

ggplot2

An implementation of the
grammar of graphics

Hadley Wickham

Outline

- Intro: installation, documentation, data and qplot
- How make to a plot
- Geoms, stats, scales, facets and coordinate systems
- Let me know if you have questions

ggplot2

- `install.packages("ggplot2")`
 - 0.5.7 should be available v. soon
- <http://had.co.nz/ggplot2>
 - documents 99 ggplot objects with over 500 examples
 - opportunities for feedback
 - link to (draft) ggplot book

Diamonds data

- ~54,000 round diamonds from <http://www.diamondse.info/>
- Carat, colour, clarity, cut
- Total depth, table, depth, width, height
- Price



qplot

- Wraps up all the details of ggplot with a familiar syntax borrowed from plot
- Additional features:
 - Automatically scales data
 - Can produce any type of plot
 - Facetting and margins
 - Creates objects that can be saved and modified

qplot

```
qplot(diamonds$carat, diamonds$price)
```

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

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

```
qplot(carat, price, data = diamonds,  
      geom=c("point", "smooth"), method=lm)
```

```
qplot(carat, data = diamonds,  
      geom="histogram")
```

```
qplot(carat, data = diamonds,  
      geom="histogram", binwidth = 100)
```

Defaults

- Layers of convenience functions
 - Multiple levels allow you to trade-off simplicity and control
 - qplot is the simplest to use, but gives the least control
- To understand more sophisticated levels you need a basic understanding of the grammar

How to make a plot

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

How to make a plot

- Want a scatterplot of length vs width

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

How to make a plot

- Want a scatterplot of length vs width
- What is a scatterplot?

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

How to make a plot

- Want a scatterplot of length vs width
- What is a scatterplot?
- Represent observations with points (geom)

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

How to make a plot

- Want a scatterplot of length vs width
- What is a scatterplot?

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

- Represent observations with points (geom)
- Linear scaling of x and y axes (scales)

How to make a plot

- Want a scatterplot of length vs width
- What is a scatterplot?

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

- Represent observations with points (geom)
- Linear scaling of x and y axes (scales)
- Cartesian coordinate system

Data

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

Mapping

length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b



x	y	colour
2	3	a
1	2	a
4	5	b
9	10	b

Scales

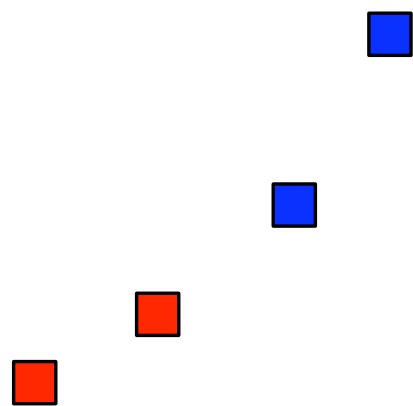
Need to convert to physical “drawing” units

x	y	colour
2	3	a
1	2	a
4	5	b
9	10	b

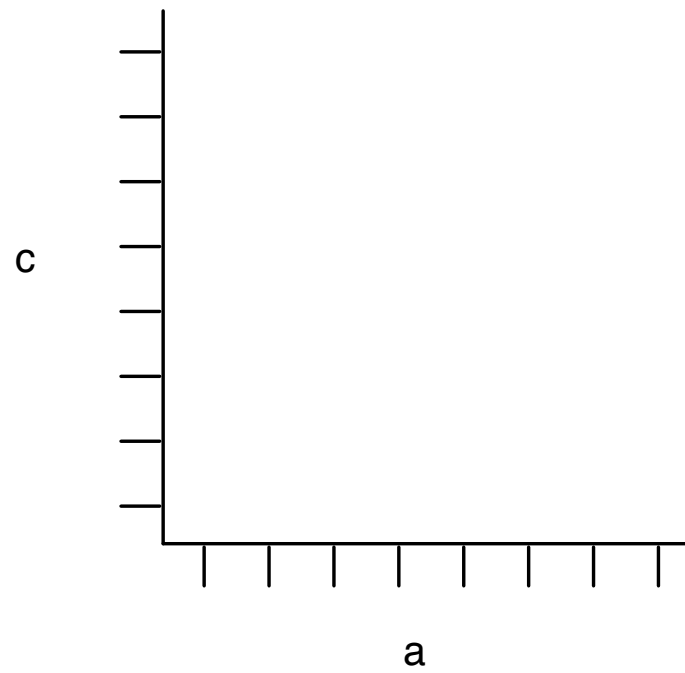


x	y	colour
25	11	red
0	0	red
75	53	blue
200	300	blue

(and coordinate system)



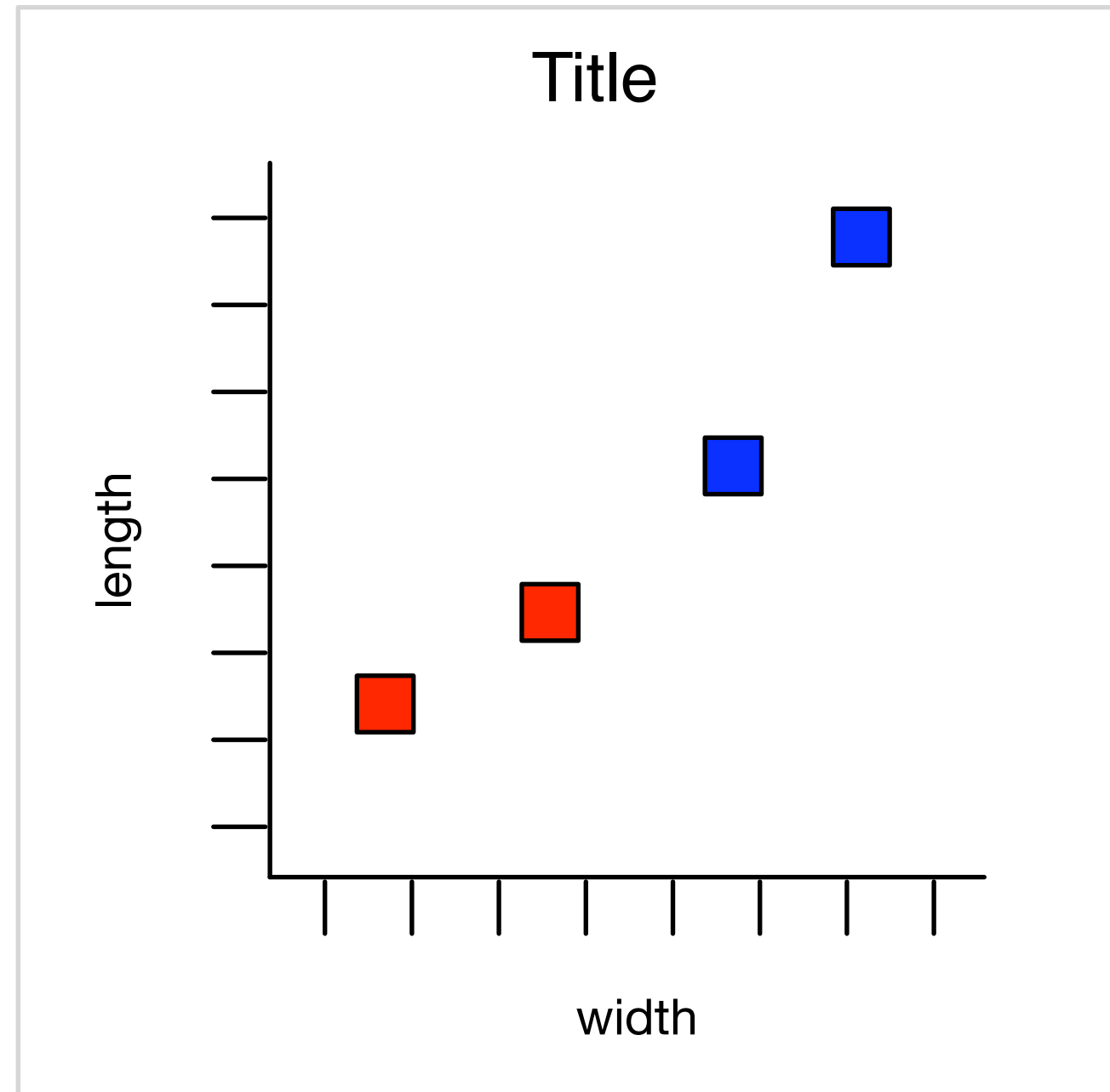
Geoms

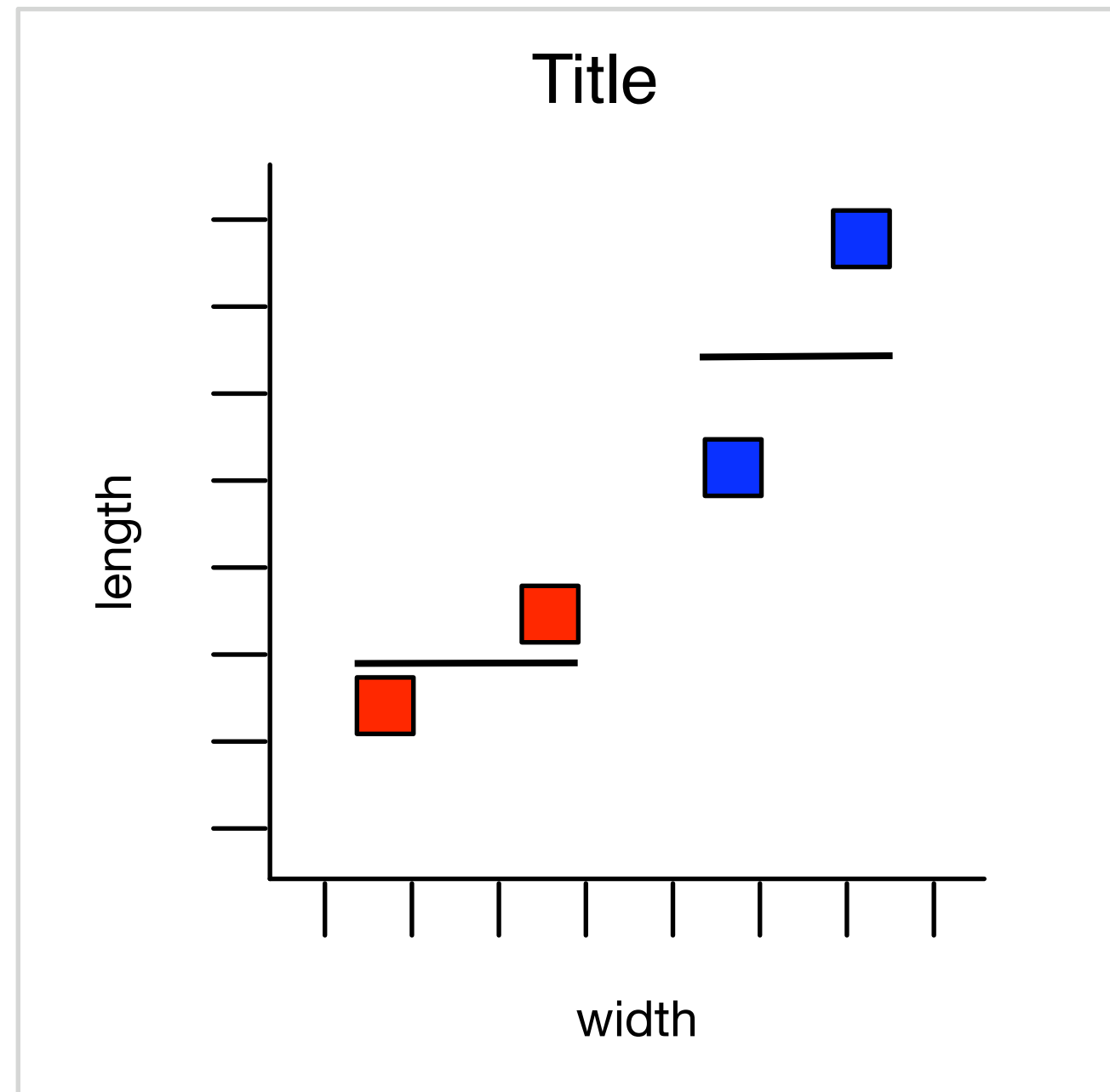


Guides
(from scales and
coordinate systems)



Plot





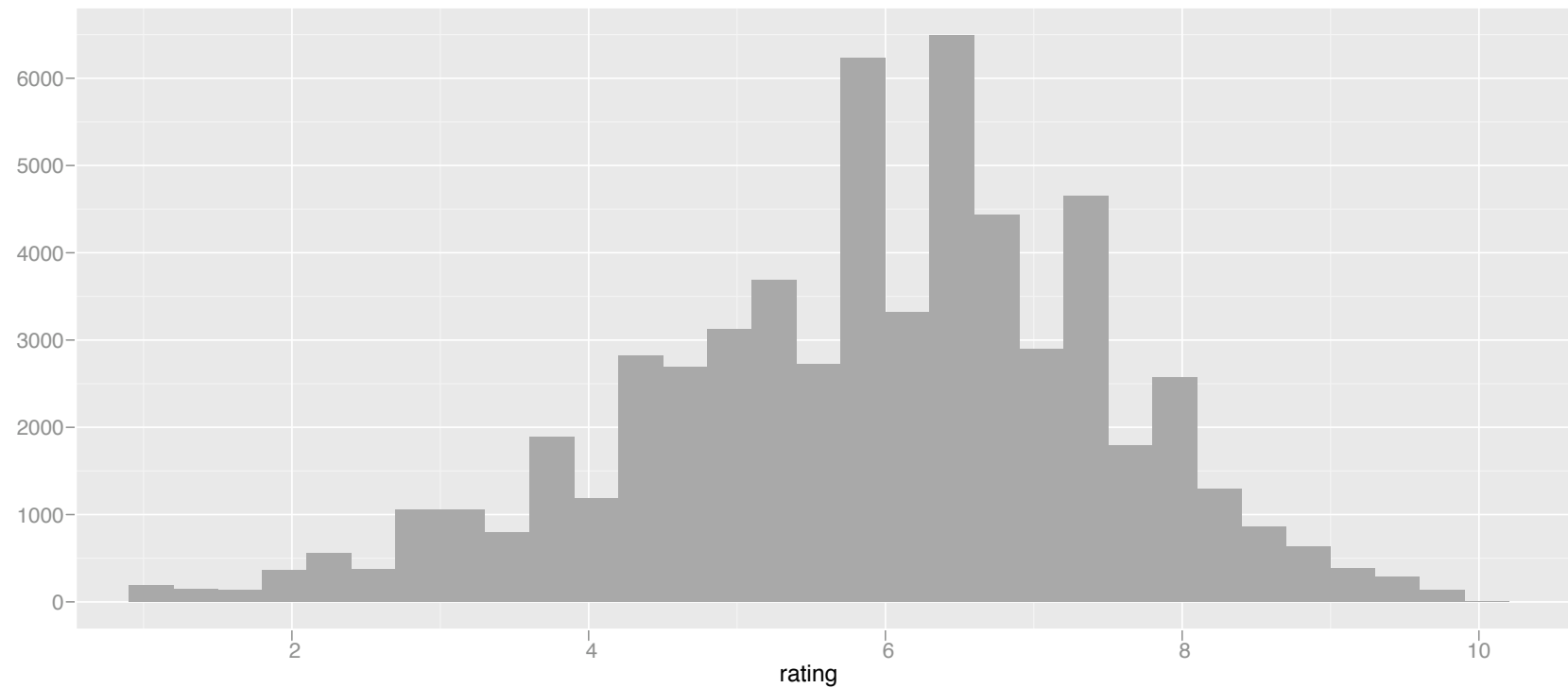
Multiple layers, statistical transformation

Components

- Data
- Geometric object (geom)
- Statistical transformation (stat)
- Scales
- Coordinate system
- (+ Position adjustment, facetting)

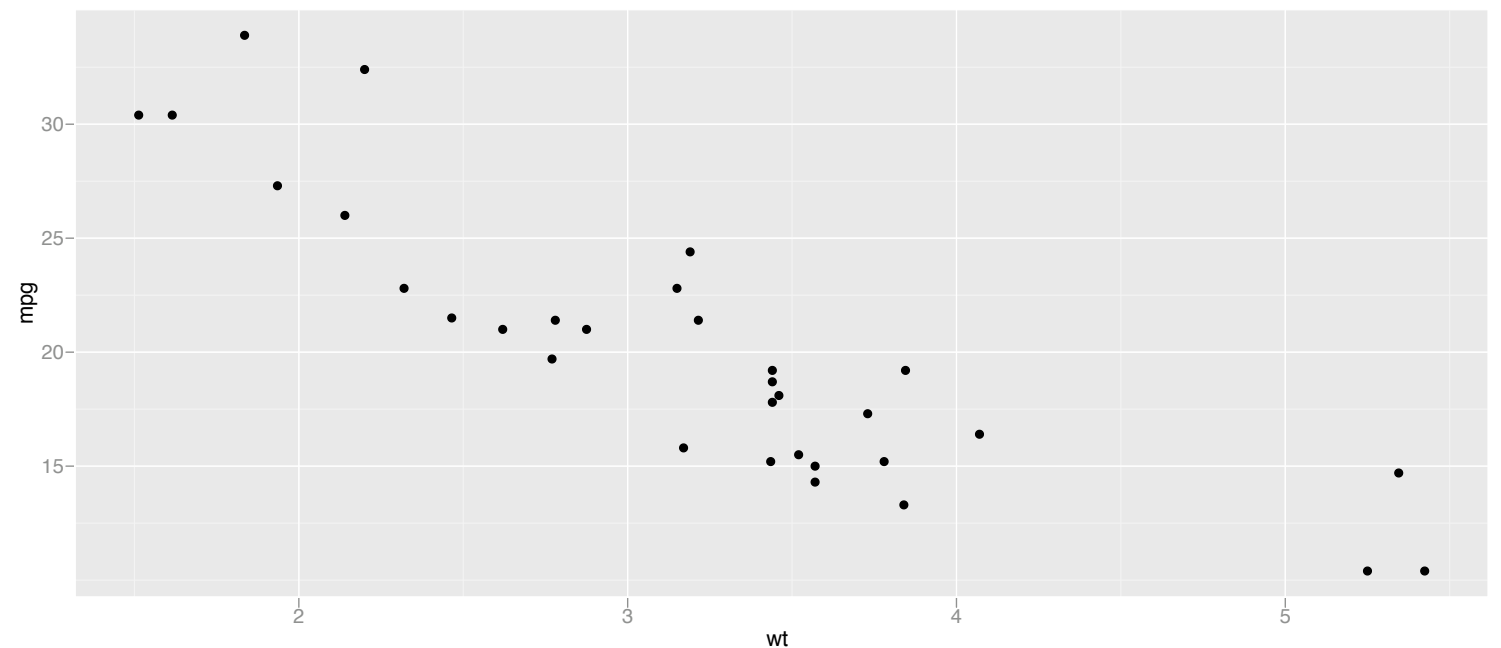
Histogram

- Geom: bar
- Stat: bin
- Scale: linear
- Coordinate system: Cartesian



Scatterplot

- Geom: point
- Stat: identity
- Scale: linear
- Coordinate system: Cartesian



Layers

- Previous description is a bit of a simplification
- Actually have: defaults + layers + scales + coordinate system
- Layer = data + mapping + geom + stat + position

Plot definition

```
ggplot(data, mapping) +  
layer(  
  stat = "",  
  geom = "",  
  position = "",  
  geom_params = list(),  
  stat_params = list(),  
)
```


Layers

- Usually won't write out the full specification, but use a shortcut:
 - `geom_smooth()`
 - `stat_summary()`
 - ...
- Every geom has a default statistic, every statistic a default geom (but can override)

Examples

```
d <- ggplot(diamonds,  
  aes(x=carat, y=price))
```

```
d + geom_point()
```

```
d + geom_point(aes(colour = carat))
```

```
d + geom_point(aes(colour = carat))  
  + scale_colour_brewer()
```

```
ggplot(diamonds) +  
geom_histogram(aes(x=price))
```

Data + mapping

- Data and mappings usually stay the same on a plot, so they are stored as defaults:
- `ggplot(data, mapping = aes(x=x, y=y))`
- `aes` function describes relationship, doesn't supply data

Geoms

- Geoms define the basic "shape" of the elements on the plot
- Basics: point, line, polygon, bar, text
- Composite: boxplot, pointrange
- Statistic: histogram, smooth, density
- Documentation

Statistics

- We haven't used explicitly, but they underlie many of the layers we have been creating - some geoms are really statistics in disguise:
 - `geom_histogram` = `stat_bin` + `geom_bar`
 - `geom_smooth` = `stat_smooth` + `geom_ribbon`
 - `geom_density` = `stat_density` + `geom_ribbon`
- Separate transformation of data from its graphical representation

Variations on a histogram

```
p <- ggplot(diamonds, aes(x=price))
```

```
p + geom_histogram()
```

```
p + stat_bin(geom="area")
```

```
p + stat_bin(geom="point")
```

```
p + stat_bin(geom="line")
```

```
p + geom_histogram(aes(fill = clarity))
```

```
p + geom_histogram(aes(y = ..density..))
```

New variables

- Some statistics produce new variables in the data (see docs for details)
 - `stat_bin` produces count and density
- If you want to map an aesthetic to one of these new variables, surround it with `..`
 - `ggplot(diamonds, aes(x=price))`
 `+ geom_histogram(aes(y = ..density..))`
 - `+ geom_histogram(aes(colour = ..count..))`

Parameters

- Parameters modify appearance of geoms and operation of statistics
 - + geom_smooth(method=lm)
 - + stat_bin(binwidth = 100)
 - + stat_summary(fun="mean_cl_boot")
 - + geom_boxplot(outlier.colour = "red")
- Any aesthetic can also be used as a parameter
 - + geom_point(colour = "red", size = 5)
 - + geom_line(linetype = 3)

Setting vs mapping

```
p <- ggplot(diamonds, aes(x=carat,y=price))
```

```
# What will this do?
```

```
p + geom_point(aes(colour = "green"))
```

```
p + geom_point(colour = "green")
```

```
p + geom_point(colour = colour)
```

Writing your own

- Fairly easy to write your own
- But not documented yet
- Don't be afraid of looking at the source

Your turn

- Look at the examples for `stat_summary` (<http://had.co.nz/ggplot2/>)
- Try them out
- How could you use them to better understand the diamonds data?

Scales

- Scales control the mapping between data and aesthetics, and control the display of the matching guide (axis or legend)
- ggplot automatically adds default scales as we need them, but we will often need to customise

Basics

- Change name and range or limits
- All scales take name as first argument
 - axis or legend name
 - can be an expression
- All position scales also take limits argument
 - Any data outside of limits is not plotted
(but is still used for computation)

Position scales

- Can be used to plot on non-linear scales
 - `scale_x_log10`, `scale_x_sqrt`, ...
- Can also control exactly where breaks occur (with `breaks` argument) and the amount of extra space on the borders (with the `expand` argument)

Scales

- Colour/fill probably most commonly manipulated
 - discrete: *hue*, *brewer*, *grey*, *manual*
 - continuous: *gradient*, *gradient2*
 - *identity*
- Also see:
 - *scale_size*, *scale_area*
 - *scale_linetype*

Your turn

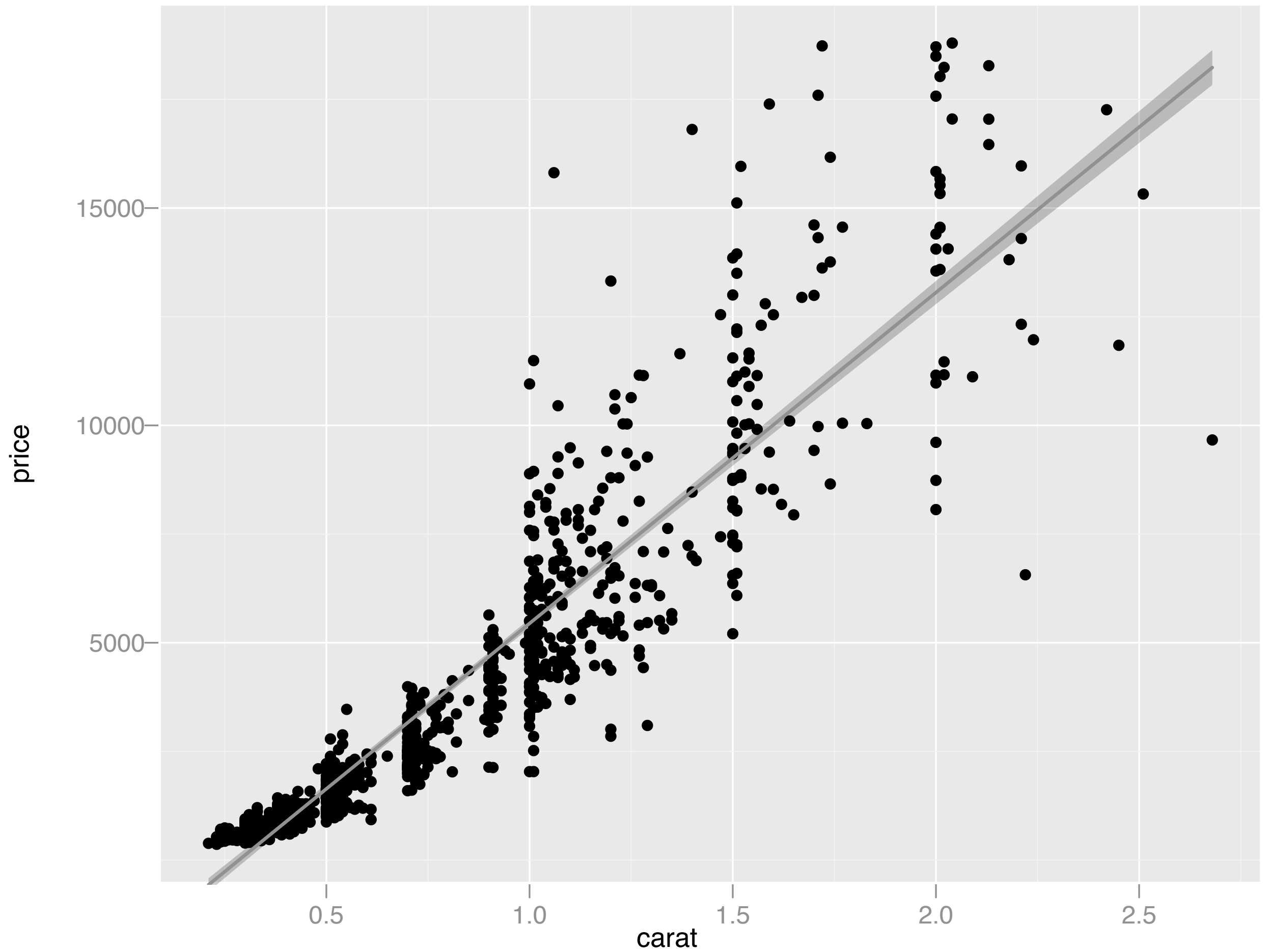
- `qplot(carat, data=diamonds, geom="histogram", fill=clarity)`
- `qplot(carat, cut, data=diamonds, geom="jitter", colour=price)`
- Look up the different colour scales in the documentation and try them out

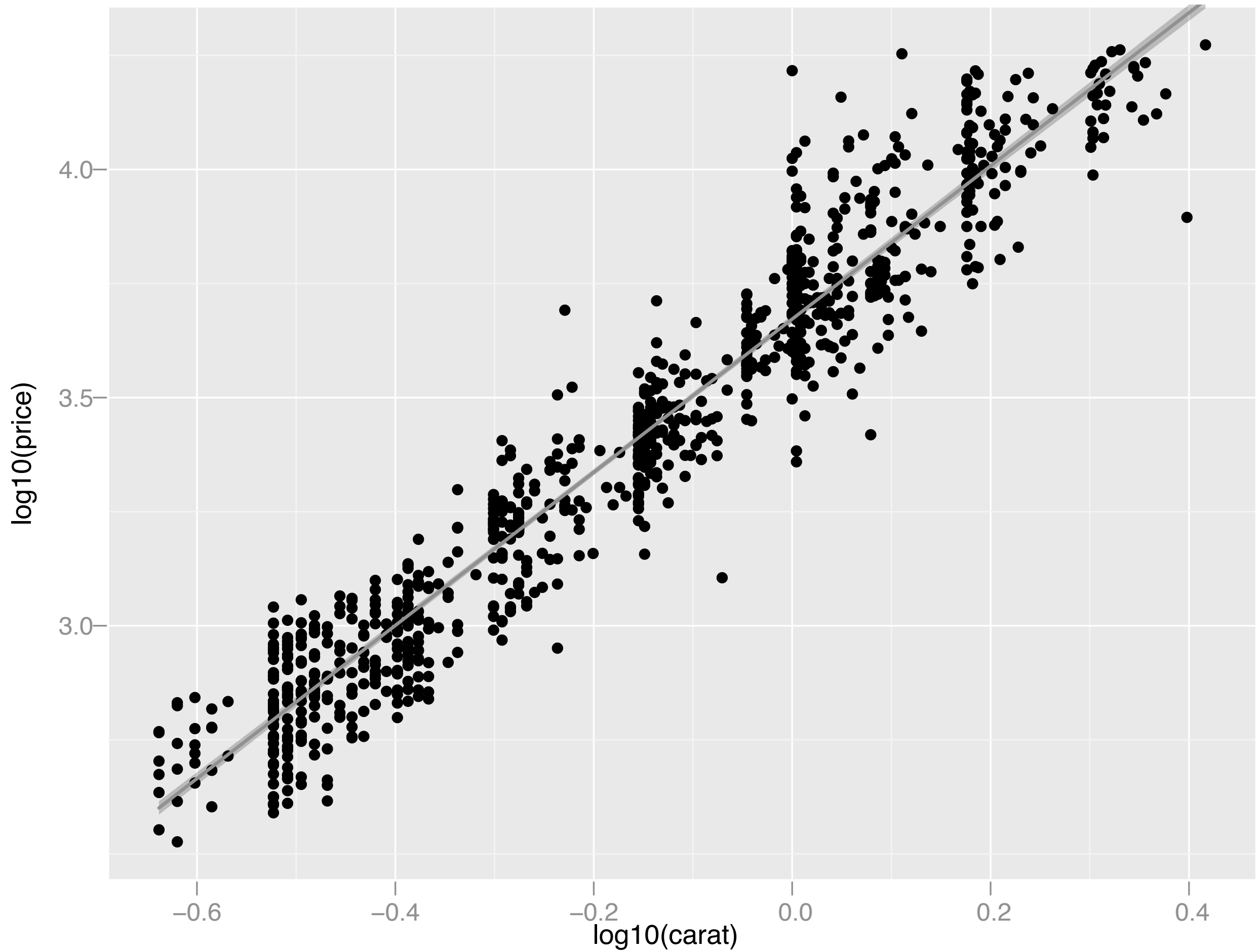
Facetting

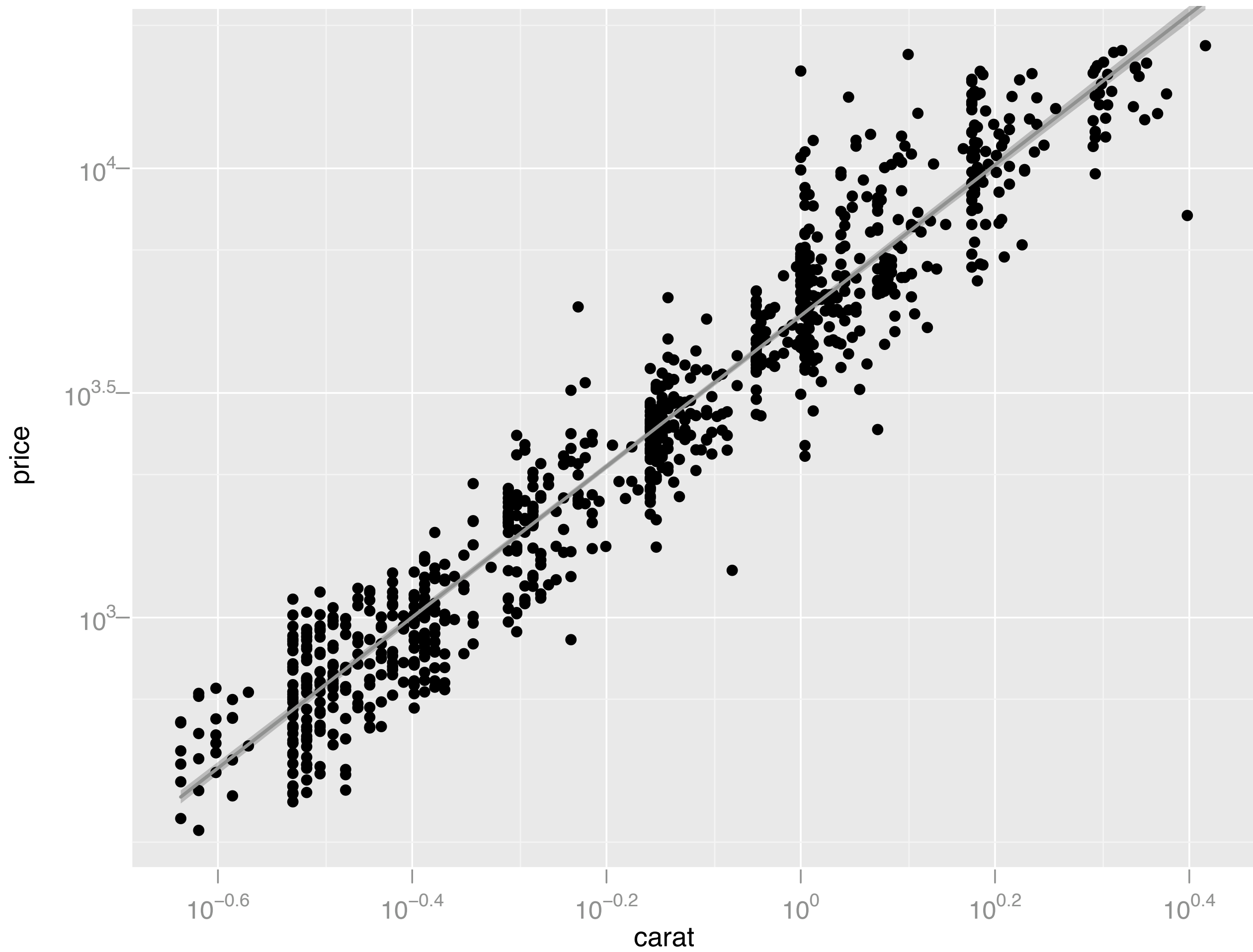
- It's often useful to draw small multiple of subsets of your data
- Currently, there is only one way to do this:
 - + `facet_grid(row ~ col, margins = TRUE)`
 - (just like in `qplot`)
- In the future there will be more

Coordinate systems

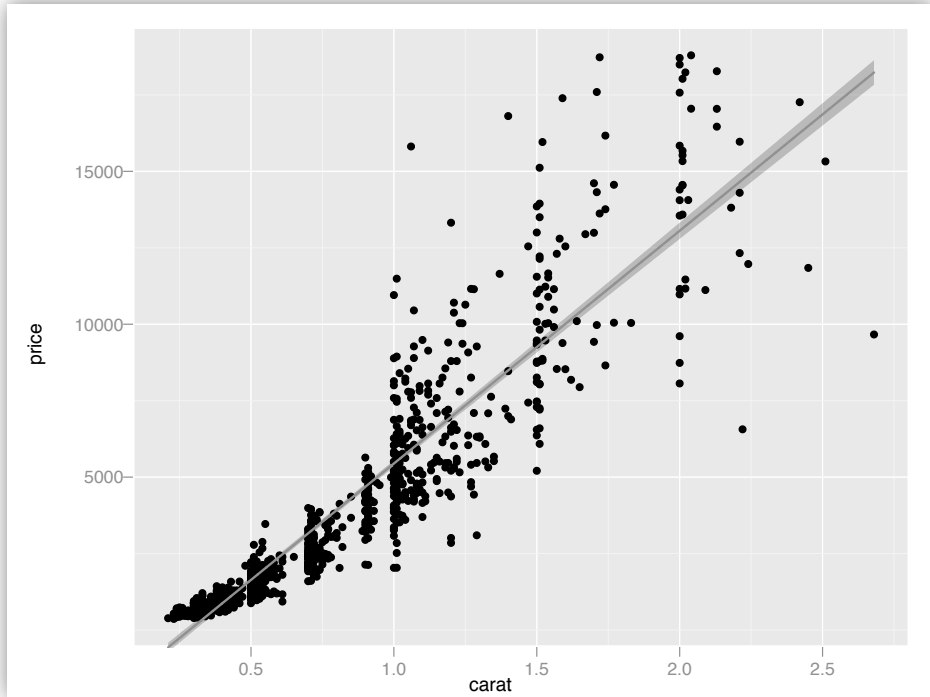
- Control how the two positions aesthetics work together (default: Cartesian)
- Others of note:
 - `coord_flip()`
 - `coord_map()`
 - `coord_polar()`
- Occur after statistics and affect the appearance of geoms



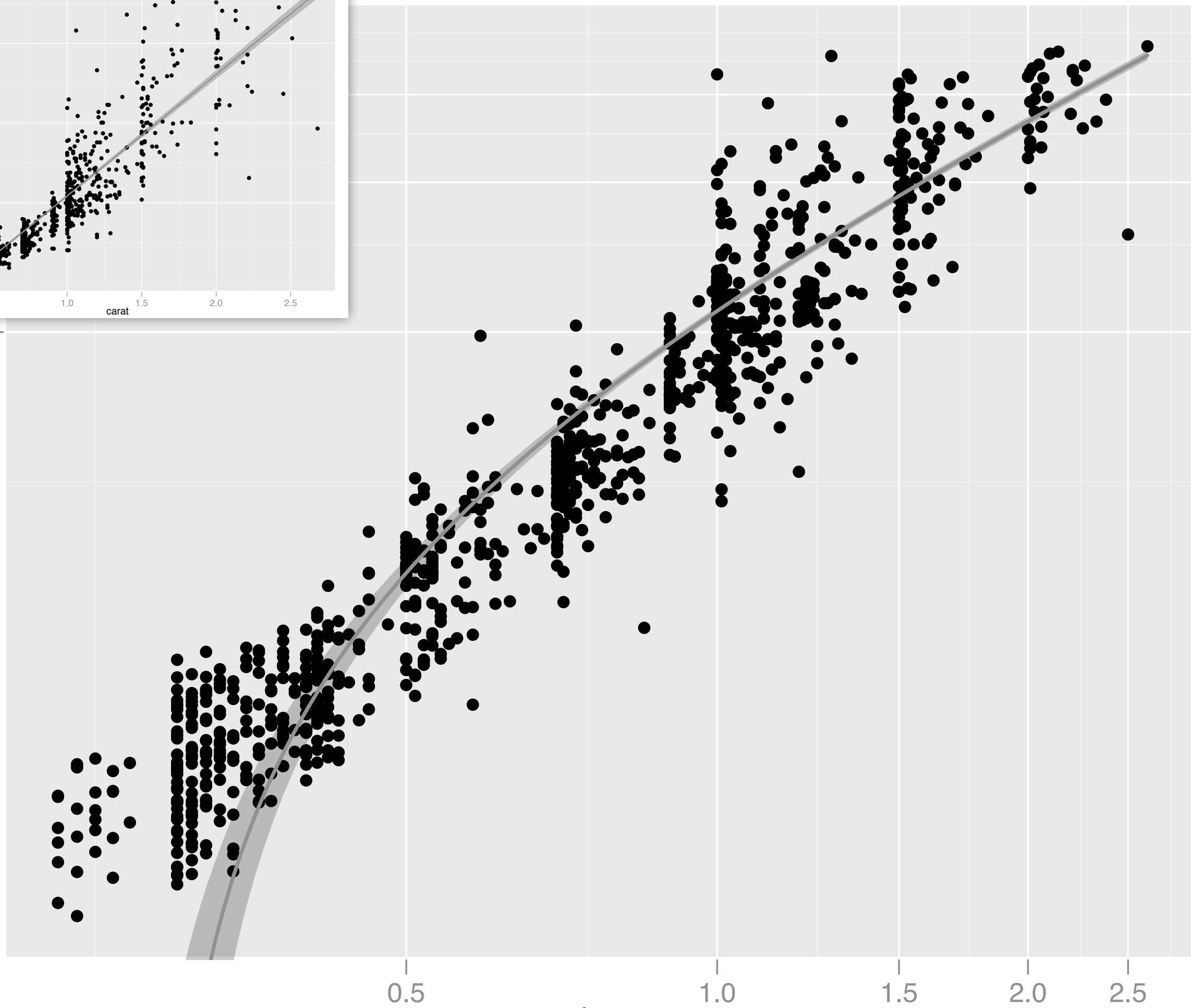


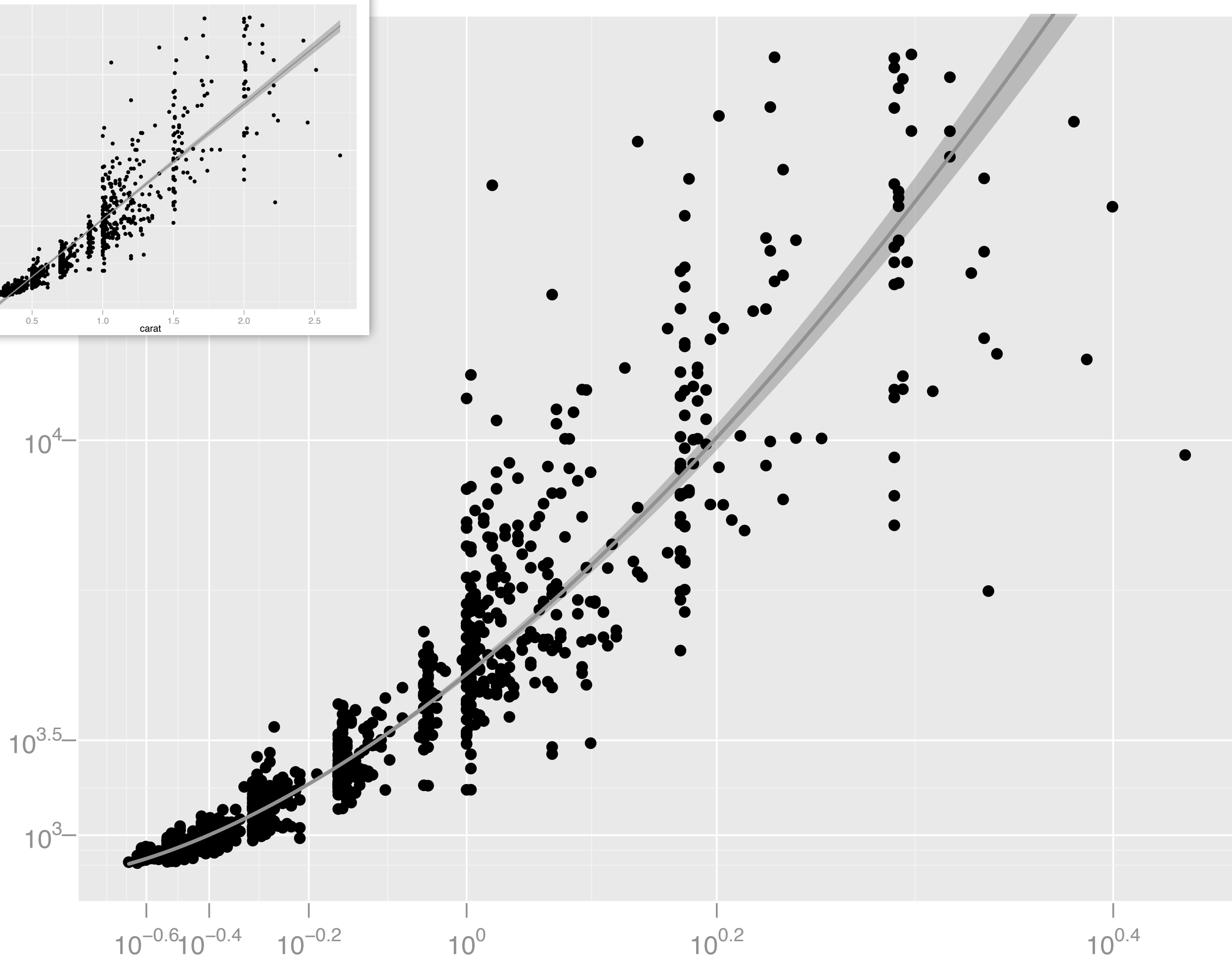
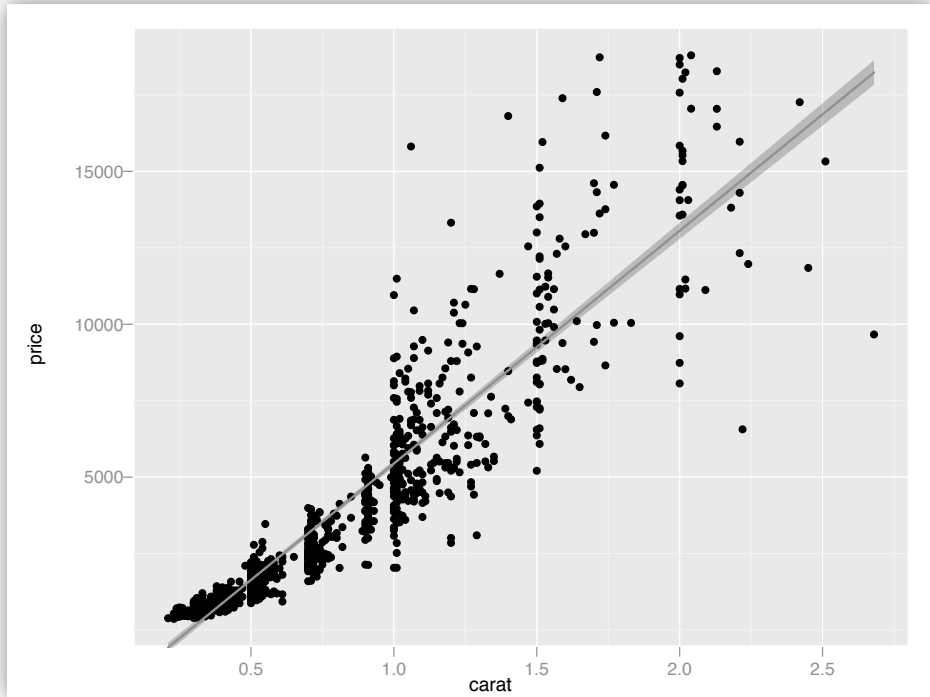


price



5000





Your turn

What's the difference?

```
qplot(log10(length), data=movies,  
geom="histogram", binwidth=0.1)
```

```
qplot(length, data=movies,  
geom="histogram", binwidth=0.1, log="x")
```

```
qplot(length, data=movies,  
geom="histogram", binwidth=10) +  
coord_trans(x="log10")
```