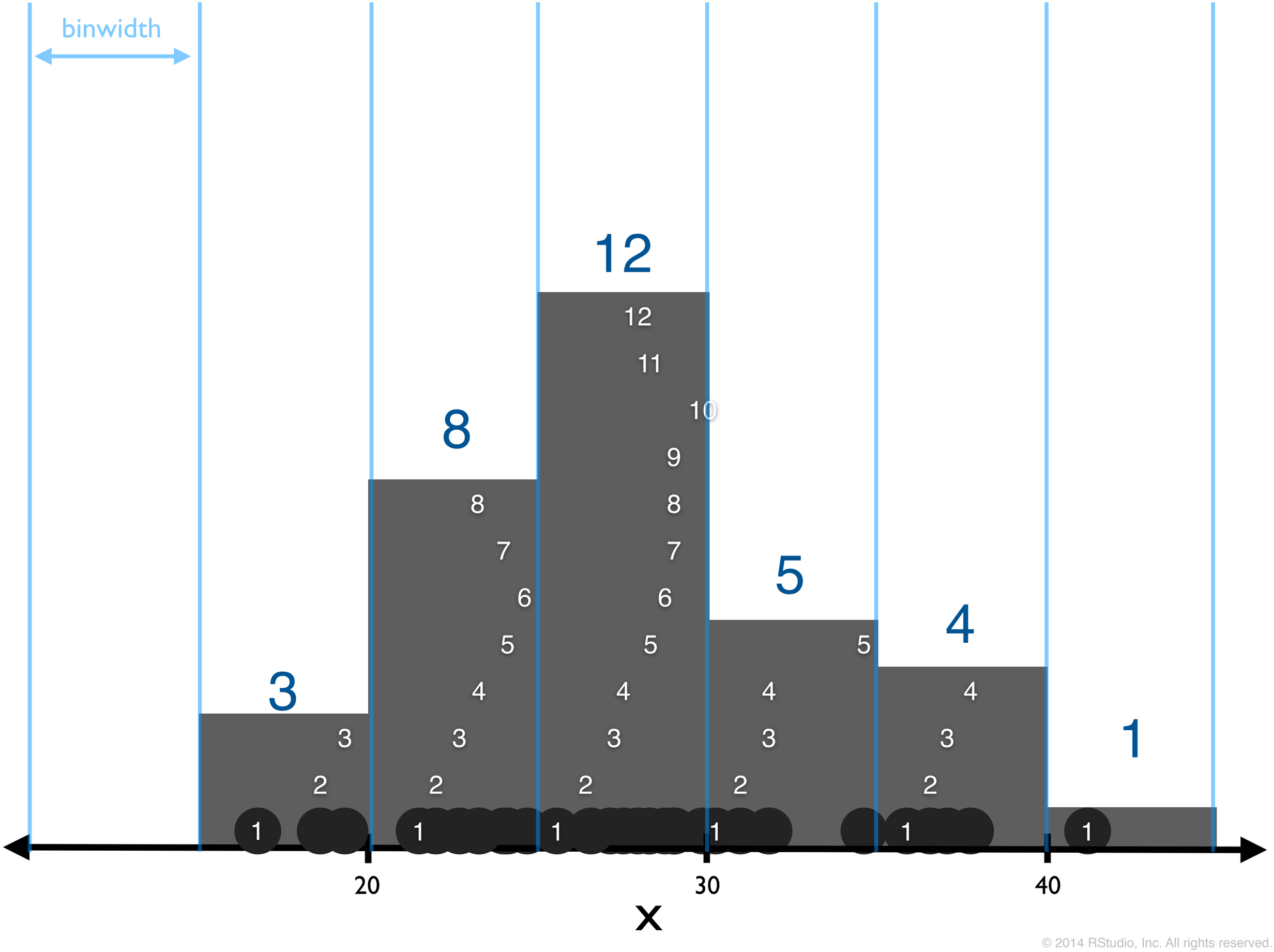
```
qplot(cty, hwy, data = mpg, geom = "point", position = "jitter")
qplot(cty, hwy, data = mpg, geom = "jitter")
```

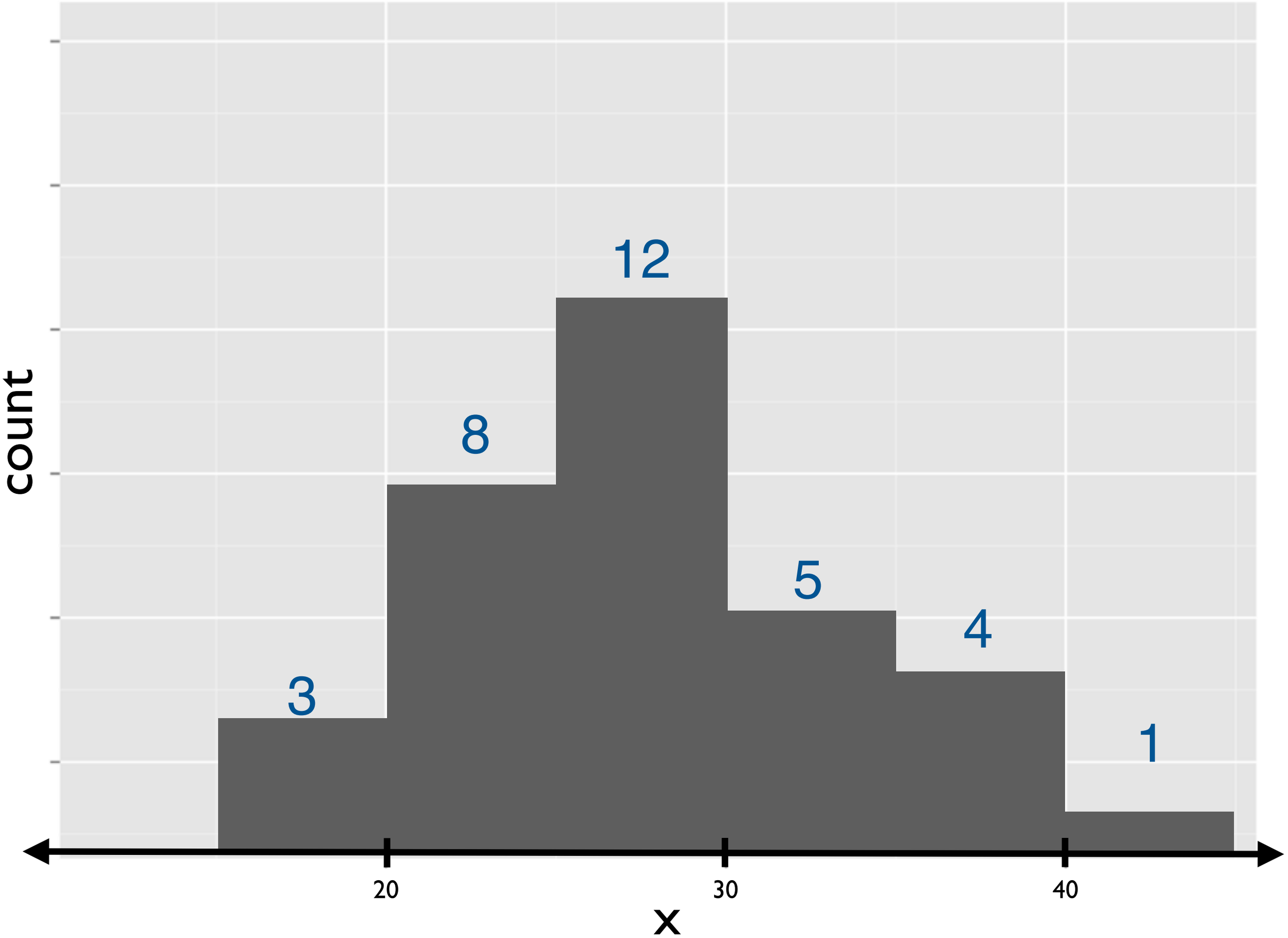
Summary

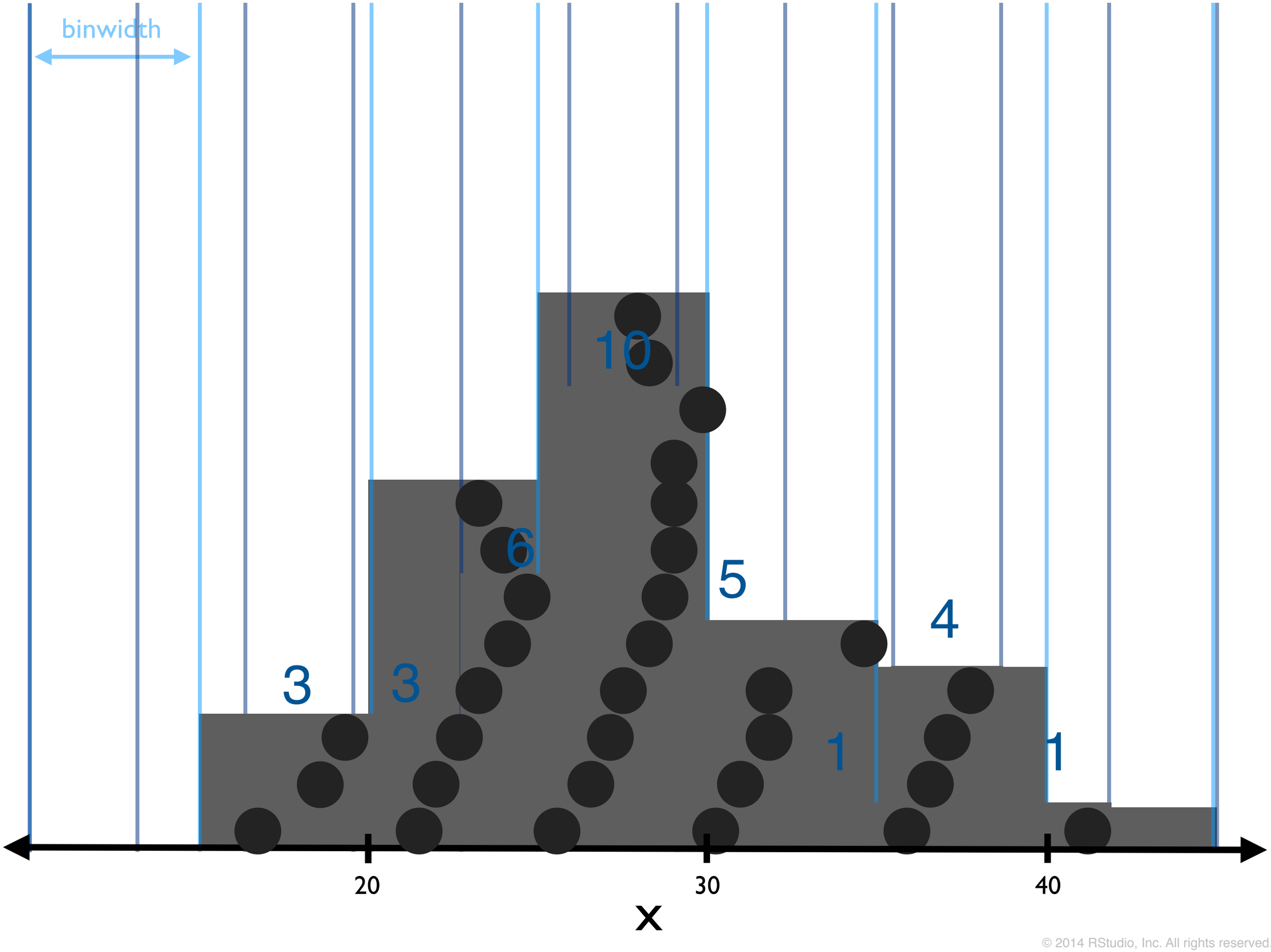
method	effect
"identity"	No adjustment. Geoms are allowed to overlap (some may be hidden behind others).
"stack"	Overlapping geoms are placed one above the other.
"dodge"	Overlapping geoms are placed beside each other.
"fill"	The available space is divided proportionately between the overlapping geoms.
"jitter"	Random noise is added to the position of each geom.

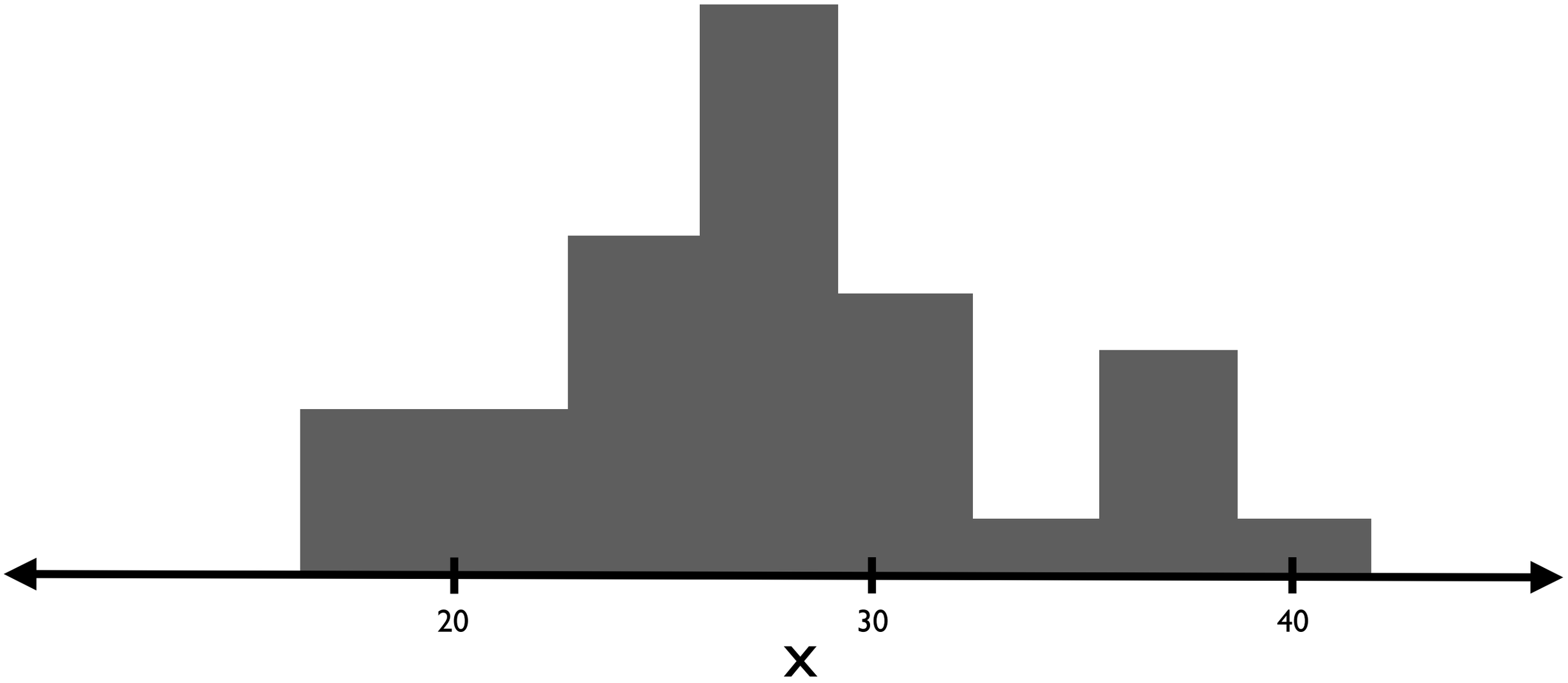
Histograms











Parameters

Similar to aesthetics.

A parameter is input that controls the appearance of the graph, *but does not map appearance to data.*

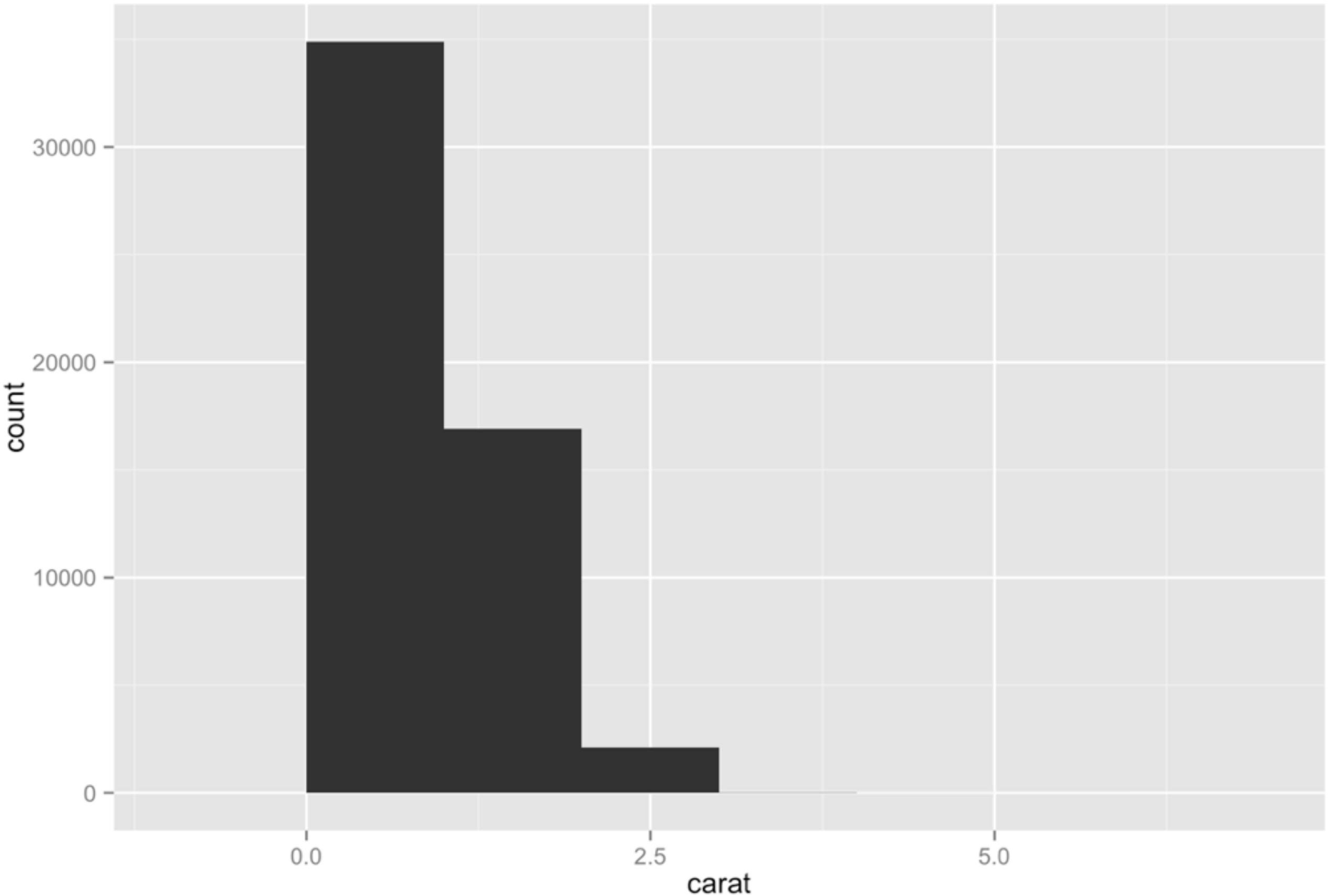
Binwidths are a parameter

Parameters

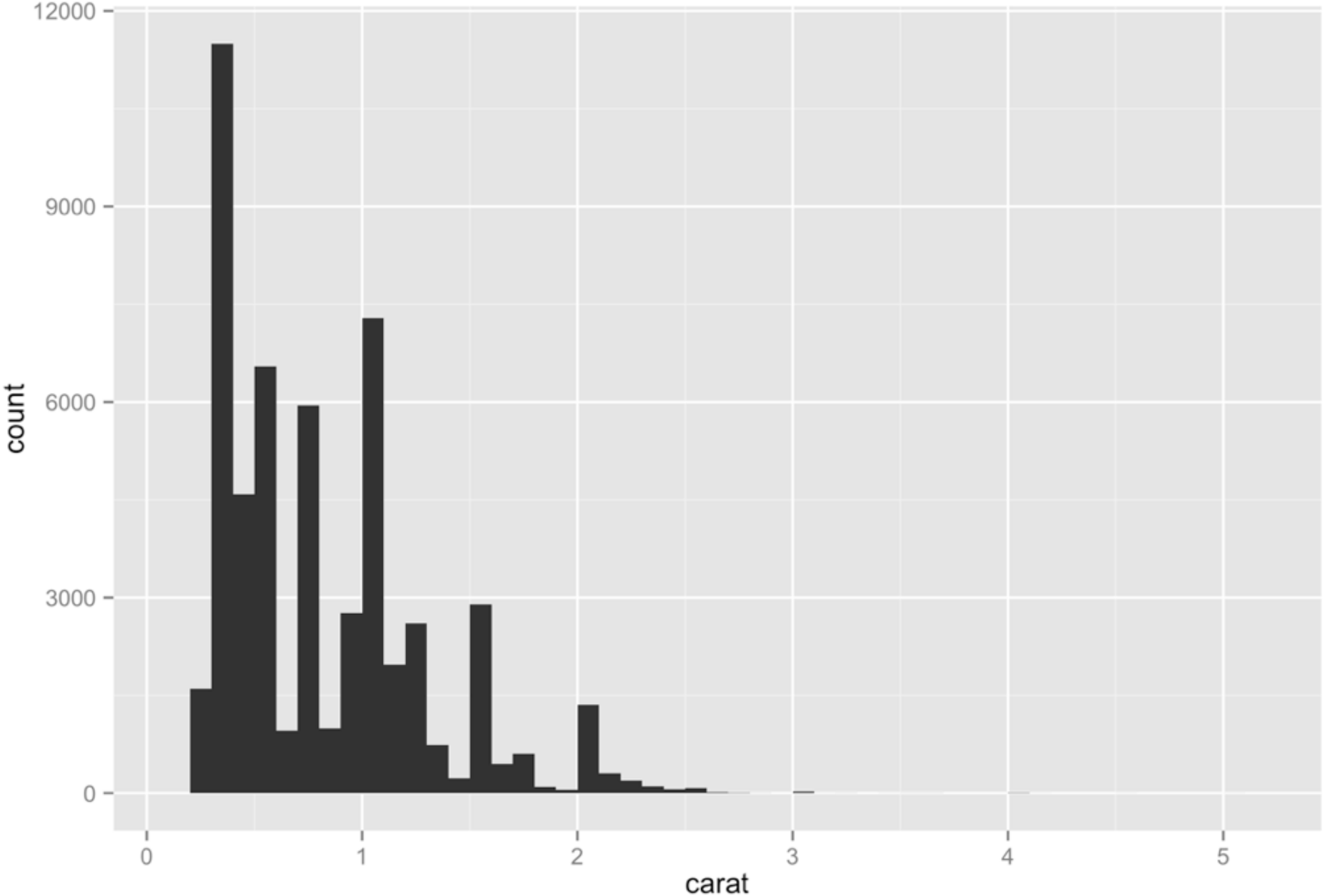
parameter name

value

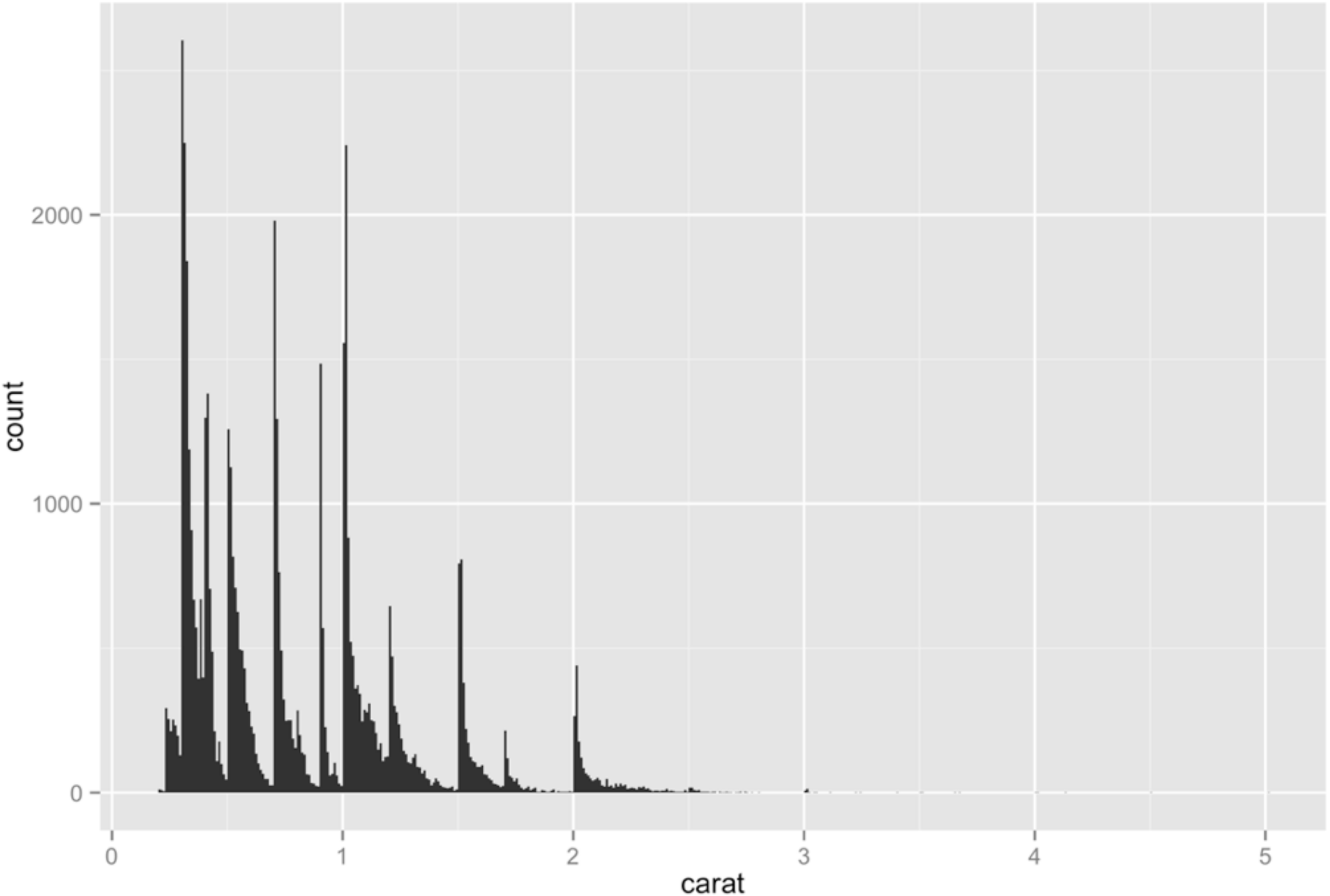
```
qplot(displ, data = mpg, binwidth = 1)
```



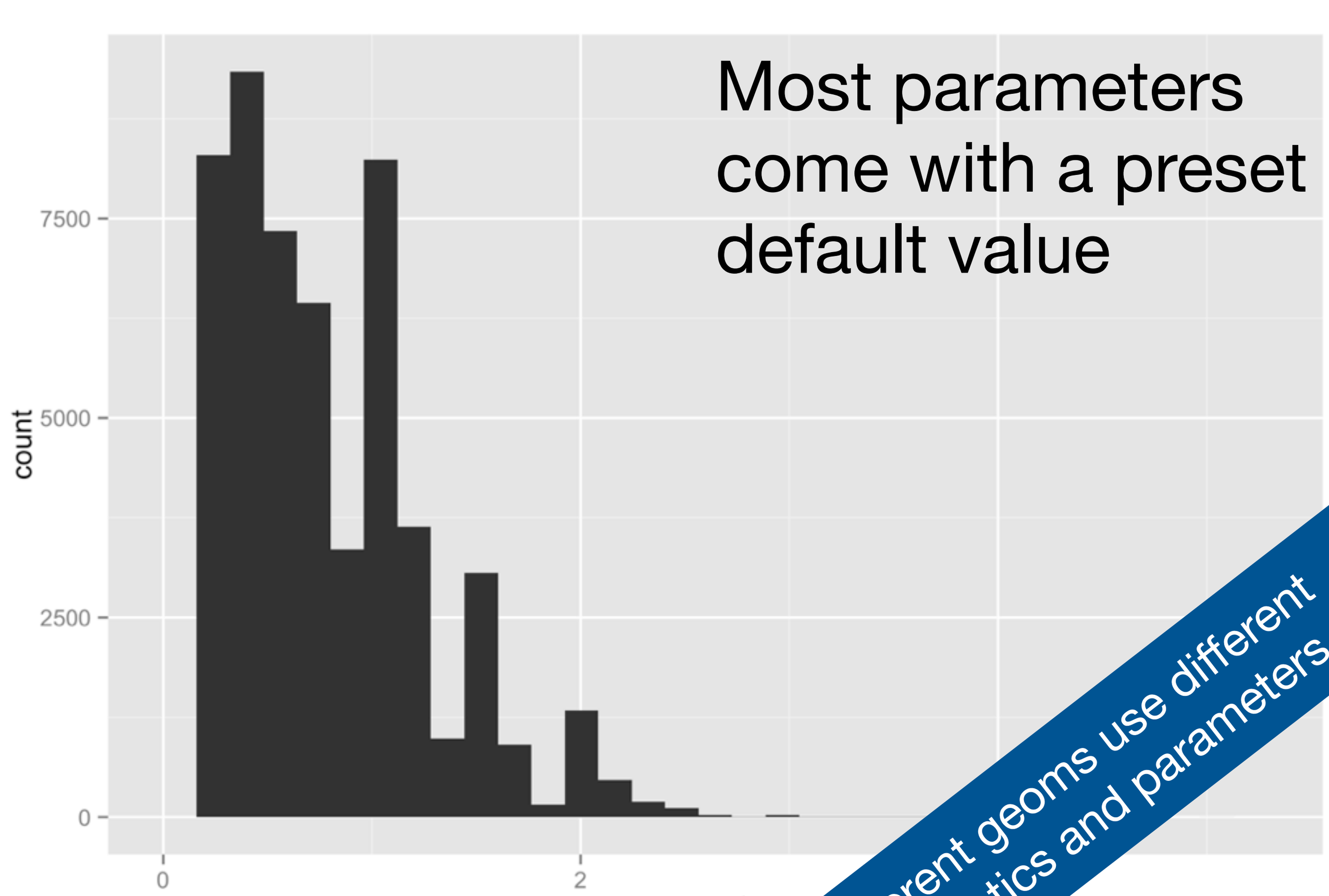

```
qplot(carat, data = diamonds, binwidth = 1)
```




```
qplot(carat, data = diamonds, binwidth = 0.1)
```




```
qplot(carat, data = diamonds, binwidth = 0.01)
```



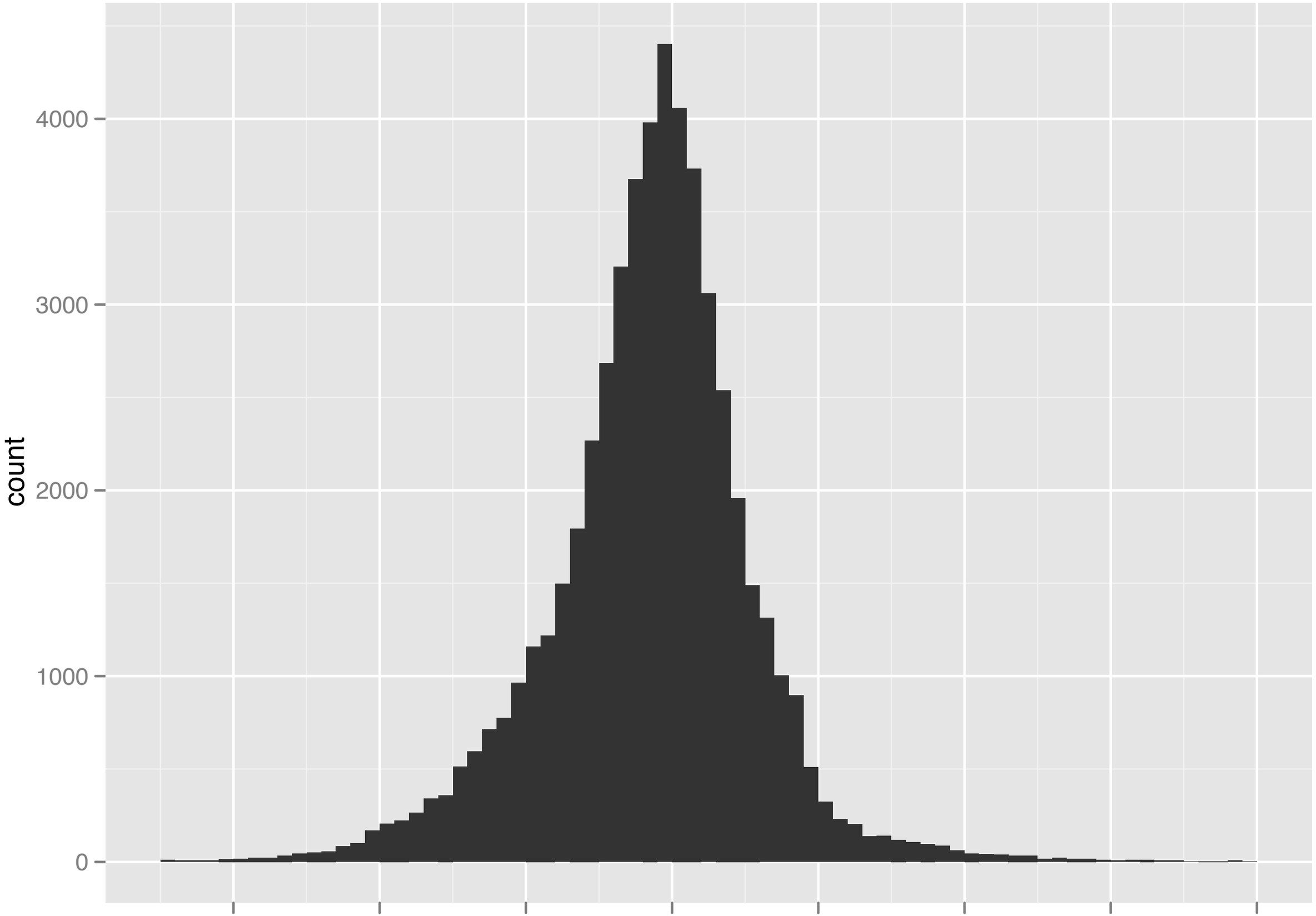
Most parameters come with a preset default value

```
qplot(carat, data = diamonds)
stat_bin: binwidth defaulted to
```

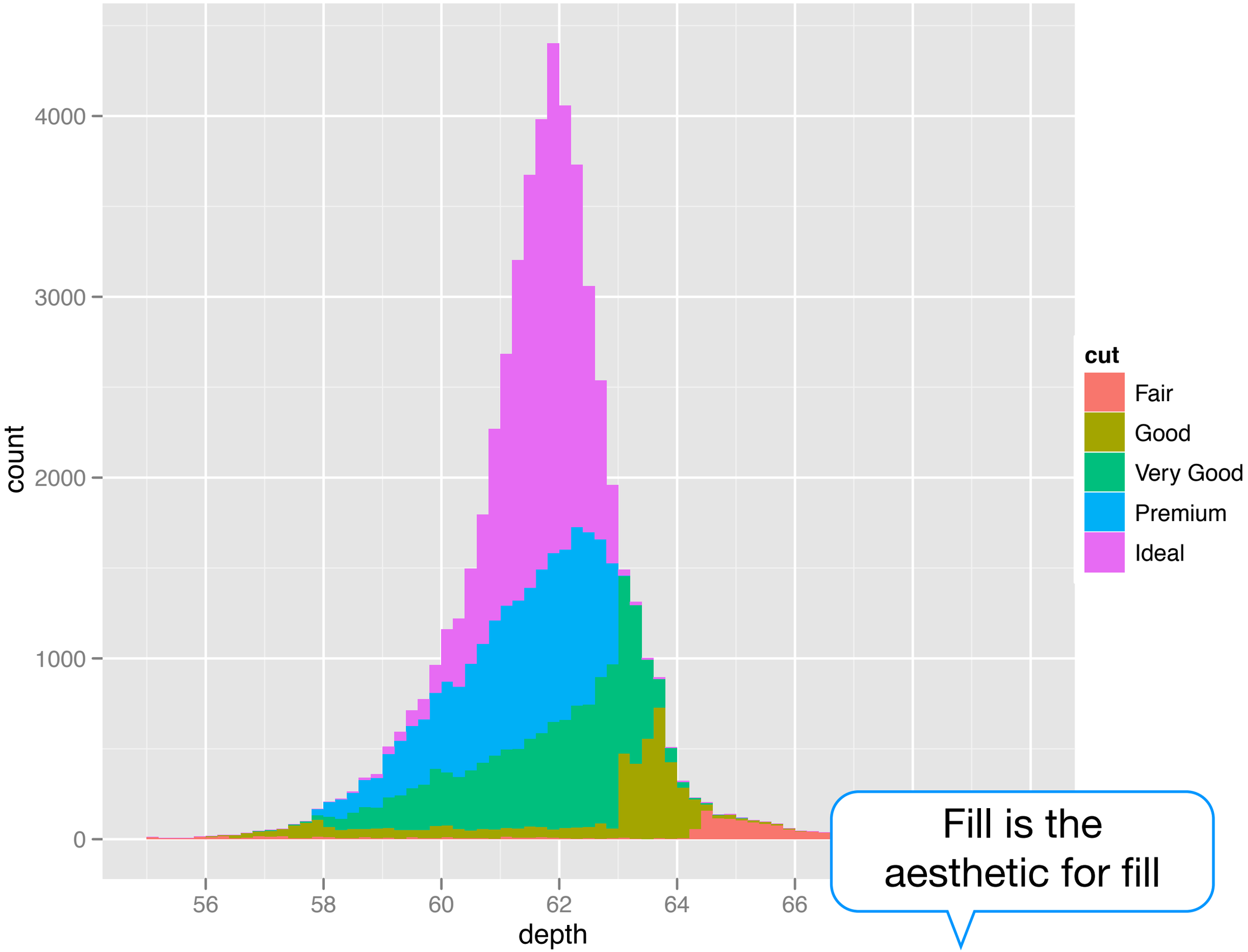
Different geoms use different aesthetics and parameters

Additional variables

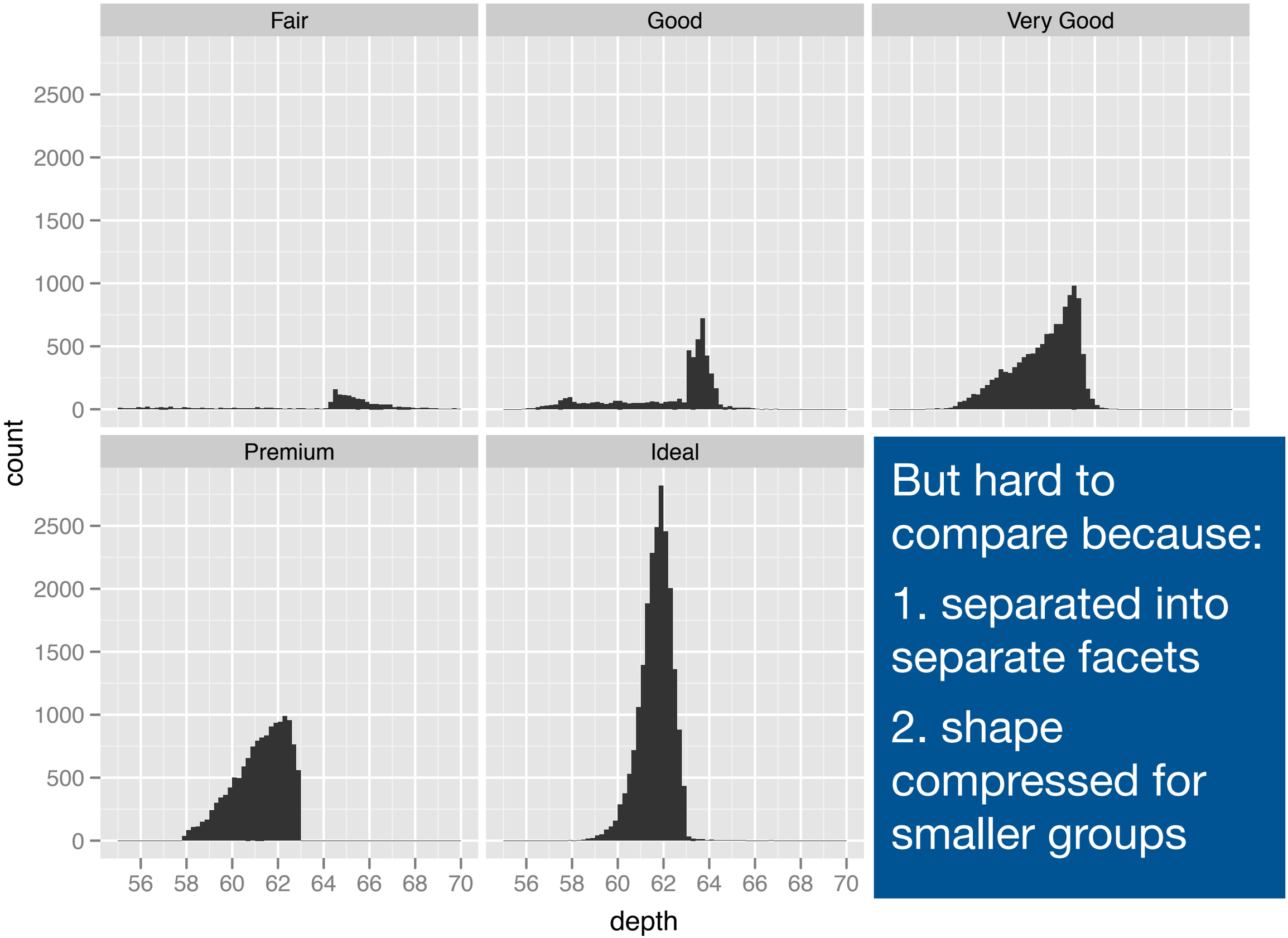
Often switching geoms is more effective than adding aesthetics or faceting to a histogram




```
zoom <- coord_cartesian(xlim = c(55, 70))
qplot(depth, data = diamonds, binwidth = 0.2) + zoom
```



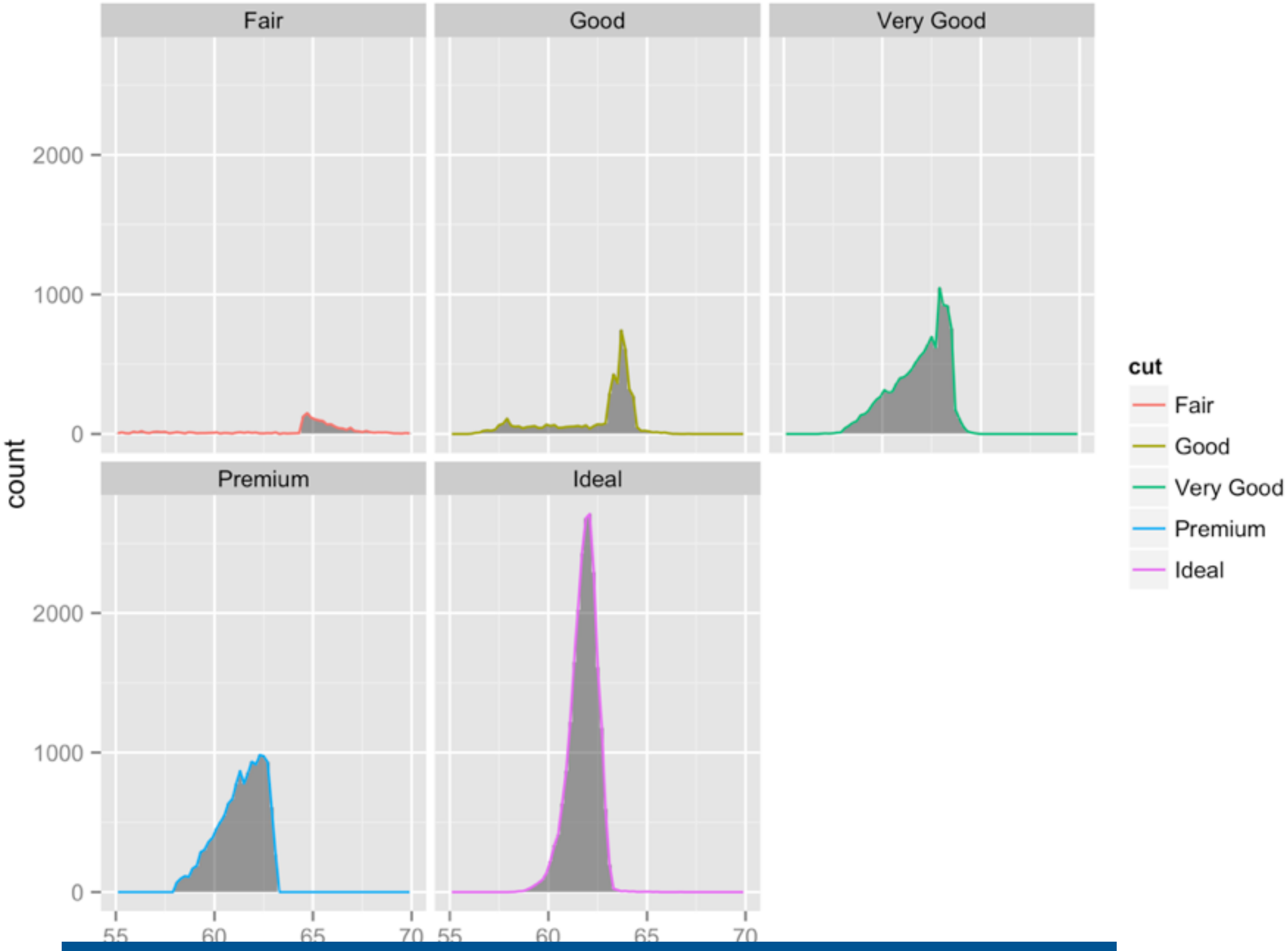

```
qplot(depth, data = diamonds, binwidth = 0.2, fill = cut) + zoom
```



But hard to compare because:

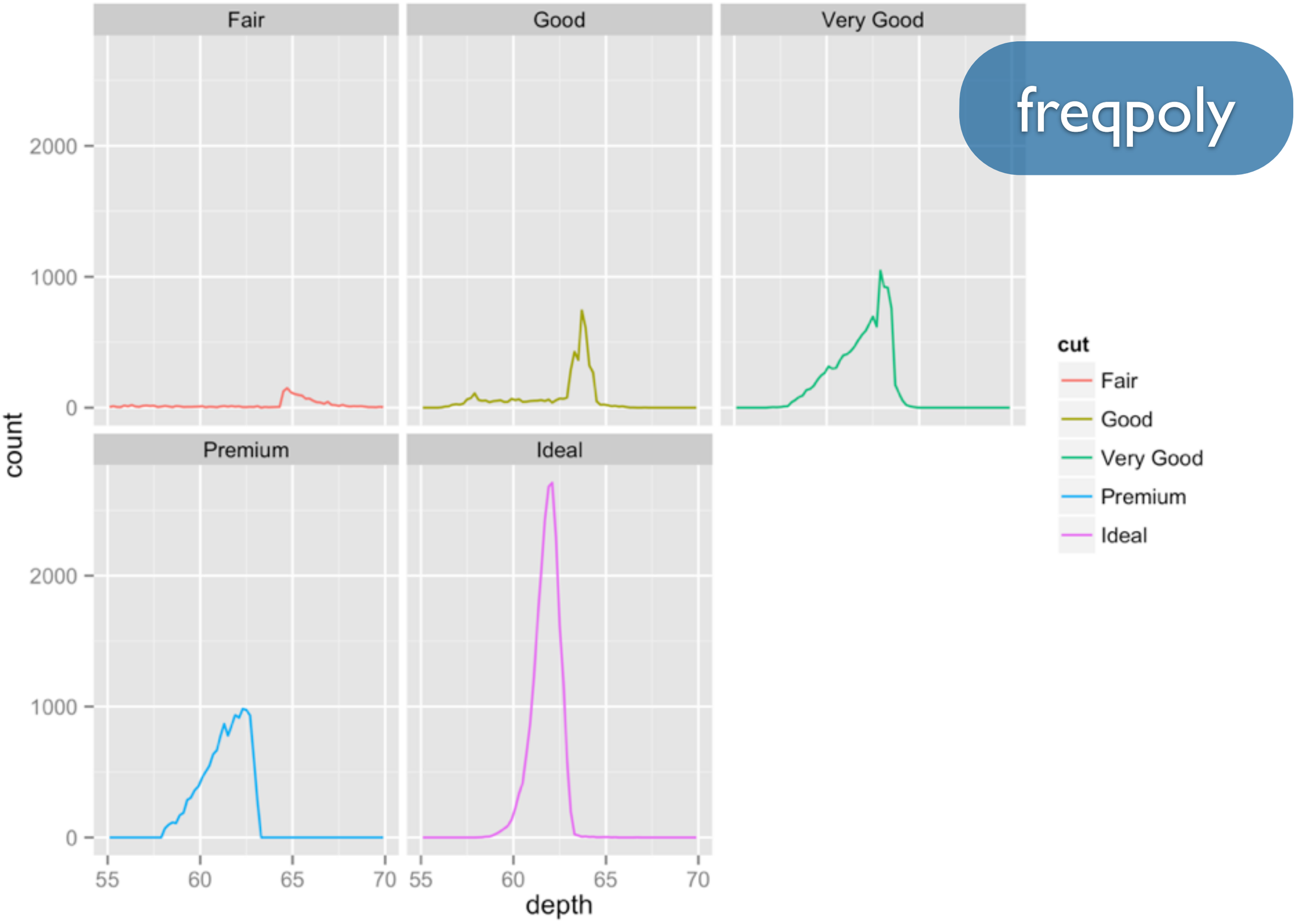
1. separated into separate facets
2. shape compressed for smaller groups

```
qplot(depth, data = diamonds, binwidth = 0.2) +
  zoom + facet_wrap(~ cut)
```

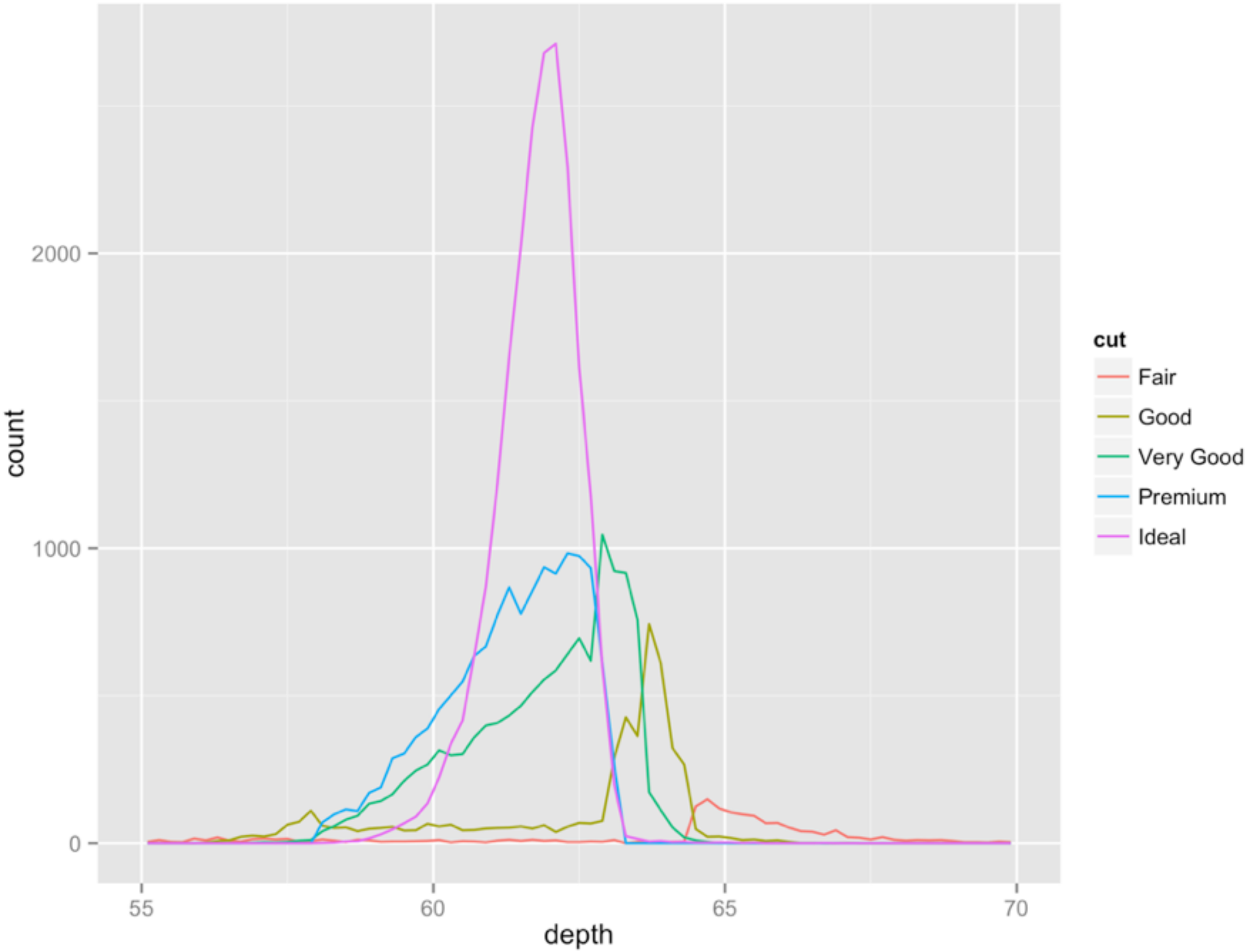


What if we just drew a line along the tops of the histograms, and threw away the bars?

freqpoly

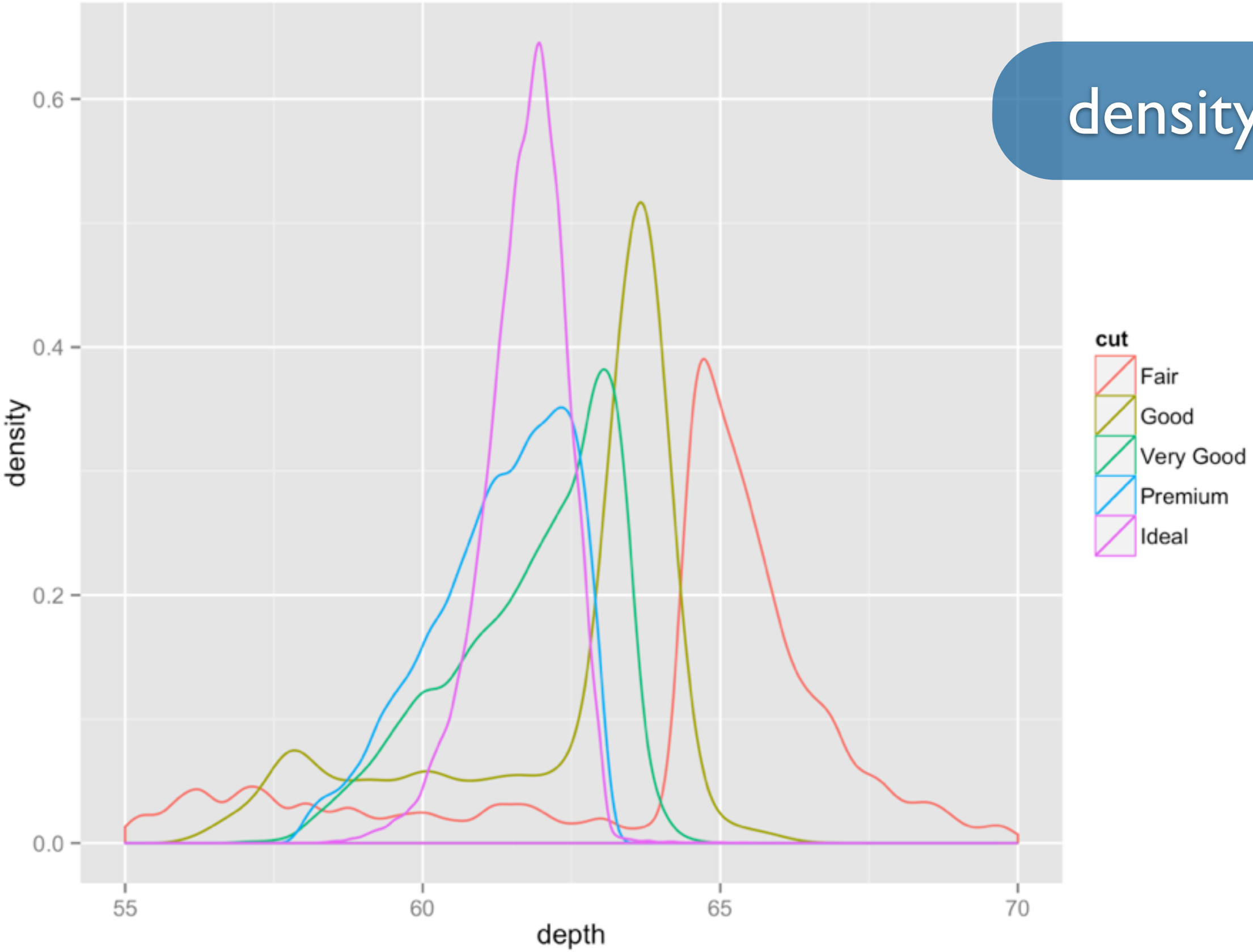



```
qplot(depth, data = diamonds, geom = "freqpoly", color = cut,
  binwidth = 0.2) + zoom + facet_wrap(~ cut)
```




```
qplot(depth, data = diamonds, geom = "freqpoly",
  color = cut, binwidth = 0.2) + zoom
```

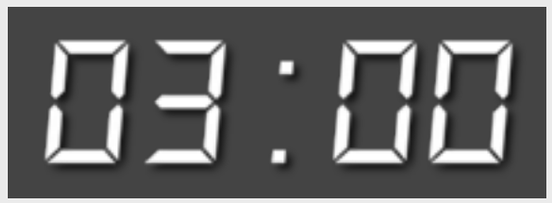
density

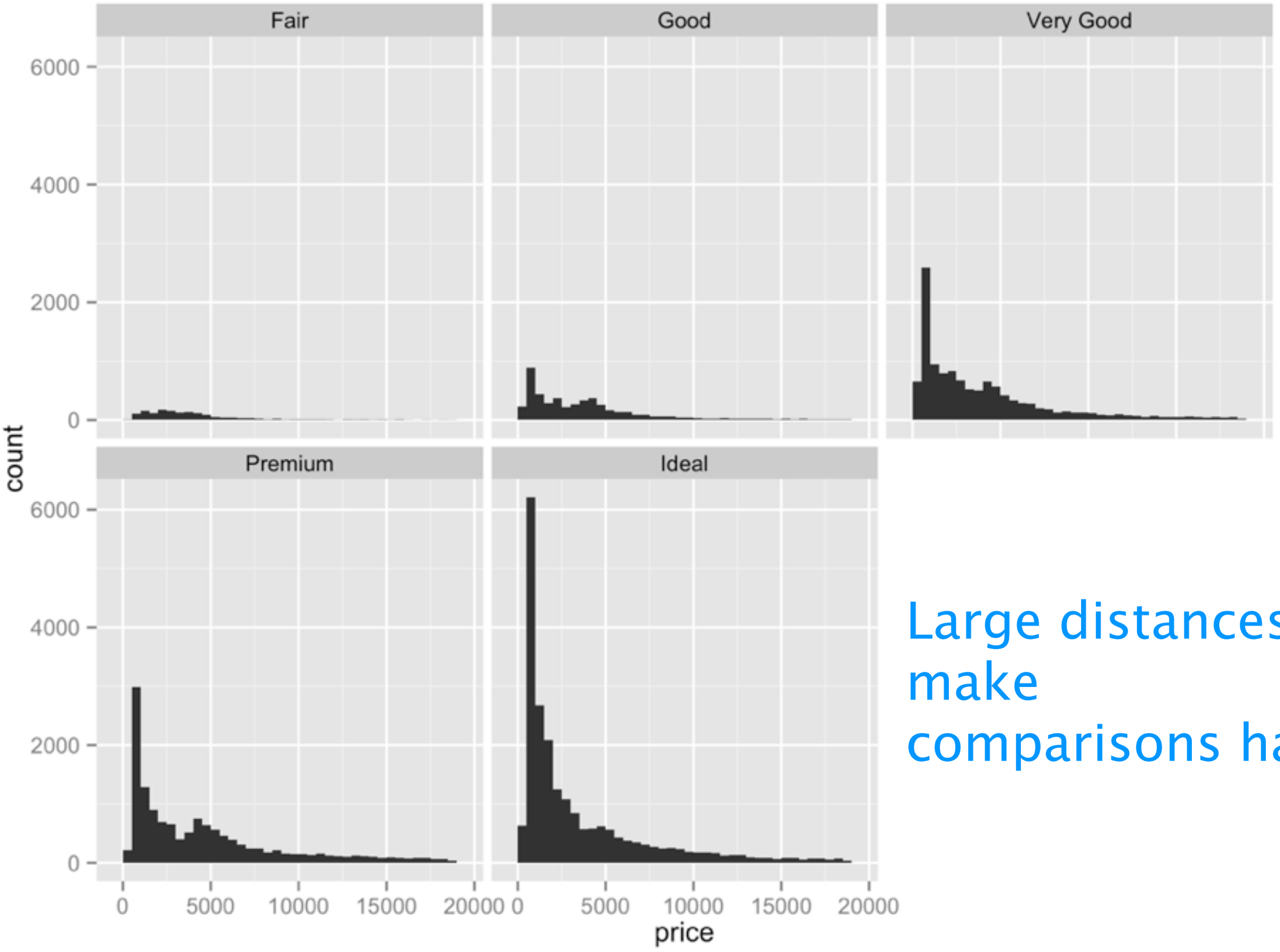



```
qplot(depth, data = diamonds, geom = "density",
  color = cut) + zoom
```

Your turn

Compare the distribution of price for the different cuts. Does anything seem unusual?

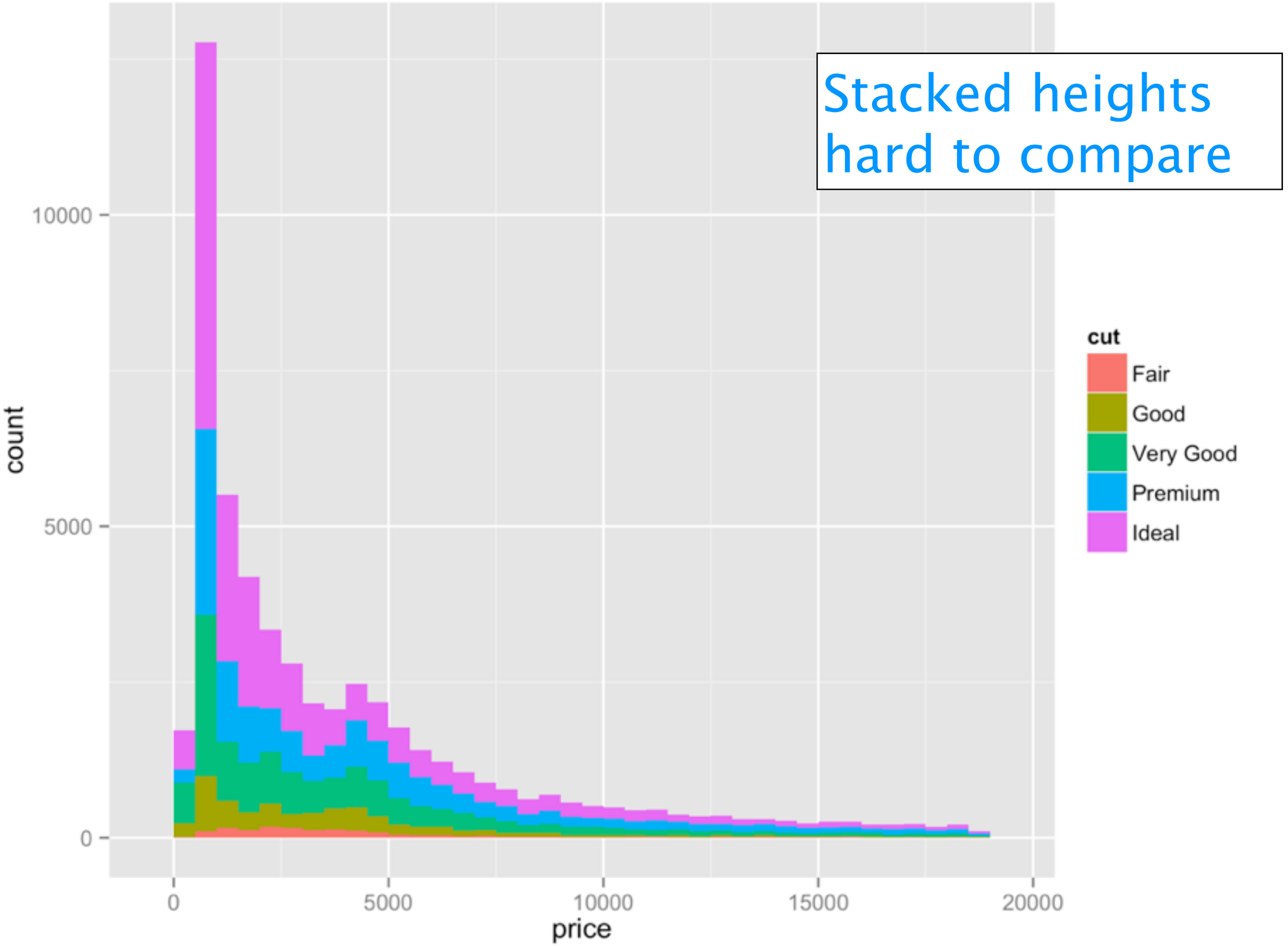




Large distances make comparisons hard

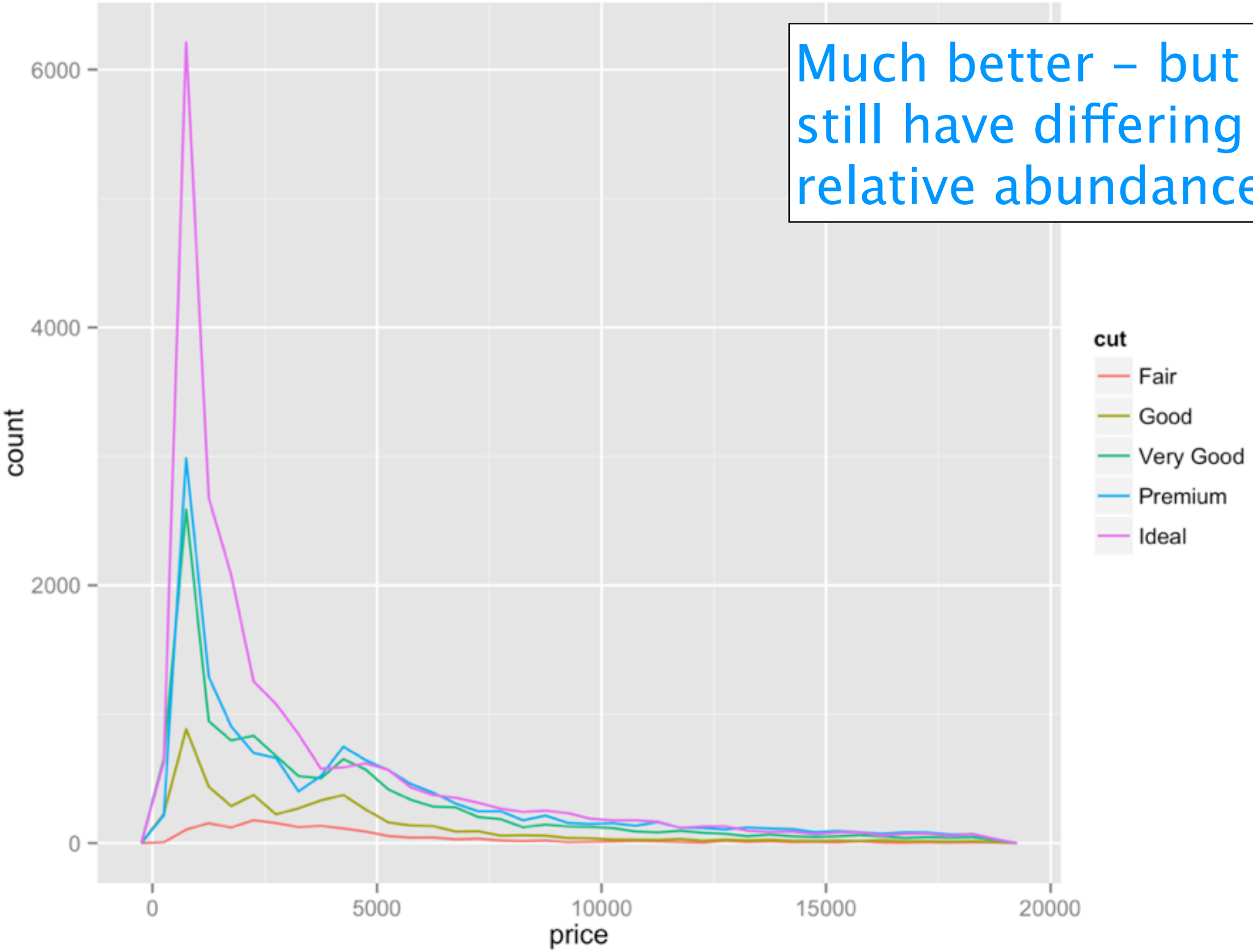
```
qplot(price, data = diamonds, binwidth = 500) +
  facet_wrap(~ cut)
```

Stacked heights hard to compare

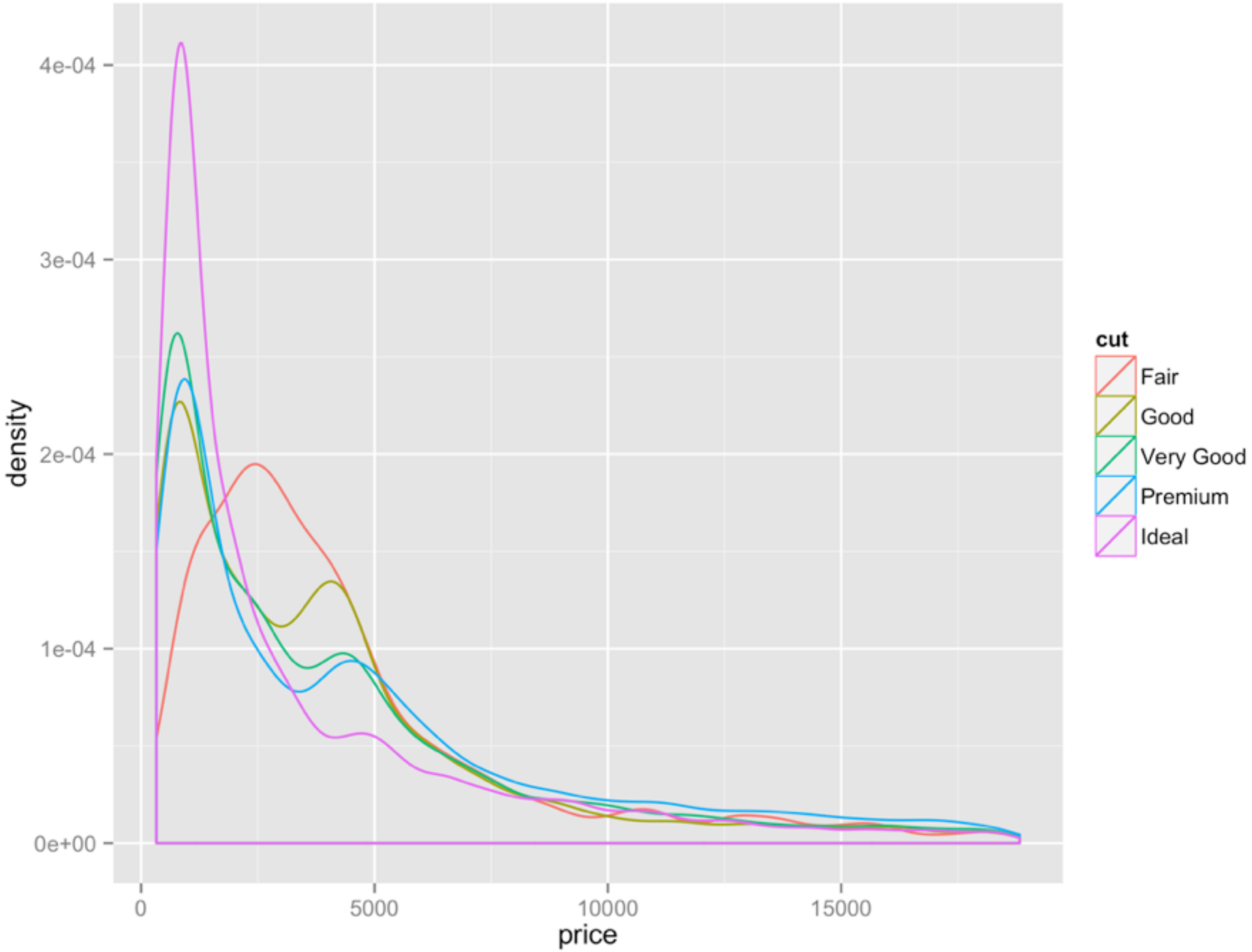



```
qplot(price, data = diamonds, binwidth = 500,
  fill = cut)
```

Much better – but still have differing relative abundance




```
qplot(price, data = diamonds, binwidth = 500,
  geom = "freqpoly", color = cut)
```

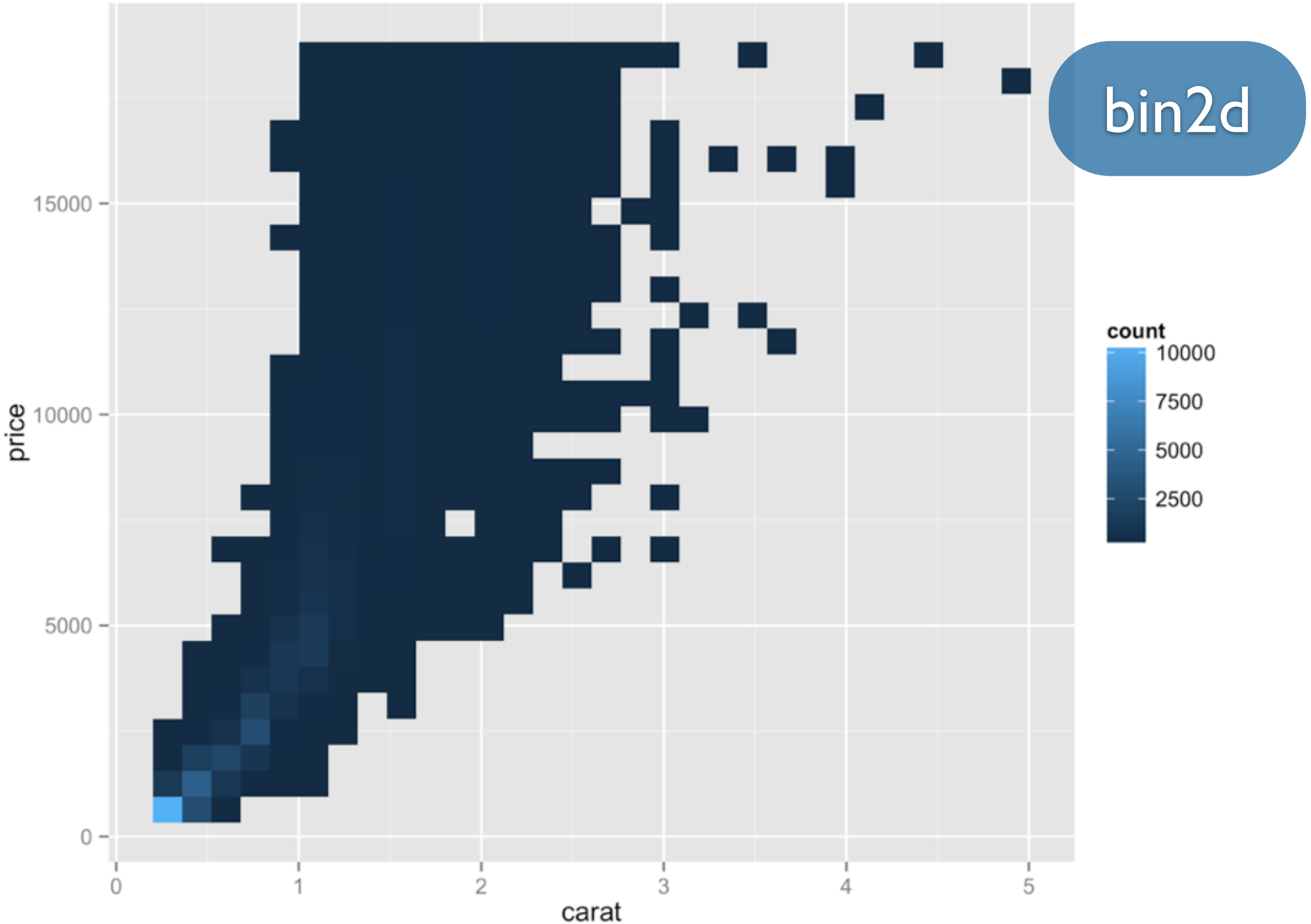



```
qplot(price, data = diamonds, geom = "density",
  color = cut)
```

Is this helpful?

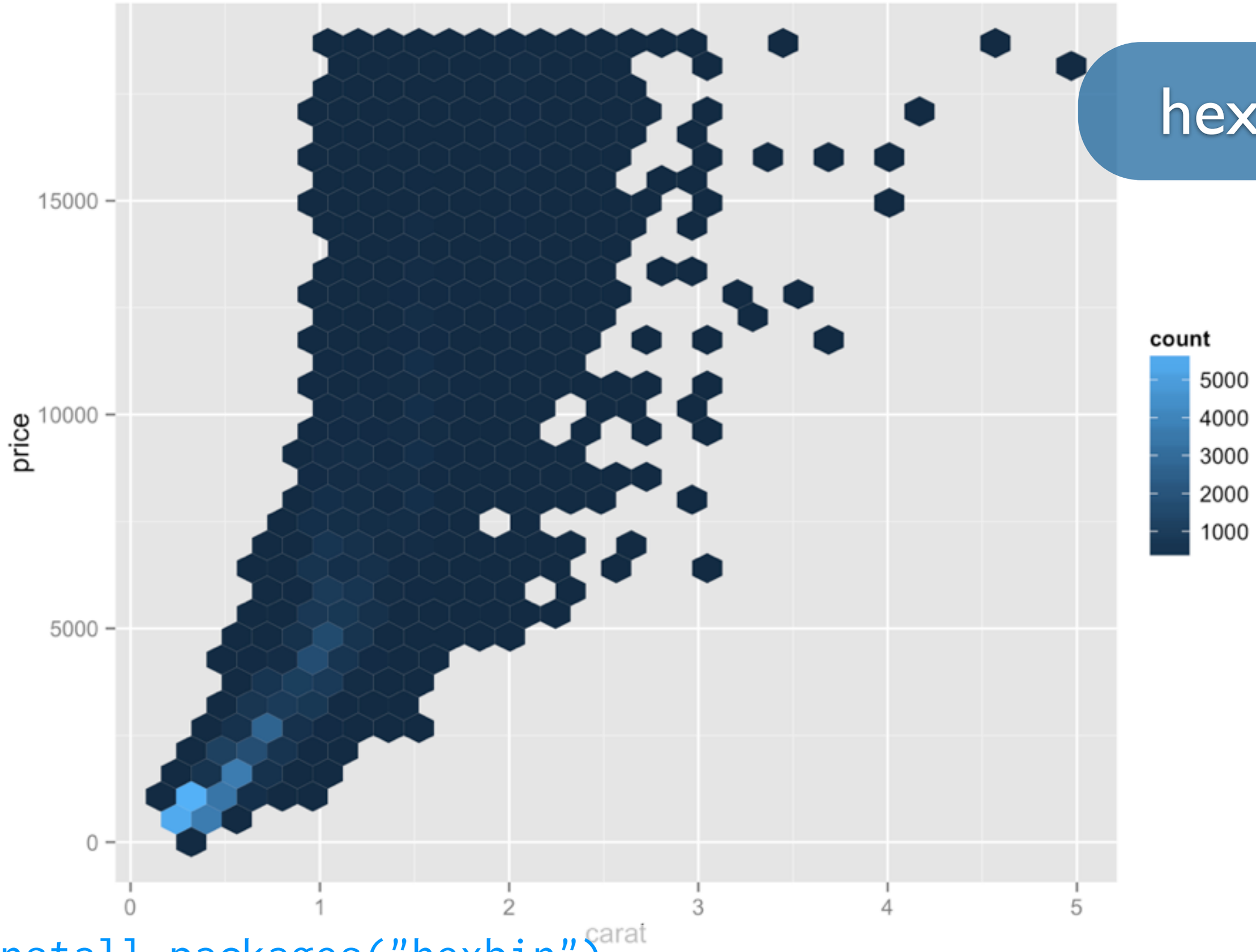
```
qplot(carat, price, data = diamonds, color = cut)
```

Geoms for Big Data



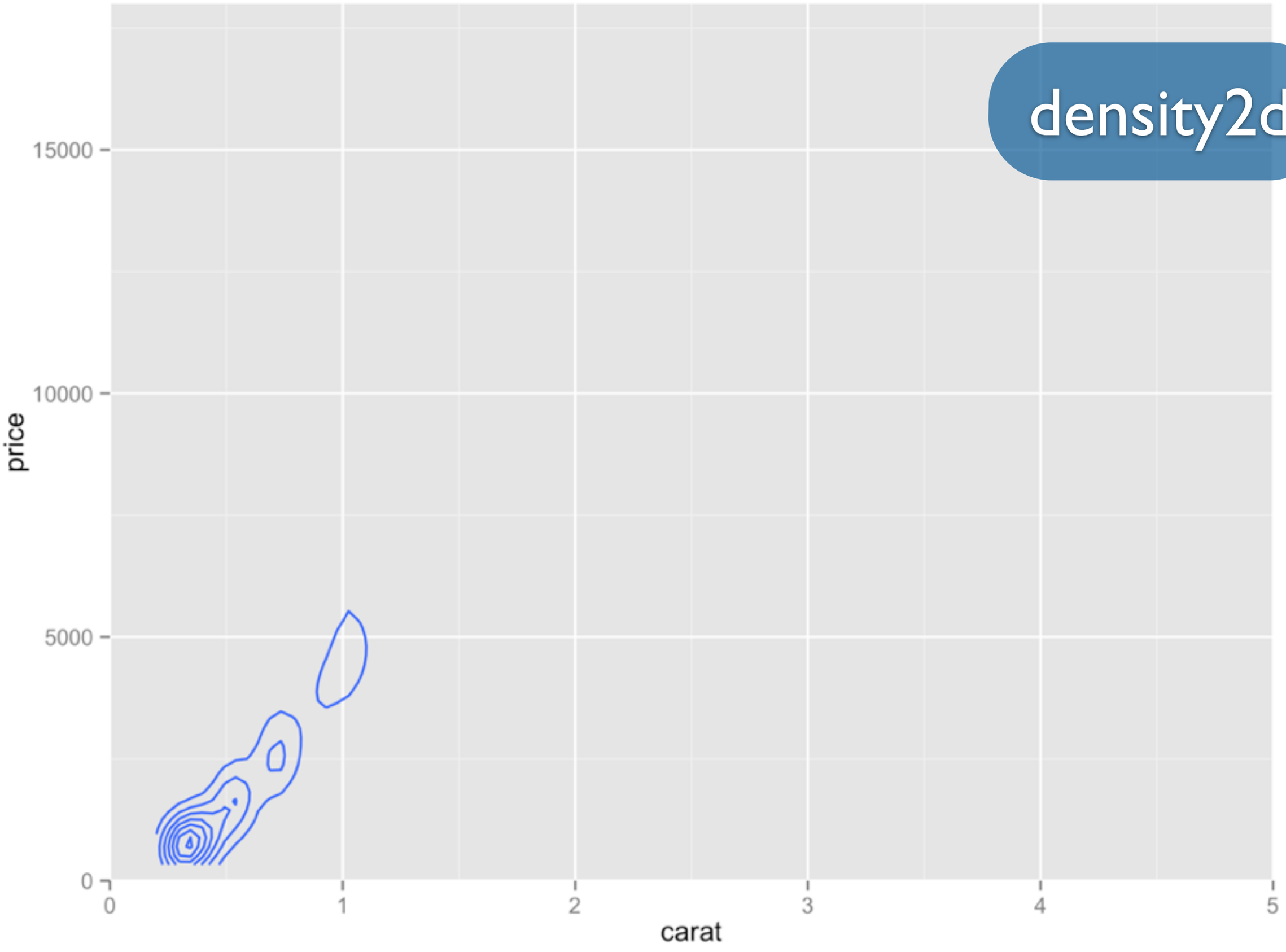

```
qplot(carat, price, data = diamonds, geom = "bin2d")
```

hex

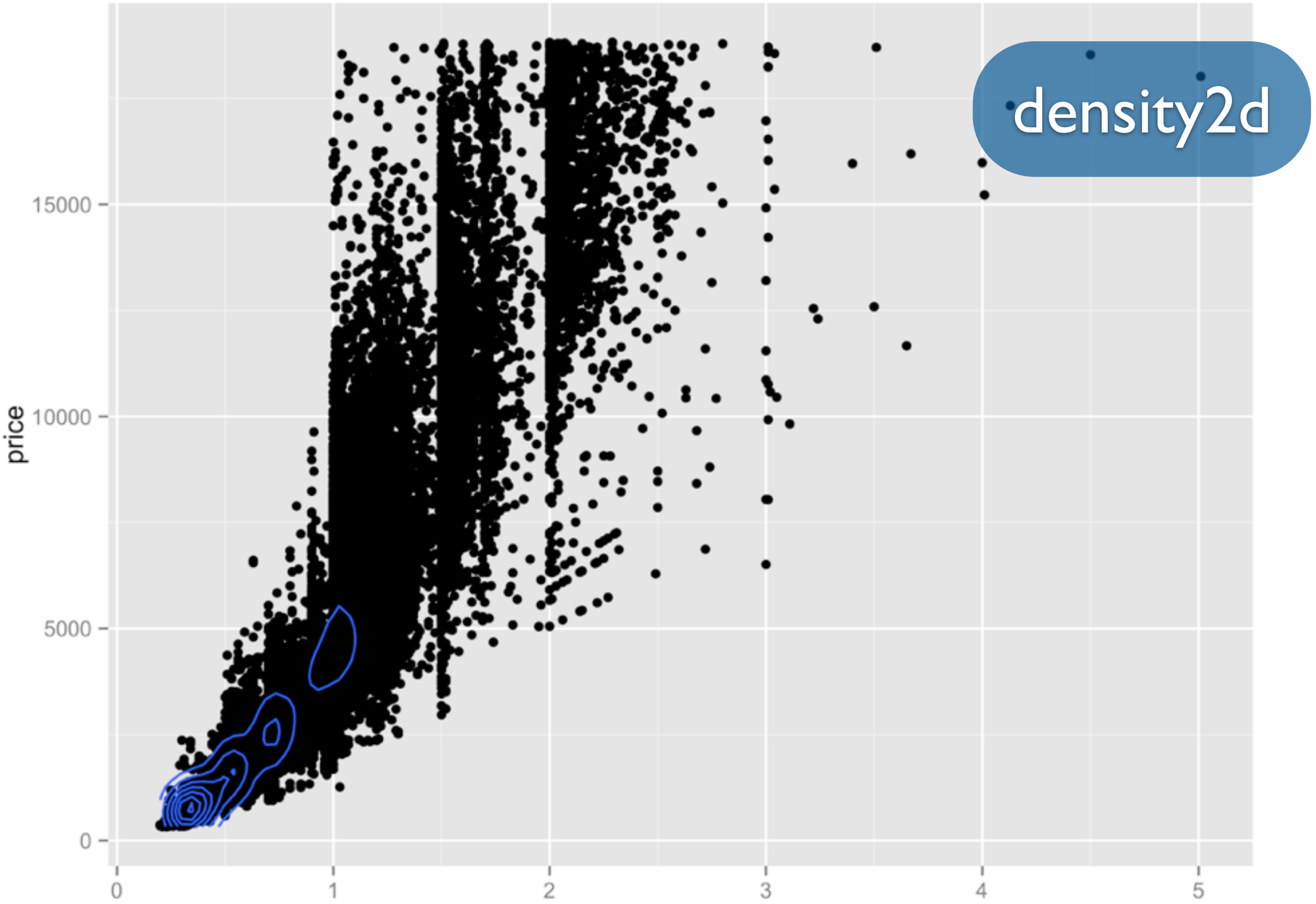



```
# install.packages("hexbin")
qplot(carat, price, data = diamonds, geom = "hex")
```

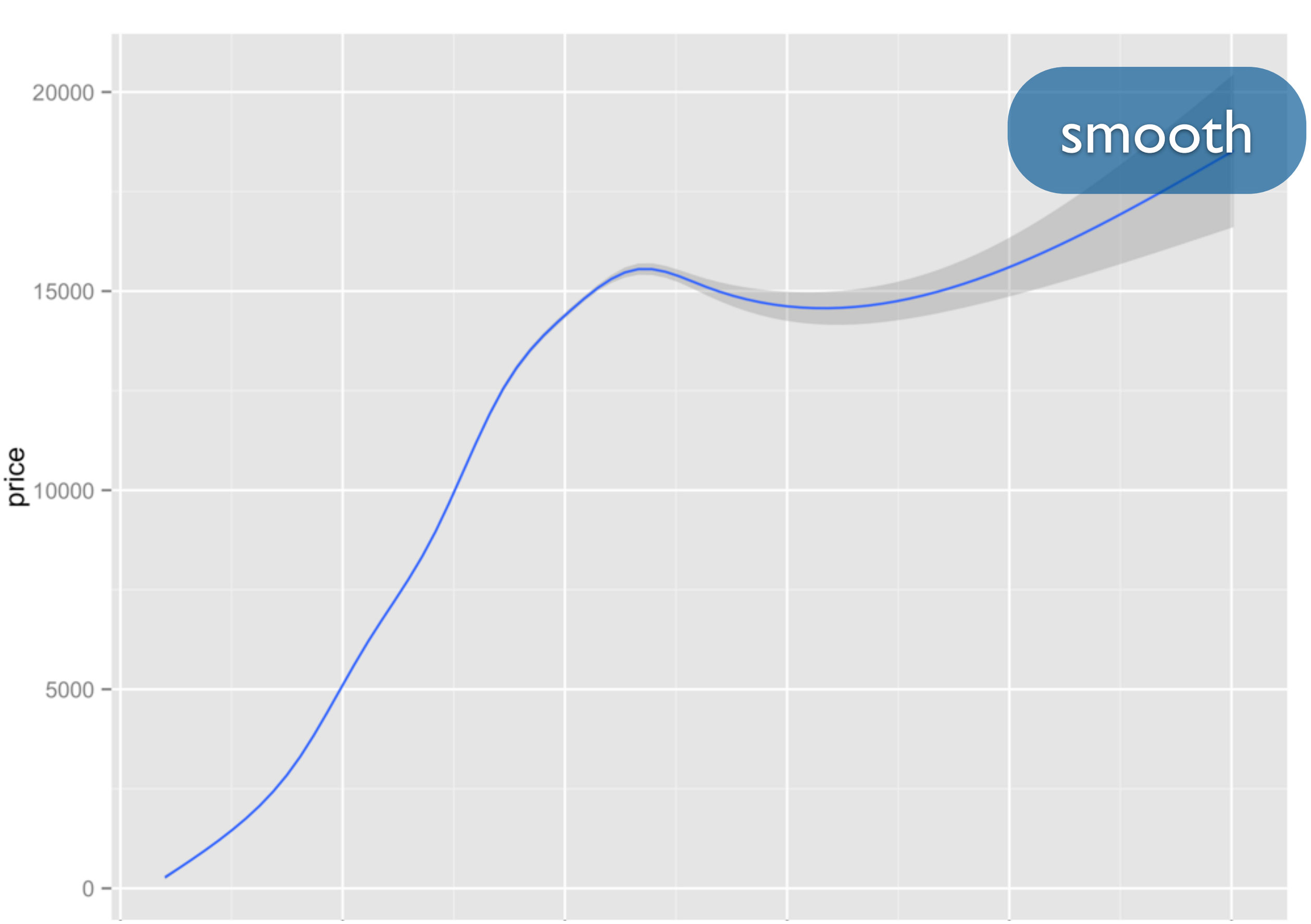
density2d



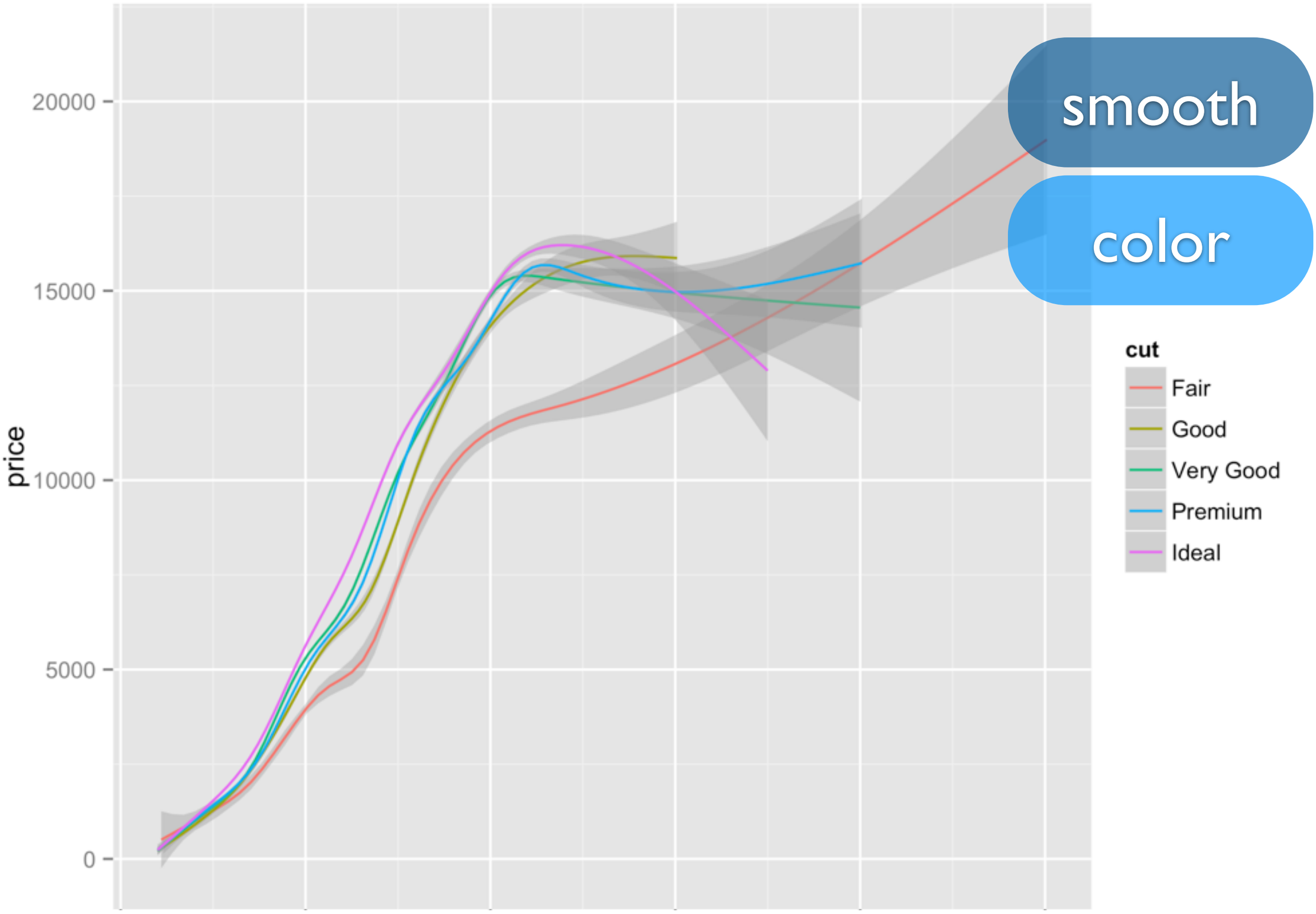

```
qplot(carat, price, data = diamonds, geom = "density2d")
```



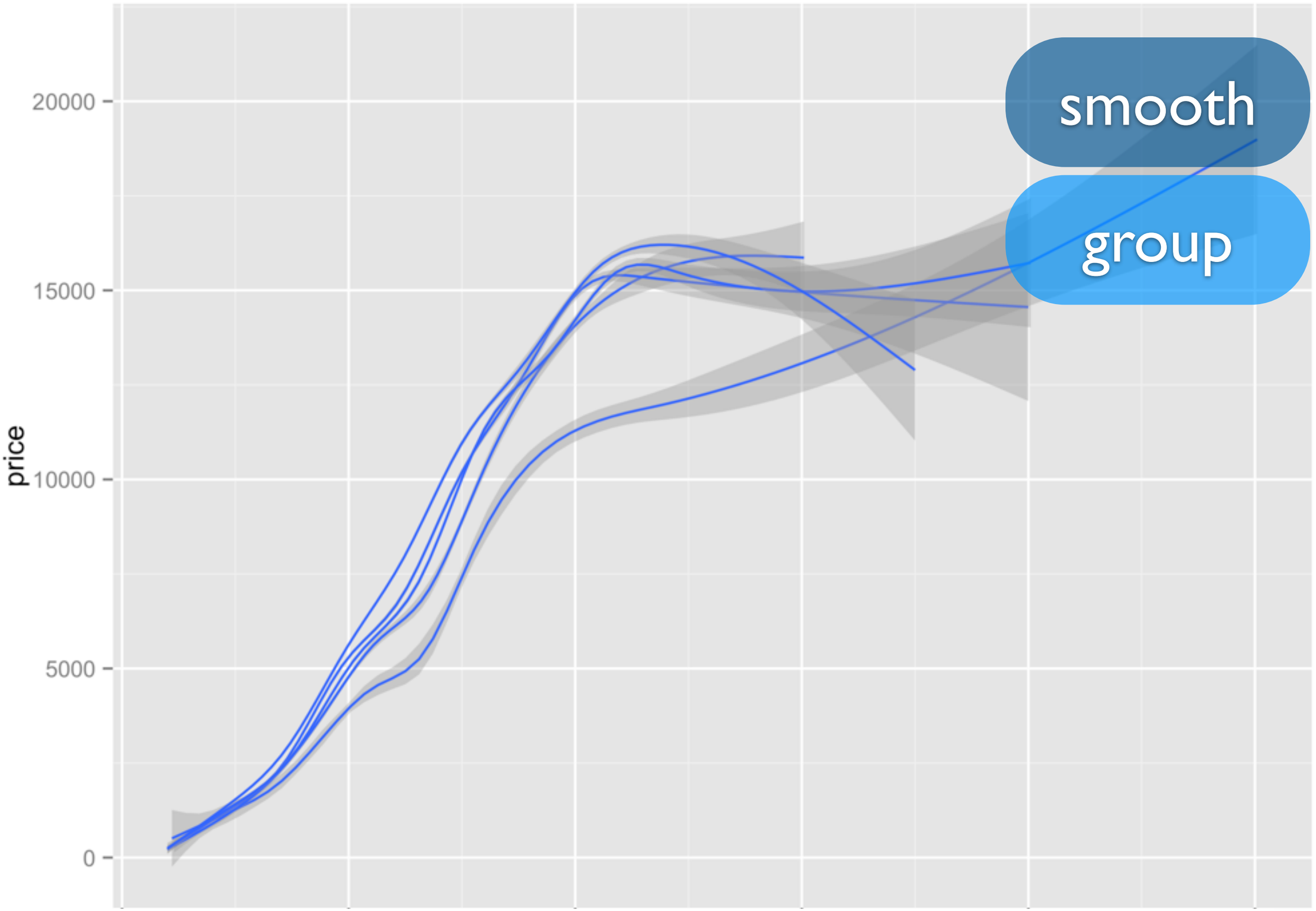

```
qplot(carat, price, data = diamonds,
  geom = c("point", "density2d"))
```



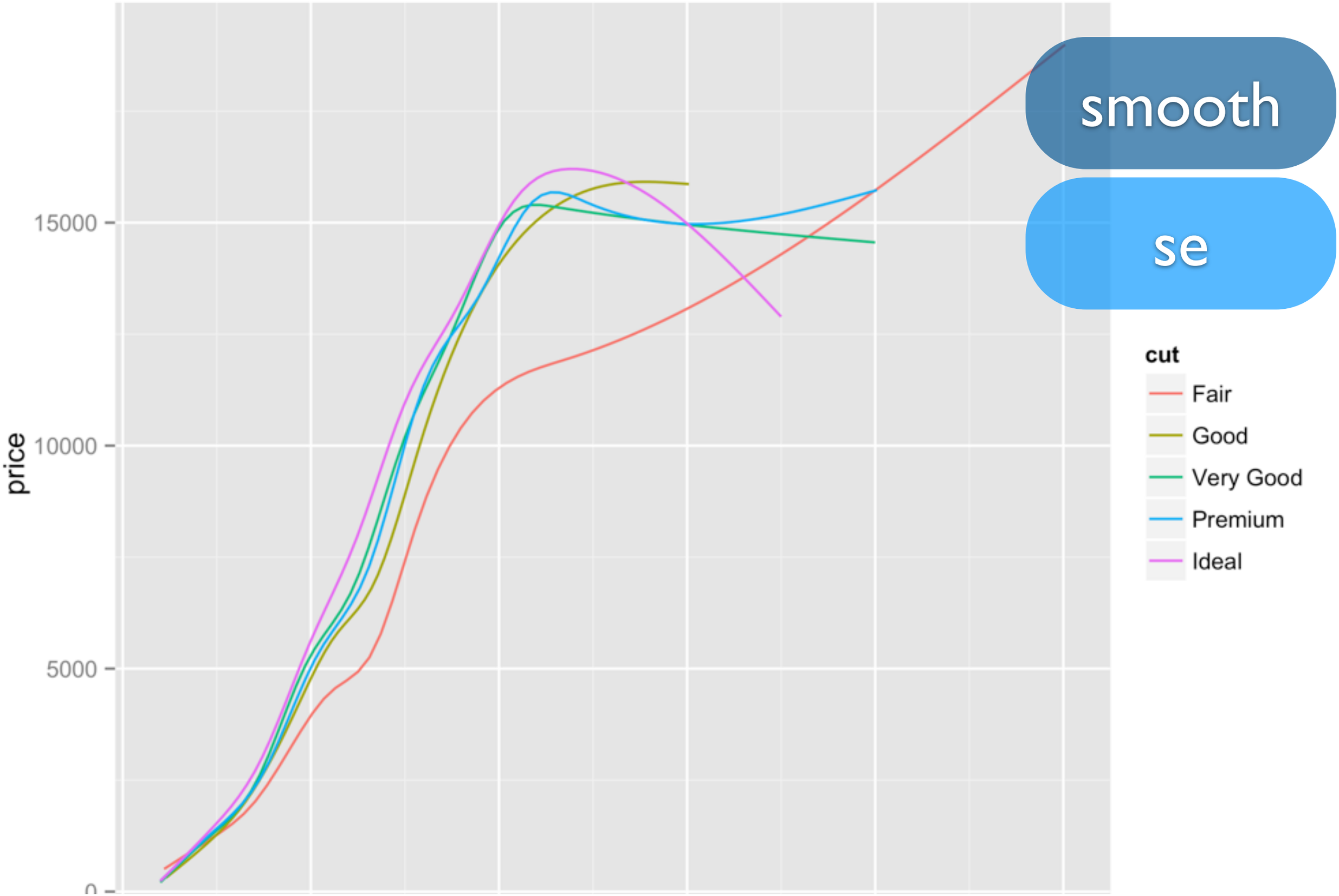

```
qplot(carat, price, data = diamonds, geom = "smooth")
```



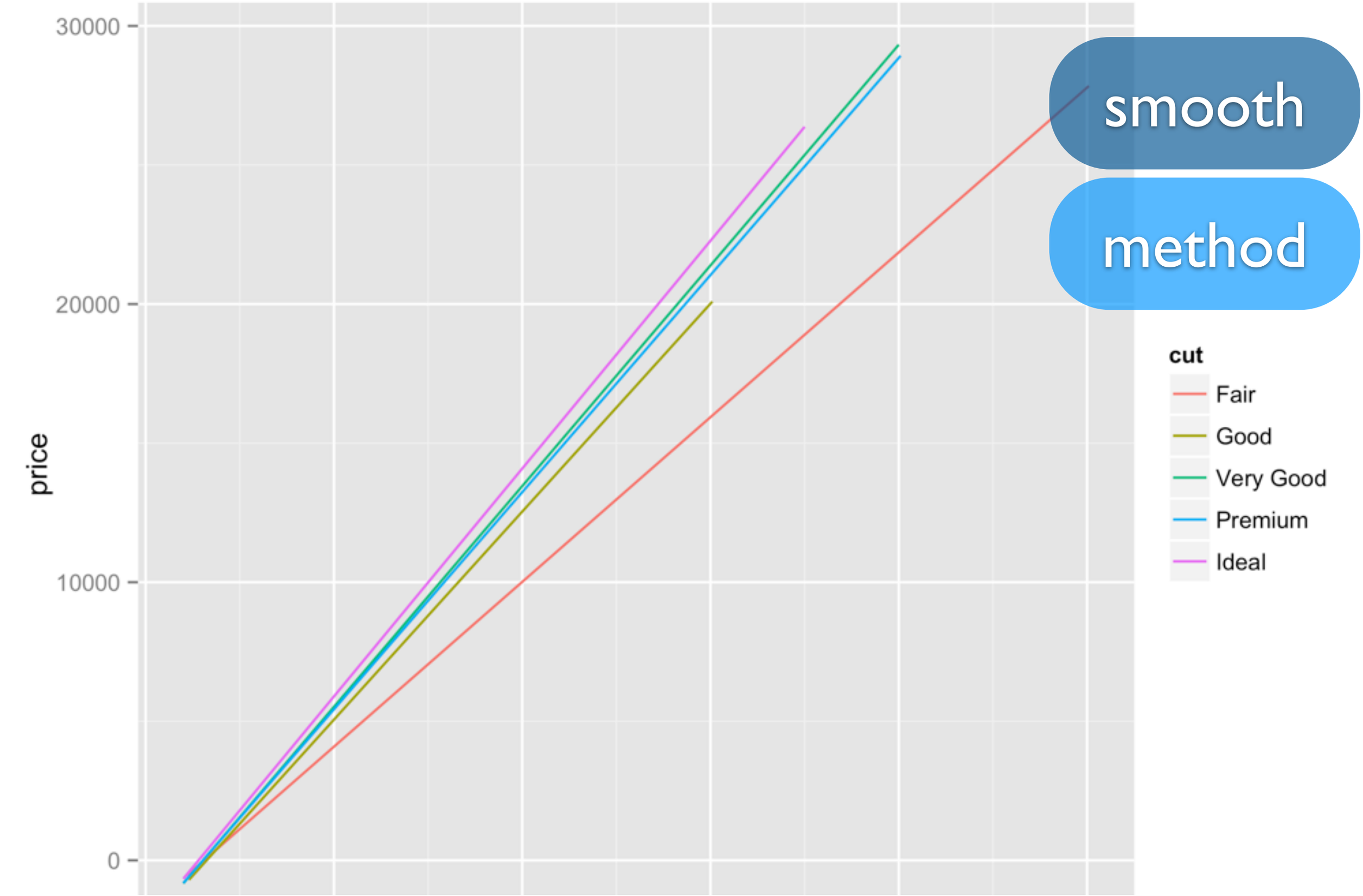

```
qplot(carat, price, data = diamonds, geom = "smooth", color = cut)
```




```
qplot(carat, price, data = diamonds, geom = "smooth", group = cut)
```

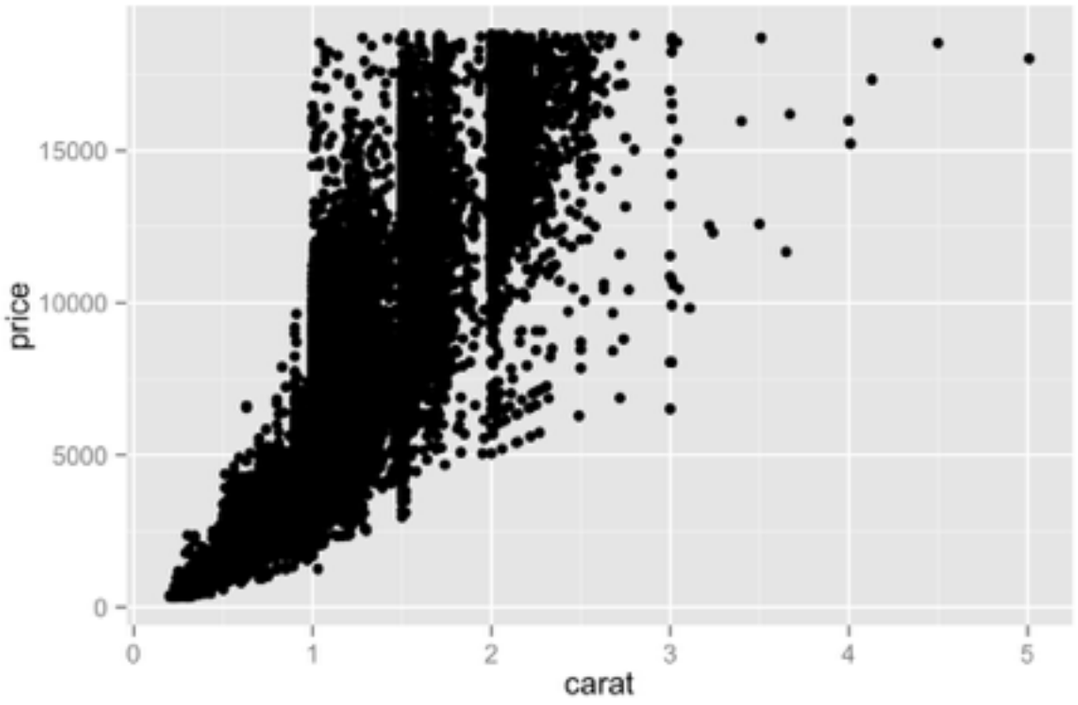


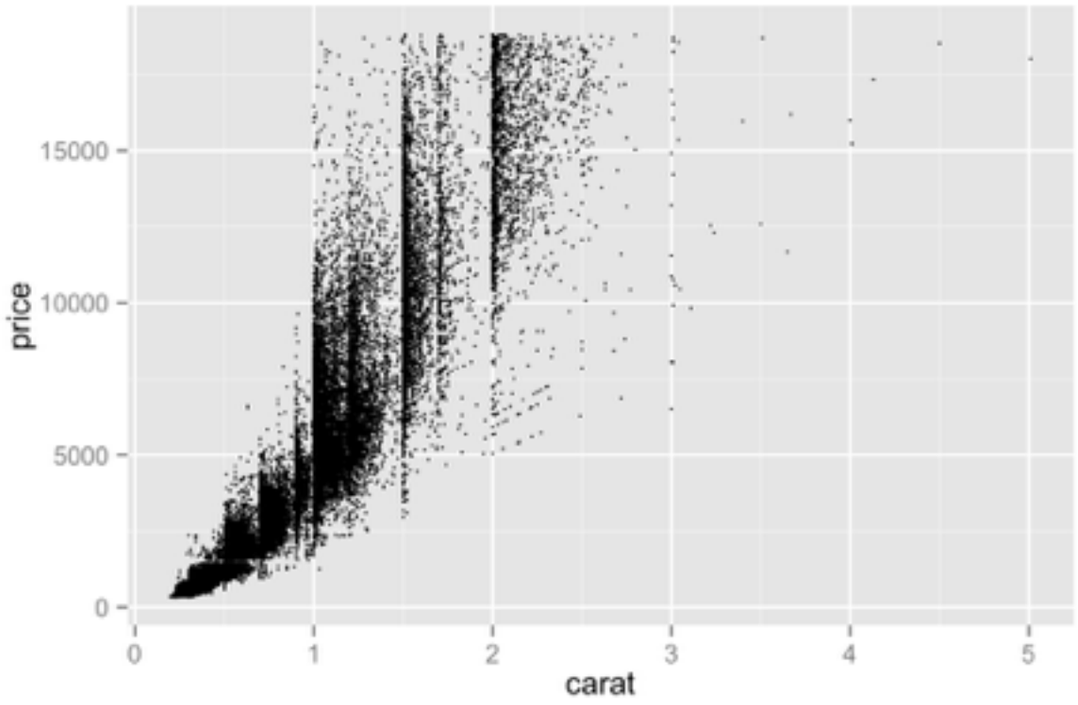

```
qplot(carat, price, data = diamonds, geom = "smooth",
  color = cut, se = FALSE)
```

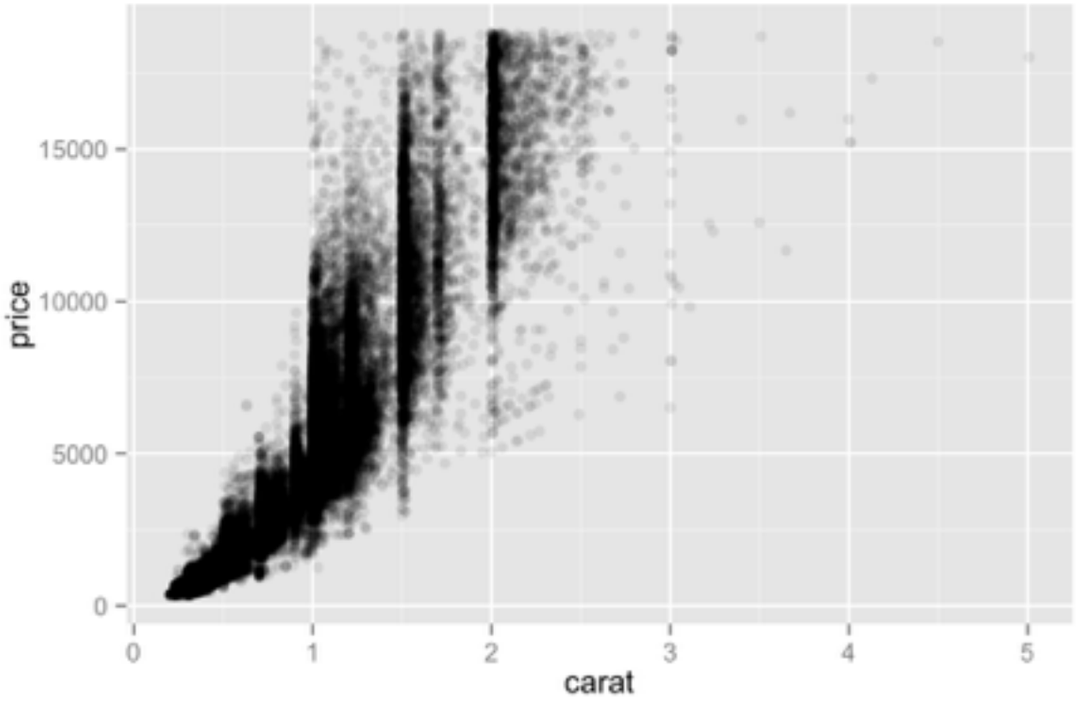


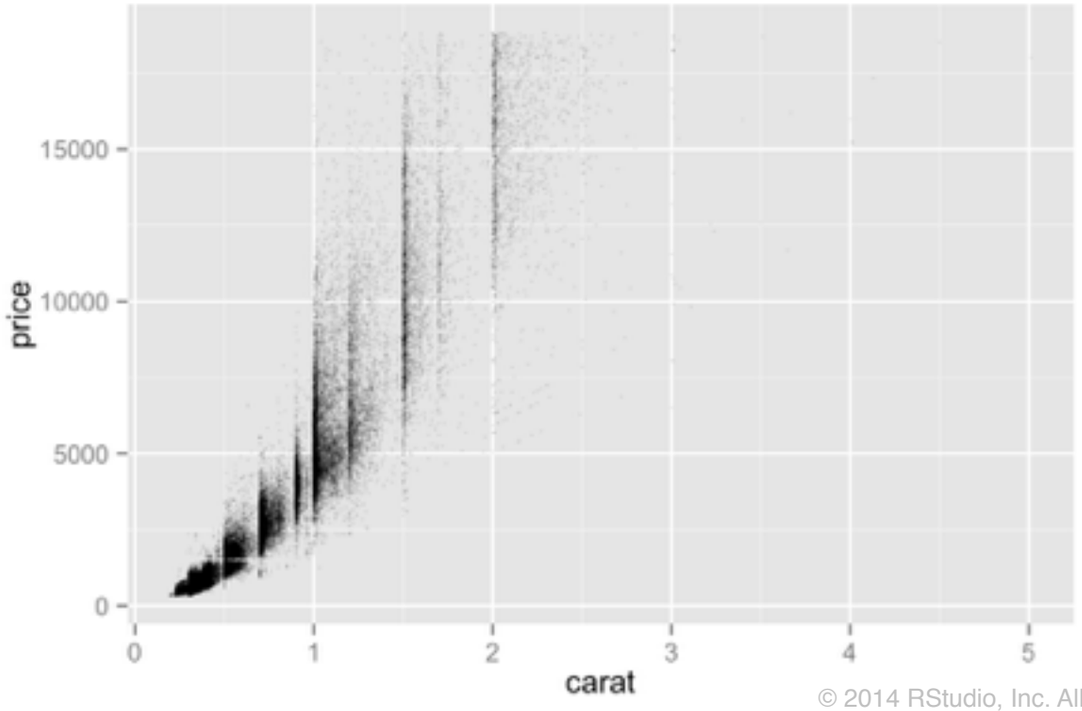

```
qplot(carat, price, data = diamonds, geom = "smooth",
  color = cut, se = FALSE, method = lm)
```

Size and transparency





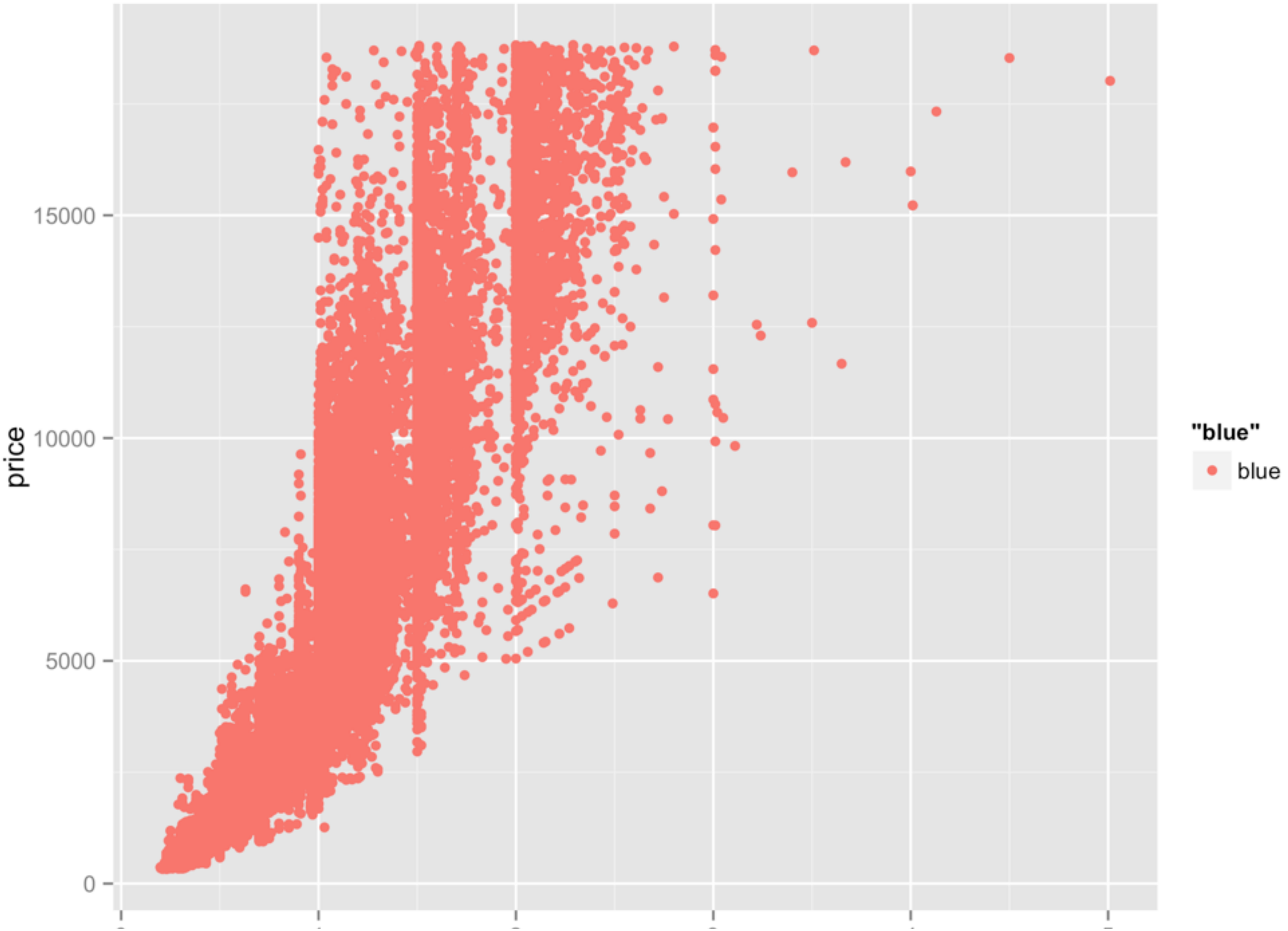




Your turn

What will this code do?

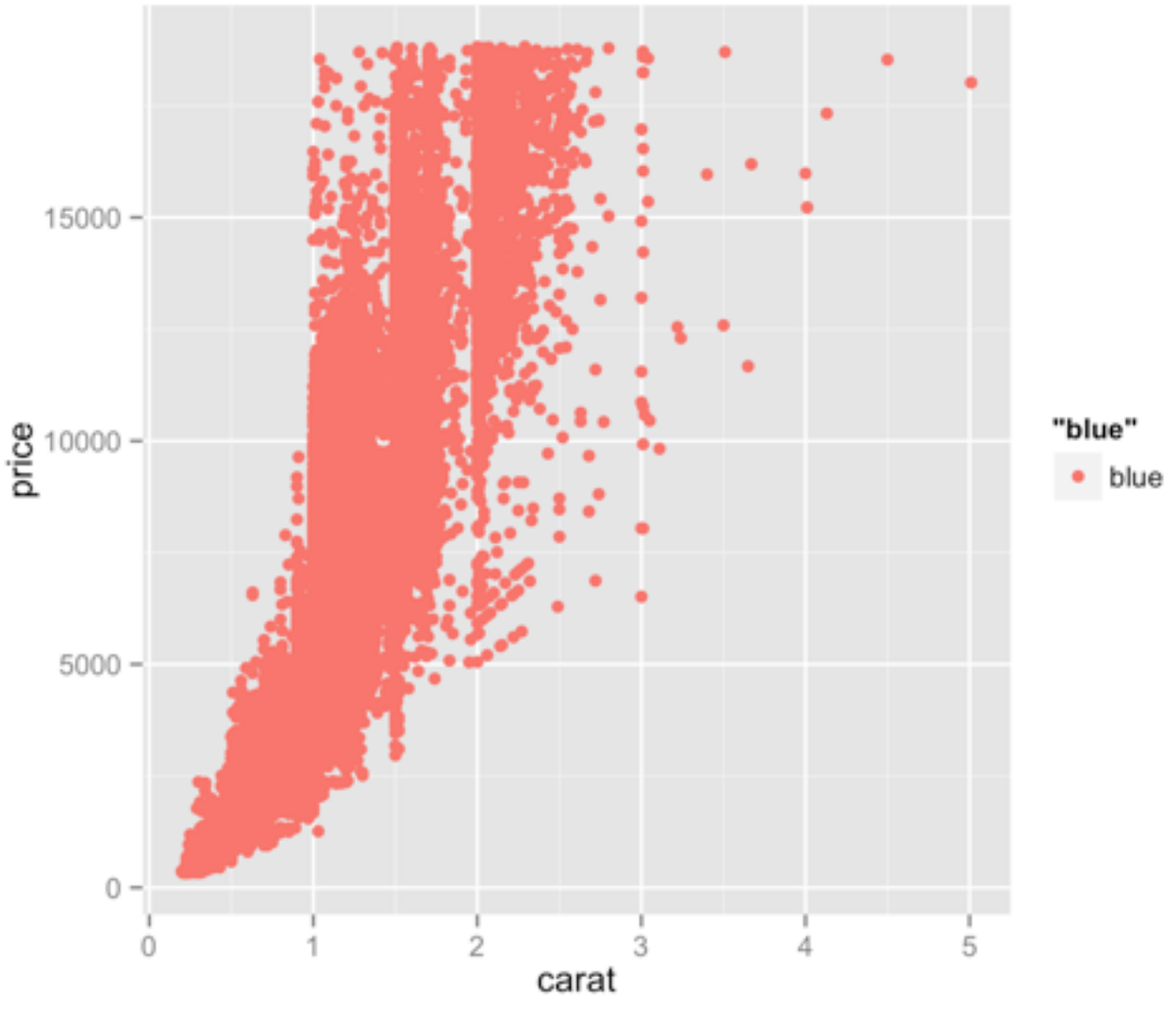
```
qplot(carat, price, data = diamonds, color = "blue")
```



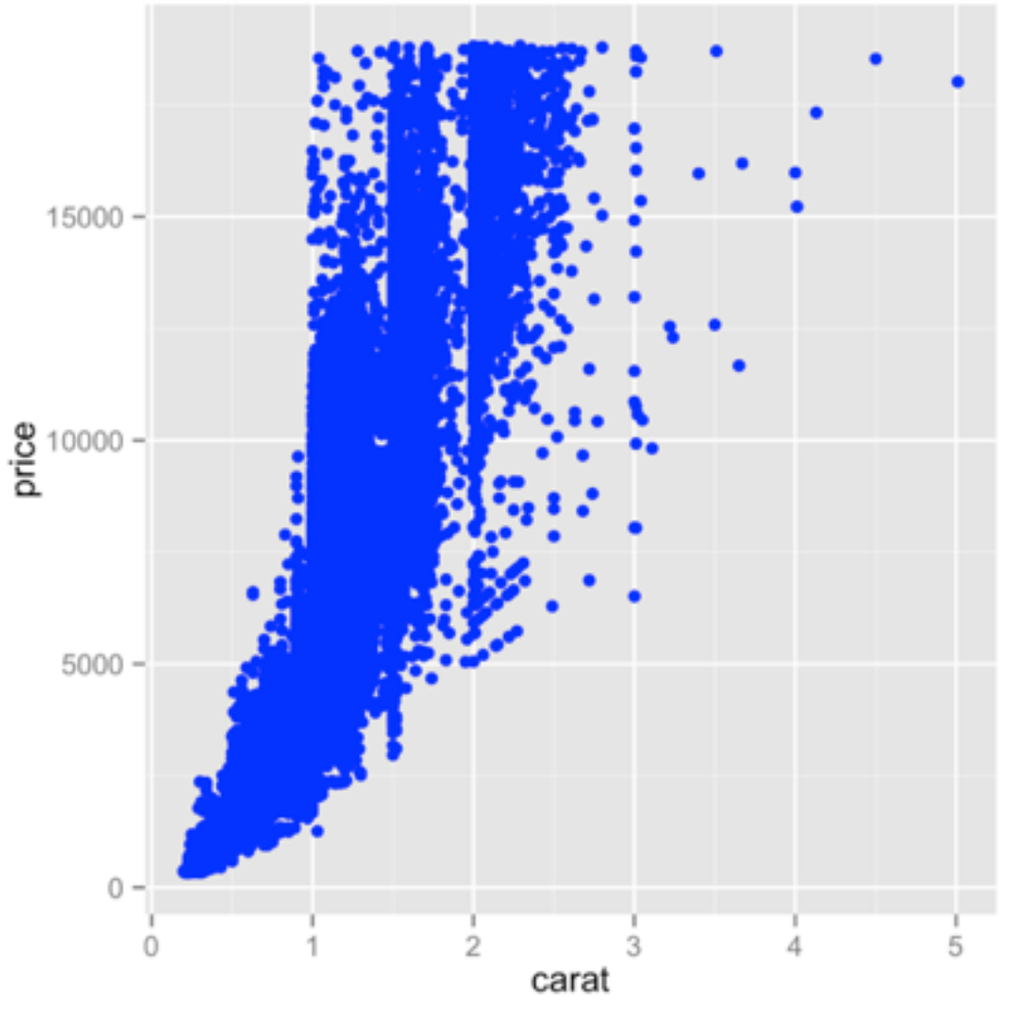

```
qplot(carat, price, data = diamonds, color = "blue")
```

You can turn an “aesthetic” into a parameter by surrounding the value with `I()`

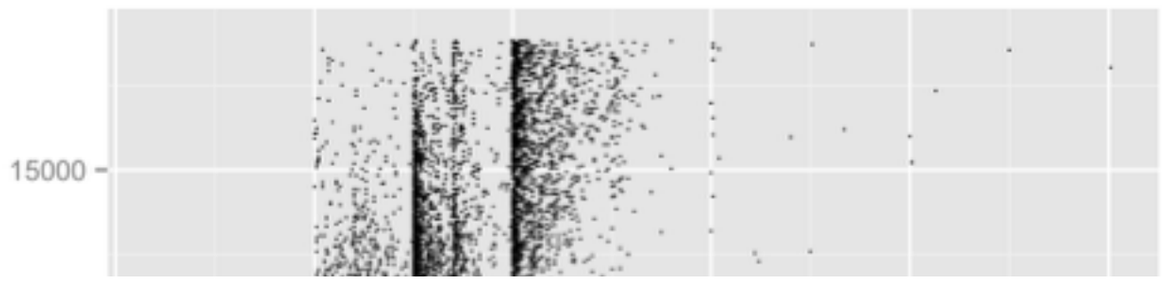
```
qplot(carat, price, data = diamonds,
      color = "blue")
```



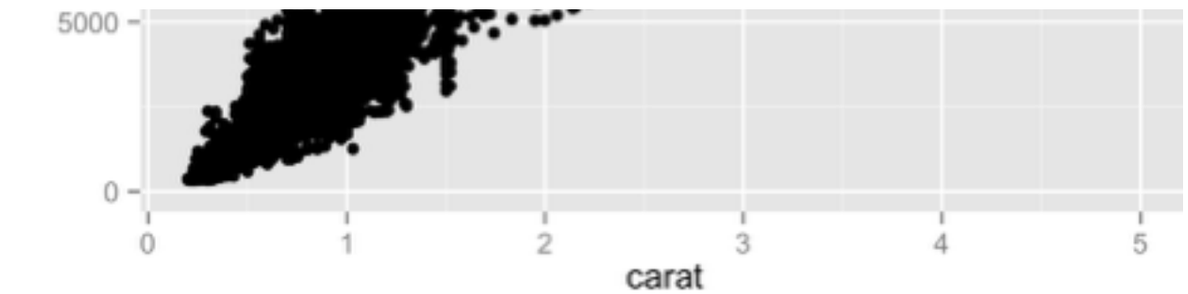

```
qplot(carat, price, data = diamonds,
      color = I("blue"))
```

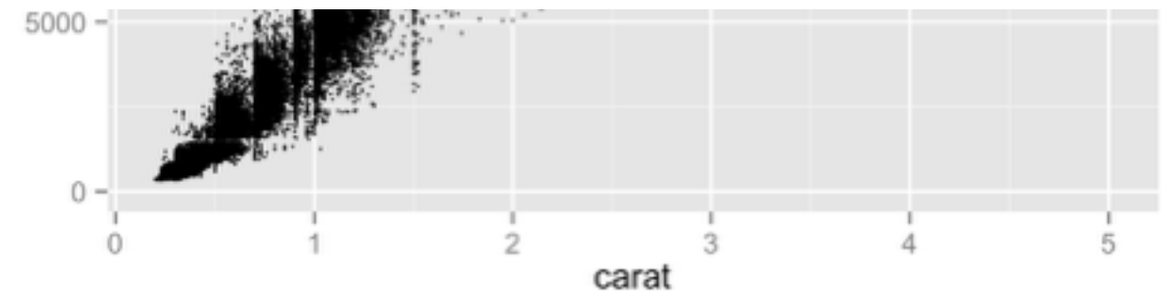


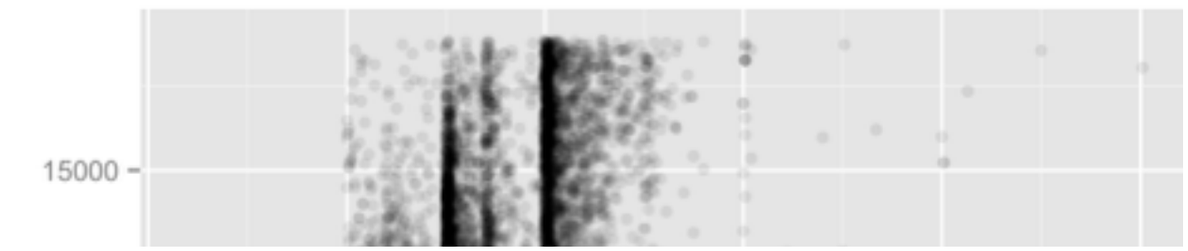

```
qplot(carat, price, data = diamonds)
```



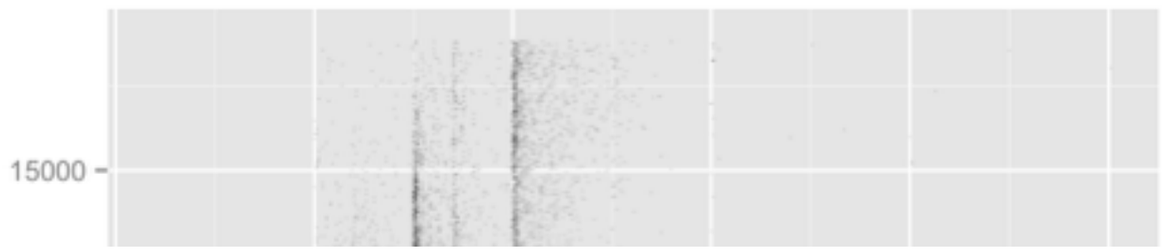

```
qplot(carat, price, data = diamonds,
      size = I(0.5))
```



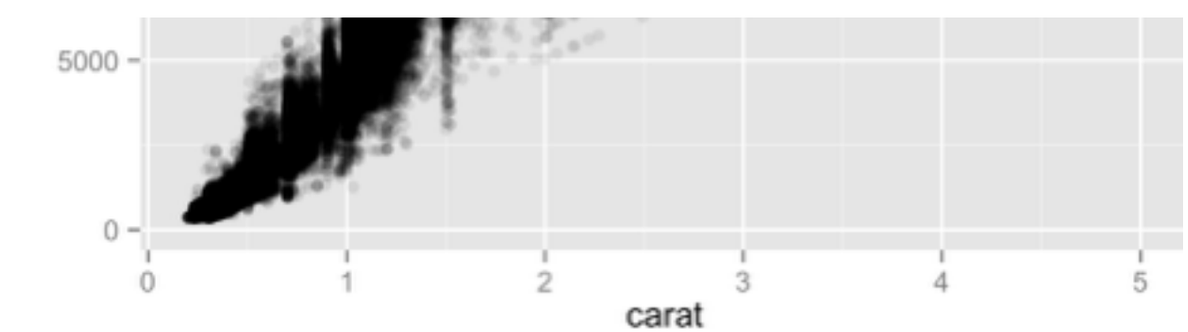


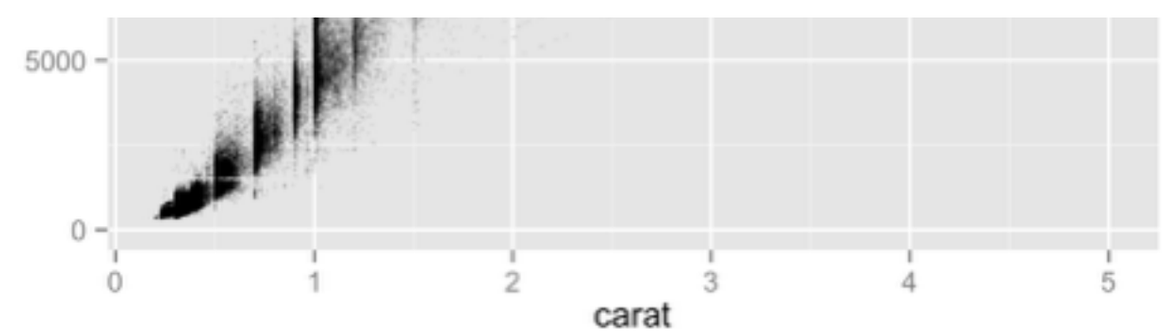



```
qplot(carat, price, data = diamonds,
      alpha = I(0.1))
```




```
qplot(carat, price, data = diamonds,
      size = I(0.5), alpha = I(0.1))
```





Saving graphs

Your turn

What does this command return?

```
getwd()
```

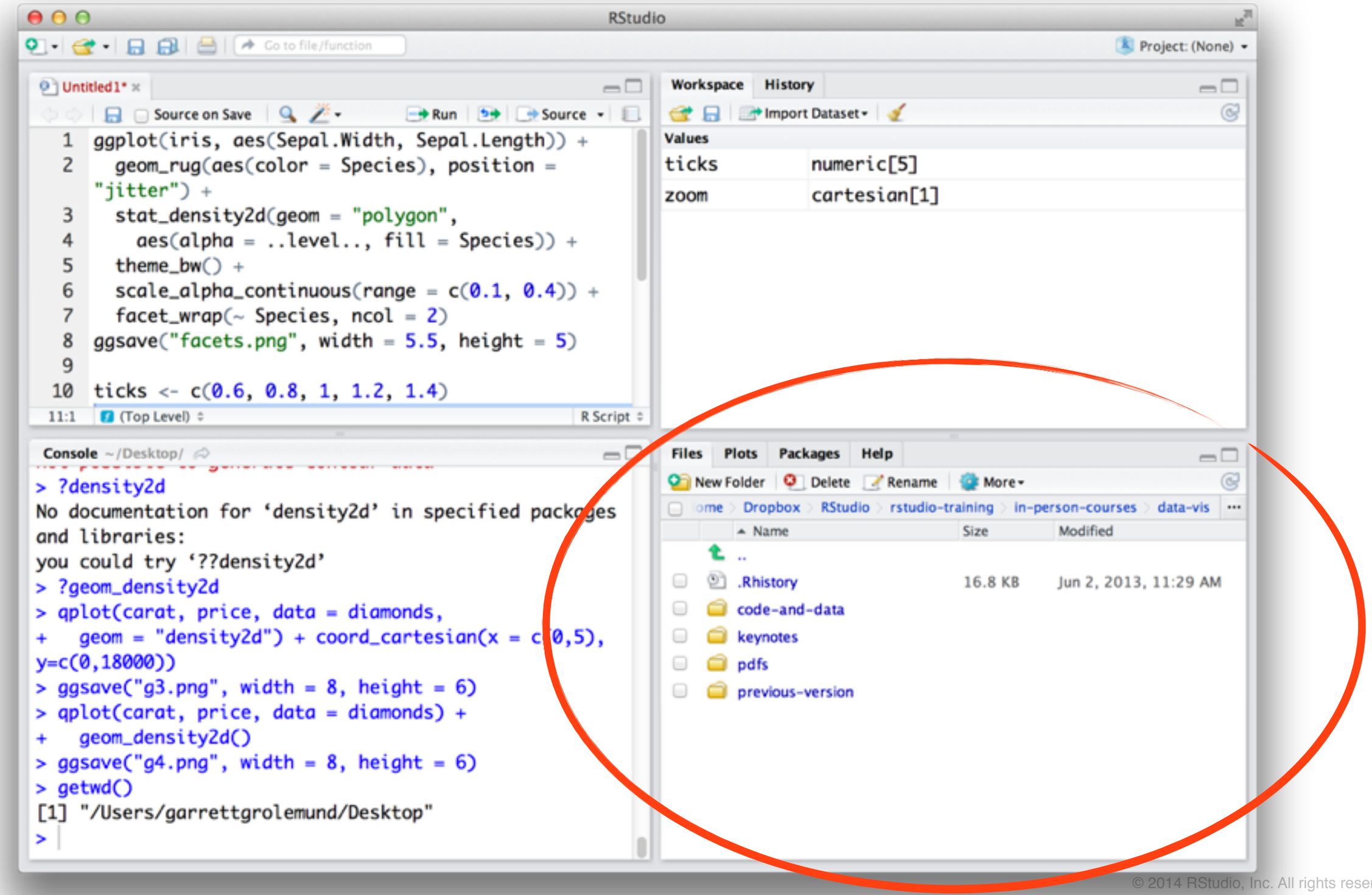
Could you find that file on your computer?

Working directory

When you start R, it associates itself with a folder (i.e, directory) on your computer.

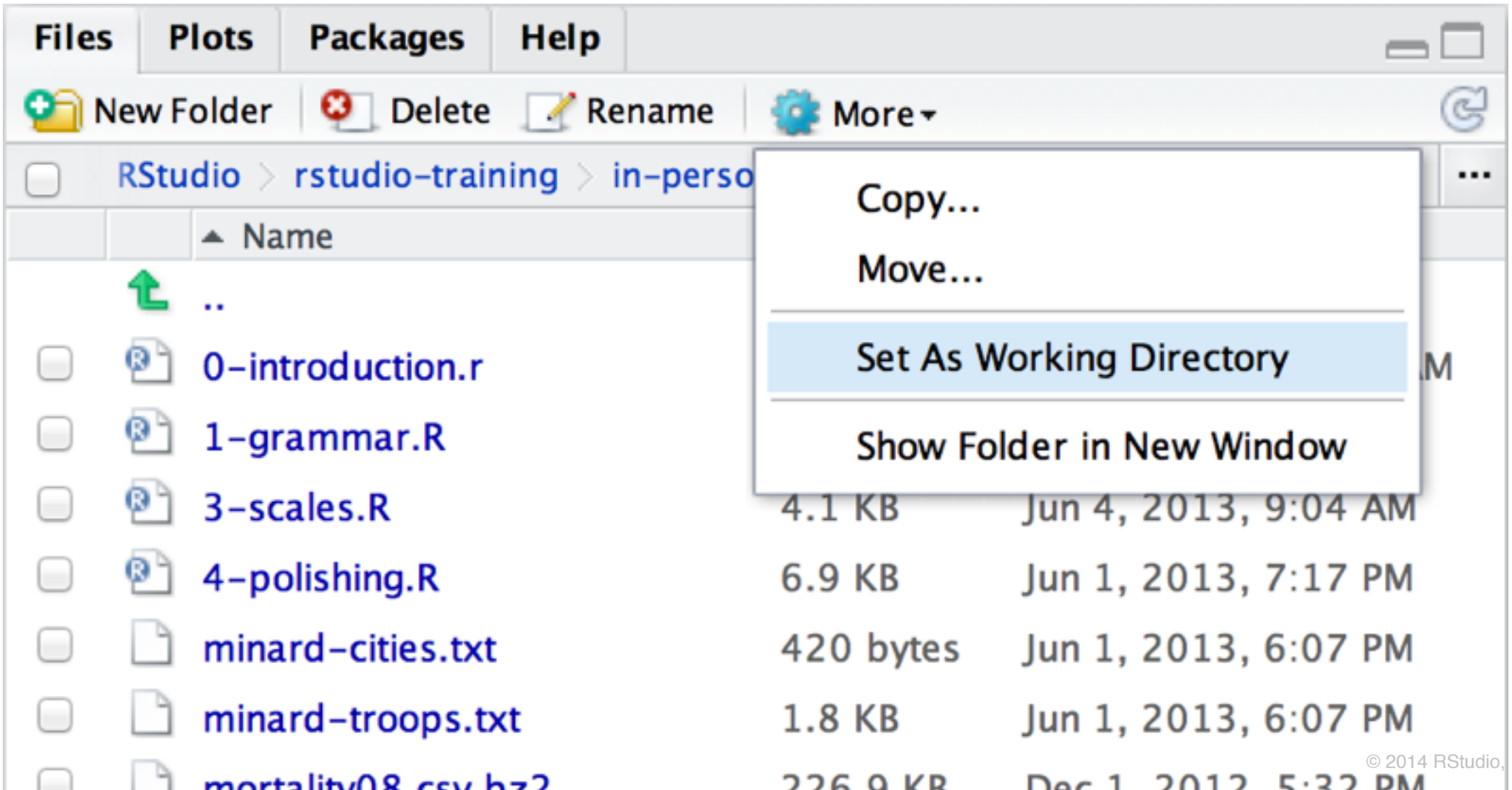
- This folder is known as your "working directory"
- When you save files, R will save them here
- When you load files, R will look for them here

The files pane of RStudio displays the contents of your working directory



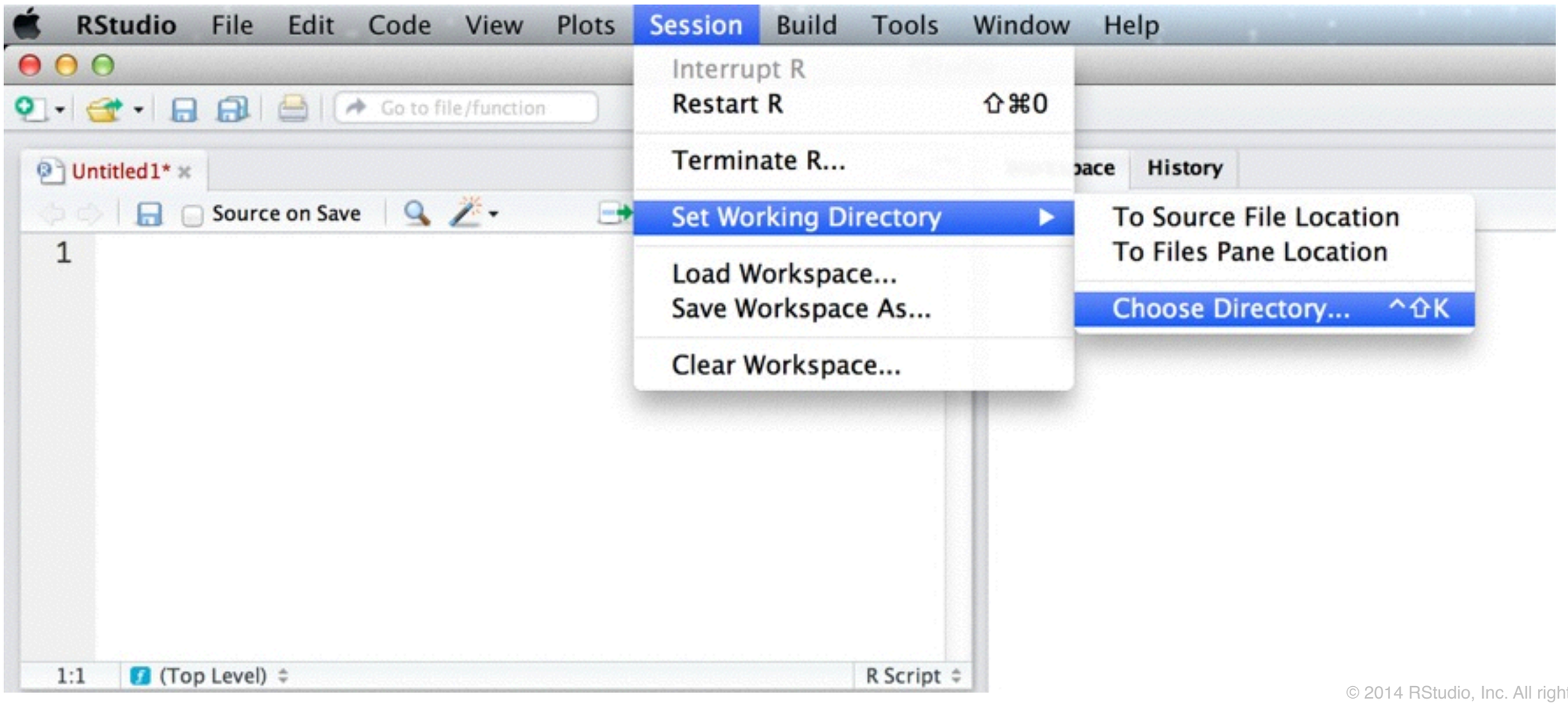
Changing the Working directory

First option: Navigate in the files pane to a new directory. Click More>Set As Working Directory



Changing the Working directory

Second option: In the toolbar, go to Session>Set Working Directory>Choose Directory...



Your turn

Change your working directory to the folder you downloaded for today's course.

Note: this folder came as a .zip archive. You must extract the .zip file before you can use it as a directory.

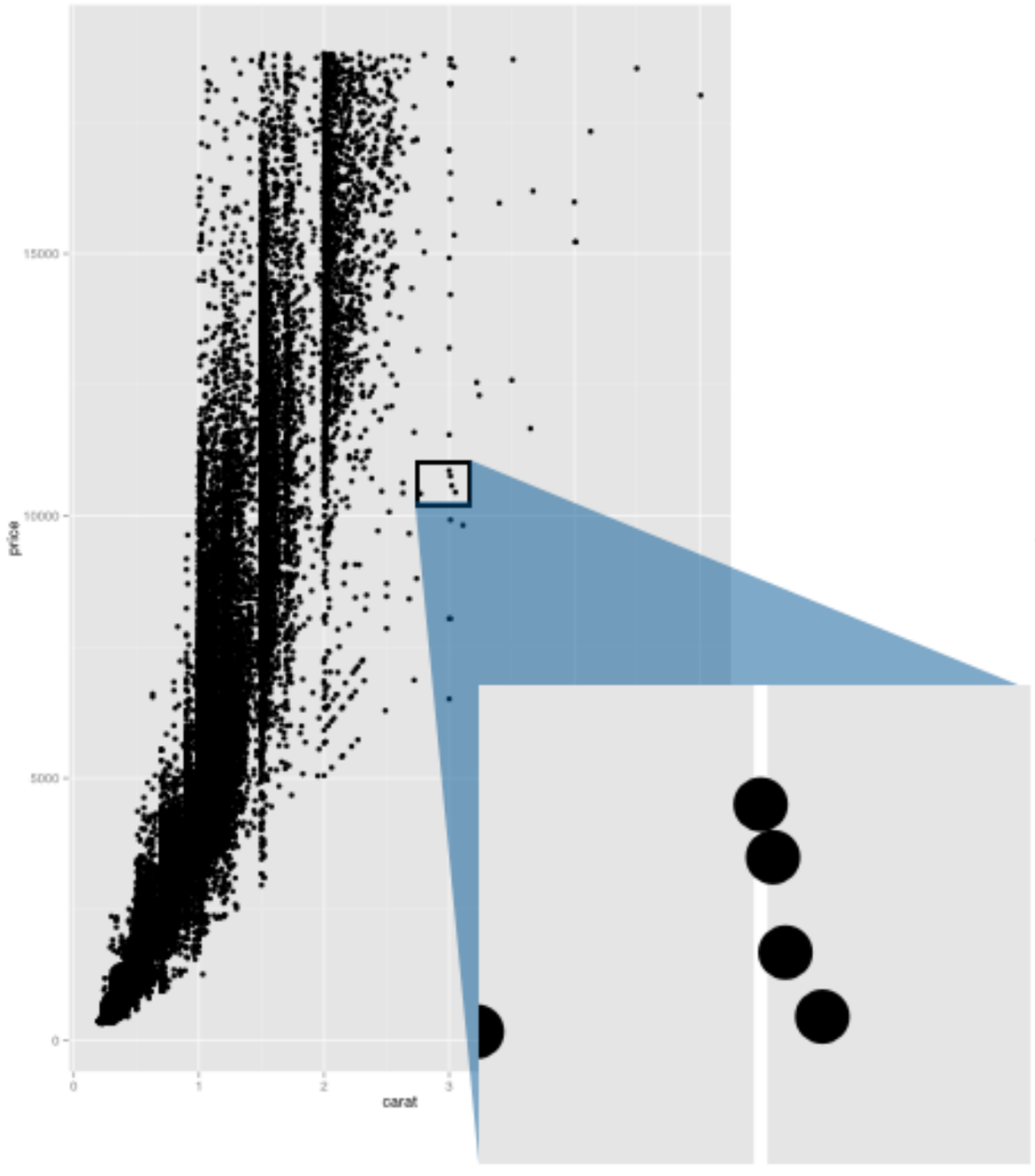
```
# Find out where your working directory is
getwd()
```

```
# List files in that directory
dir()
```

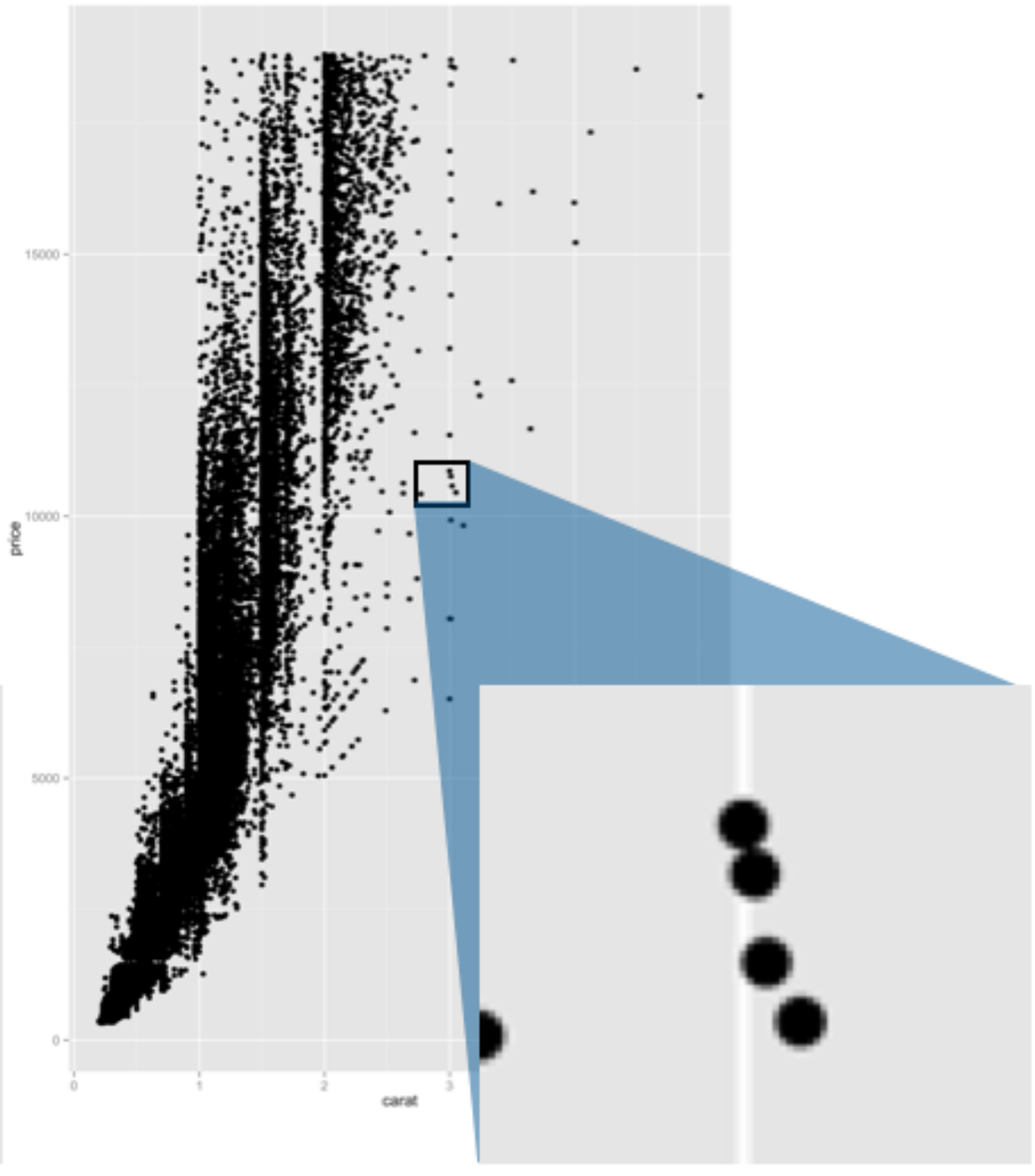
Saving plots

```
# Uses size on screen:
ggsave("my-plot.pdf")
ggsave("my-plot.png")

# Specify size in inches
ggsave("my-plot.pdf", width = 6, height = 6)
```



pdf



png

PDF	PNG
Vector based (can zoom in infinitely)	Raster based (made up of pixels)
Good for most plots	Good for plots with thousands of points

Summary

qplot + aesthetics = 100's of plots

qplot + geoms = 100's of plots

qplot + geoms + aesthetics = 1000's of plots

qplot + geoms + aesthetics + parameters = 100,000's