Stat405

Graphical extensions & subsetting

Hadley Wickham

1. Interesting facts
2. Reading scatterplots
3. Scatterplots for big data
4. Introduction to subsetting

Interesting facts

Your interesting facts

6x speak three or more lanugages

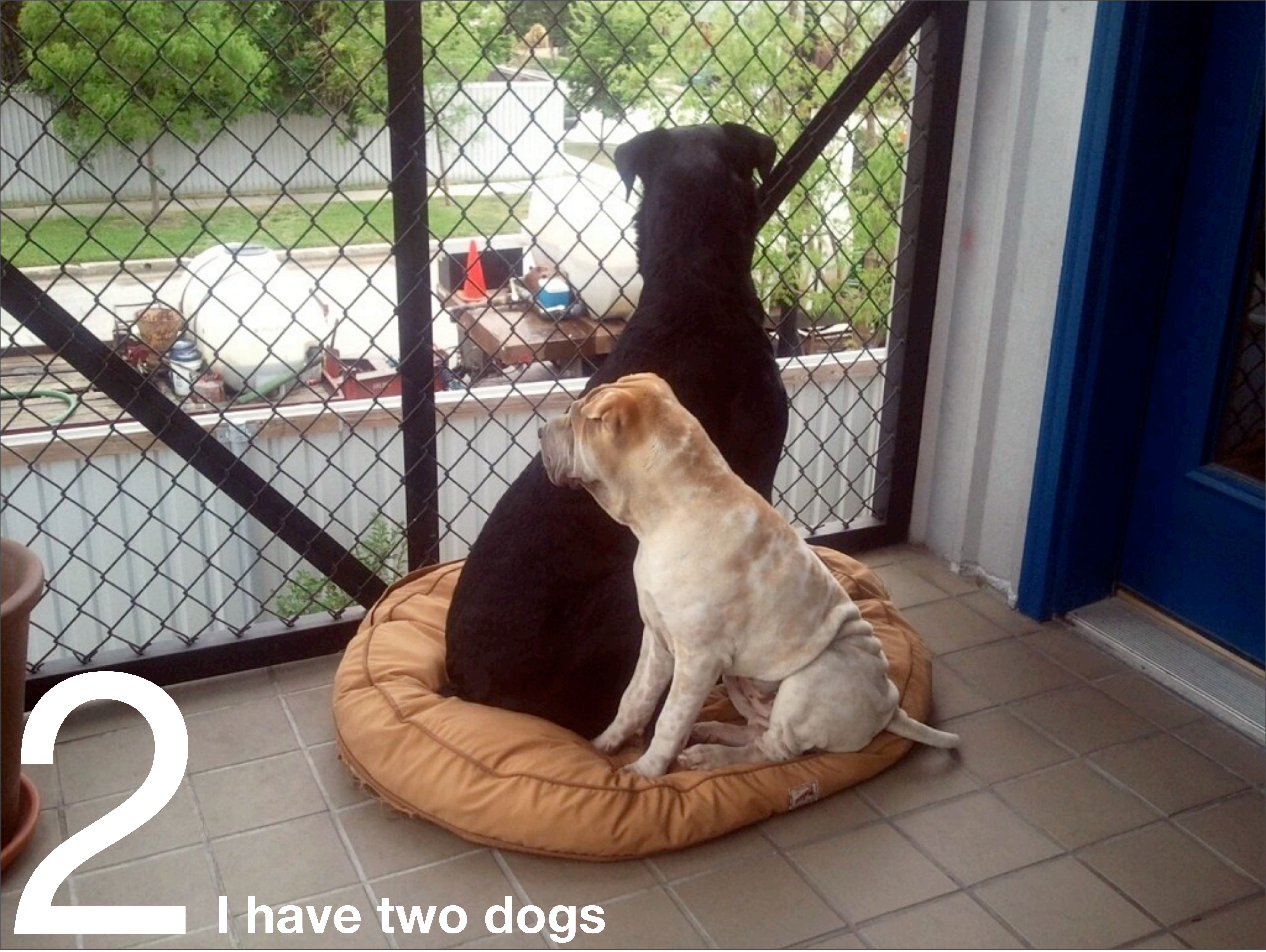
3x mentioned canada

2x black belt in taekwondo

2x likes alpacas/llamas

1x ridden an ostrich


1
I’m from New Zealand

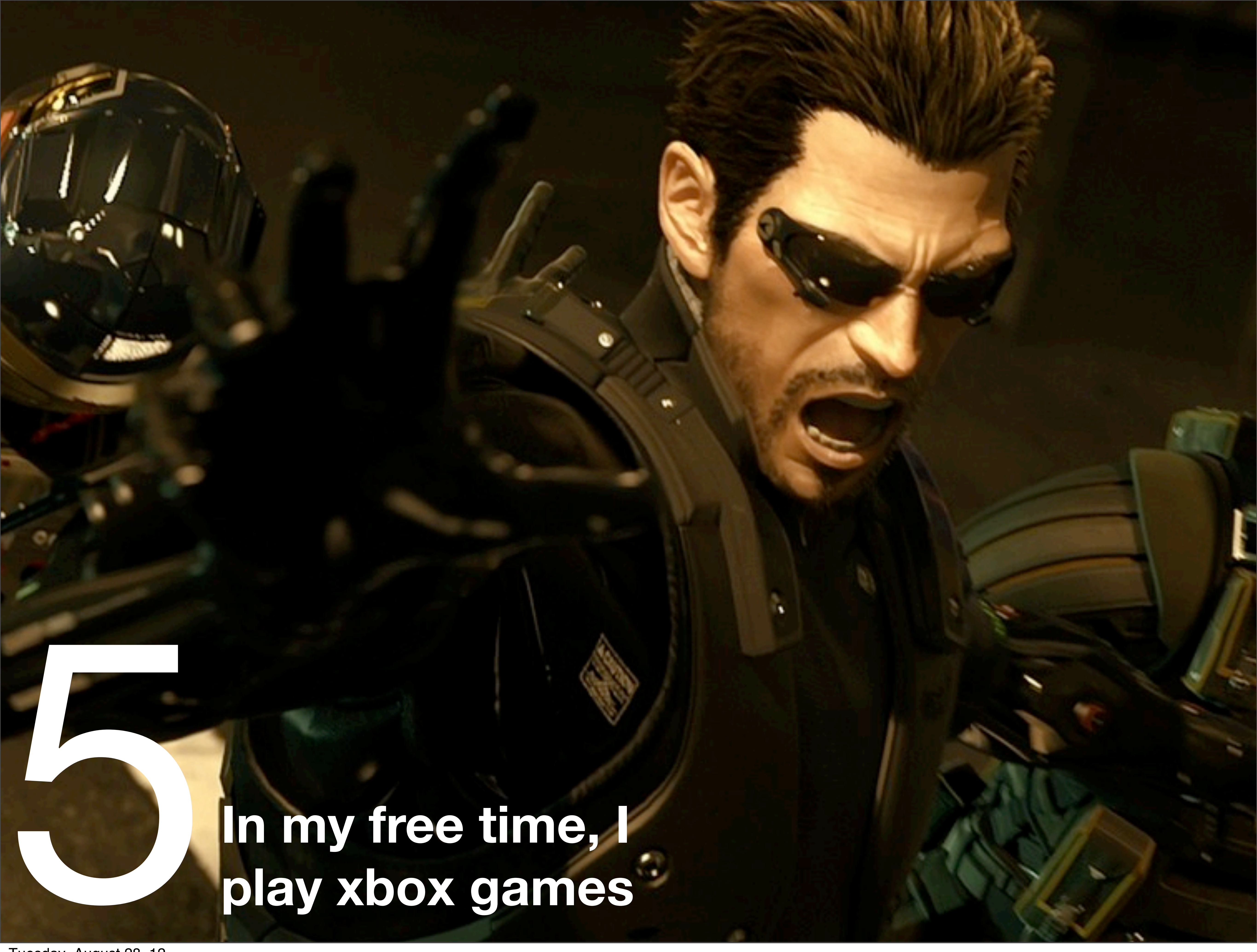


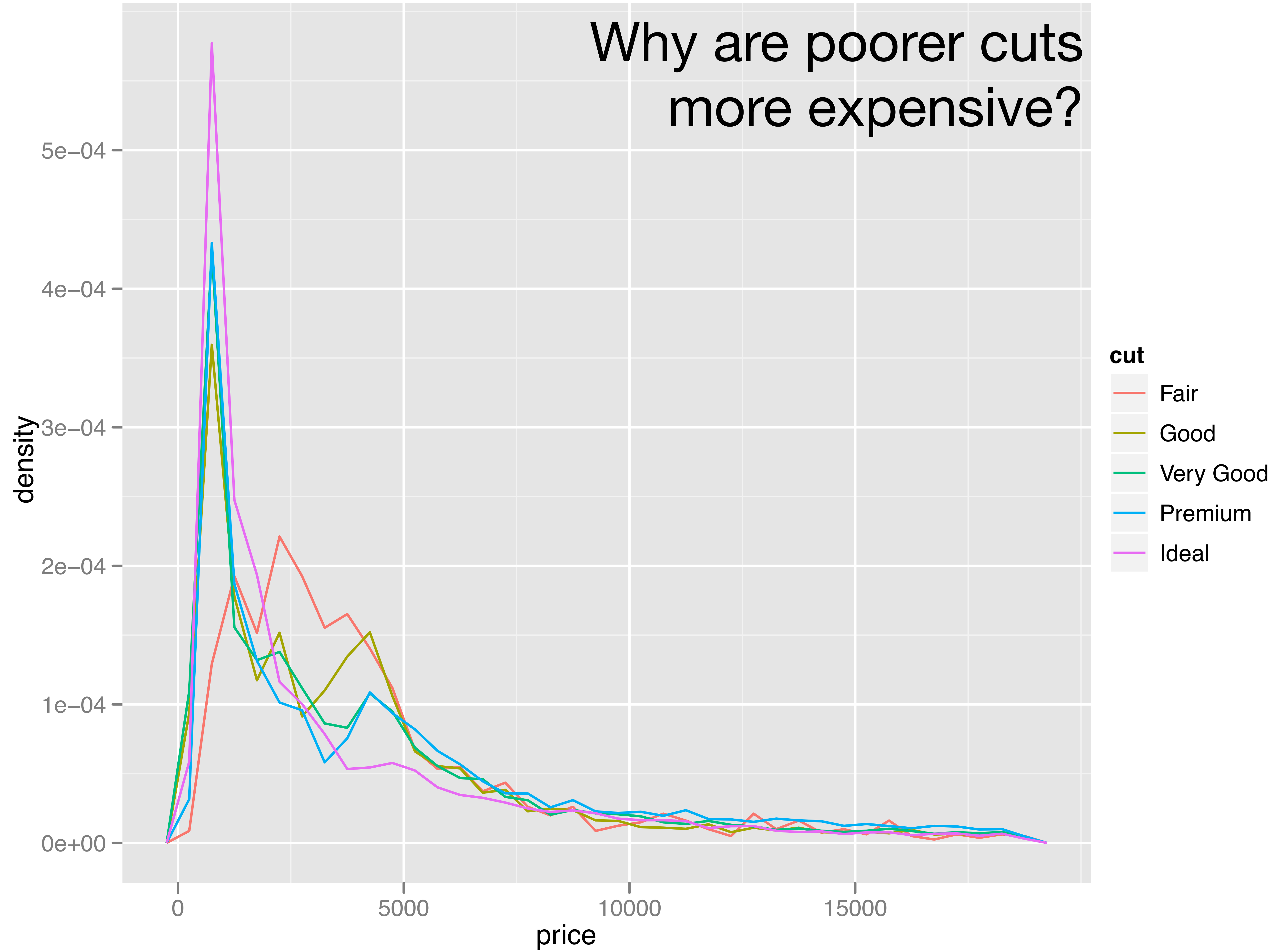
2 I have two dogs

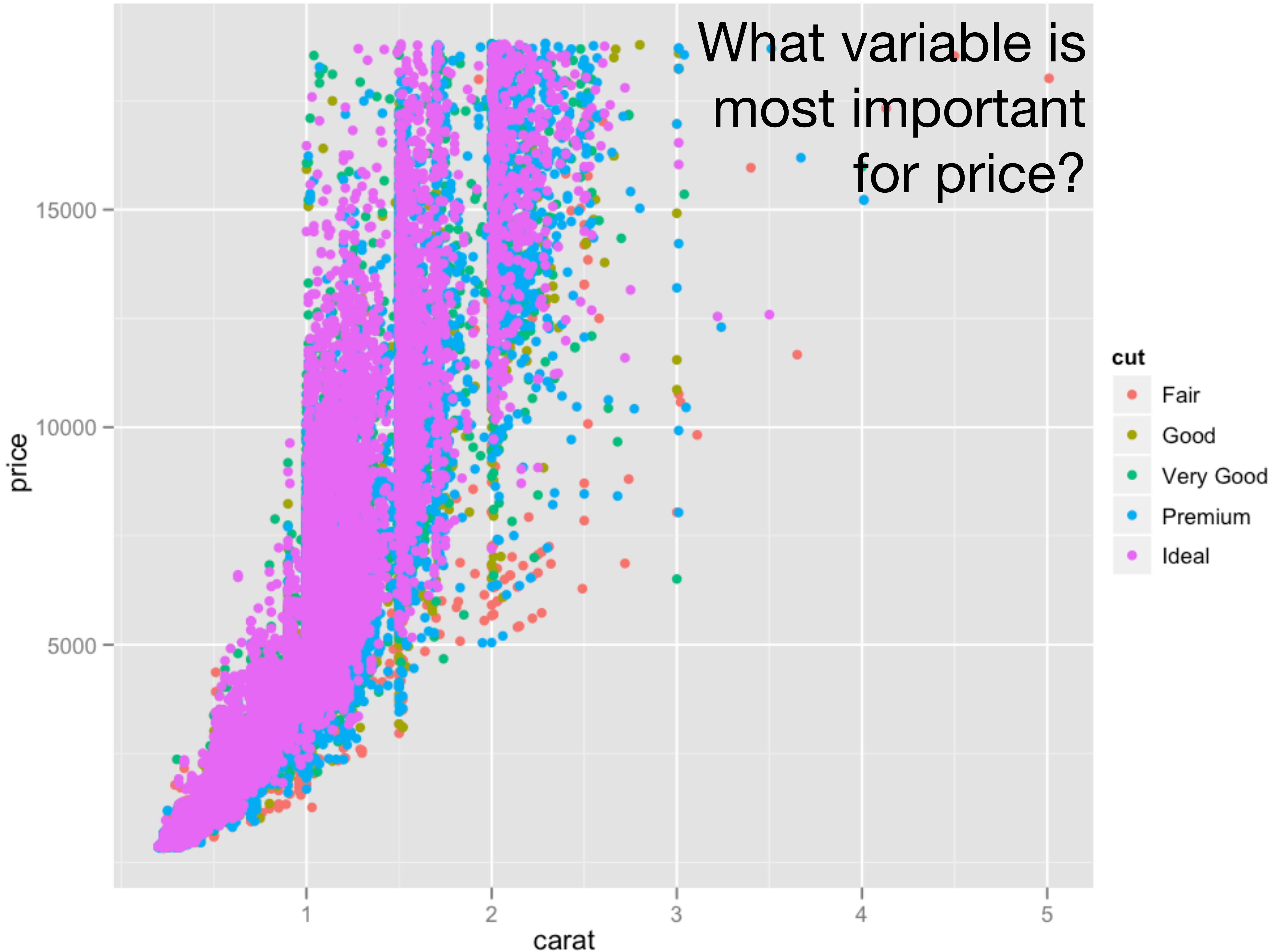

3
I like to bake

4

At Rice, I'm a major advisor for statistics, and a divisional advisor for McMurtry


5
In my free time, I play xbox games

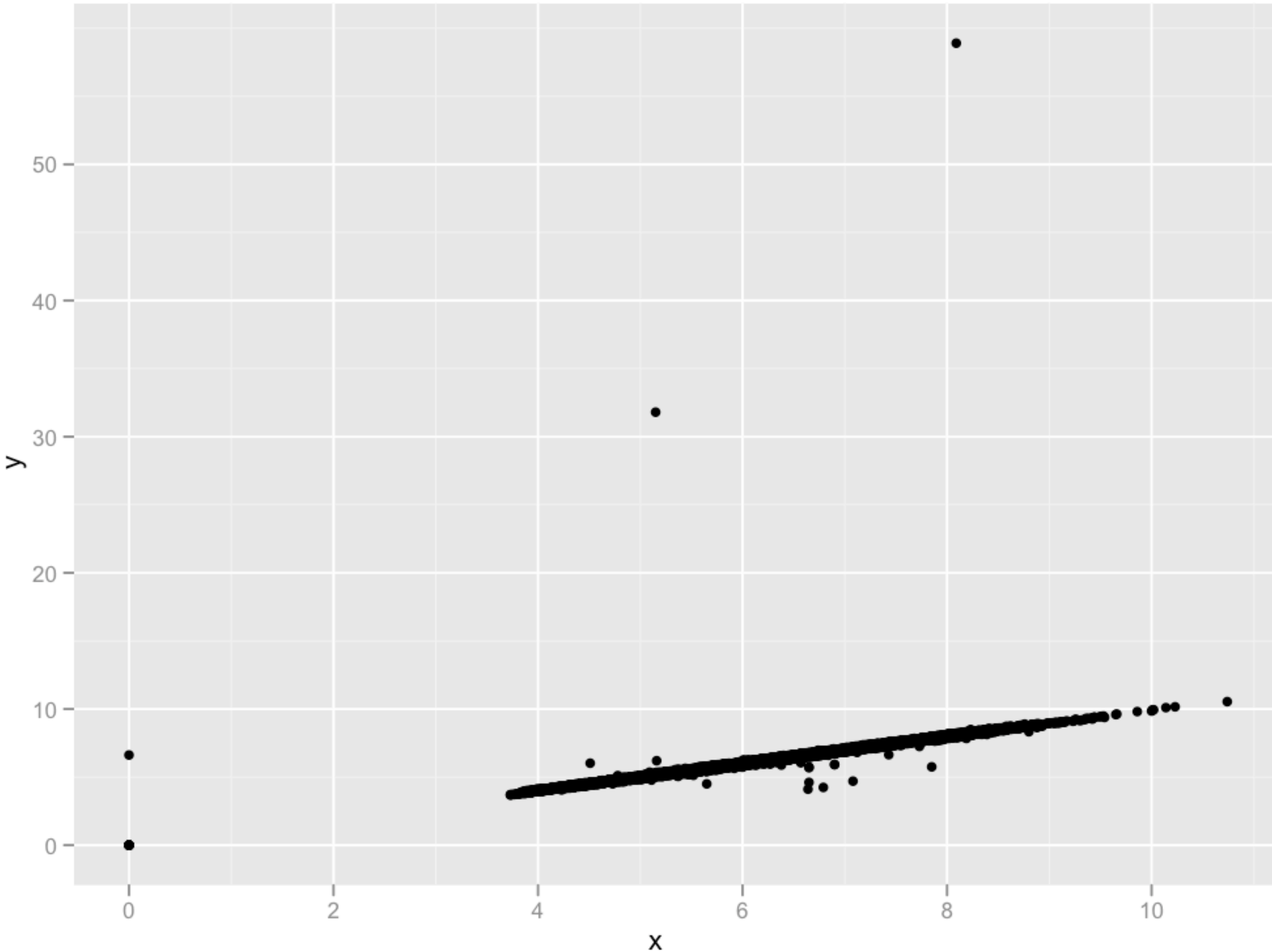

Why are poorer cuts
more expensive?
5e−04
4e−04
3e−04
2e−04
1e−04
0e+00
density
0
5000
10000
15000
price
cut
Fair
Good
Very Good
Premium
Ideal


What variable is most important for price?
15000
10000
5000
price
1
2
3
4
5
carat
cut
Fair
Good
Very Good
Premium
Ideal

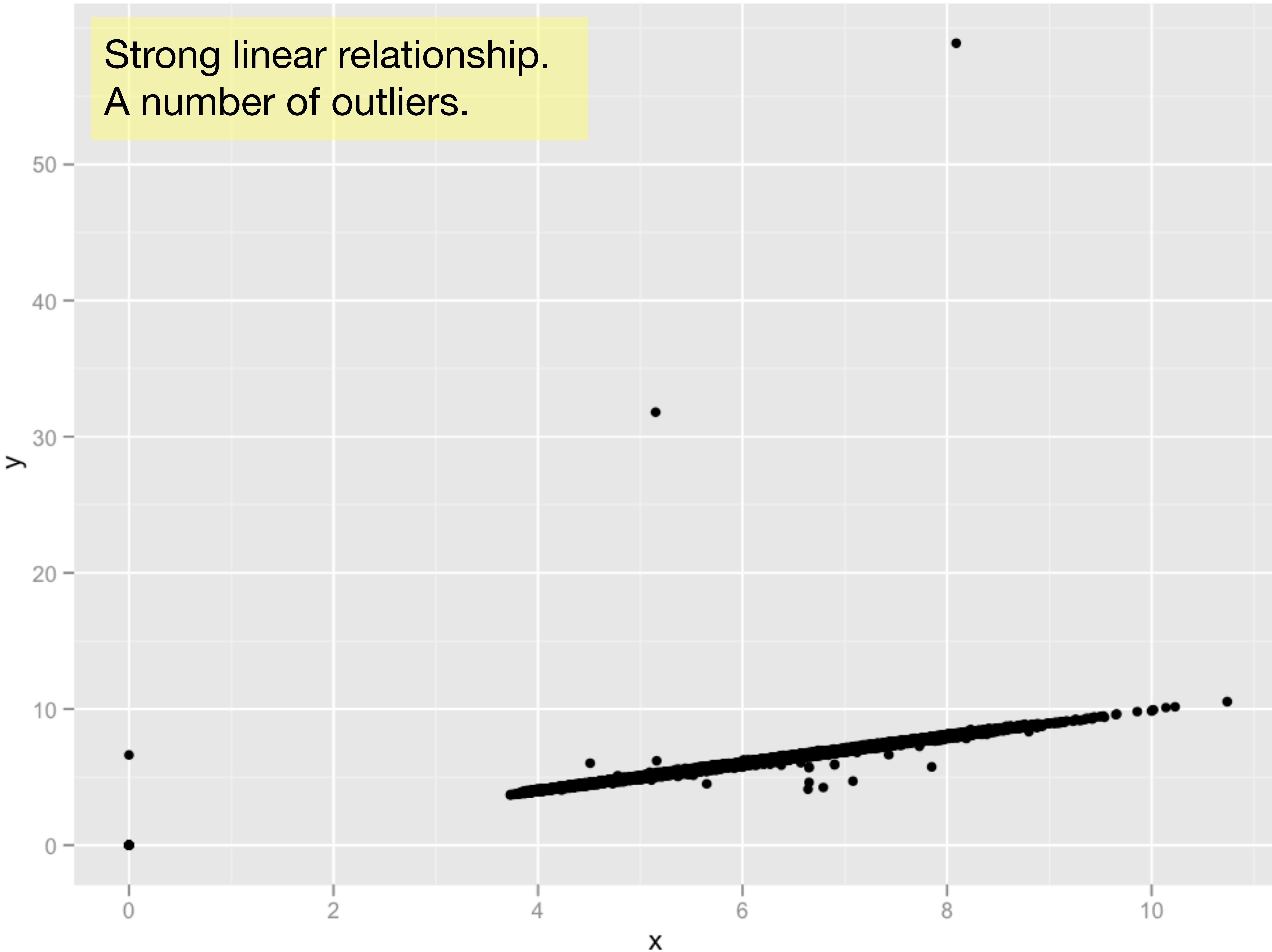
Reading scatterplots

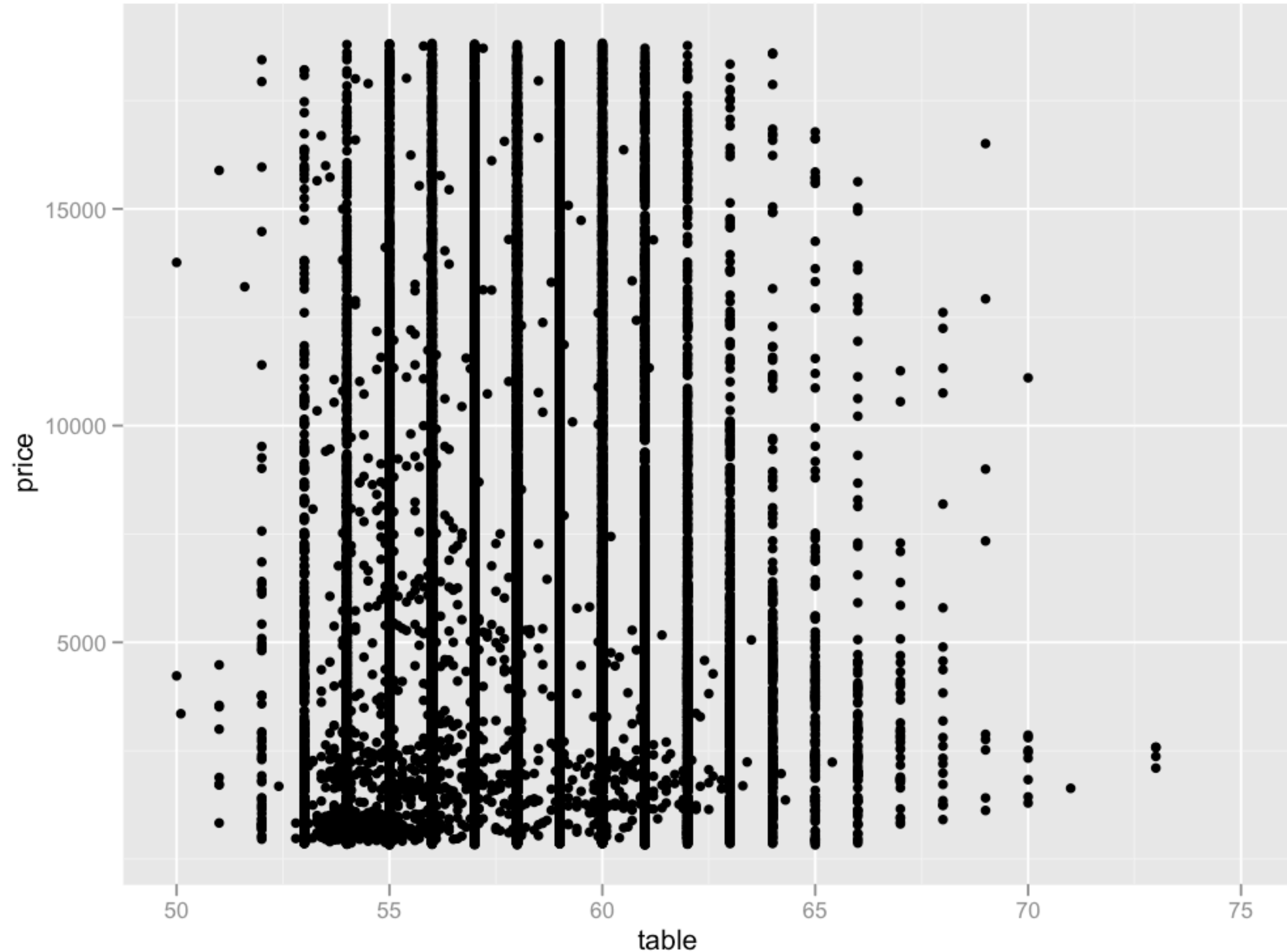
Look for structure

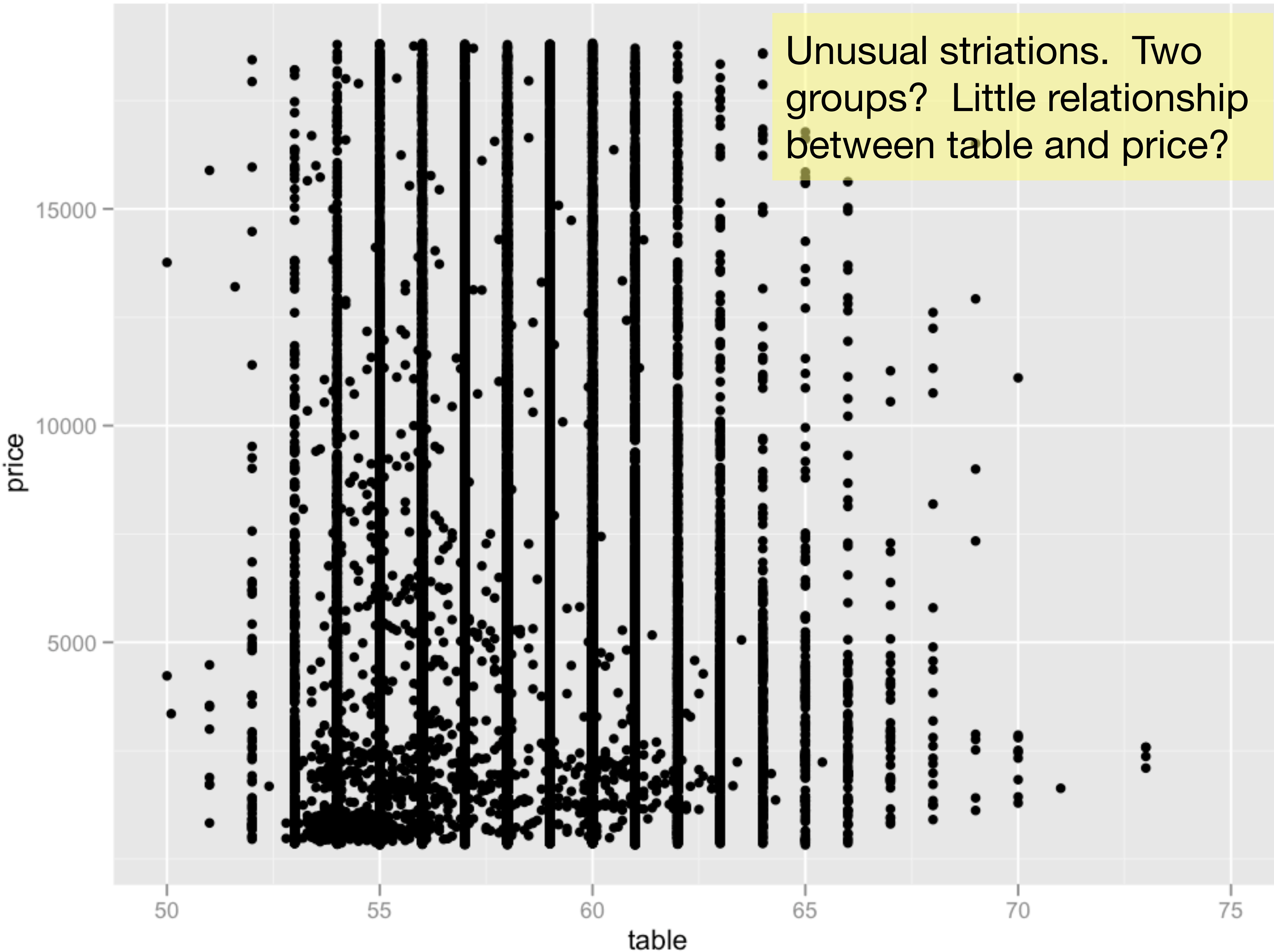
- Global patterns
- Local patterns
- Deviations



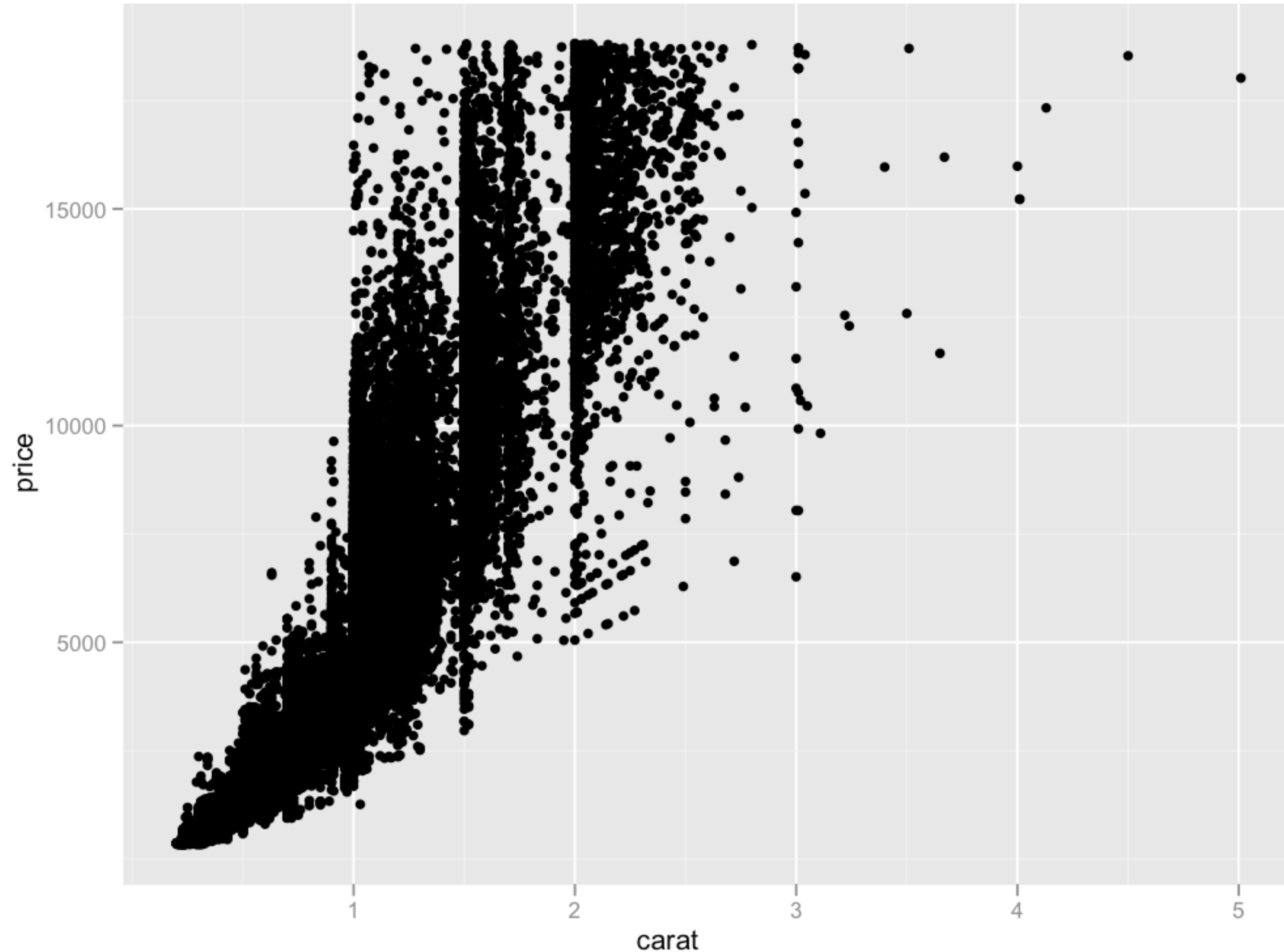
50
40
30
20
10
0
y
0
2
4
6
8
10
x


Strong linear relationship.
A number of outliers.
y
x
0
10
20
30
40
50
0
2
4
6
8
10

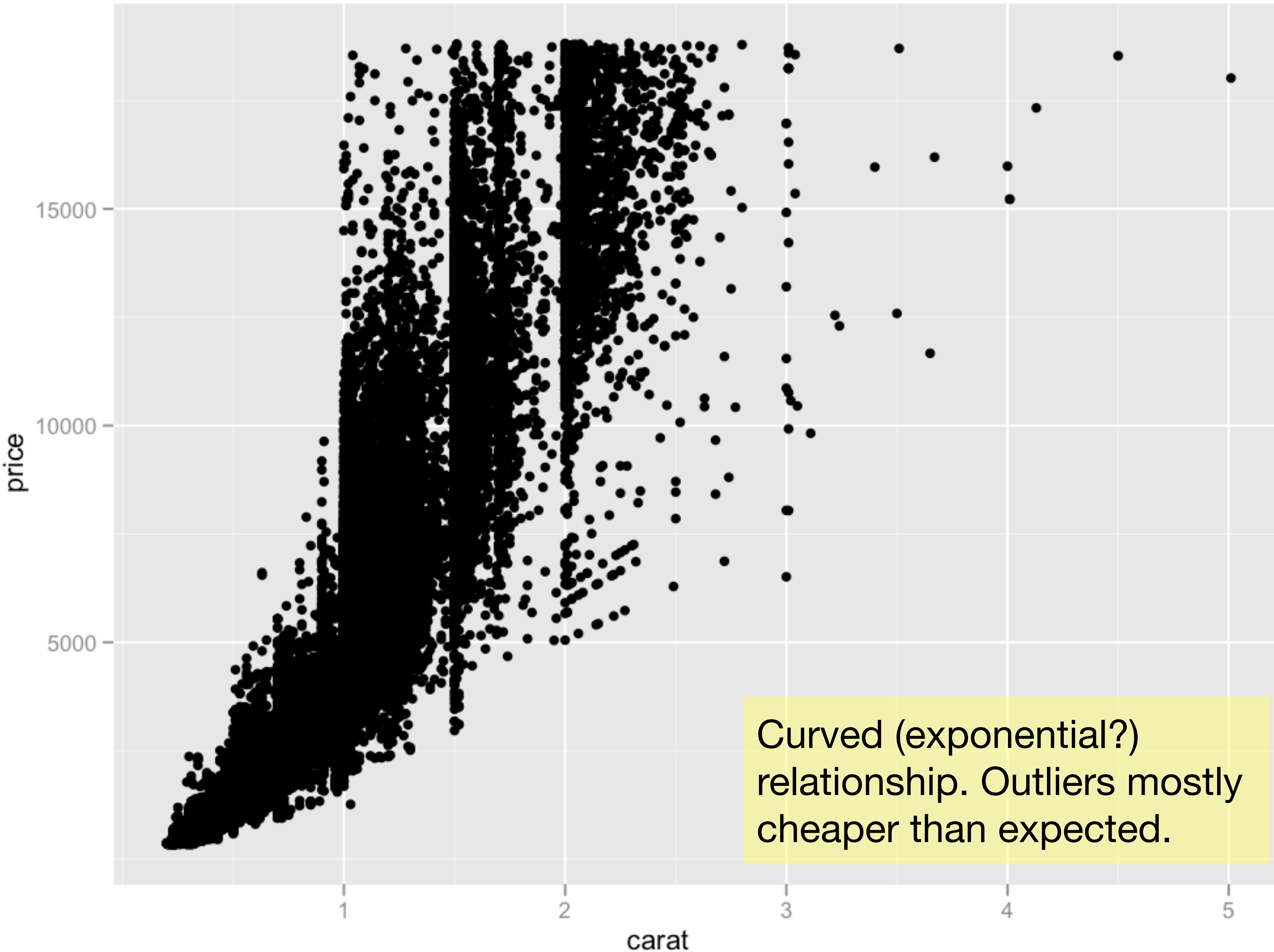

price
15000
10000
5000
50
55
60
65
70
75
table

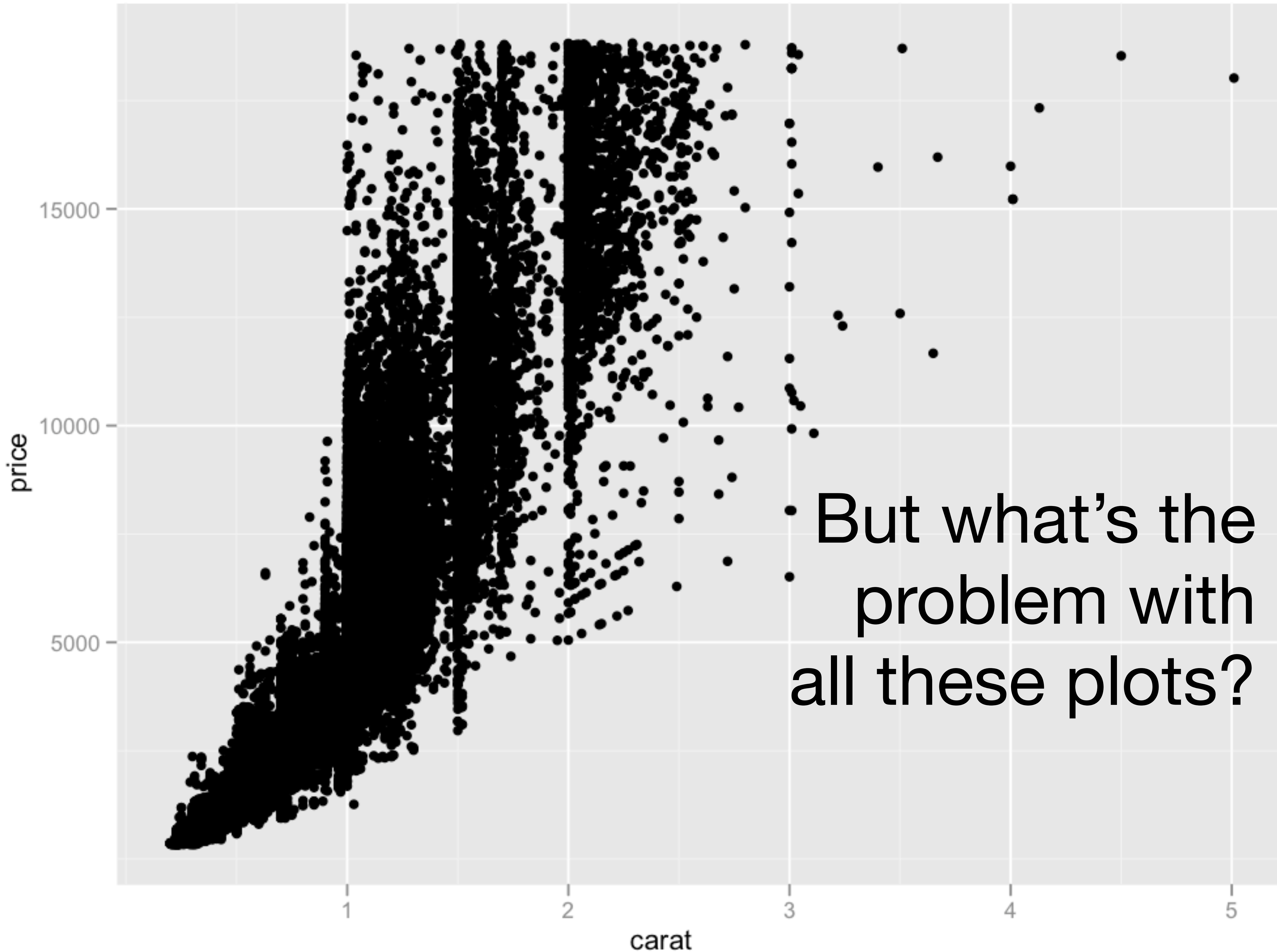


price
15000
10000
5000
50
55
60
65
70
75
table
Unusual striations. Two groups? Little relationship between table and price?

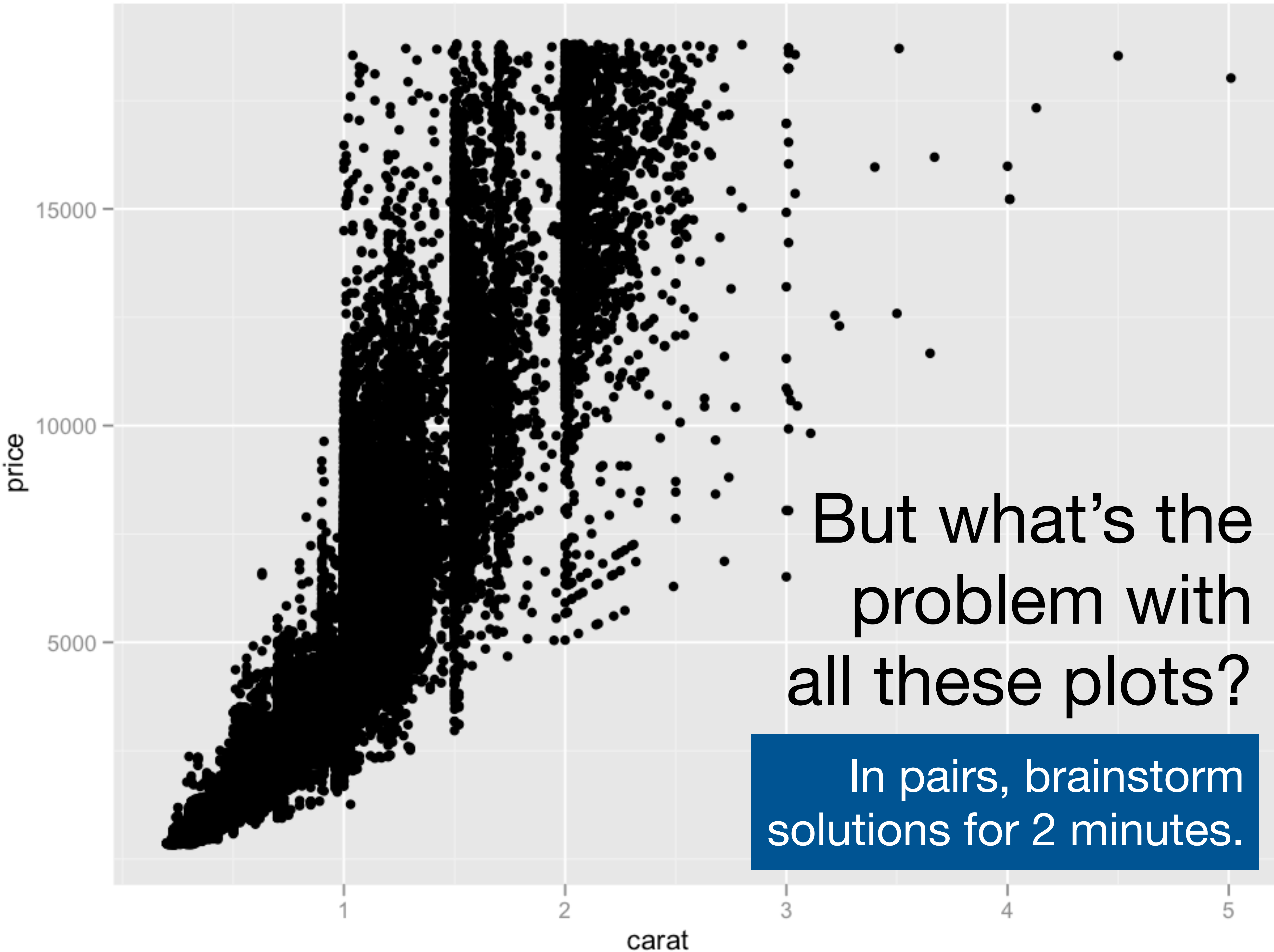


15000
10000
5000
price
1
2
3
4
5
carat


price
15000
10000
5000
carat
1
2
3
4
5
Curved (exponential?) relationship. Outliers mostly cheaper than expected.



But what’s the problem with all these plots?



But what’s the problem with all these plots?

In pairs, brainstorm solutions for 2 minutes.

Scatterplots for big data

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"`
Boxplots	`geom = "boxplot" +` `group = ...`

```
# 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 is how you get help about a specific geom:
# ?geom_smooth
```

```r
# 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(carat, price, data = diamonds, alpha = I(1/10))
qplot(carat, price, data = diamonds, alpha = I(1/50))
qplot(carat, price, data = diamonds, alpha = I(1/100))
qplot(carat, price, data = diamonds, alpha = I(1/250))
```

```
qplot(table, price, data = diamonds)
qplot(table, price, data = diamonds,
  geom = "boxplot")

# Need to specify grouping variable: what determines
# which observations go into each boxplot
qplot(table, price, data = diamonds,
  geom = "boxplot", group = round_any(table, 1))

qplot(table, price, data = diamonds,
  geom = "boxplot", group = round_any(table, 1)) +
  xlim(50, 70)
```

Your turn

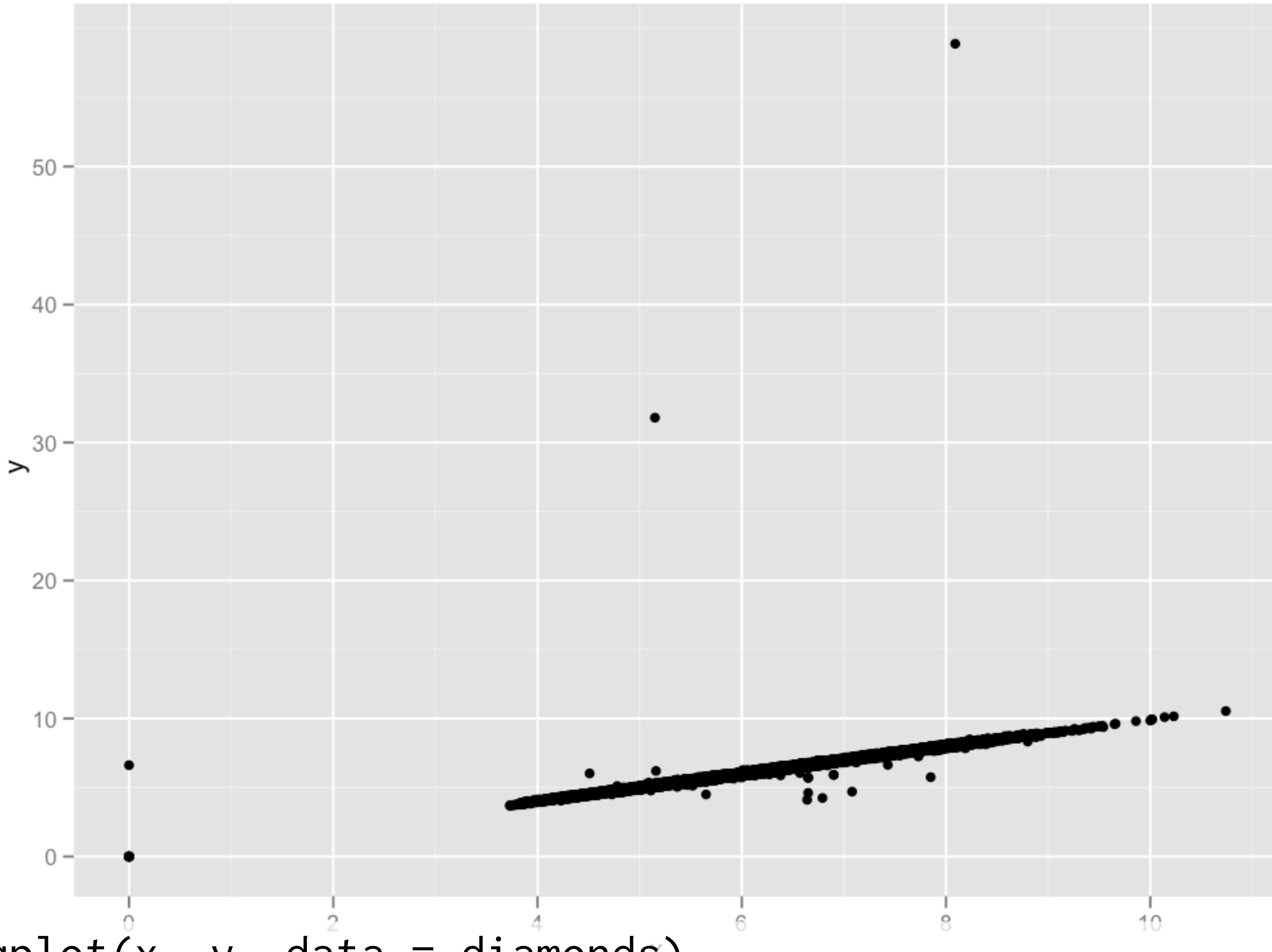
Explore the relationship between carat, price and cut using these techniques.

(i.e. make this plot more informative:

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

Which did you find most useful?

Subsetting



qplot(x, y, data = diamonds)

Motivation

To explore the relationship between x and y we need to remove the obvious incorrect values.

To modify, must first know how to extract, or **subset**.

Many different methods available in R. We'll start with most explicit then learn some shortcuts next time.

Subsetting

Run the following two lines of code to create two vectors:

```
x <- sample(1:10)
y <- setNames(x, letters[1:10])
```

Next, run the code on the following slide and figure out the four types of things you can subset with.

```
x[1:4]
x[x == 5]
y[order(y)]
x[]
x[-1]
y["a"]
x[x]
x[x > 2 & x < 9]
x[sample(10)]
x[order(x)]
x[-(1:5)]
x["a"]
y[letters[10:1]]
x[x < 2 | x >= 8]
x[-1:5]
x[0]
```

blank	include all
integer	**+ve**: include **0**: include none **-ve**: exclude
logical	keep TRUEs
character	lookup by name

```
# Everything
str(diamonds[, ])

# Positive integers & nothing
diamonds[1:6, ]  # same as head(diamonds)
diamonds[, 1:4]  # watch out!

# Two positive integers in rows & columns
diamonds[1:10, 1:4]

# Repeating input repeats output
diamonds[c(1,1,1,2,2), 1:4]

# Negative integers drop values
diamonds[-(1:53900), -1]
```

```
# Use logical comparisons to describe which values
you want. Comparison functions:
# < > <= >= != == %in%

x_big <- diamonds$x > 10

head(x_big)
sum(x_big)
mean(x_big)
table(x_big)

diamonds$x[x_big]
diamonds[x_big, ]
```

```
small <- diamonds[diamonds$carat < 1, ]
lowqual <- diamonds[diamonds$clarity
  %in% c("I1", "SI2", "SI1"), ]
```

```
# Boolean operators: & | !
small <- diamonds$carat < 1 &
  diamonds$price > 500
lowqual <- diamonds$colour == "D" |
  diamonds$cut == "Fair"
```

	a
	b
	a | b
	a & b
	a & !b
	xor(a, b)

```
# Common mistakes

diamonds[diamonds$color == ”D” | ”E” | ”F”, ]
diamonds[diamonds$carat < 1 & > 2]

diamonds[diamonds$cut = ”Good”, ]
```

Your turn

Select the diamonds that have:

Equal x and y dimensions.

Depth between 55 and 70.

Carat smaller than the mean.

Cost more than $10,000 per carat.

Are of good quality or better.