



NYC DATA SCIENCE  
**ACADEMY**

# Data Visualization with 'ggplot2'

---

Data Science Bootcamp

# Outline

---

- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ Constructing a ggplot2 plot
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving Graphs

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

- John Tukey

## Why ggplot2?

---

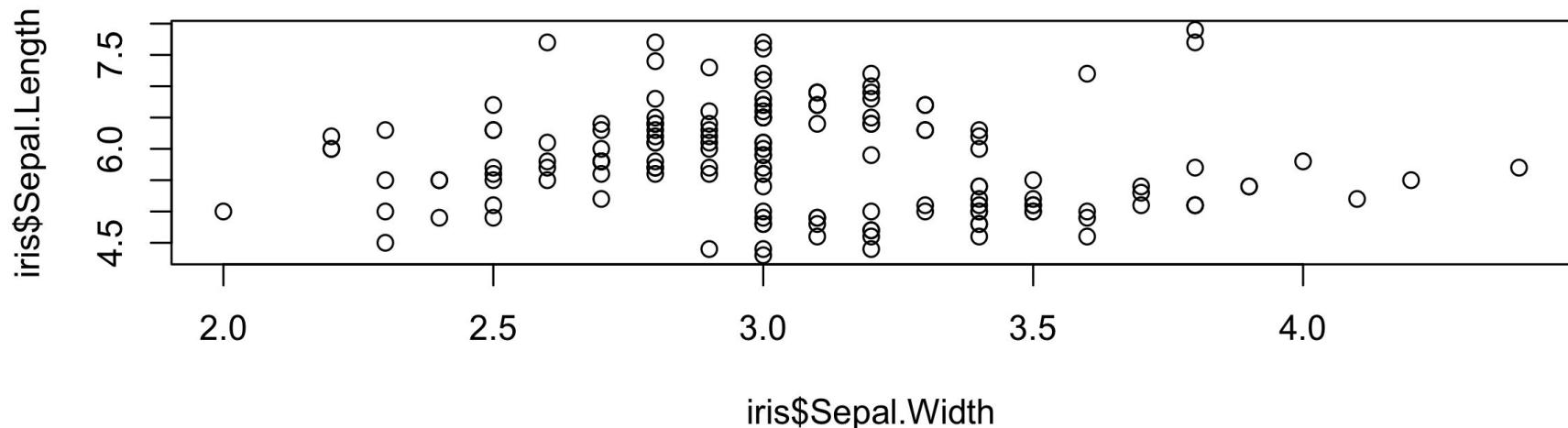
- ❖ In the next few slides, we'll see some examples of plots you can make in ggplot2...
- ❖ ...but first, let's see an example using base R's plot function...

# Why ggplot2?

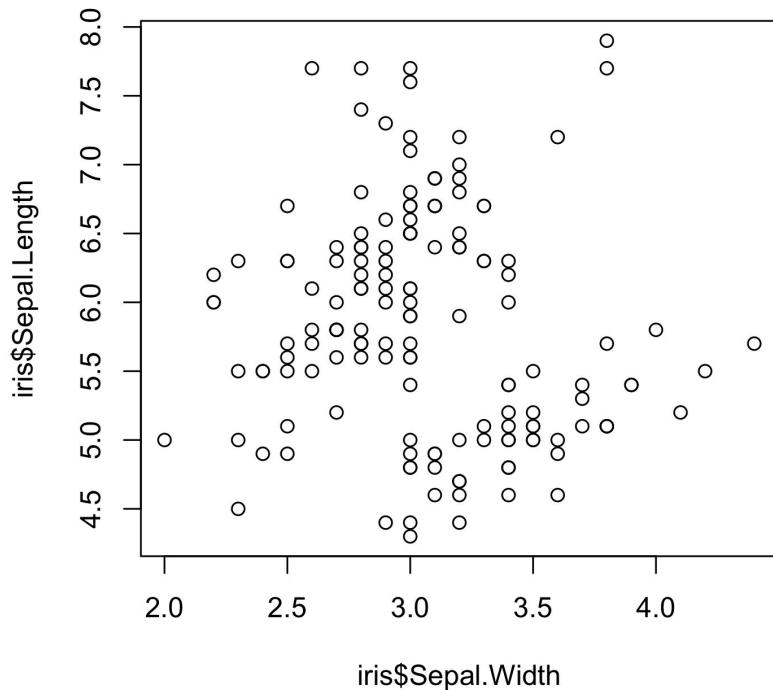
---

- ❖ Using base R's plot function:

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



# Why ggplot2?

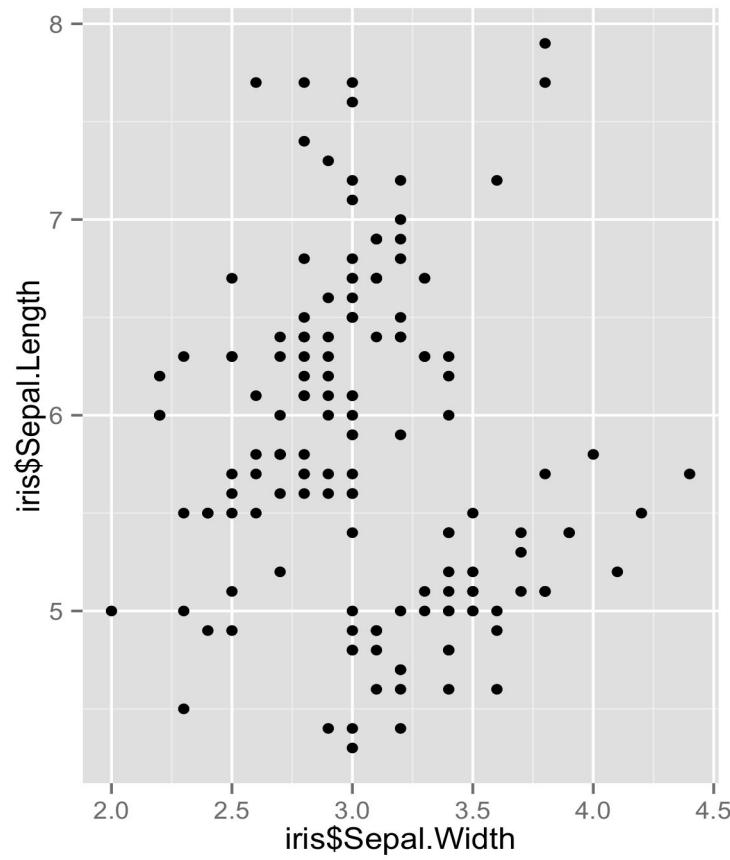
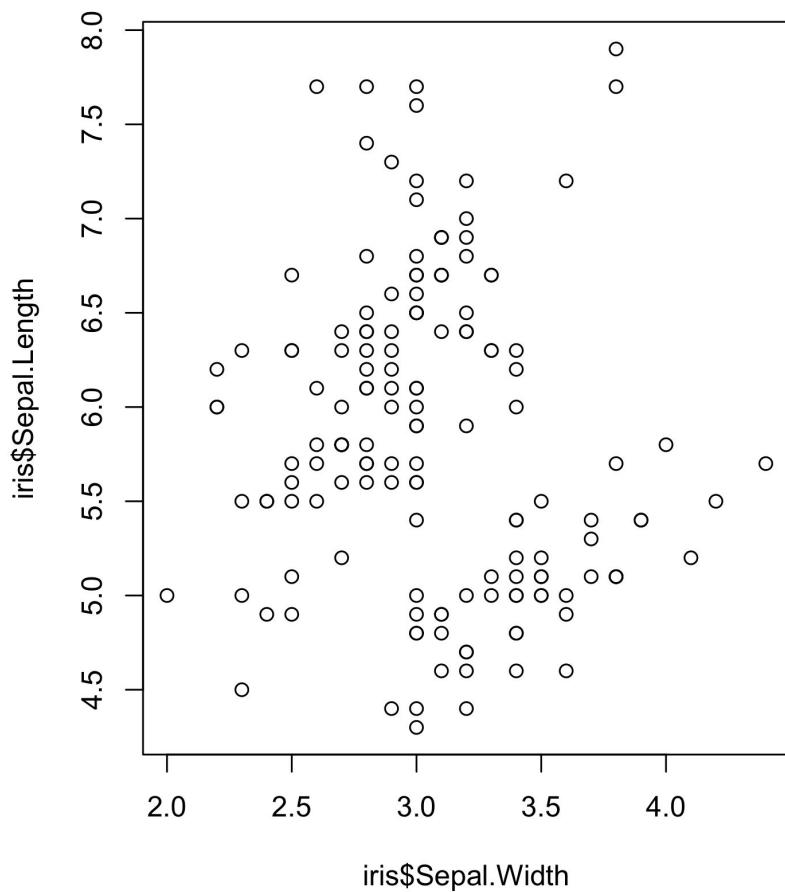


```
plot(iris$Sepal.Width,  
     iris$Sepal.Length)
```

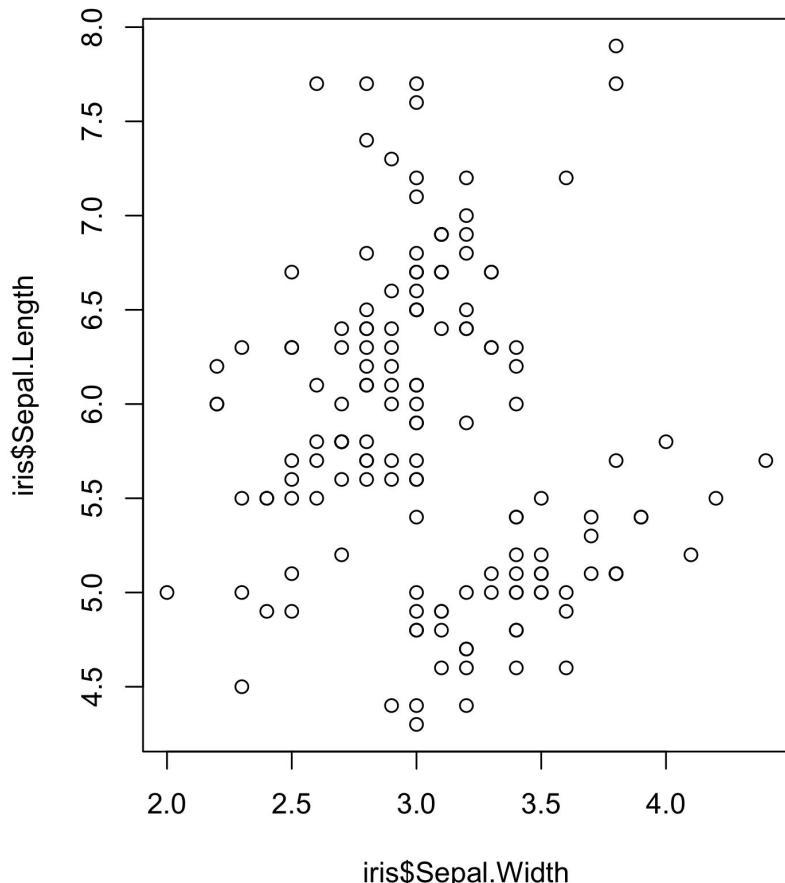
- ❖ R's basic plot method
- ❖ simple
- ❖ does different things in different contexts
- ❖ (usually in a helpful way)
- ❖ difficult to customize

# Scatterplot

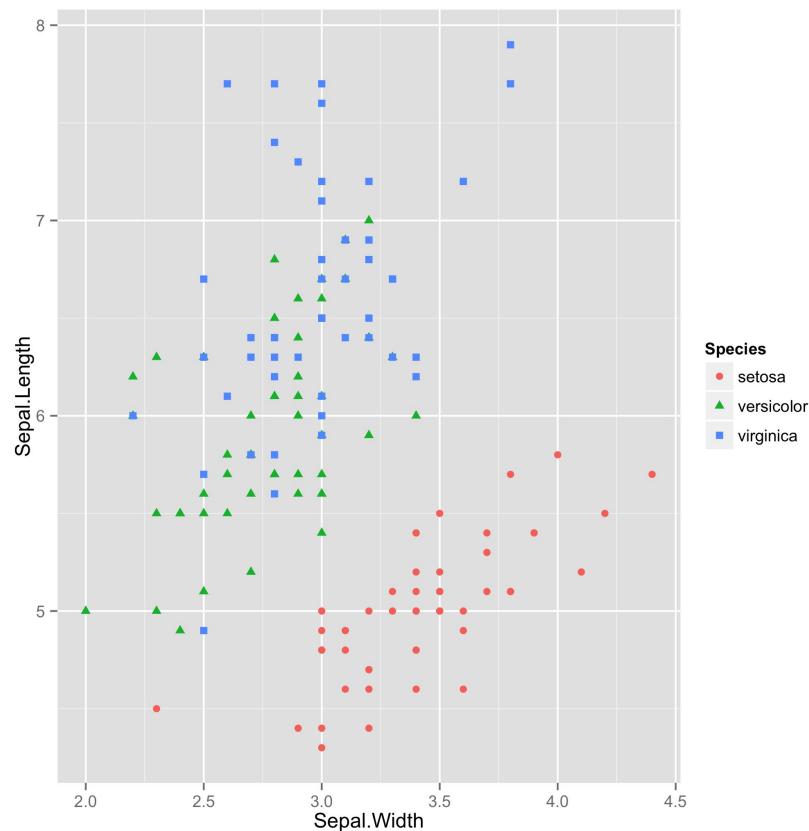
---



# Scatterplot

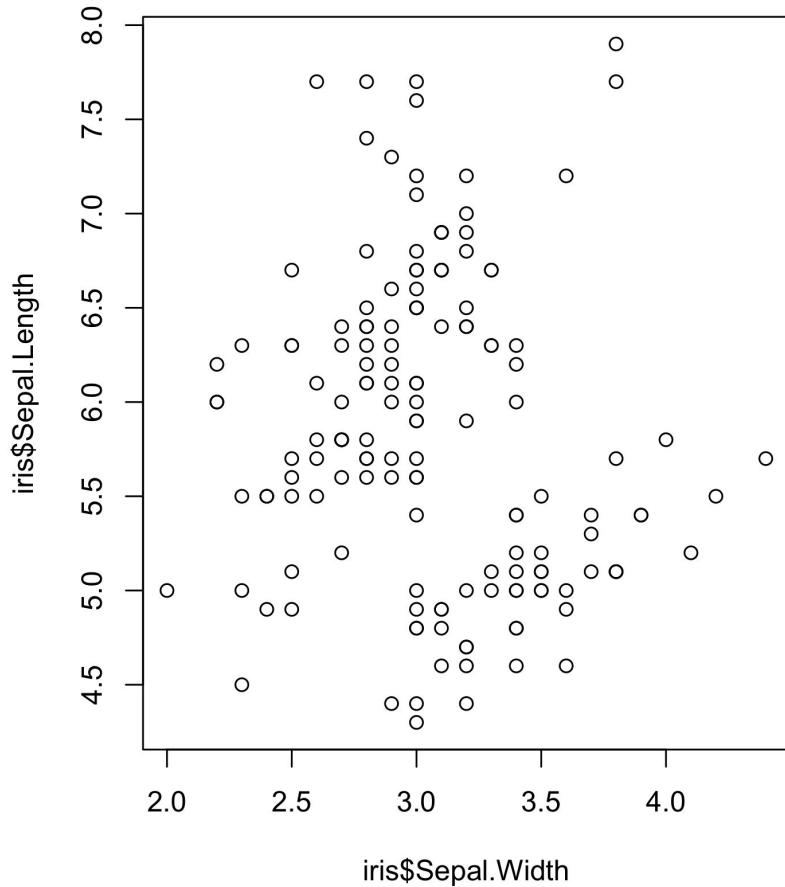


Same, but with species mapped to different colors and shapes.

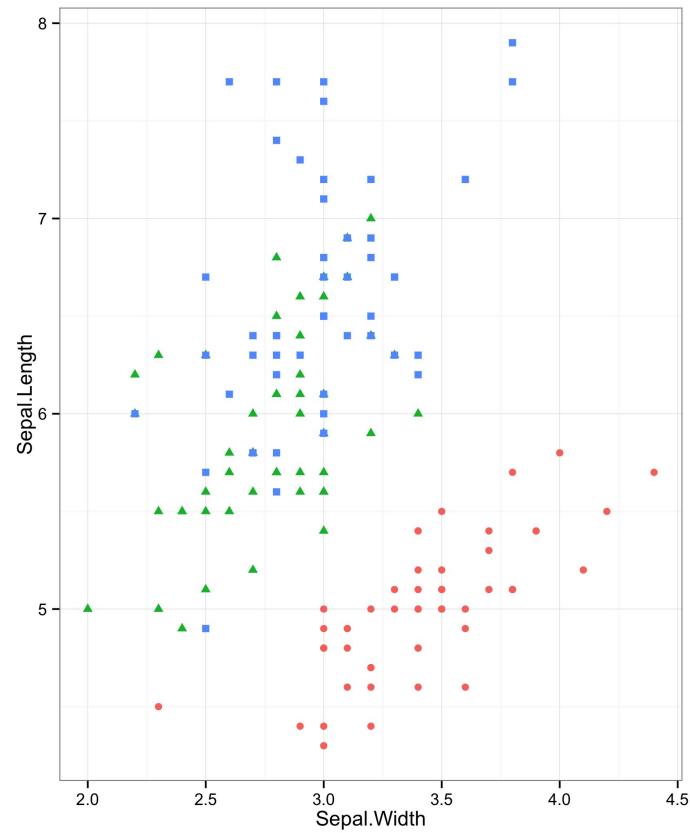


# Scatterplot

---

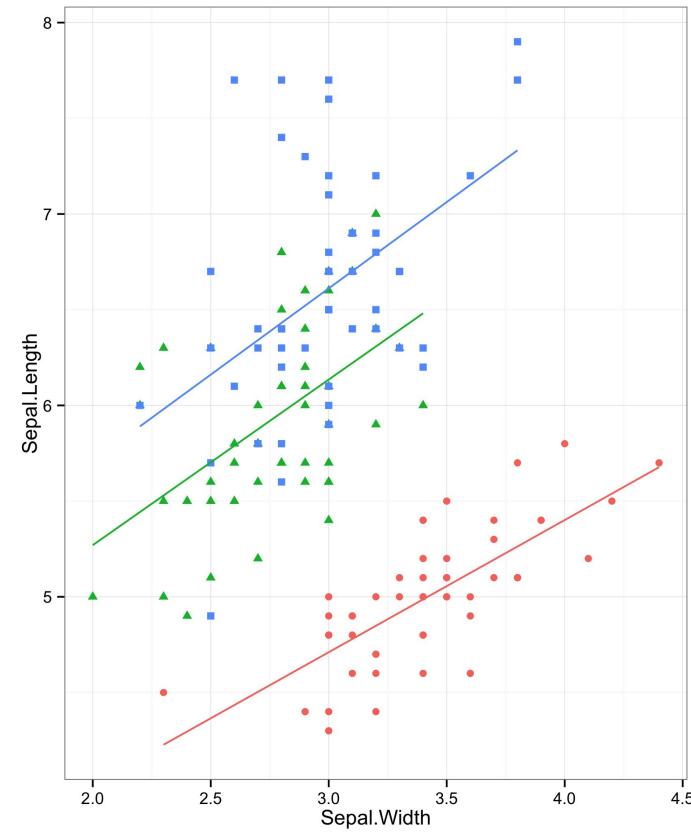
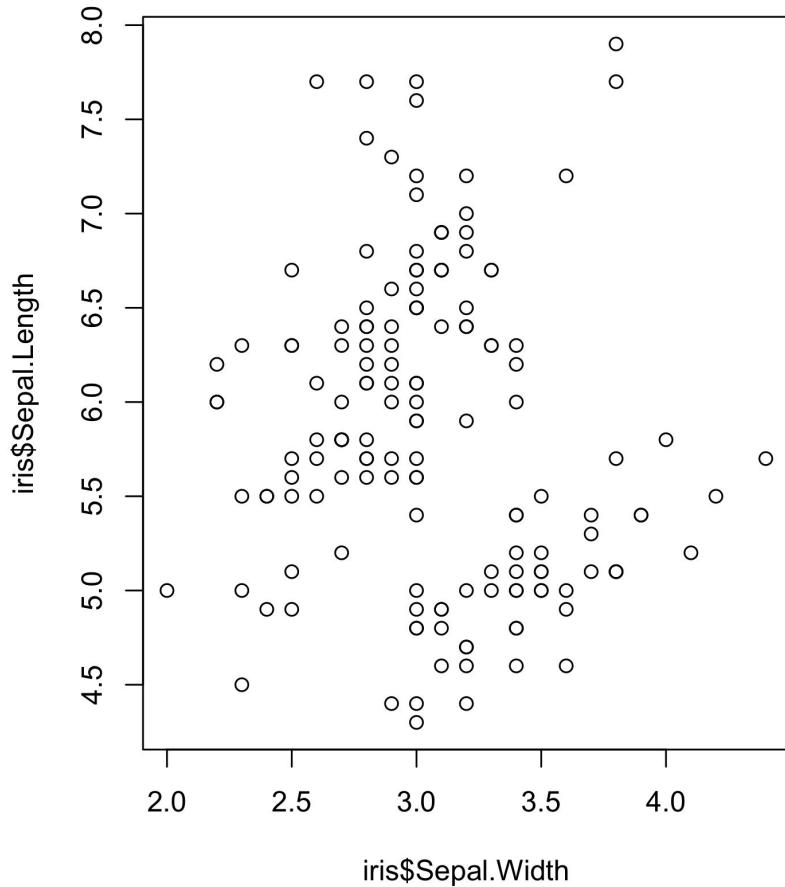


A different theme.



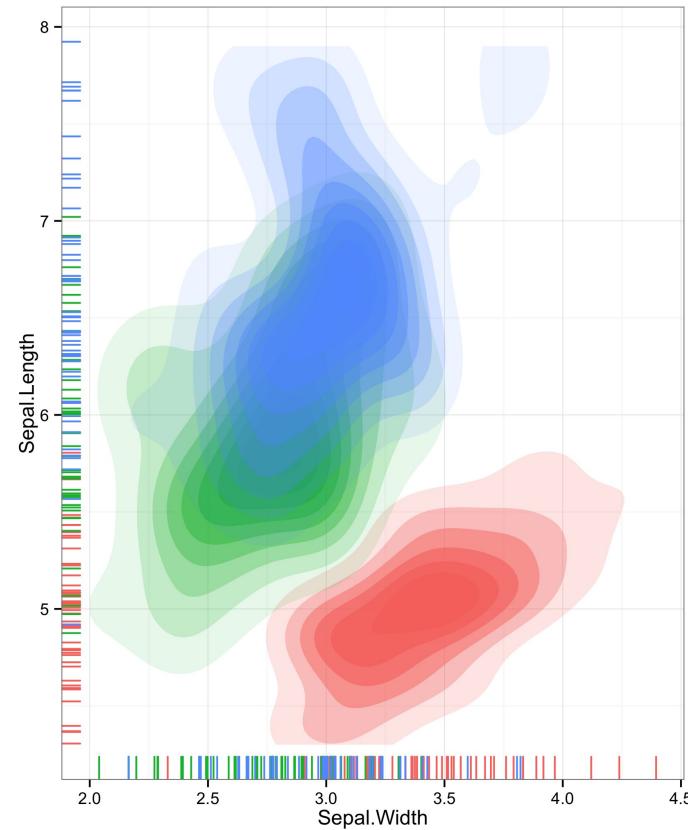
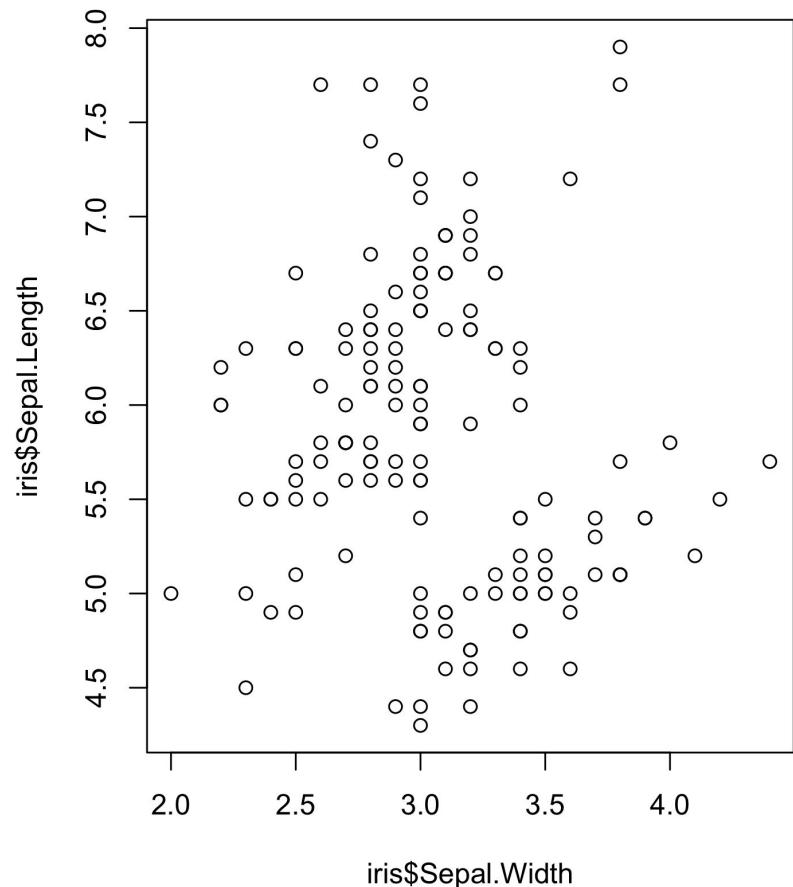
# Scatterplot with Regression Line

---

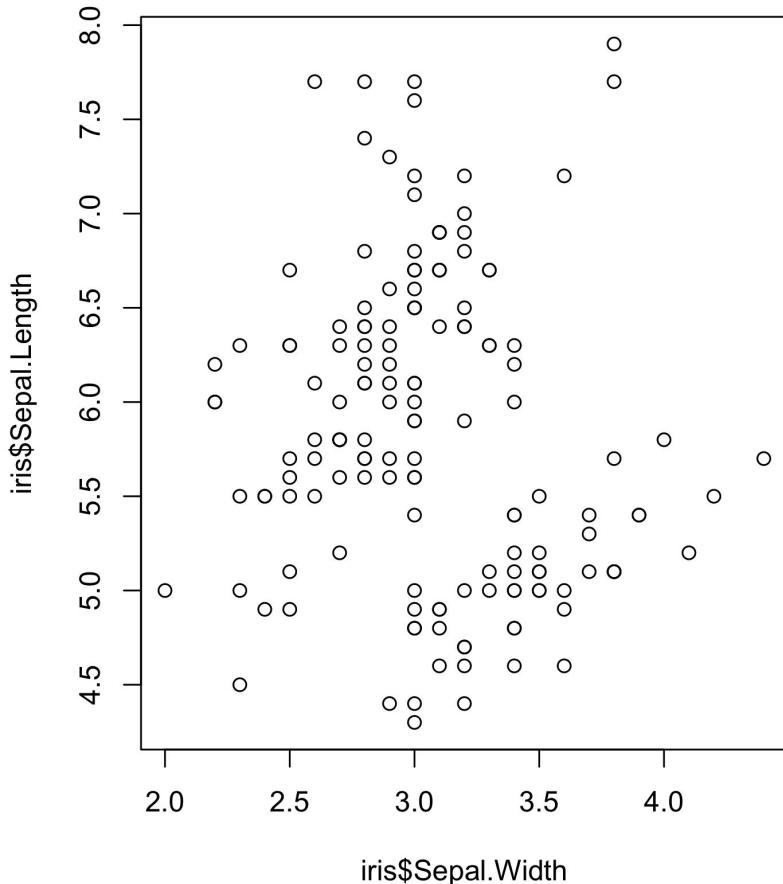


## 2D Density Plot with Rug Plot

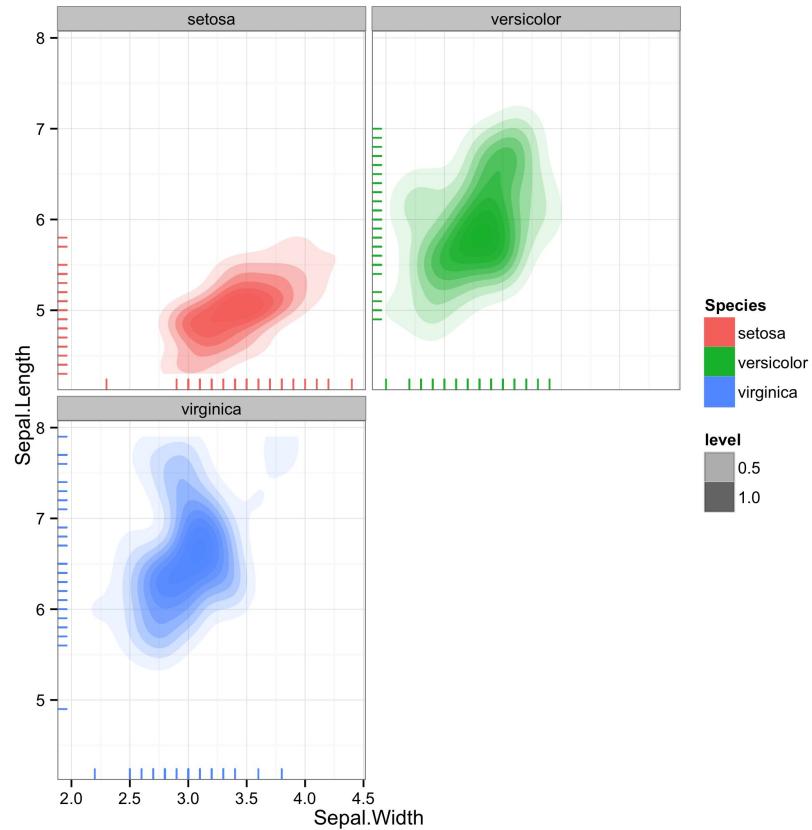
---



# Grid of 2D Density Plots



Each species has its own plot.



# Vector Field

---

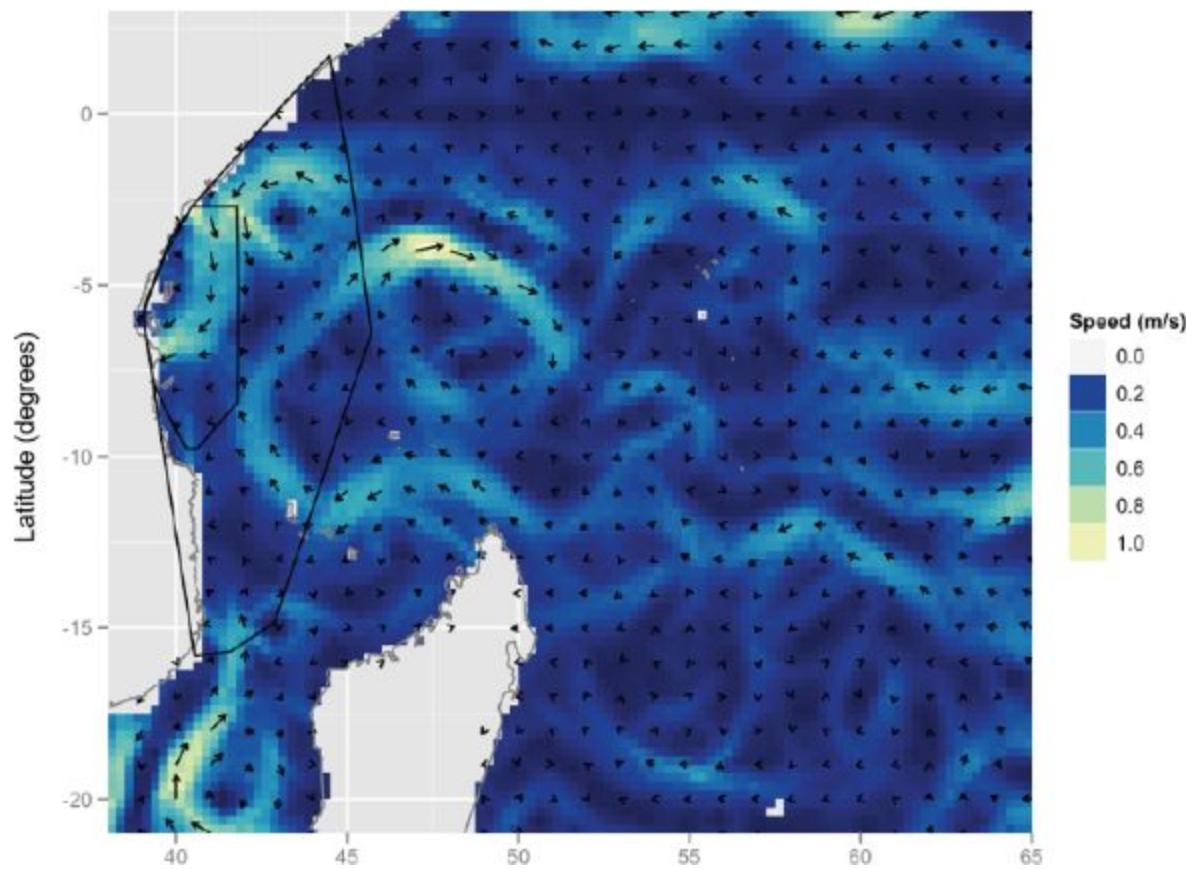
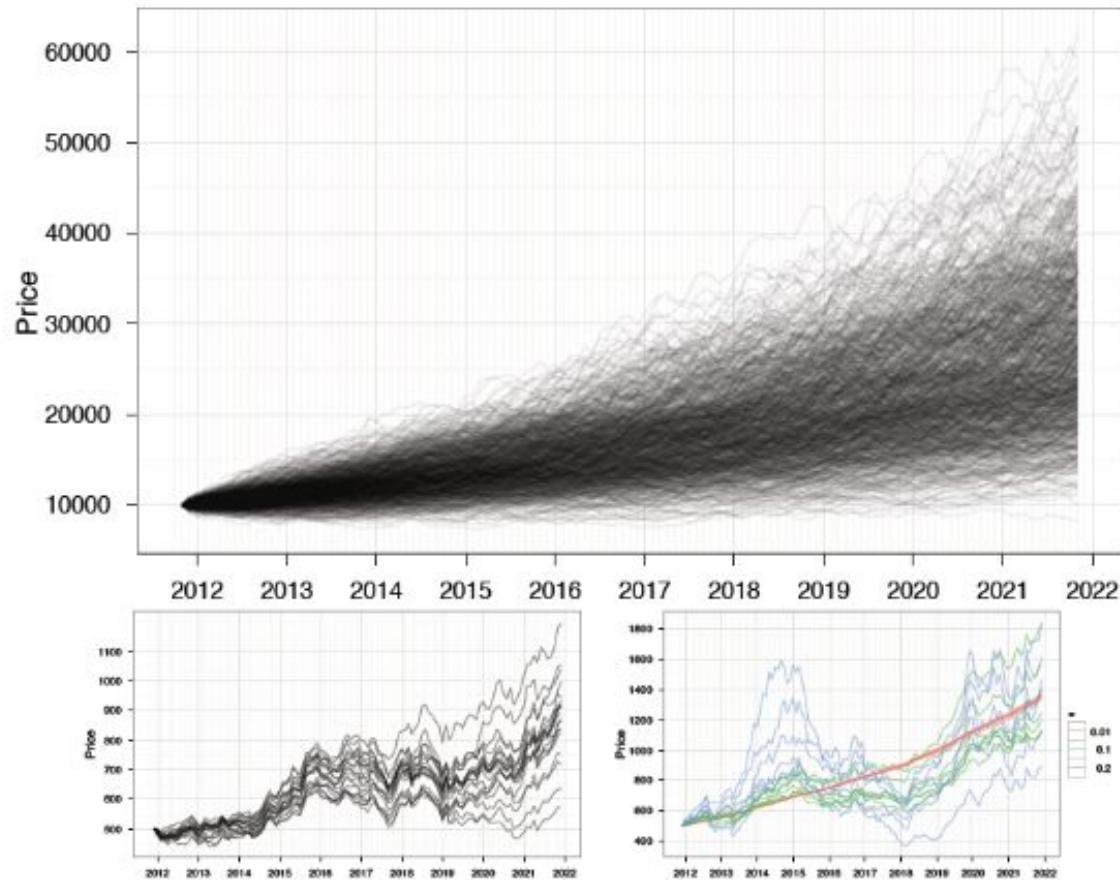


Figure: Charlotte Wickham, <http://cwick.co.nz/>

# Time Series

---



# Coefficient Plot

---

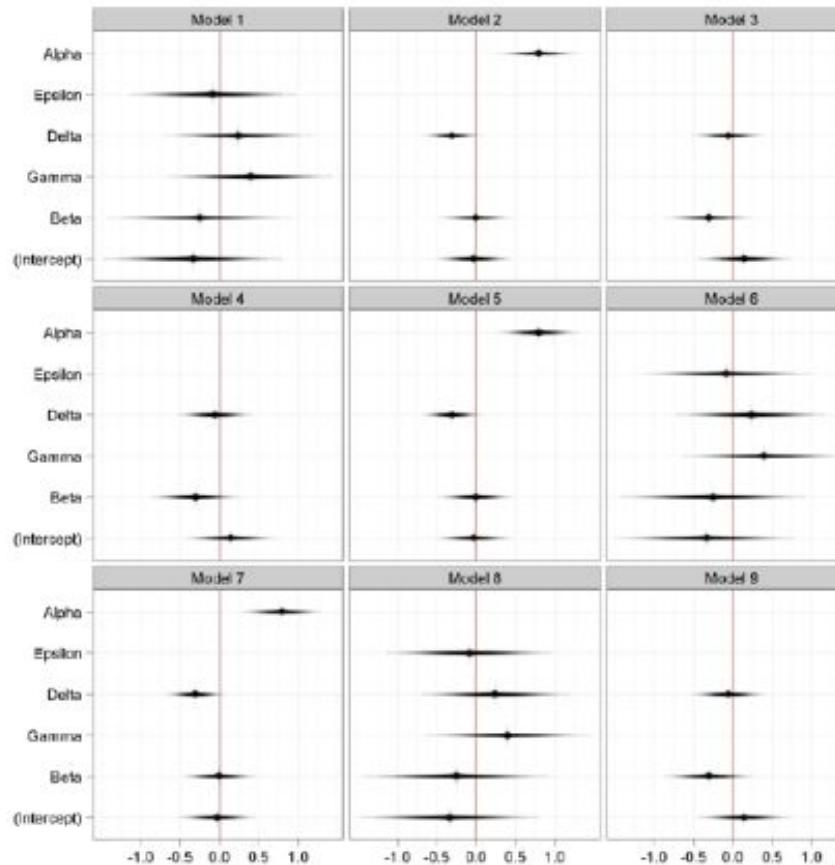


Figure: David B Sparks, <http://bit.ly/hn54NW>

# Maps

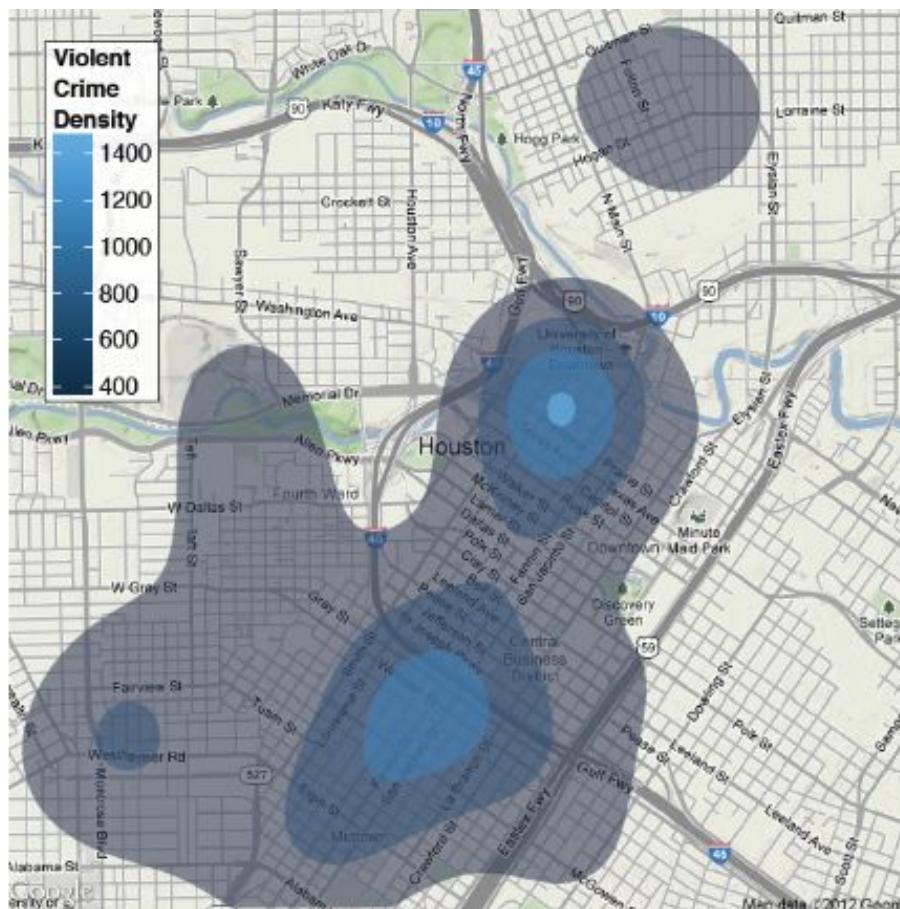


Figure: David Kahle, <https://dl.dropbox.com/u/24648660/ggmap%20useR%202012.pdf>

# Maps

---



Figure: 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

# Outline

---

- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ Constructing a ggplot2 plot
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving Graphs

# The “Grammar of Graphics”

## The “Grammar of Graphics”

---

- ❖ An abstraction which makes thinking about, reasoning about, and communicating graphics easier.
- ❖ Think “verb”, “noun”, “adjective” for graphics.
- ❖ Permits a “theory” of graphics on which to build new graphics and graphics objects.

## The “Grammar of Graphics”

---

“In brief, the grammar tells us that a statistical graphic is a **mapping** from data to **aesthetic** attributes (color, shape, size) of **geometric** objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system.”

-- *ggplot2: Elegant Graphics for Data Analysis*

## The “Grammar of Graphics”

---

- ❖ A *ggplot2 graphic* consists of:
  - ❖ **data**: an R data frame
  - ❖ **aesthetics**: values that will be plotted, e.g. x and y values, plus color and shape of dots, for a scatter plot
  - ❖ **geoms**: the type of plot
  - ❖ **facets**: side-by-side plots distinguished by a categorical variable
  - ❖ **stats**: statistical transformations like binning or smoothing
  - ❖ **scales**: adjust sizes (and other attributes)
  - ❖ **coordinates**: what coordinate system to use
  - ❖ **themes**: the overall presentation of the graphic

# Outline

---

- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ **Constructing a ggplot2 plot**
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving graphs

# Constructing a `ggplot2` Plot

## Constructing a ggplot2 Plot

---

- ❖ A ggplot2 graphic is constructed by building up **layers**.
- ❖ The ggplot function is used to initialize the basic graph structure, and then we add to it:

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

g <- ggplot(data = <name of R data frame>,
             aes(x = <name of default x variable>,
                 y = <name of default y variable>, ...,
                 <other aesthetic mappings>),
             <other plot defaults>)
```

- ❖ Note that this doesn't graph anything, because we haven't told it *how* to display these data!

## Constructing a ggplot2 Plot

---

- ❖ Next, we add a **geom** layer to the base graph with the `+` operator.

```
p <- g + geom_<geom type>(aes(  
  <aesthetic mappings for this geom>),  
  <other arguments>)
```

- ❖ This is the minimum requirement to plot something.

## Constructing a ggplot2 Plot

---

One can then add other layers as desired, e.g.,

- ❖ **other geoms**
- ❖ **facets** with `facet_grid(...)` or `facet_wrap(...)`
- ❖ **stats** with `stat_<transformation>(...)`
- ❖ **scales** with `scale_<aesthetic>_<type>(...)`
- ❖ **coordinate systems** with `coord_<system>(...)`
- ❖ **themes** with `theme(...)`
- ❖ **titles and labels** with `ggtitle(...)` or `labs(...)`

A complete list of the various features of ggplot2 plots can be found at <http://docs.ggplot2.org/>

# Outline

---

- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ Constructing a ggplot2 plot
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving Plots

# Scatterplots

## Scatterplots

---

The mpg data set comes in the ggplot2 package. Always read the help page and take a quick look at the data.

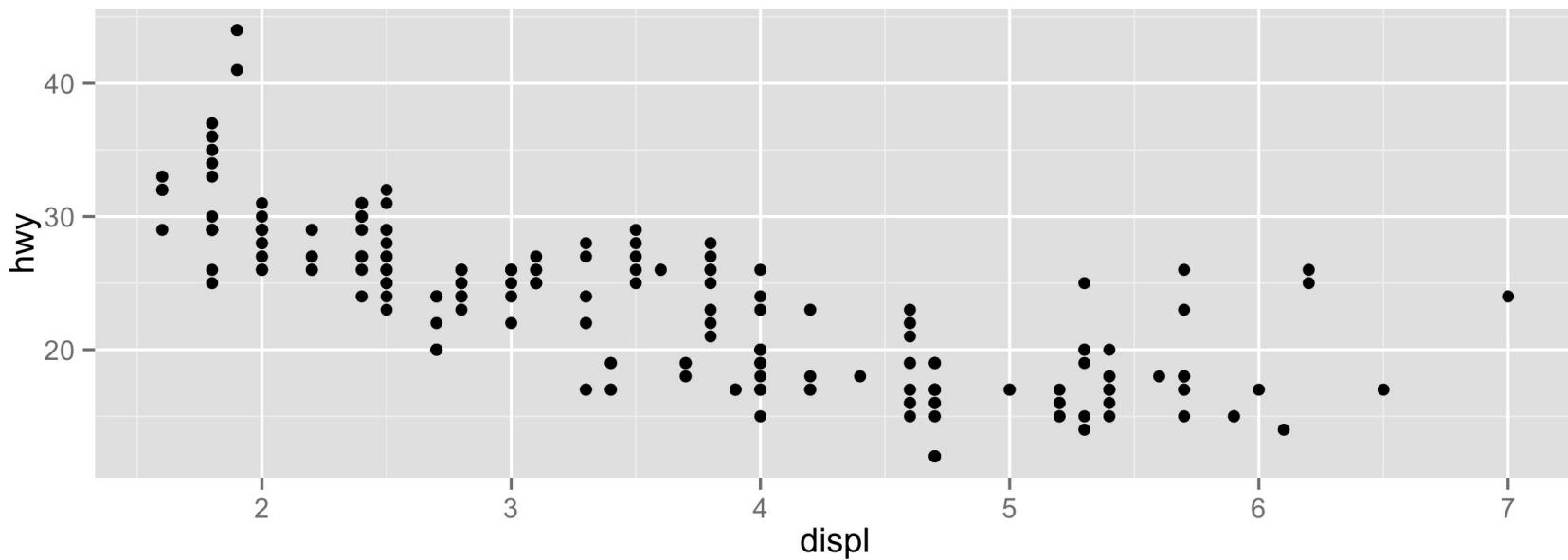
```
?mpg  
head(mpg)
```

What relationship would you expect to see between:

- ❖ engine size (displ)
- ❖ highway mileage (hwy)

## Scatterplots

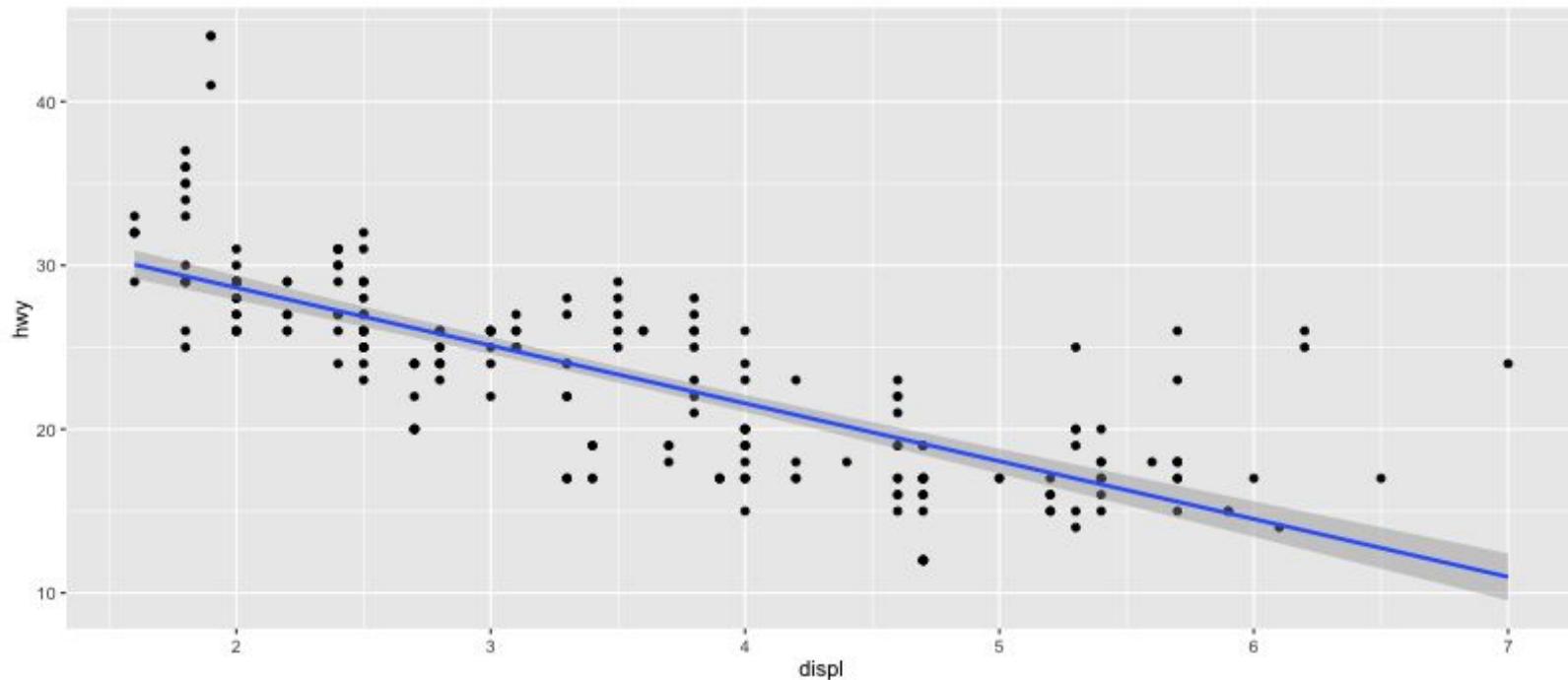
```
g <- ggplot(data = mpg, aes(x = displ, y = hwy))  
g + geom_point()
```



## Scatterplots

We can add a regression line to the existing plot to visualize this trend:

```
g + geom_point() + geom_smooth(method = "lm")
```



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

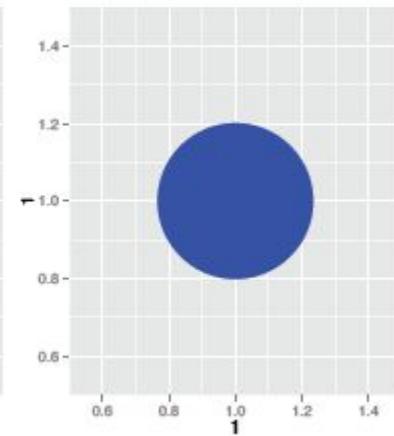
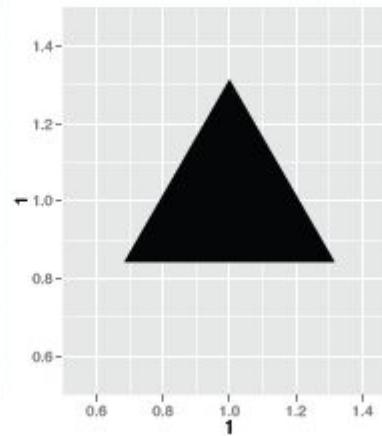
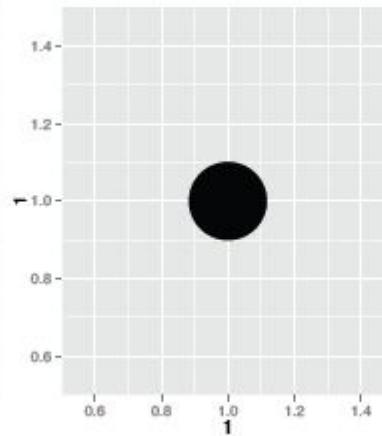
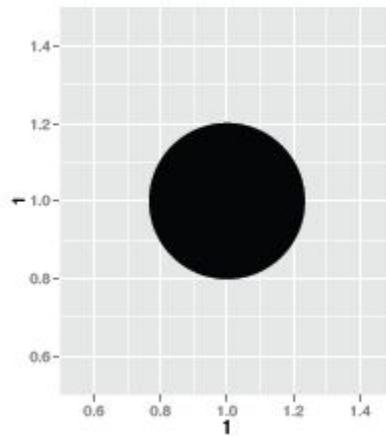
- John Tukey

# Aesthetics

# Aesthetics

---

Visual characteristics that data are mapped to.



## Aesthetics

---

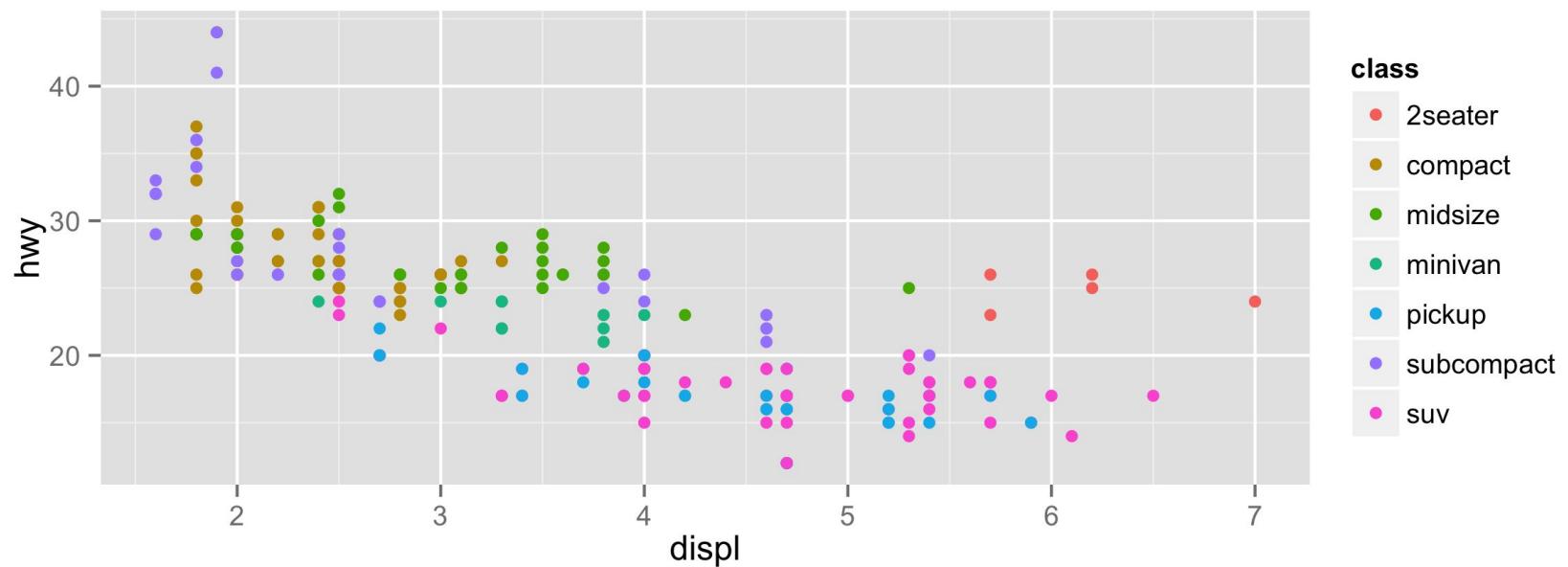
- ❖ color, size, shape, alpha
- ❖ Add color, size, and shape aesthetics to your graph. Experiment.
- ❖ Do different things happen for discrete and continuous variables?
- ❖ What happens when you use more than one aesthetic?
- ❖ What are the warning messages trying to tell us?

```
g + geom_point(aes(color = class))
g + geom_point(aes(size = class))
g + geom_point(aes(shape = class))
g + geom_point(aes(alpha = class))
```

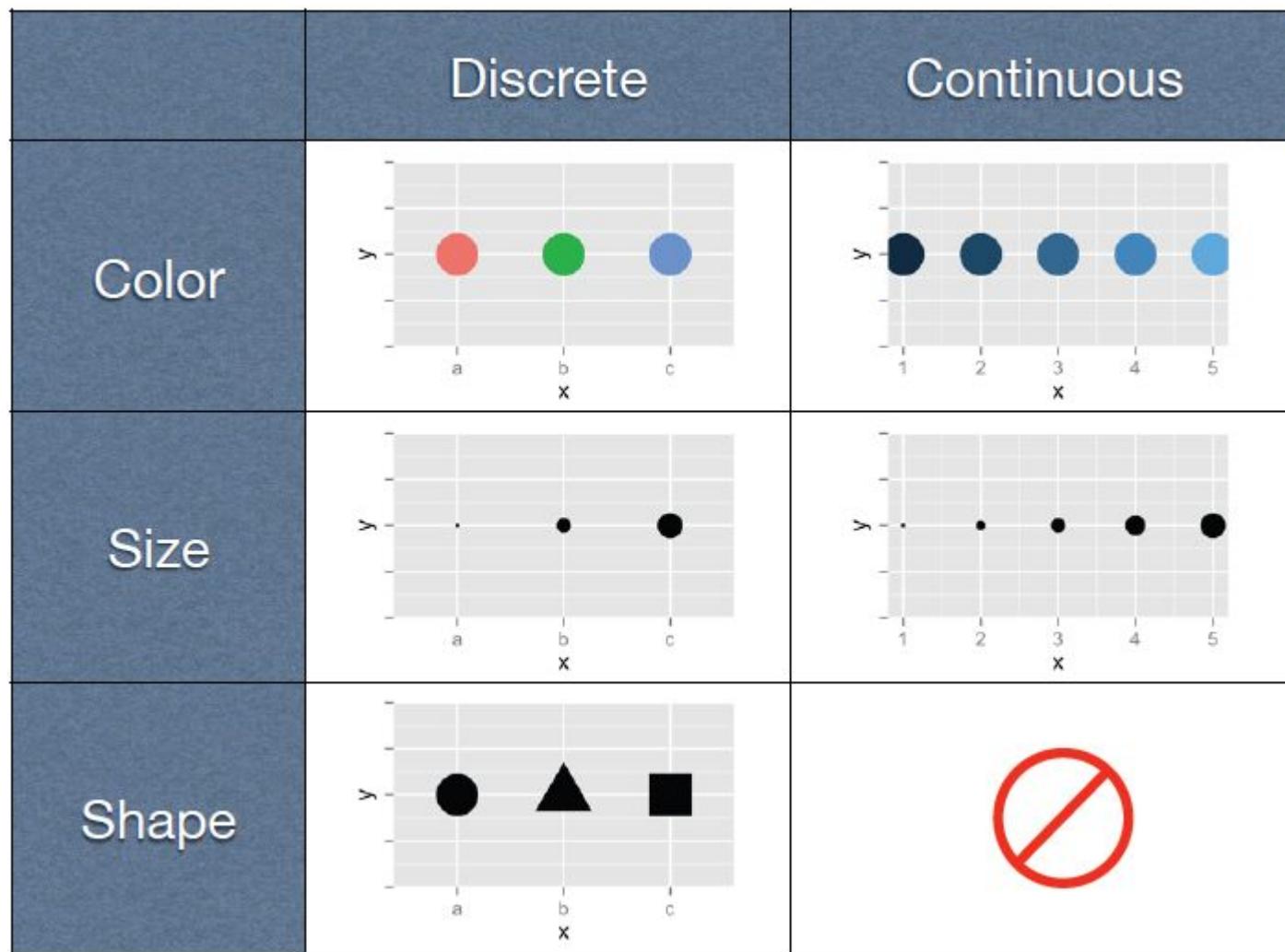
## Aesthetics

Notice: The legend is chosen and displayed automatically.

```
g + geom_point(aes(color = class))
```



# Aesthetics



# Faceting

## Faceting

---

- ❖ Smaller plots that display different subsets of the data.
- ❖ Also useful for exploring conditional relationships.
- ❖ Useful for large data.
- ❖ `facet_grid` function.
- ❖ `facet_wrap` function.

## Faceting

---

- ❖ `facet_grid()`: 2d grid, rows ~ cols, . is a placeholder that can be used if you don't want a split.
- ❖ `facet_wrap()`: 1d ribbon wrapped into 2d.

# Faceting

---

## Your turn

```
g + geom_point() + facet_grid(. ~ cyl)
```

```
g + geom_point() + facet_grid(drv ~ .)
```

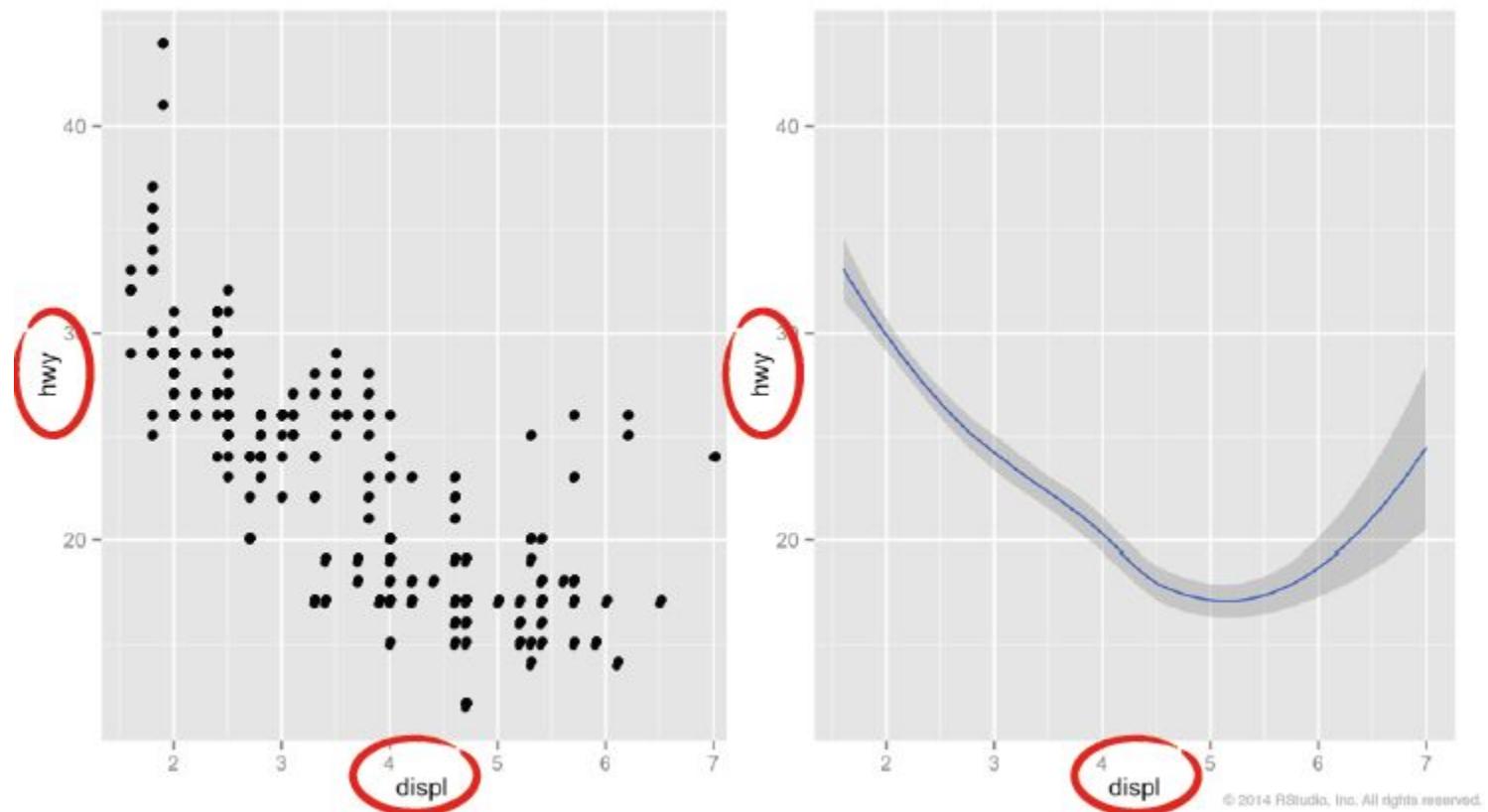
```
g + geom_point() + facet_grid(drv ~ cyl)
```

```
g + geom_point() + facet_wrap( ~ class)
```

# Geoms

# Geoms

How are these plots similar?



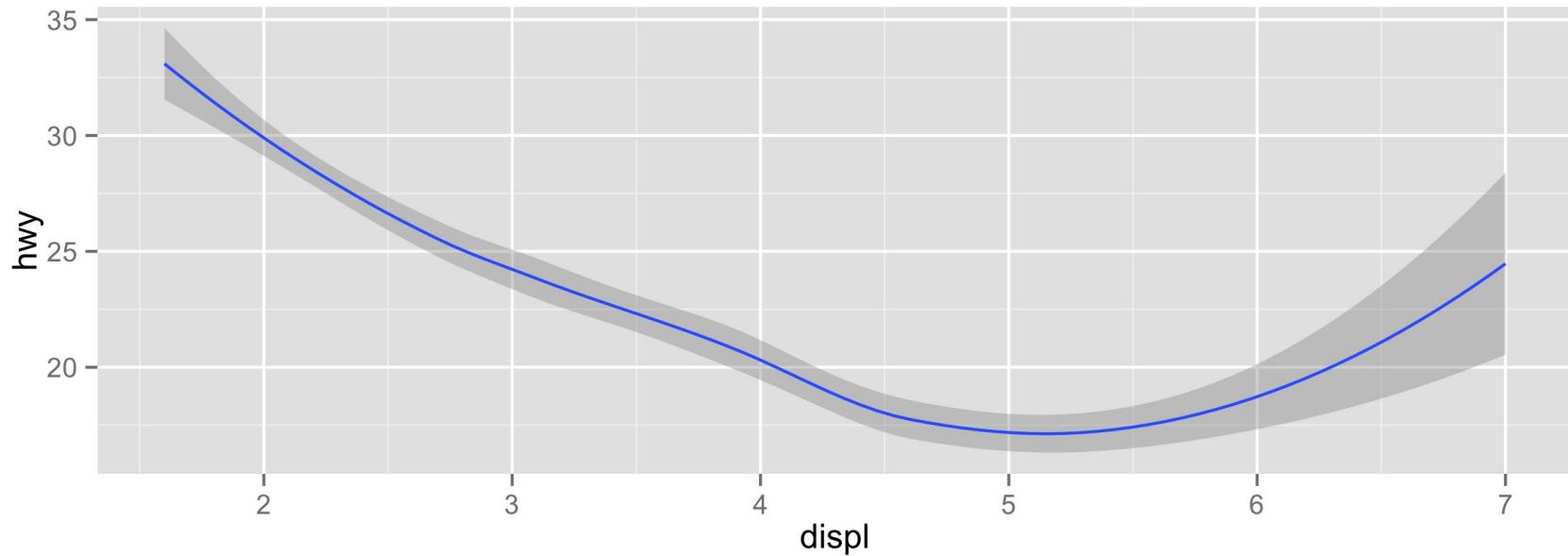
© 2014 RStudio, Inc. All rights reserved.

## Geoms

---

The `geom_<type>(...)` function determines the "type" of graph.

```
g + geom_smooth() # method = "auto" is the default
```

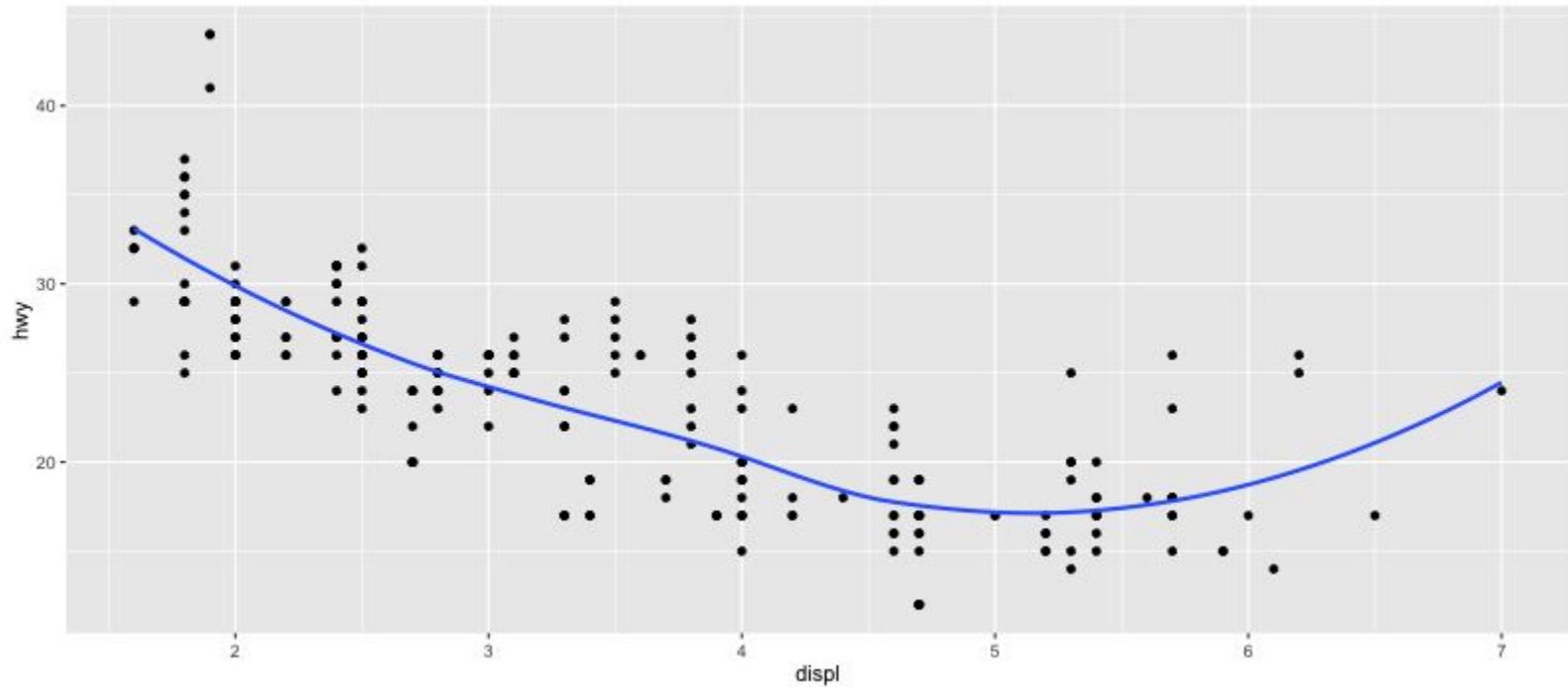


## Geoms

---

As we saw earlier, we can layer multiple geoms.

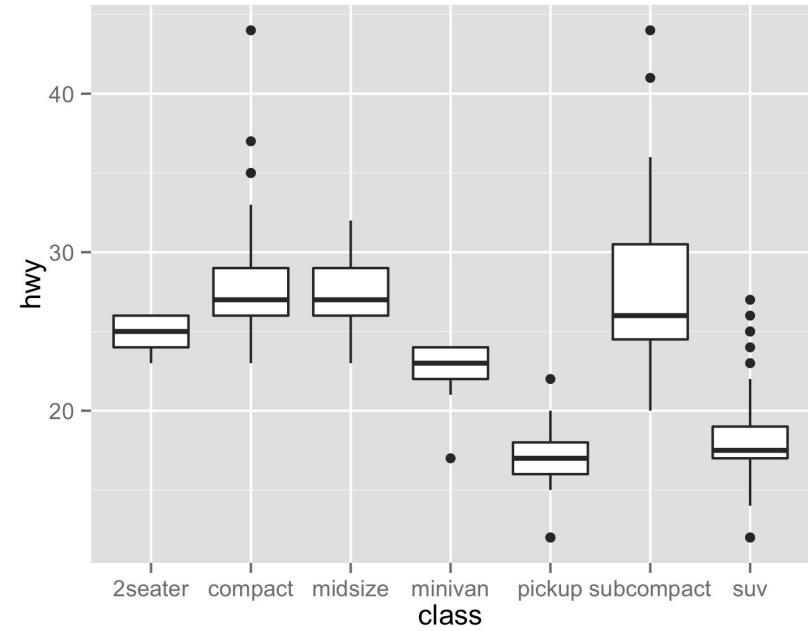
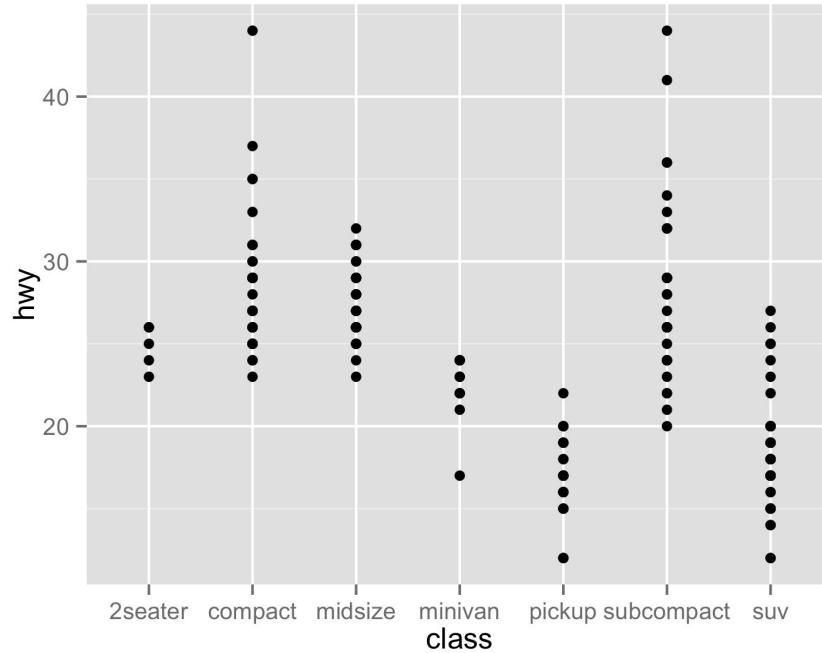
```
g + geom_point() + geom_smooth(se = FALSE) # Turn off  
confidence band
```



# Geoms

How would you replace this scatterplot with one that draws boxplots?

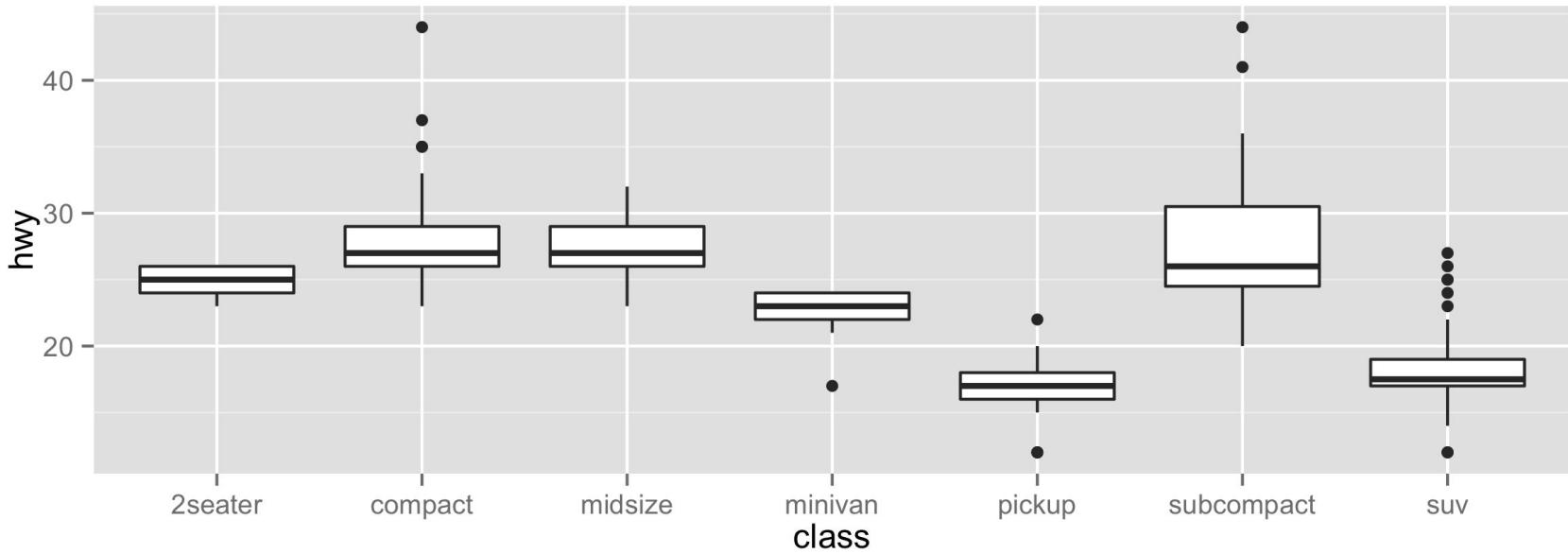
```
ggplot(data = mpg, aes(x = class, y = hwy)) + geom_point()
```



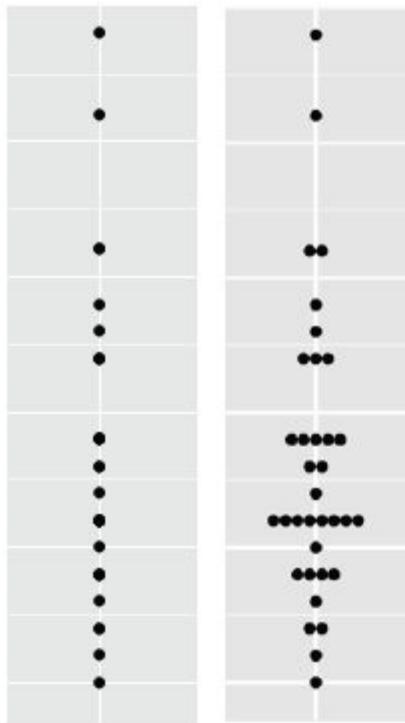
## Geoms

---

```
g <- ggplot(data = mpg, aes(x = class, y = hwy))  
g + geom_boxplot()
```

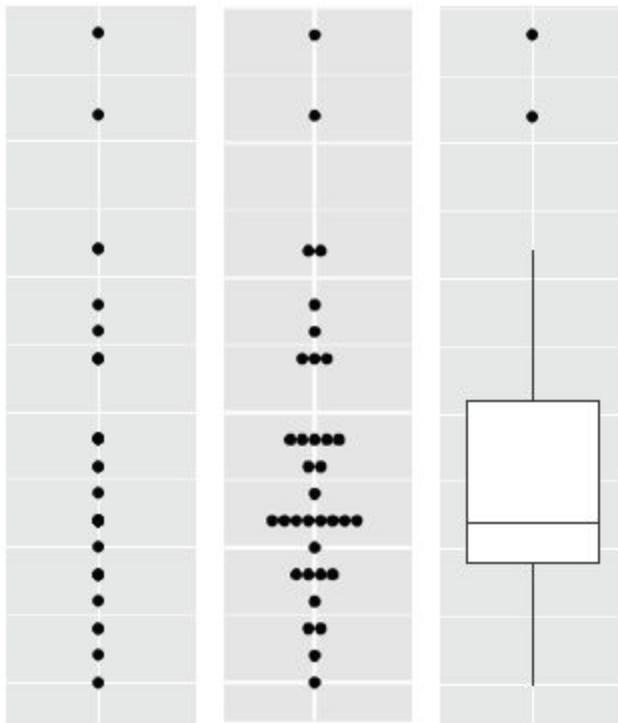


# boxplots



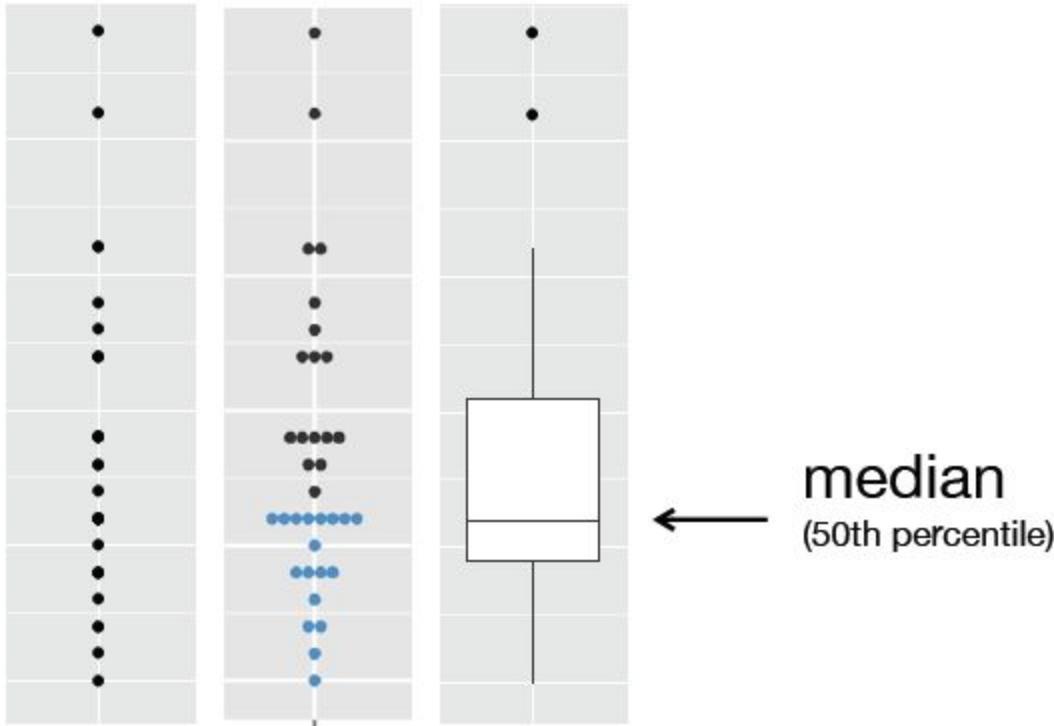
© 2014 RStudio, Inc. All rights reserved.

# boxplots



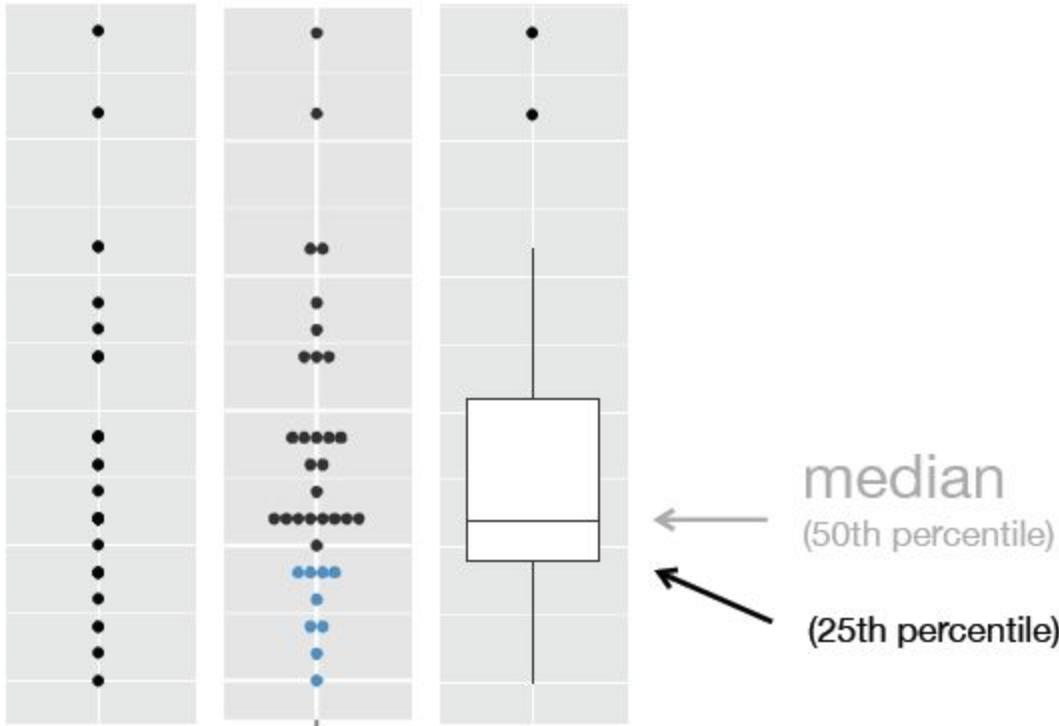
© 2014 RStudio, Inc. All rights reserved.

# boxplots



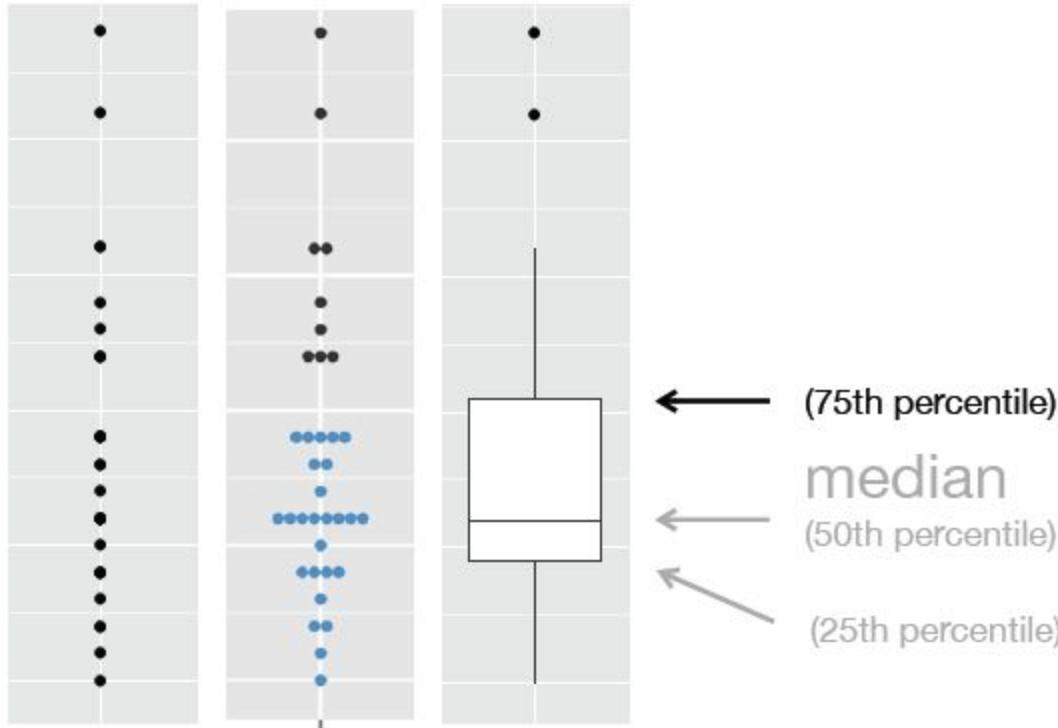
© 2014 RStudio, Inc. All rights reserved.

# boxplots



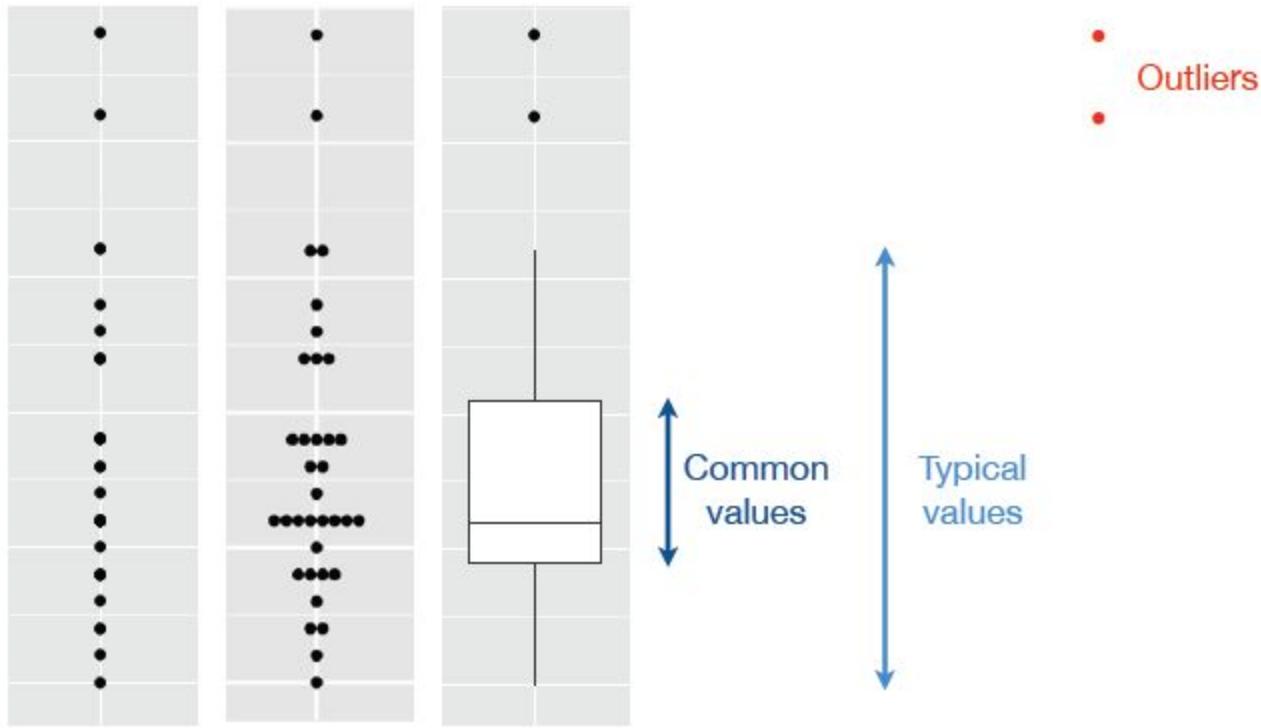
© 2014 RStudio, Inc. All rights reserved.

# boxplots



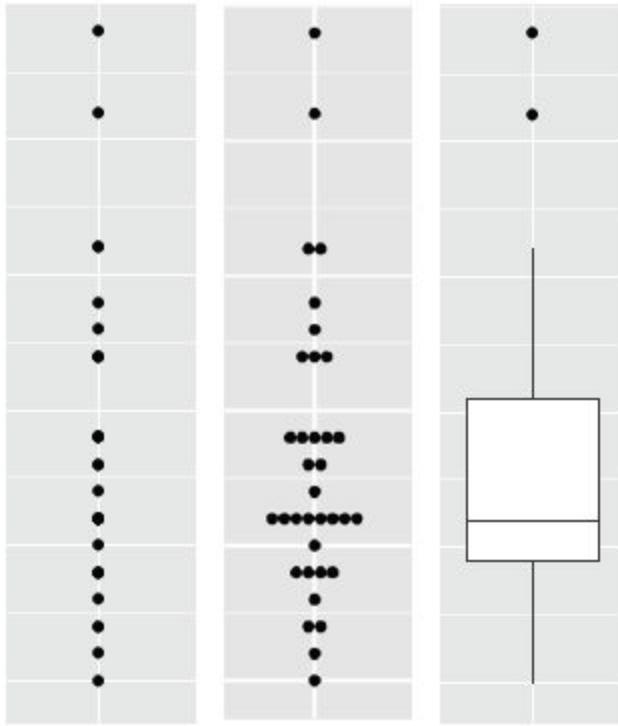
© 2014 RStudio, Inc. All rights reserved.

# boxplots



© 2014 RStudio, Inc. All rights reserved.

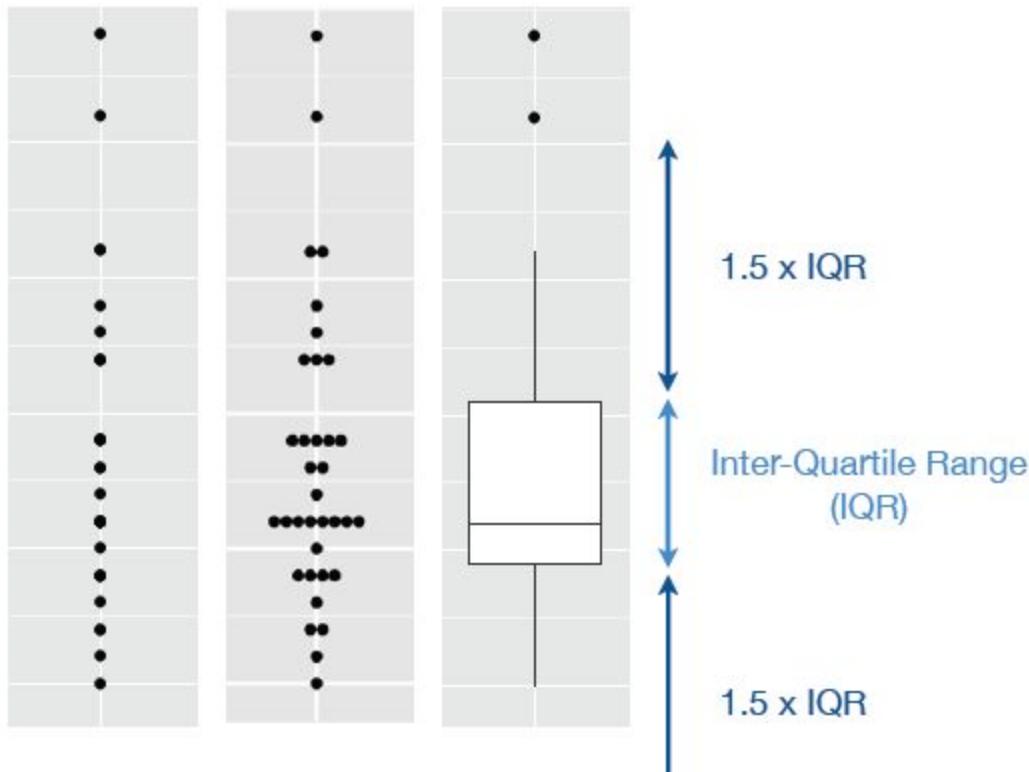
# boxplots



Inter-Quartile Range  
(IQR)

© 2014 RStudio, Inc. All rights reserved.

# boxplots

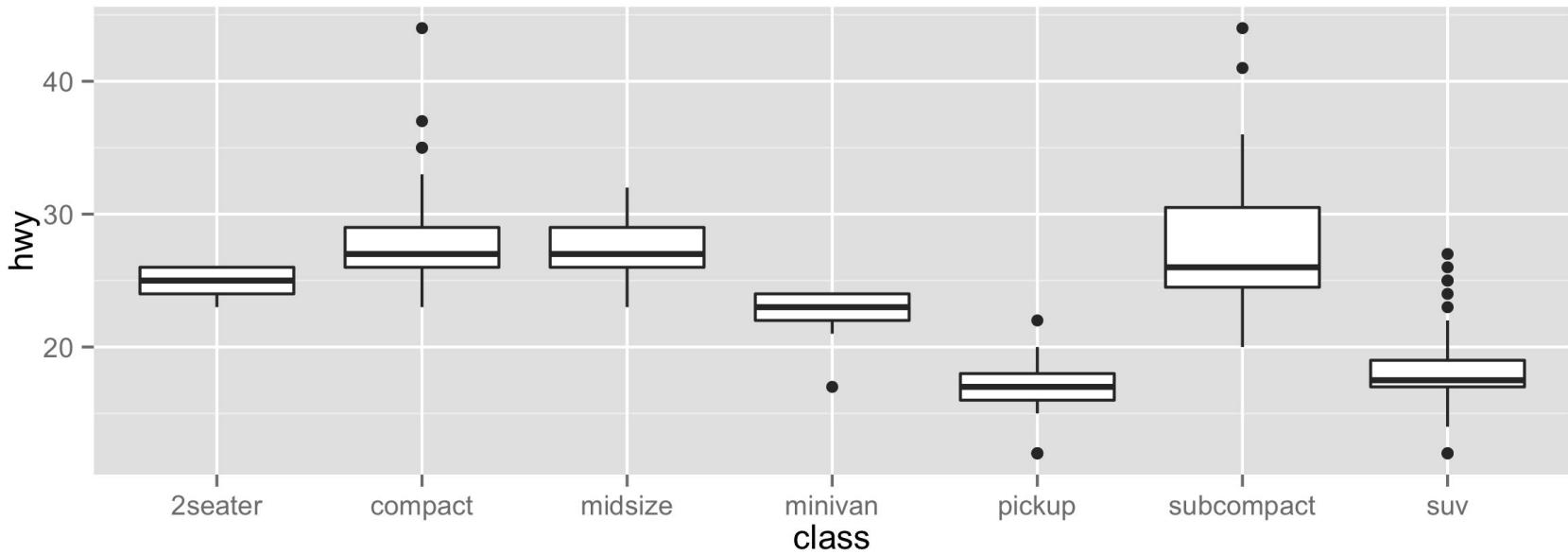


© 2014 RStudio, Inc. All rights reserved.

## Geoms

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

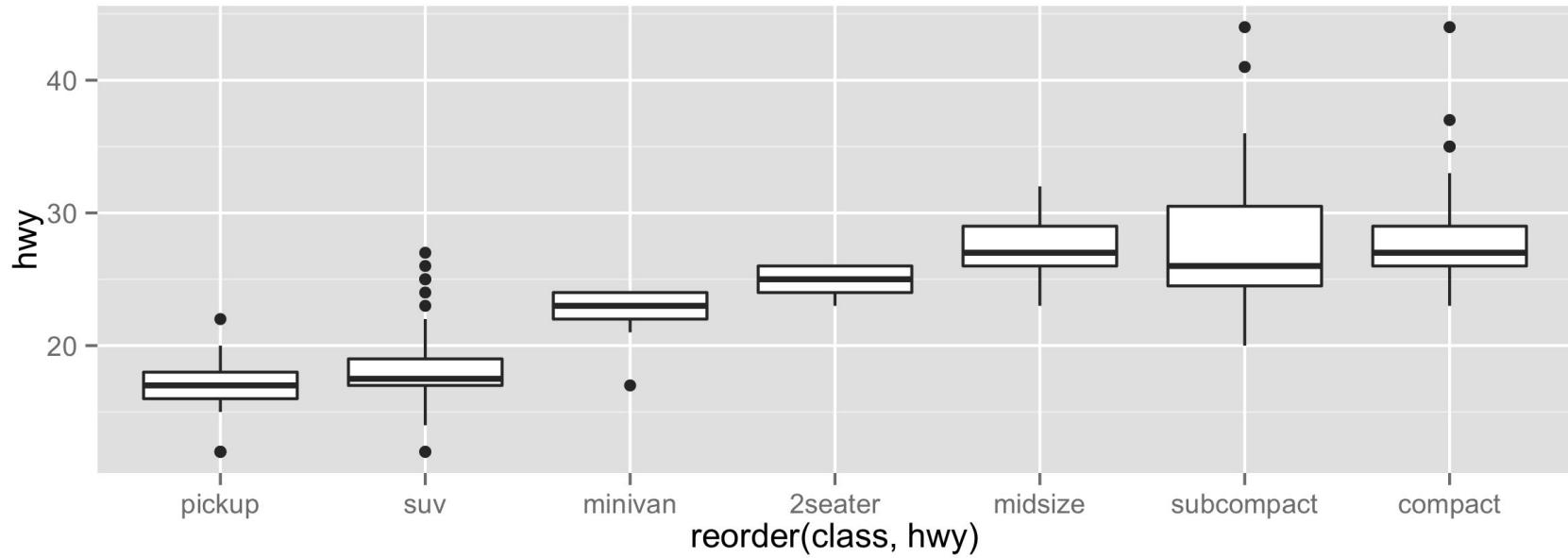
```
g + geom_boxplot()
```



## Geoms

Use the **reorder** function to make it easier to compare classes.

```
g <- ggplot(data = mpg, aes(x = reorder(class, hwy), y = hwy))  
g + geom_boxplot()
```



## Geoms

---

- ❖ Read the help for the `reorder` function.
- ❖ Redraw the previous plots with the classes ordered by the `median`.

# Geoms

## Description

Useful overview

## Usage

Good place to spot default values

## Arguments

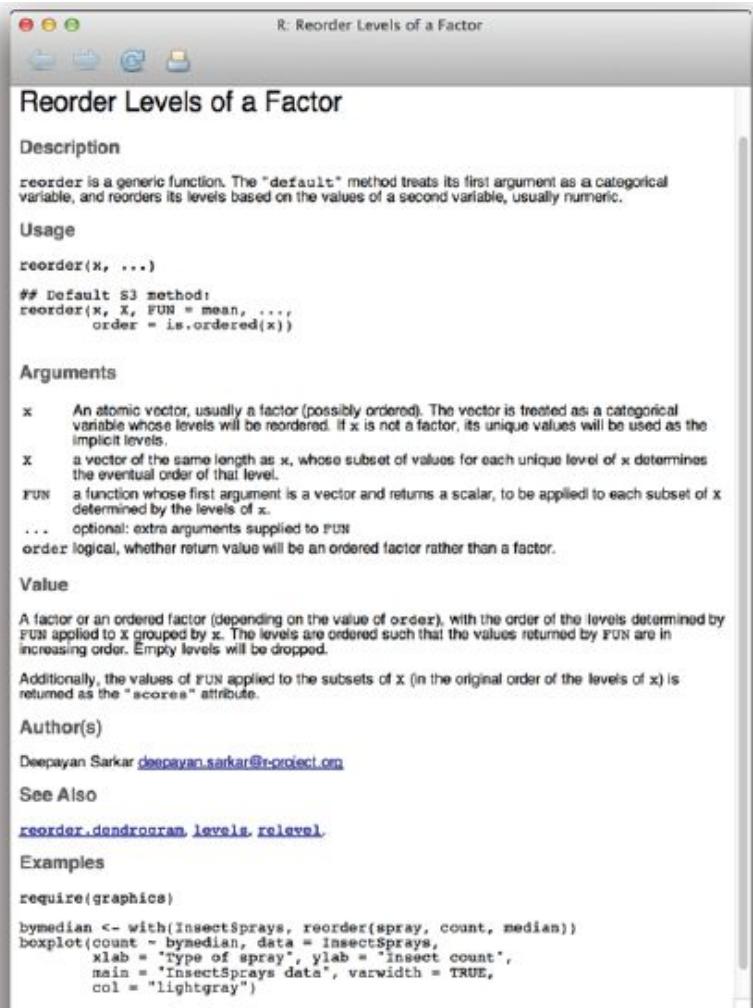
explanation of each argument

## Value

what the function returns

## Examples

Most helpful section!

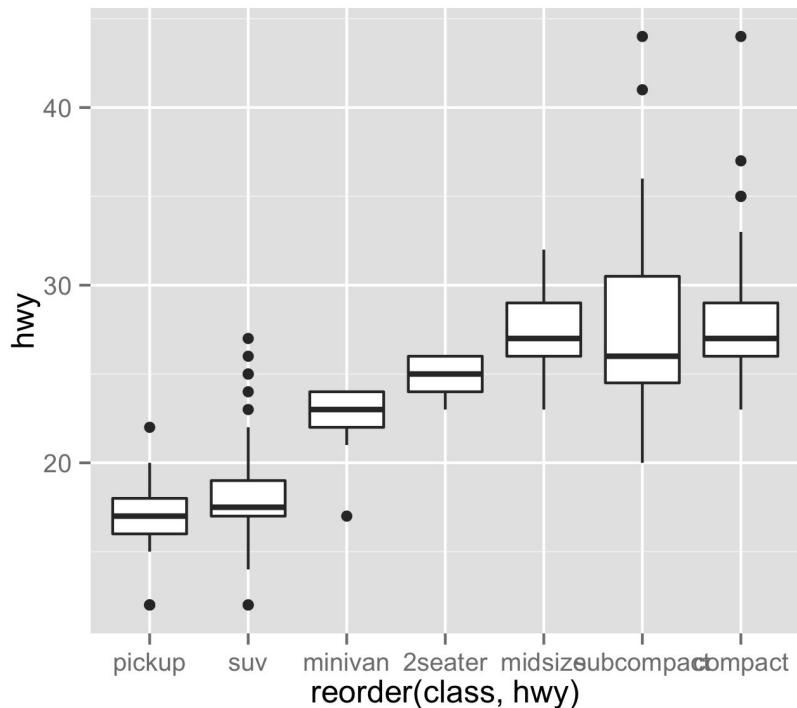


The screenshot shows the R help documentation for the 'reorder' function. The title is 'R: Reorder Levels of a Factor'. The 'Description' section states that 'reorder' is a generic function where the 'default' method treats its first argument as a categorical variable and reorders its levels based on the values of a second variable, usually numeric. The 'Usage' section shows the function signature: `reorder(x, ...)`. It includes a note about the S3 method: `# Default S3 method:` `reorder(x, X, FUN = mean, ..., order = is.ordered(x))`. The 'Arguments' section details the parameters: `x` (an atomic vector, usually a factor), `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`), and `order` (logical, whether return value will be an ordered factor rather than a factor). The 'Value' section describes the output as a factor or ordered factor with levels ordered by `FUN` grouped by `x`, with empty levels dropped. It also notes that the 'scores' attribute contains the values of `FUN` applied to the subsets of `x`. The 'Author(s)' section credits Deepayan Sarkar ([deepayan.sarkar@r-project.org](mailto:deepayan.sarkar@r-project.org)). The 'See Also' section links to `reorder.dendrogram`, `levels`, and `relevel`. The 'Examples' section provides an R code example:

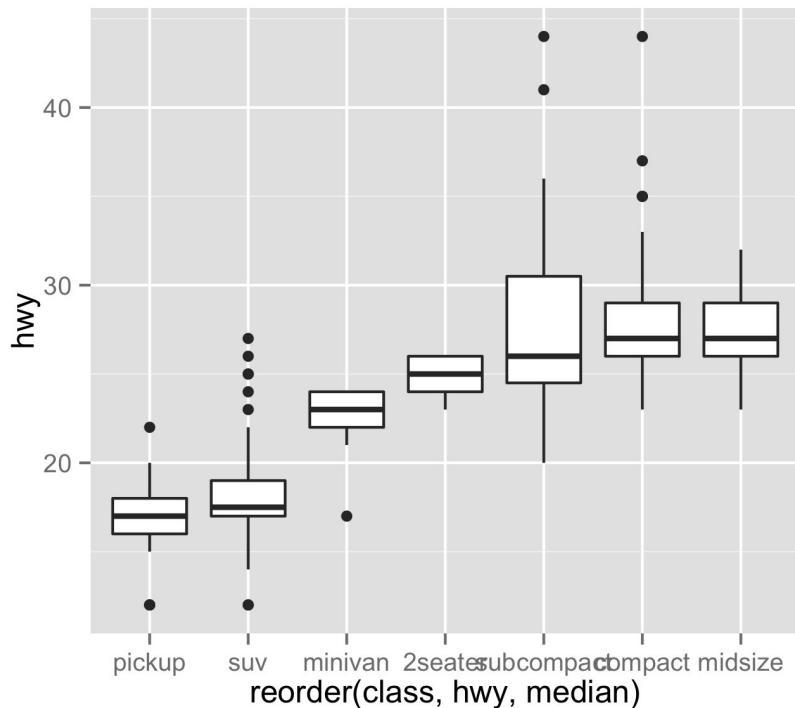
```
require(graphics)
bymedian <- with(InsectSprays, reorder(spray, count, median))
boxplot(count ~ bymedian, data = InsectSprays,
       xlab = "Type of spray", ylab = "Insect count",
       main = "InsectSprays data", varwidth = TRUE,
       col = "lightgray")
```

# Geoms

```
ggplot(data = mpg, aes(x =  
reorder(class, hwy), y = hwy))  
+ geom_boxplot()
```



```
ggplot(data = mpg, aes(x =  
reorder(class, hwy, median), y  
= hwy)) + geom_boxplot()
```



# Geoms

As a reminder, there are always the help pages:

- ❖ <http://docs.ggplot2.org/>

ggplot2 0.9.3.1    [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`

### Dependencies

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

  
geom\_abline

  
geom\_area

  
geom\_bar

  
geom\_bin2d

  
geom\_blank

  
geom\_boxplot

  
geom\_contour

  
geom\_crossbar

  
geom\_density

  
geom\_density2d

# Diamonds Data Set

## Diamonds Data Set

---

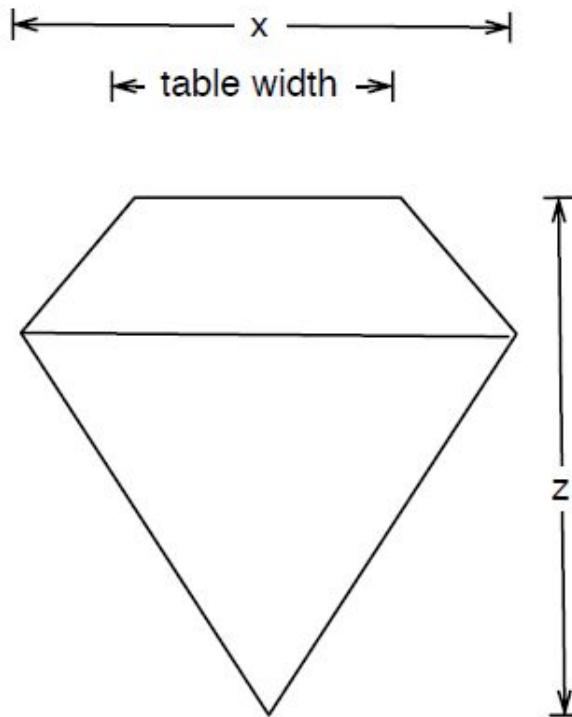
- ❖ Approx. 54,000 round diamonds from <http://www.diamondse.info/>
- ❖ comes in the ggplot2 package

```
names(diamonds)
```

```
[1] "carat"   "cut"      "color"    "clarity"  "depth"  
"table"    "price"    "x"        "y"  
[10] "z"
```

# Diamonds Data Set

---



depth =  $z / \text{diameter}$   
table =  $\text{table width} / x * 100$

© 2014 RStudio, Inc. All rights reserved.

# Diamonds Data Set

---



Figure: <http://www.mlcl.com/colour.aspx>

## Diamonds Data Set

---

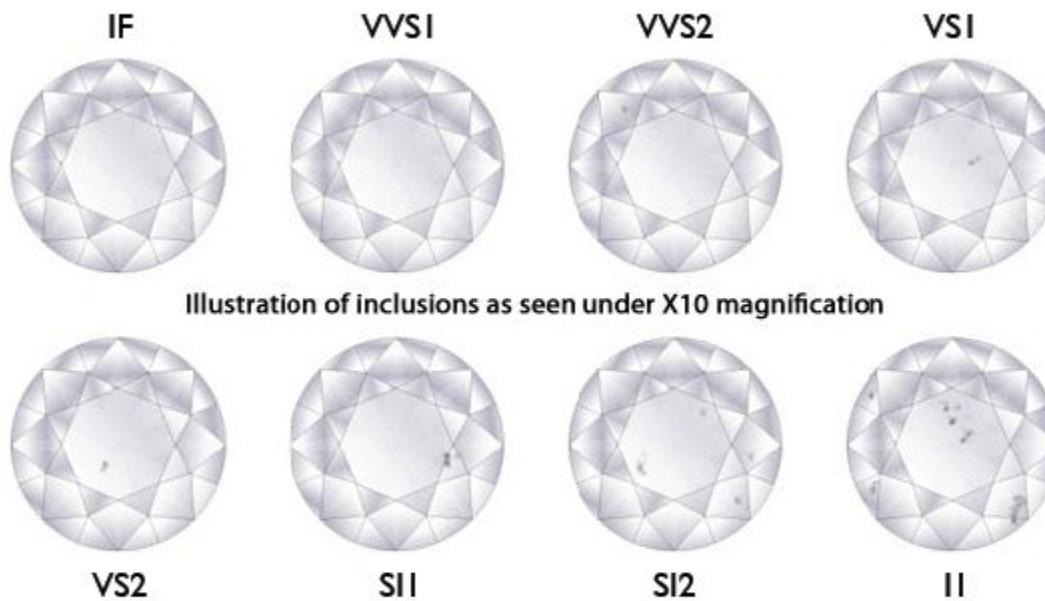


Figure: <http://www.thediamondsexperts.com/diamonds-guide>

# Outline

---

- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ Constructing a ggplot2 plot
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving Graphs

# Bar Charts

## Bar Charts

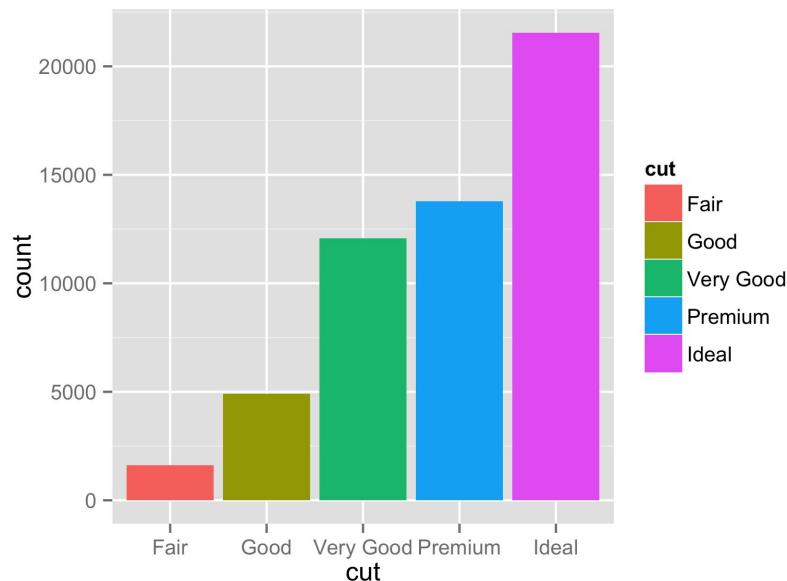
---

What would be a sensible way to display:

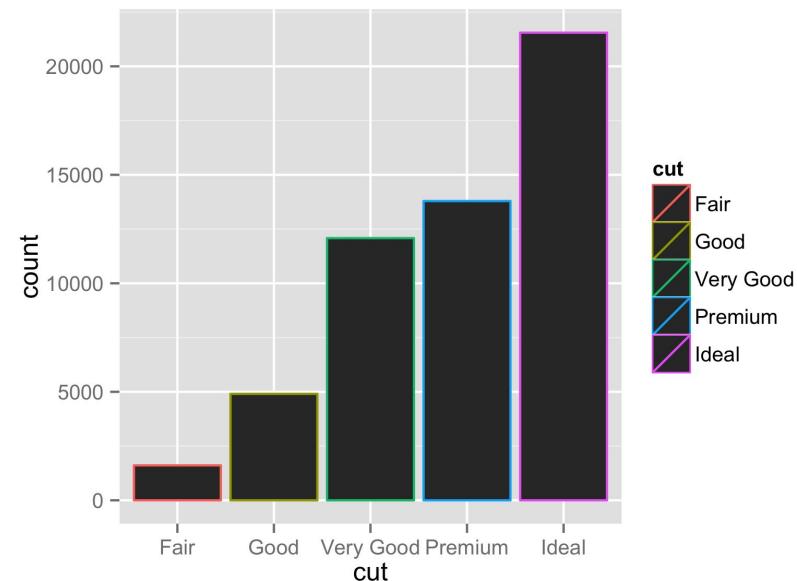
- ❖ The relationship between the x and z variables (two continuous variables)?
- ❖ Just the cut variable (a categorical variable)?
- ❖ Just the x variable (a continuous variable)?

## Bar Charts

```
ggplot(data = diamonds,aes(x = cut)) + geom_bar(aes(fill = cut))
```



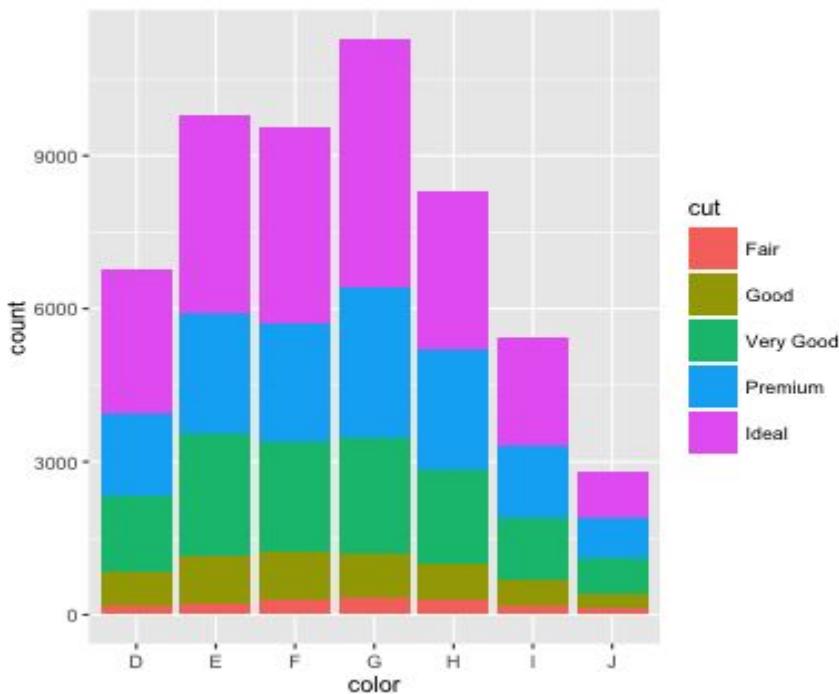
```
ggplot(data = diamonds, aes(x = cut)) + geom_bar(aes(color = cut))
```



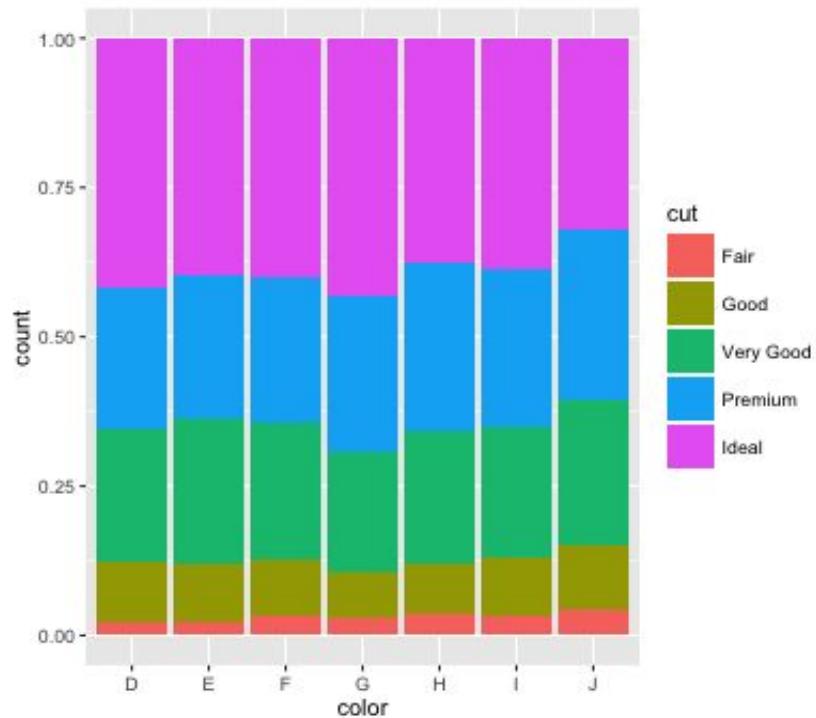
# Position Adjustments

# Position Adjustments

```
ggplot(data = diamonds, aes(x = color)) + geom_bar(aes(fill = cut))
```

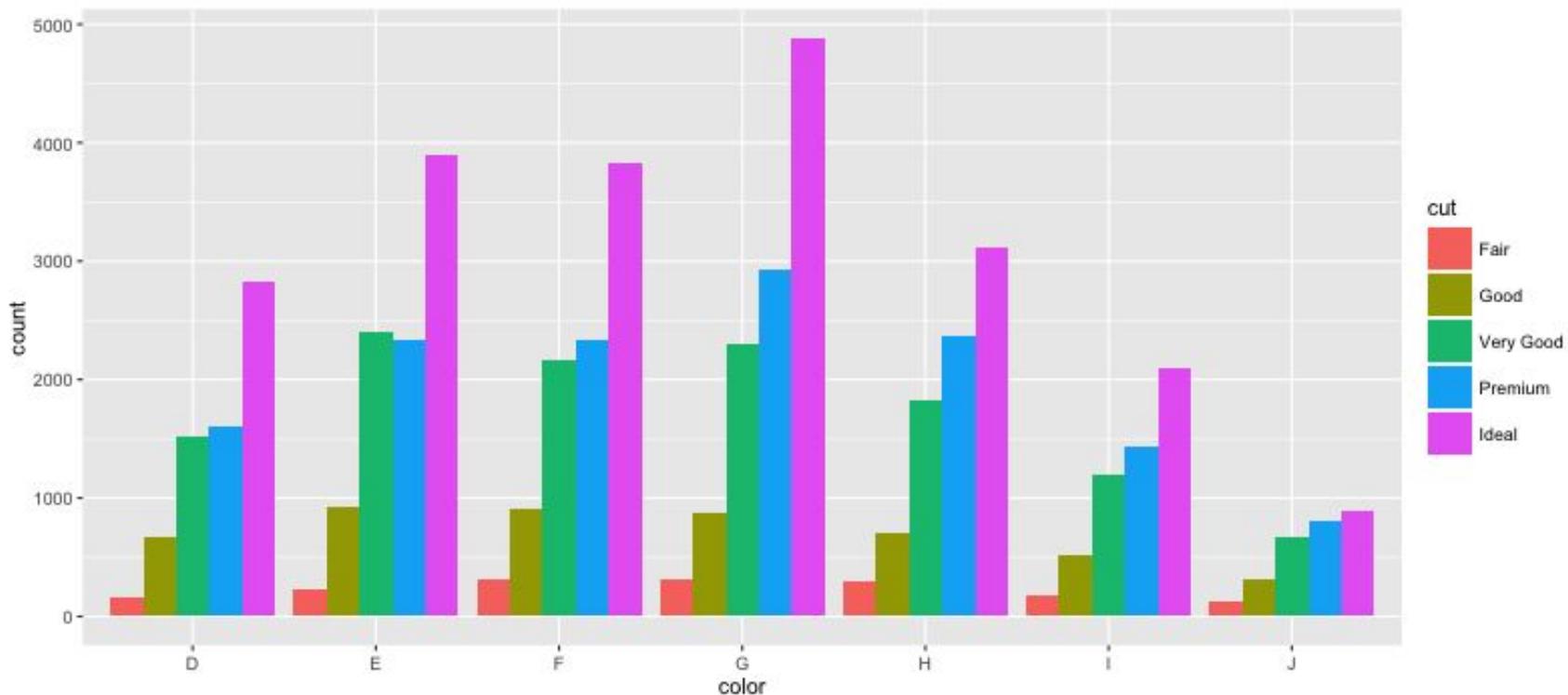


```
ggplot(data = diamonds, aes(x = color)) + geom_bar(aes(fill = cut), position = "fill")
```



# Position Adjustments

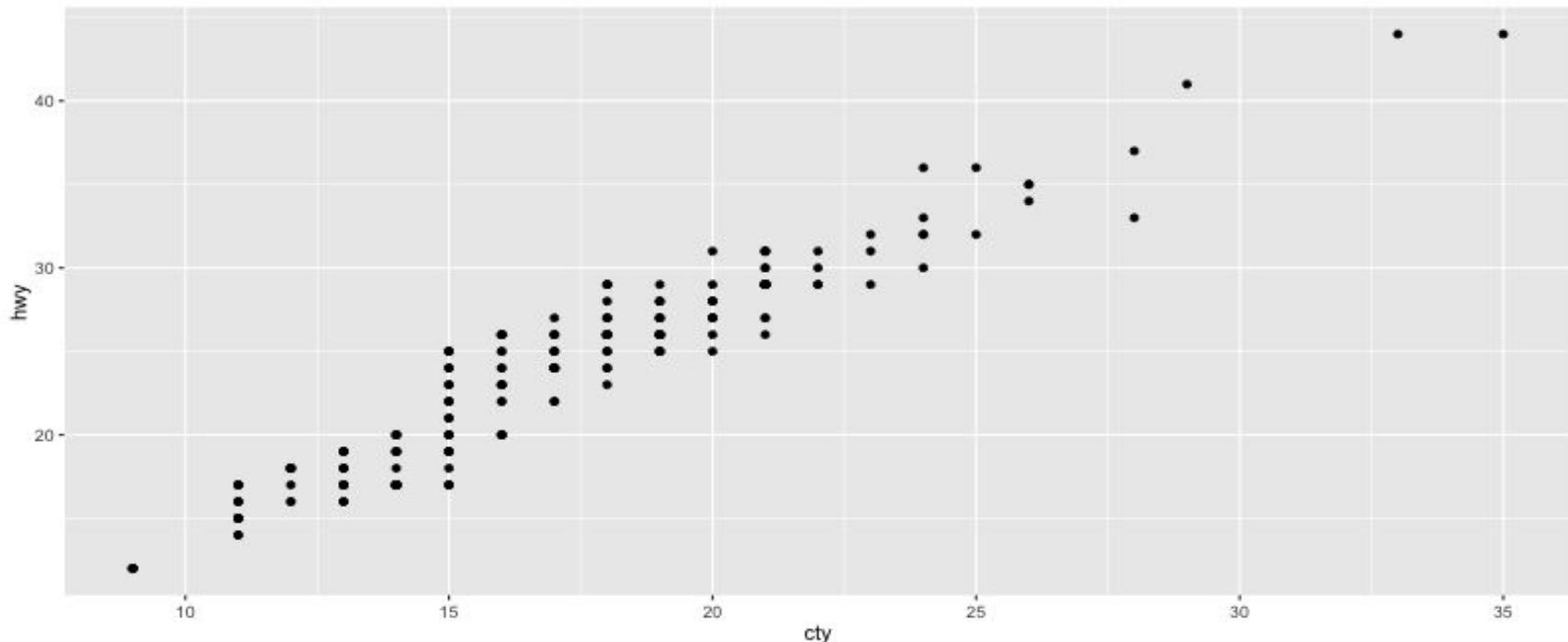
```
ggplot(data = diamonds, aes(x = color)) +  
  geom_bar(aes(fill = cut), position = "dodge")
```



# Position Adjustments

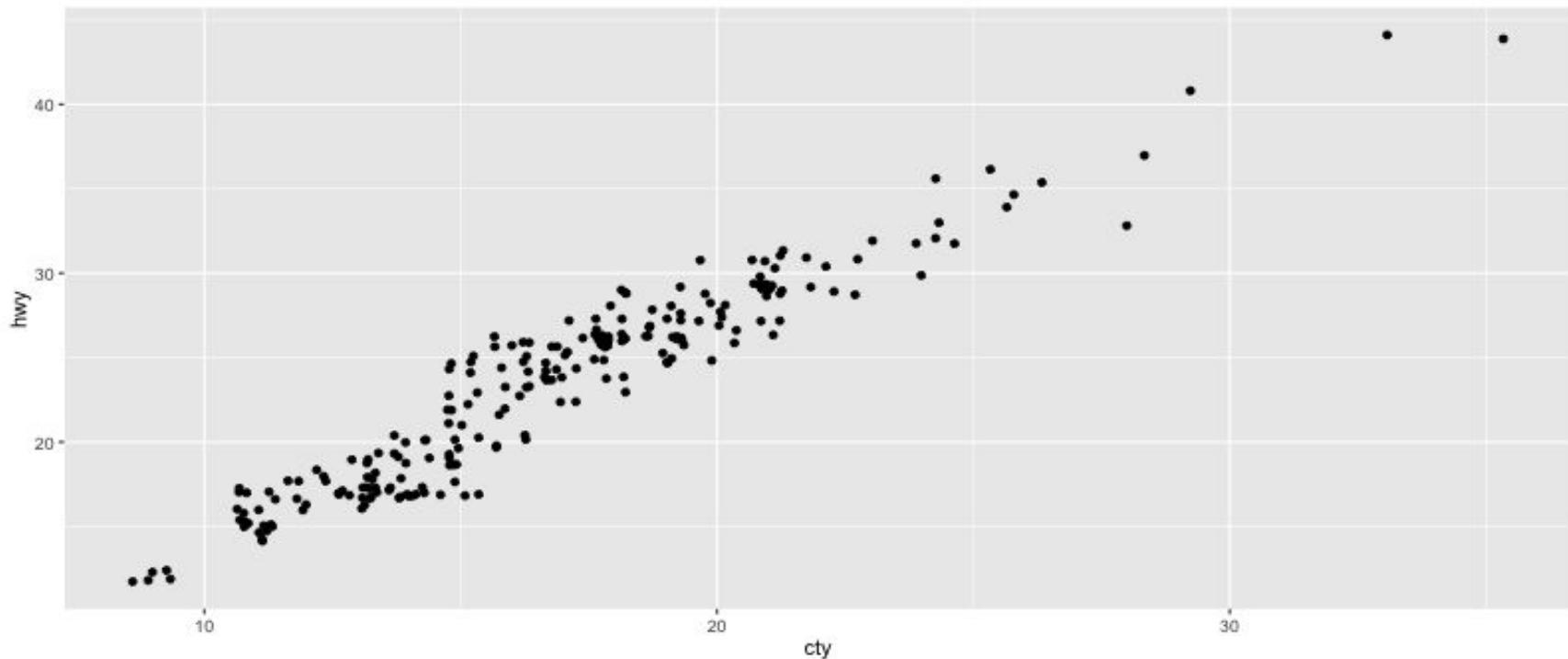
What is strange about this graph?

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



## Position Adjustments

```
ggplot(data = mpg, aes(x = cty, y = hwy)) +  
  geom_point(position = "jitter")
```



# Position Adjustments

---

## 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.
- ❖ It's just a short cut for a point **geom** with a jitter position adjustment.
- ❖ For example, these do the same thing:

```
ggplot(data = mpg, aes(x = cty, y = hwy)) +  
  geom_point(position = "jitter")
```

```
ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_jitter()
```

# Outline

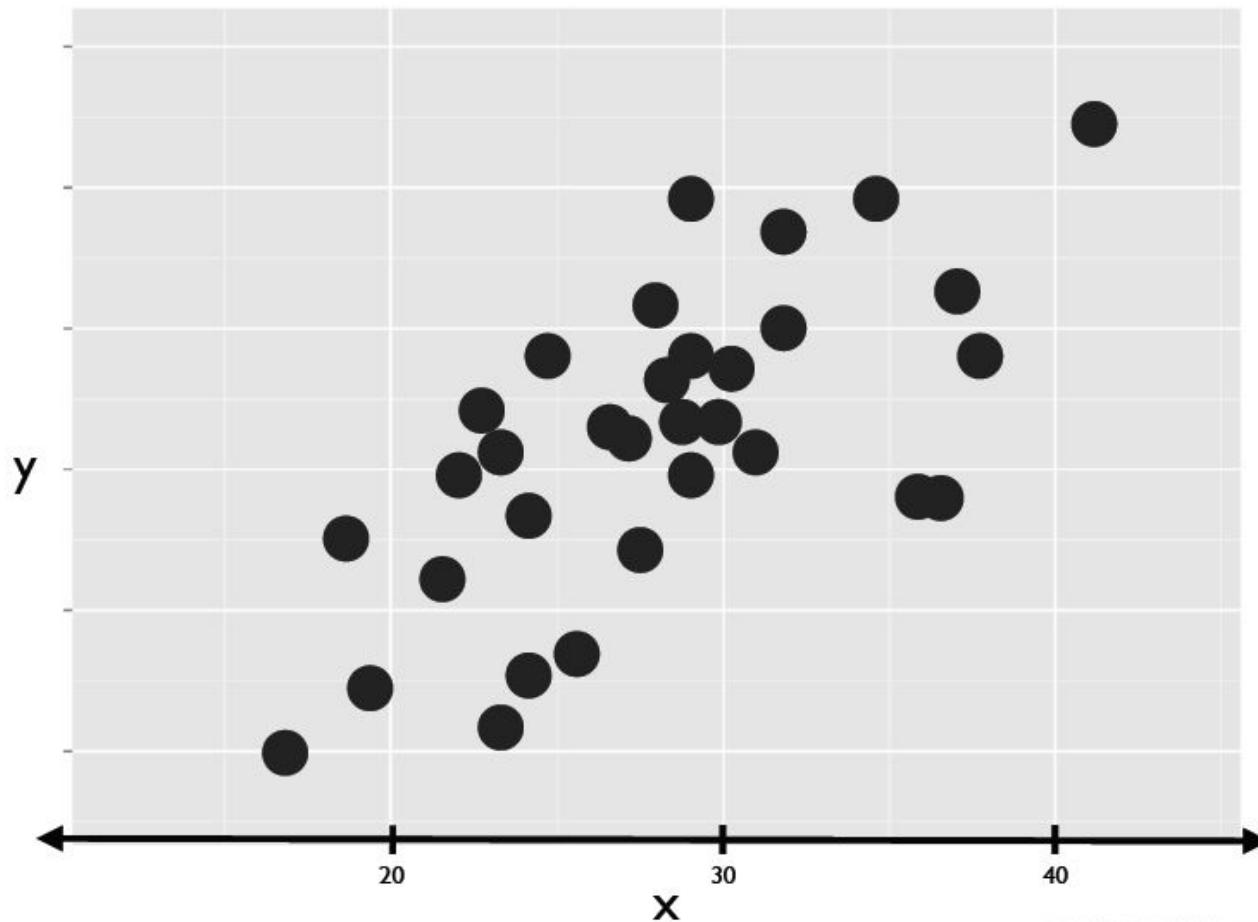
---

- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ Constructing a ggplot2 plot
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving Graphs

# Histograms

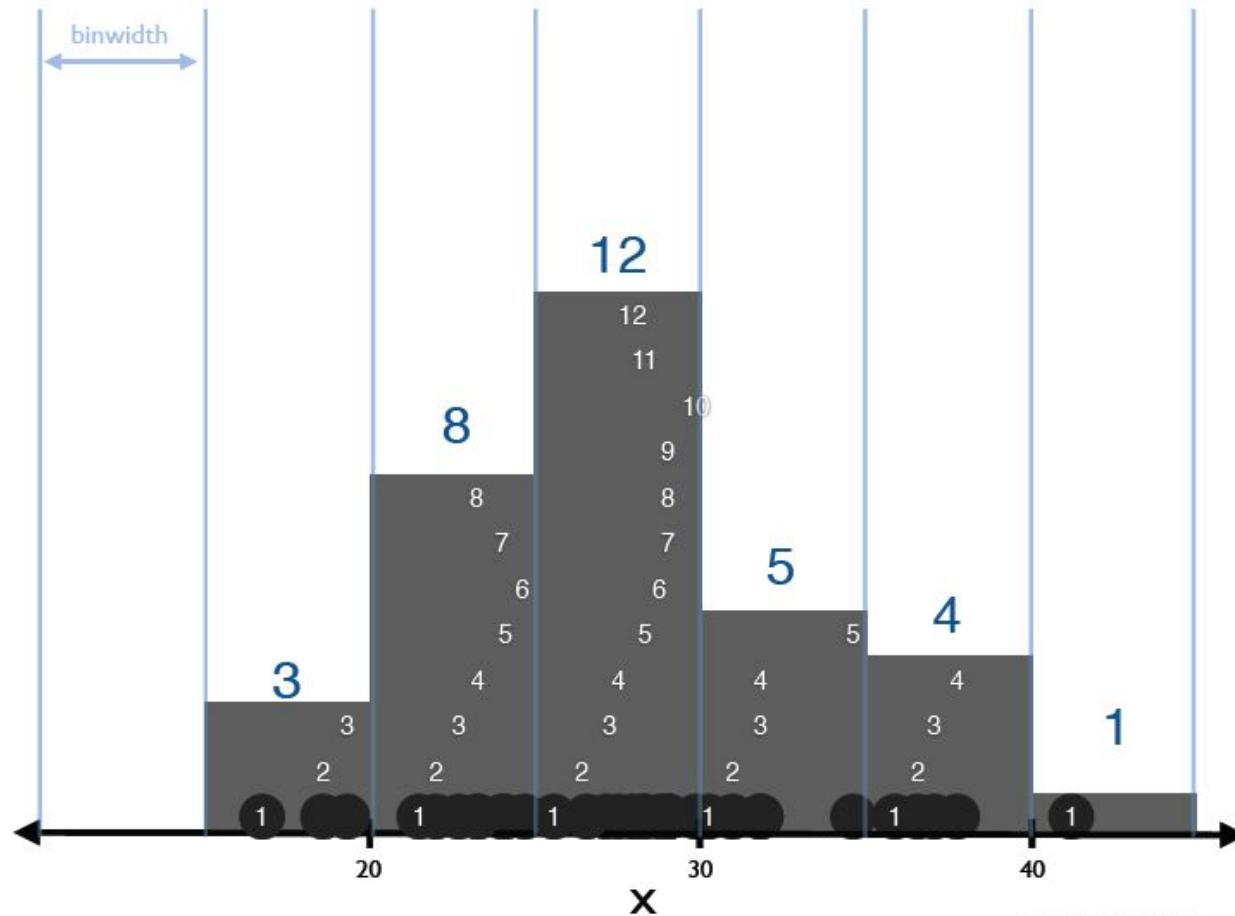
# Histograms

---



© 2014 RStudio, Inc. All rights reserved.

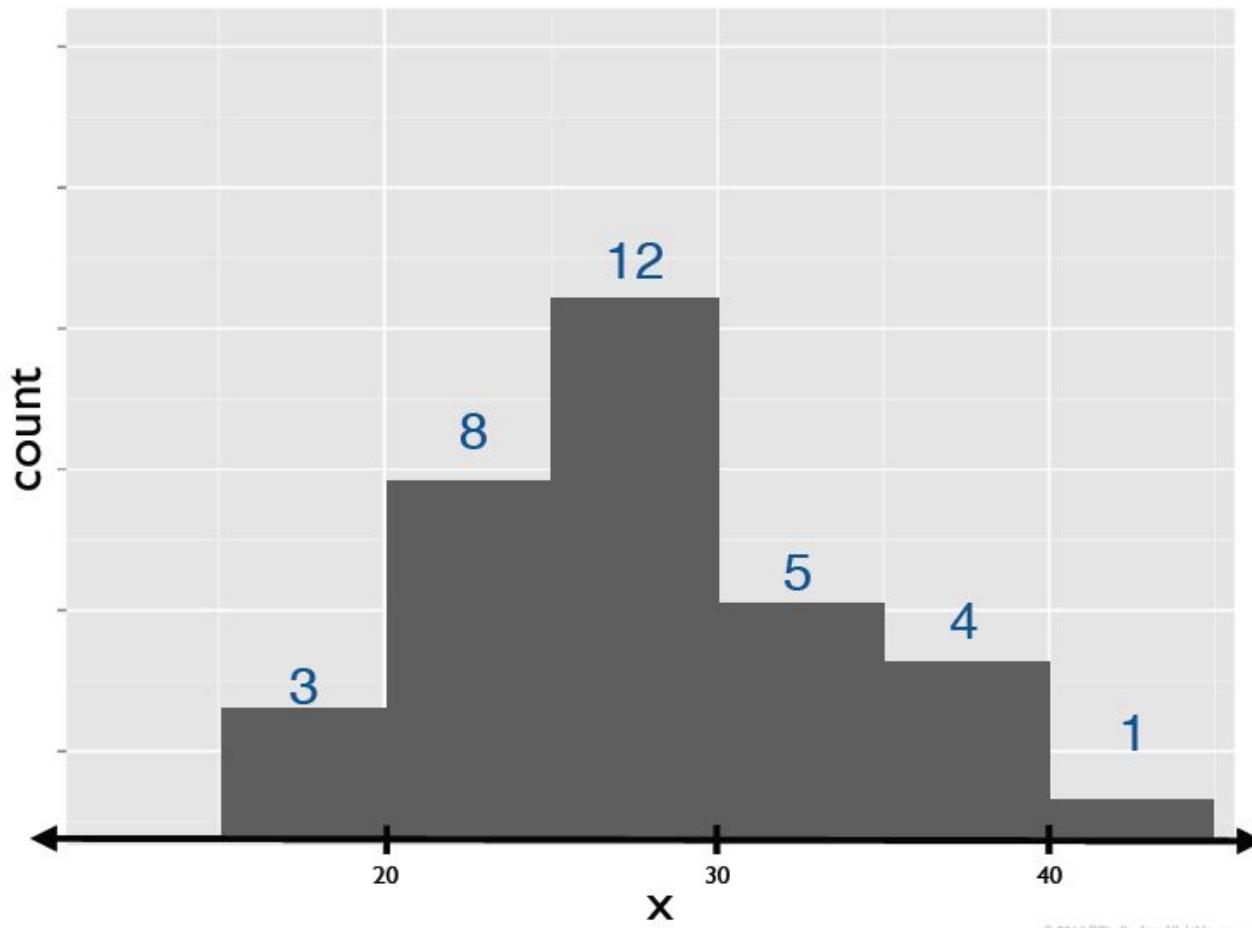
# Histograms



© 2014 RStudio, Inc. All rights reserved.

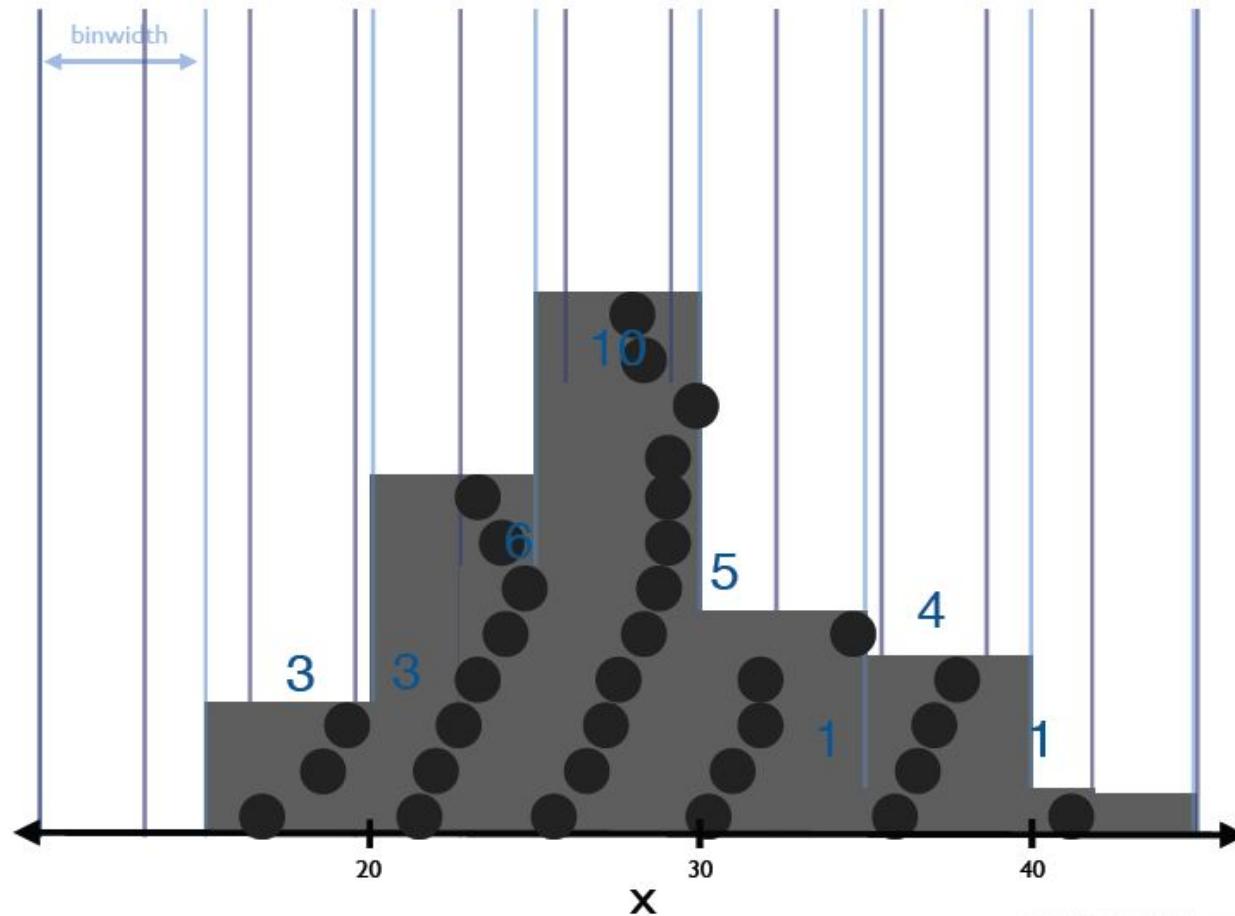
# Histograms

---



© 2014 RStudio, Inc. All rights reserved.

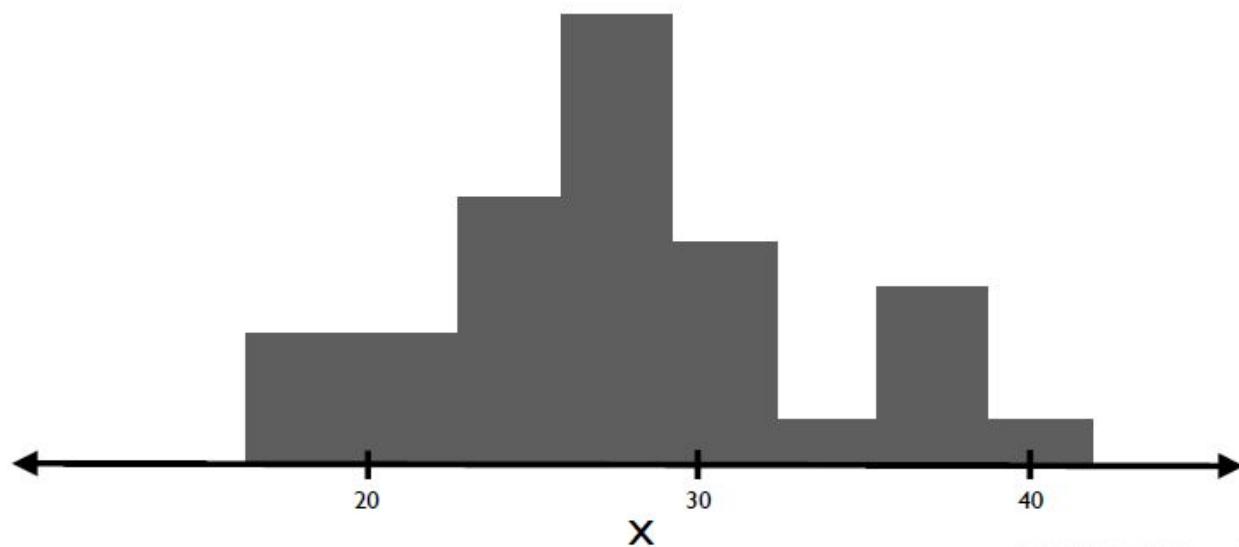
# Histograms



© 2014 RStudio, Inc. All rights reserved.

# Histograms

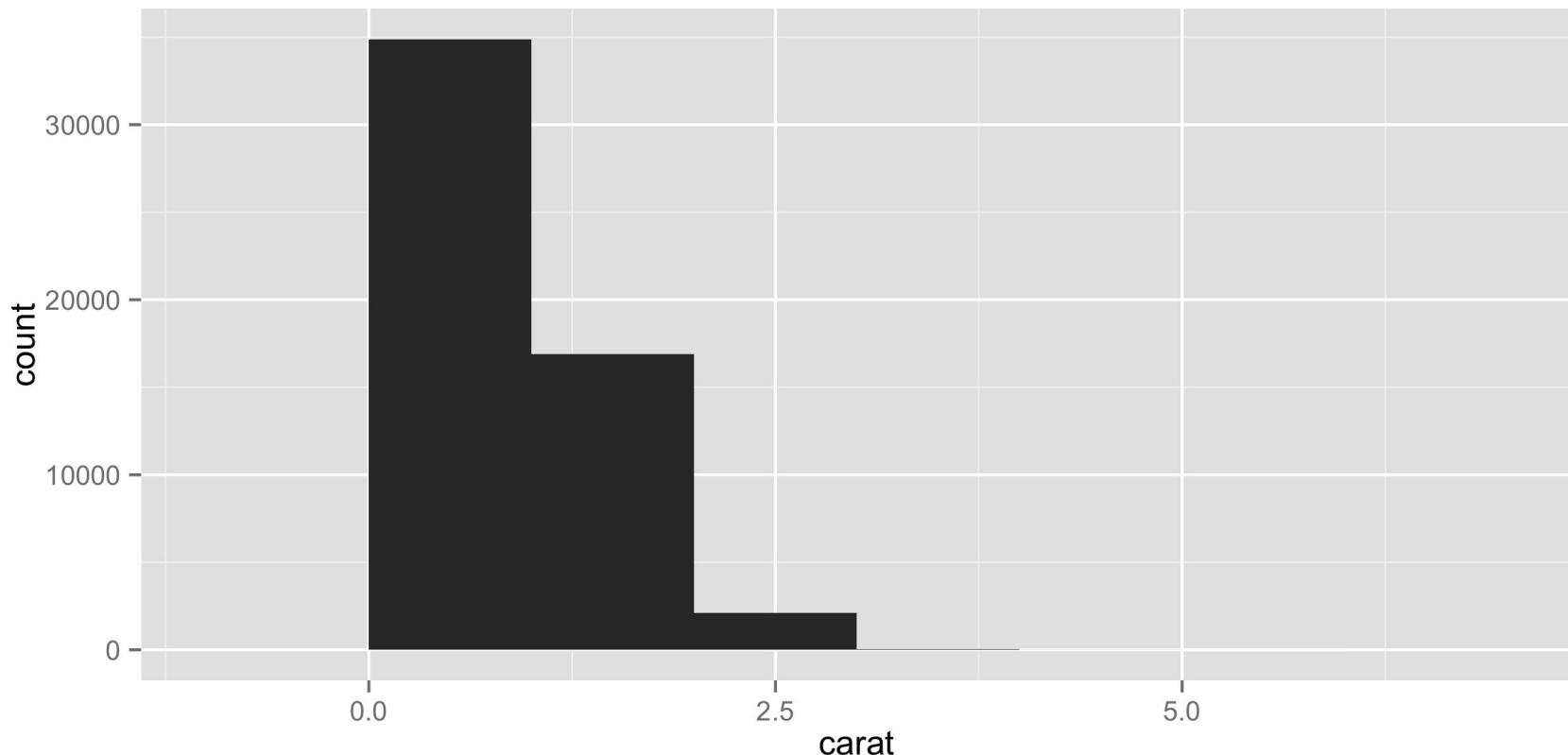
---



© 2014 RStudio, Inc. All rights reserved.

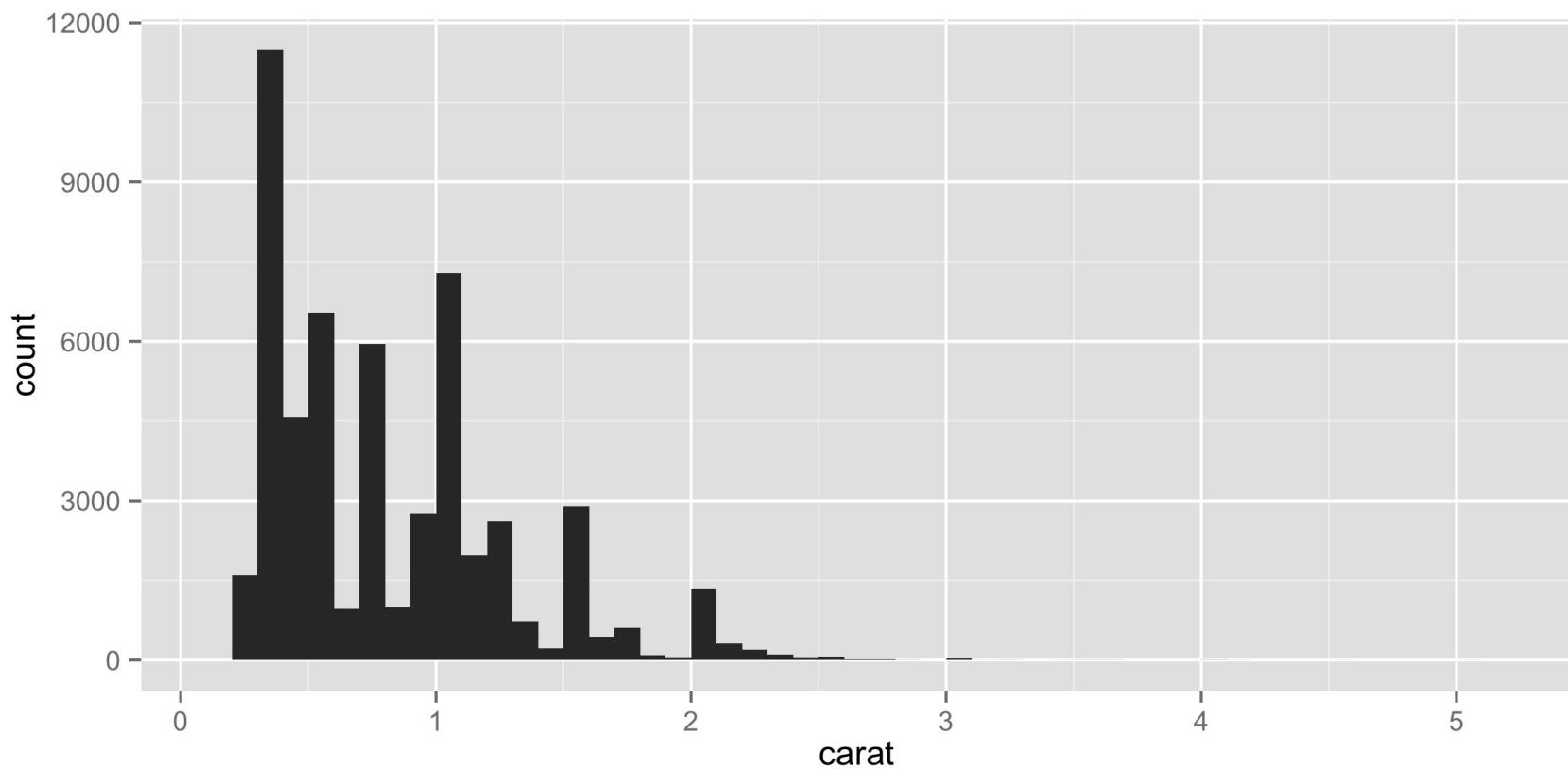
## Histograms

```
g <- ggplot(data = diamonds, aes(x = carat))  
g + geom_histogram(binwidth = 1)
```



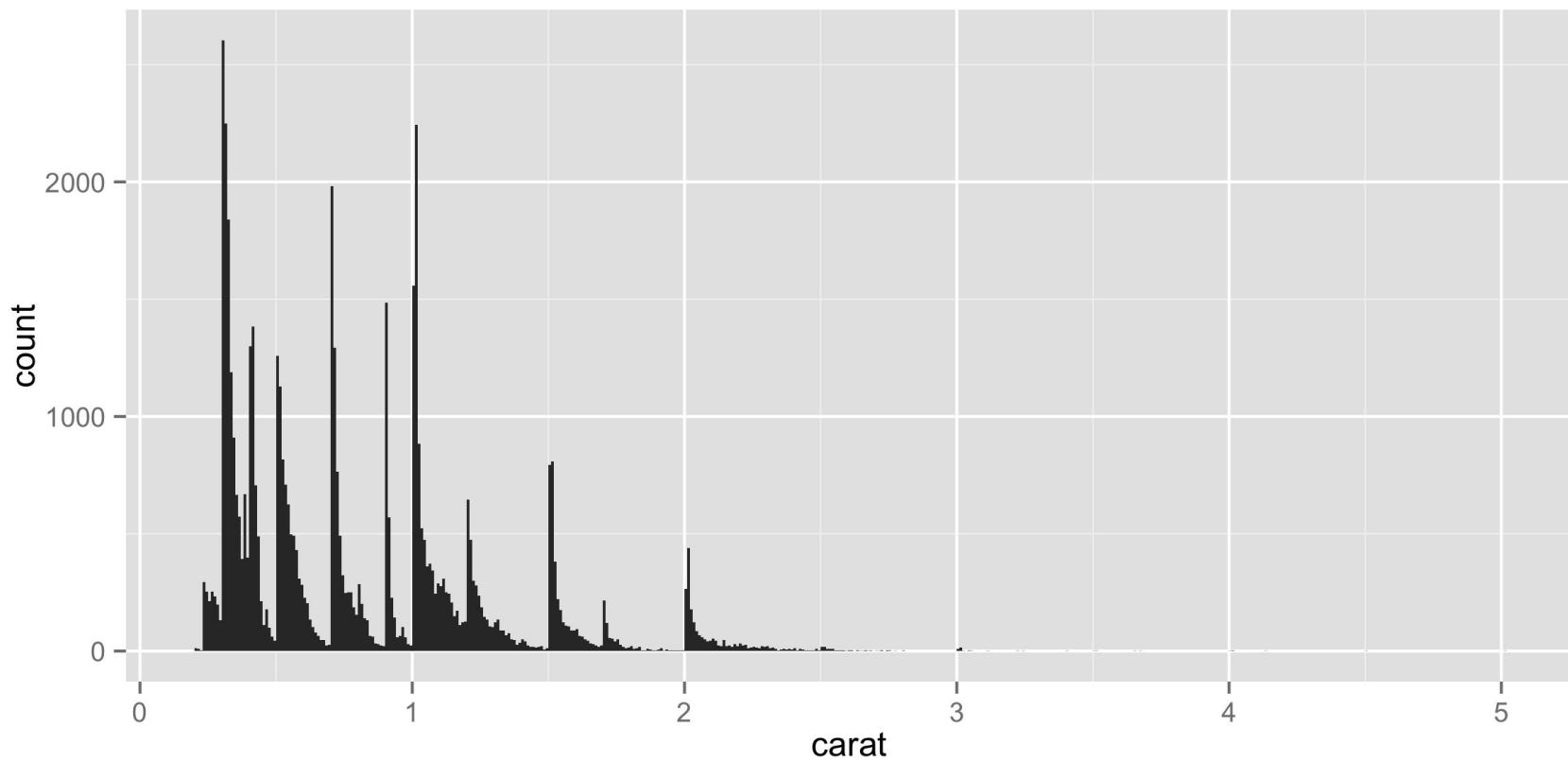
# Histograms

```
g + geom_histogram(binwidth = 0.1)
```



## Histograms

```
g + geom_histogram(binwidth = 0.01)
```

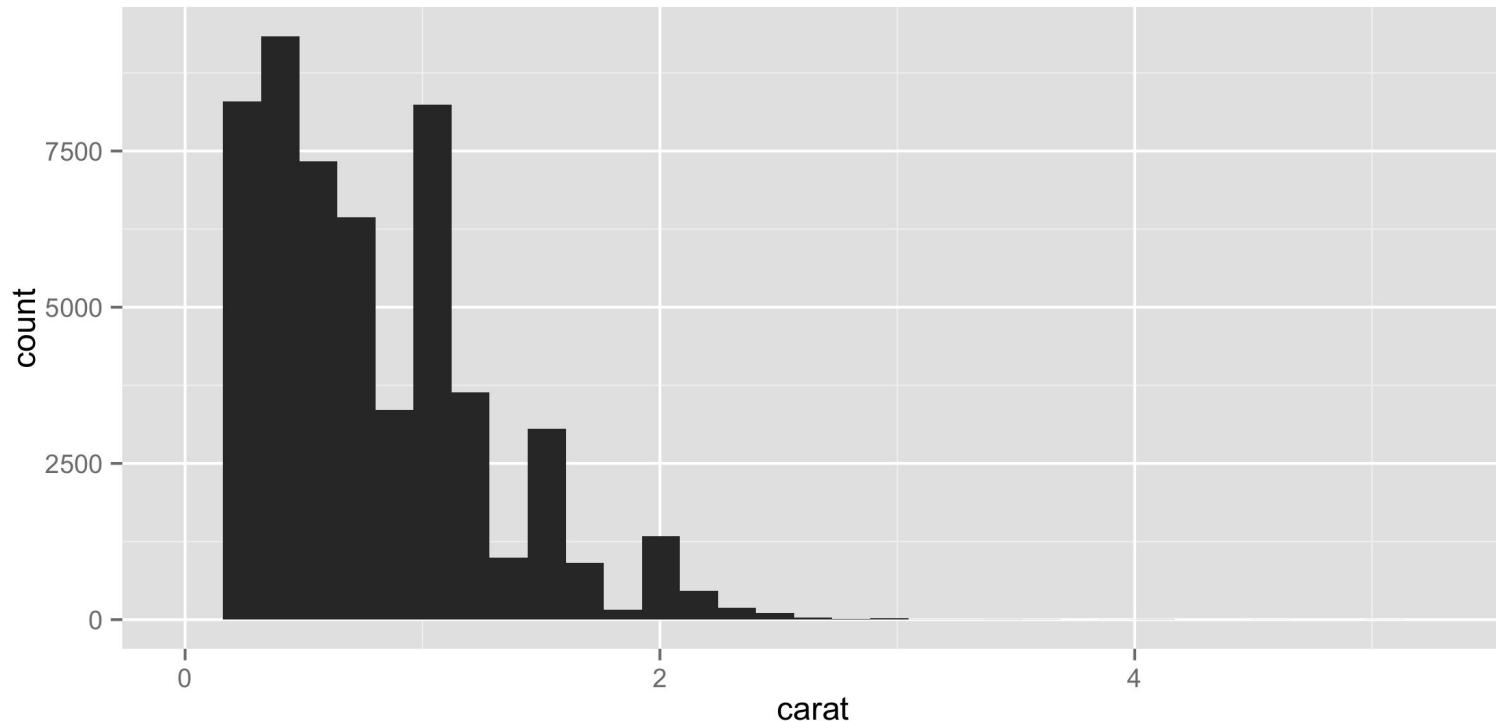


# Histograms

---

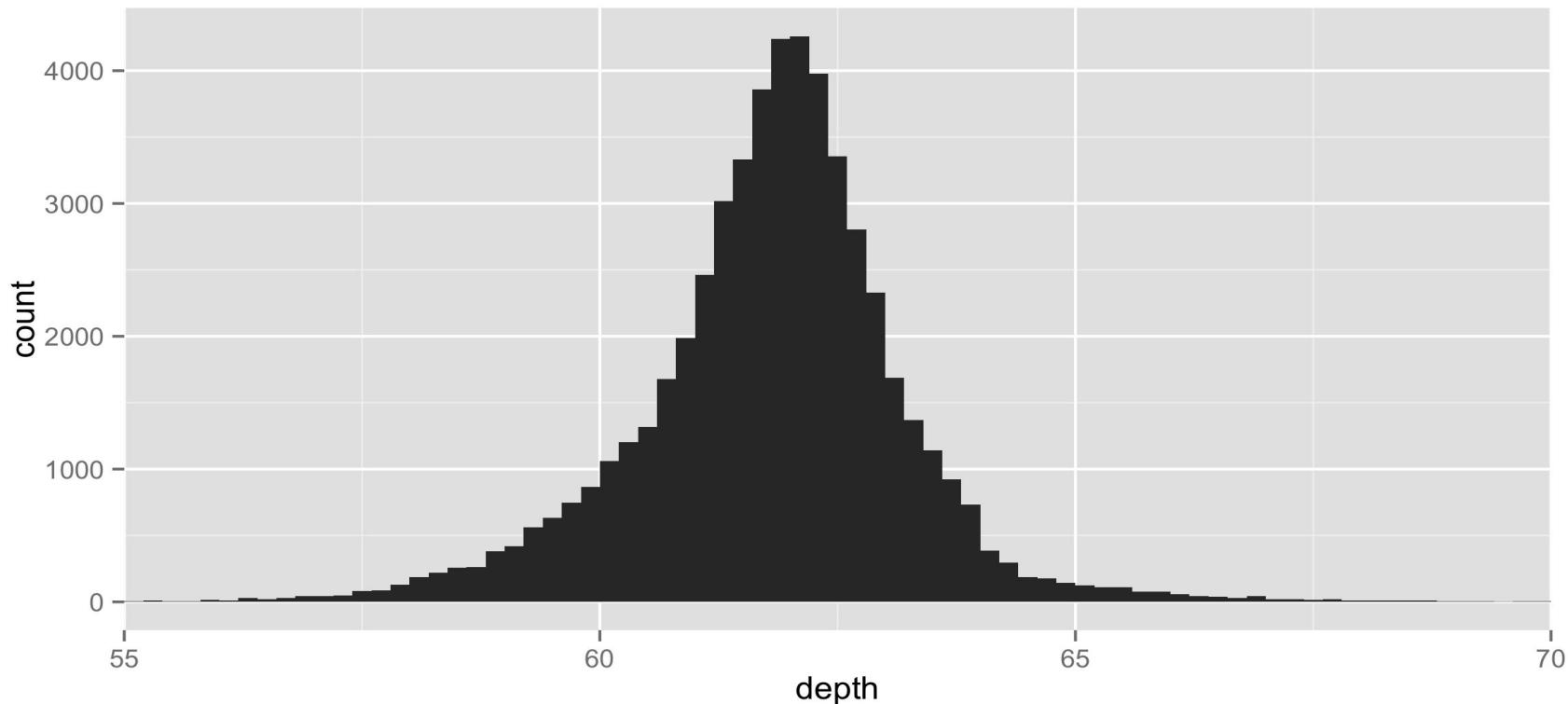
Most parameters come with a preset [default value](#).

```
#stat_bin: binwidth defaulted to range/30.  
g + geom_histogram()
```



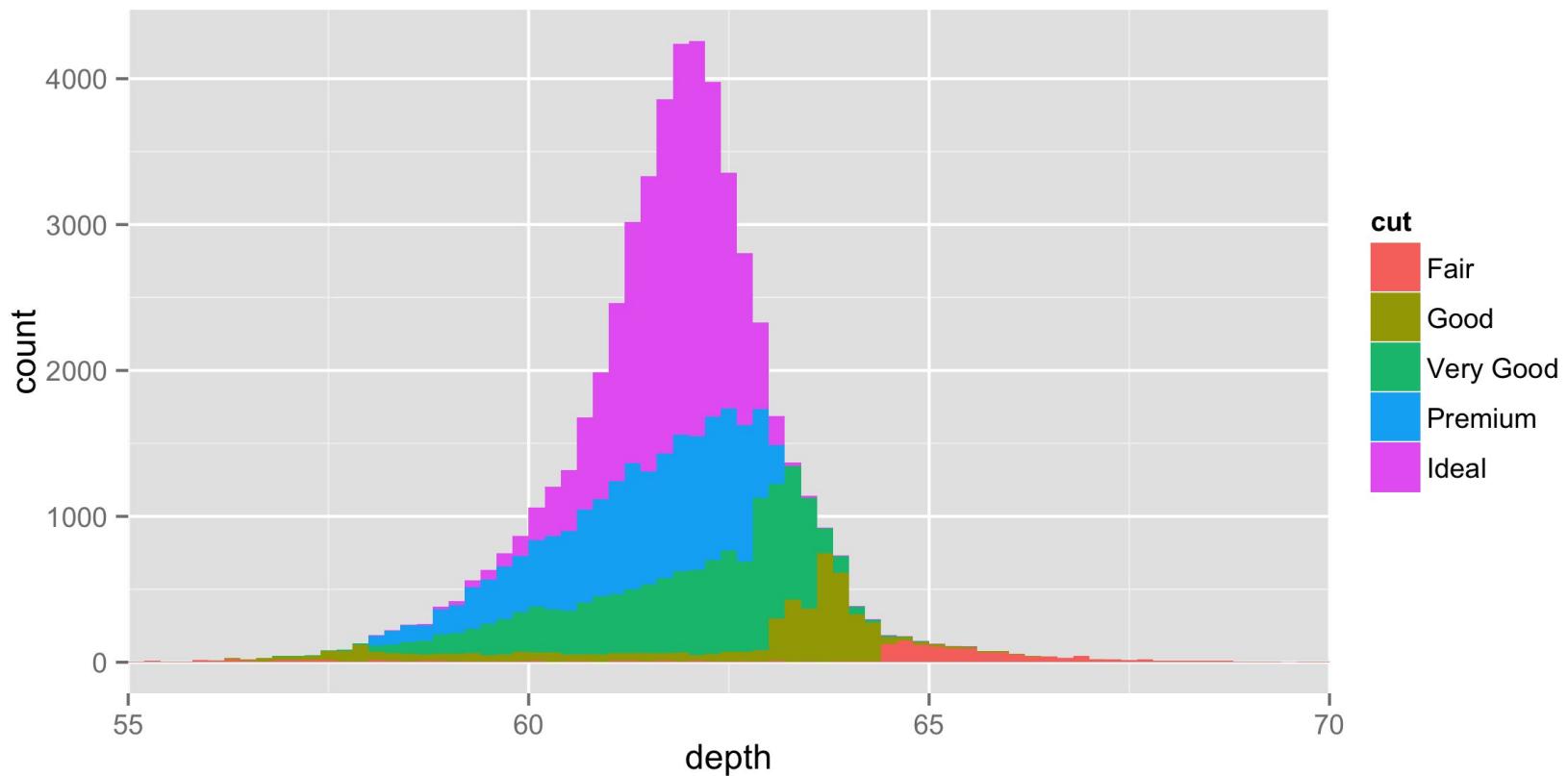
## Histograms

```
g <- ggplot(data = diamonds, aes(x = depth))  
zoom <- coord_cartesian(xlim = c(55, 70))  
g + geom_histogram(binwidth = 0.2) + zoom
```



# Histograms

```
g + geom_histogram(aes(fill = cut), binwidth = 0.2) + zoom
```



# Comparing Variables

## Comparing Variables

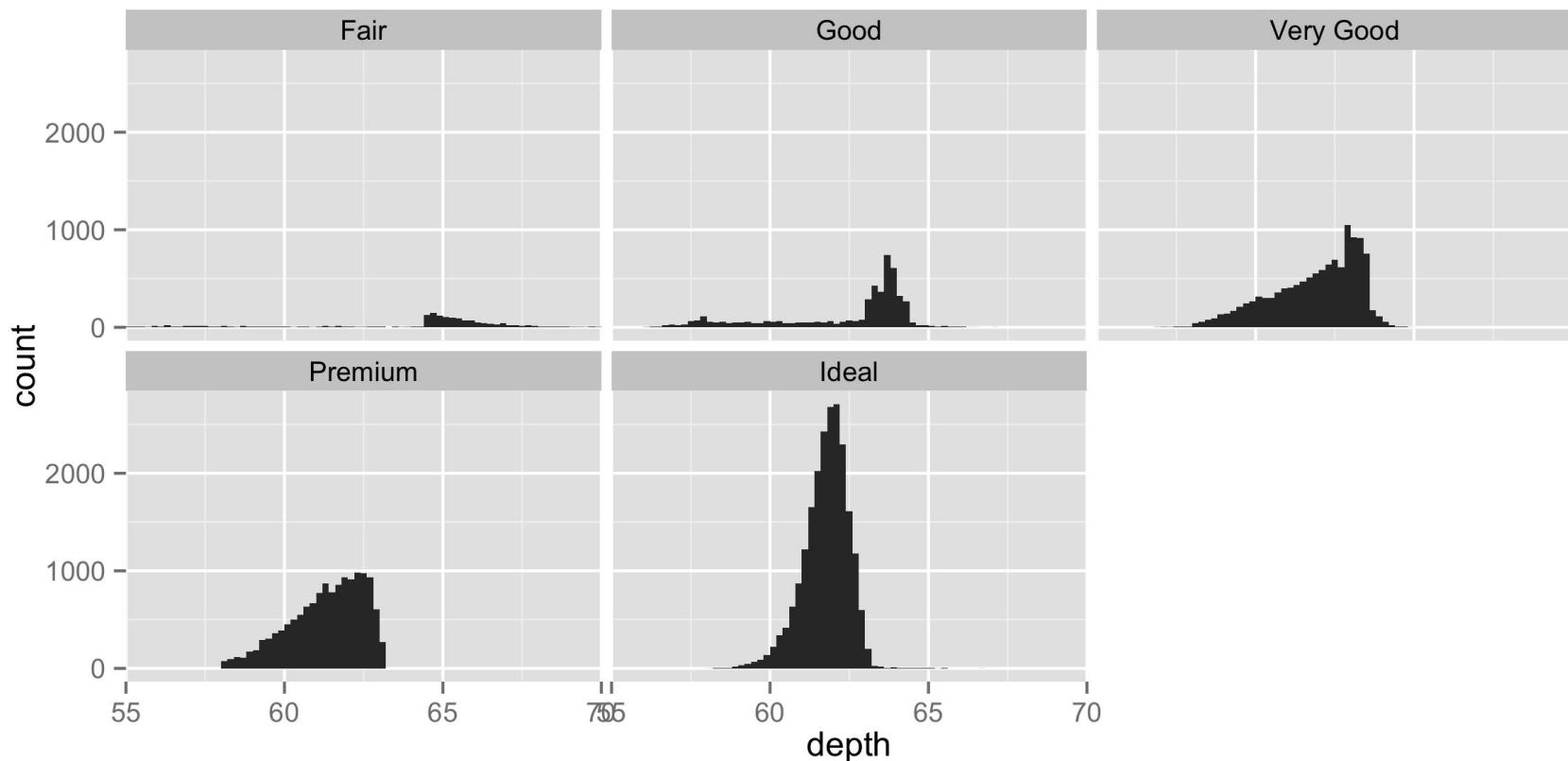
---

Additional variables are sometimes hard to compare if they are separated into facets because the shape for smaller groups can be diminished or overshadowed by larger groups.

```
g <- ggplot( data = diamonds, aes(x = depth))  
zoom <- coord_cartesian(xlim = c(55, 70))  
g + geom_histogram(binwidth = 0.2) + facet_wrap(~ cut) + zoom
```

## Comparing Variables

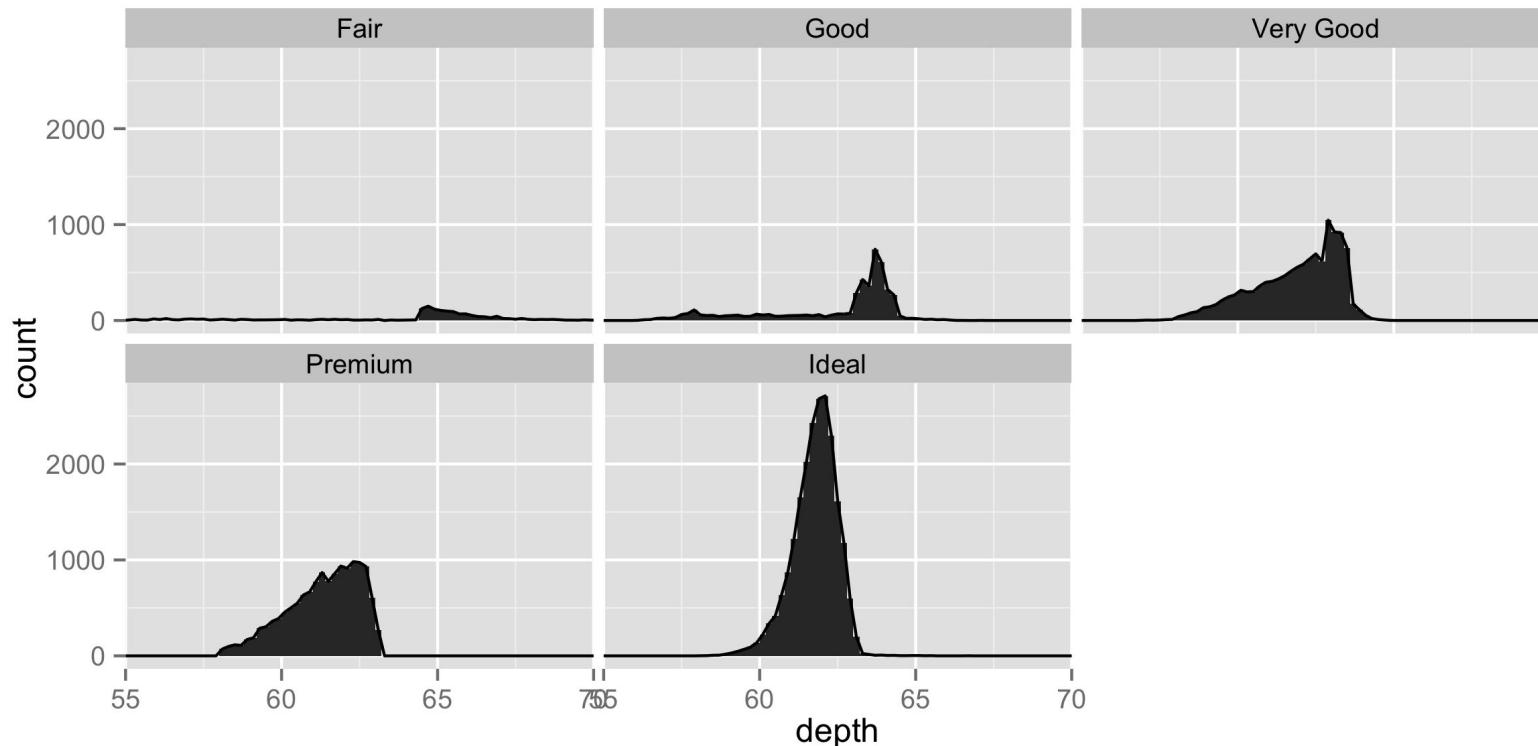
```
g + geom_histogram(binwidth = 0.2) + facet_wrap(~ cut) + zoom
```



## Comparing Variables

What if we just drew a line along the tops of the histograms?

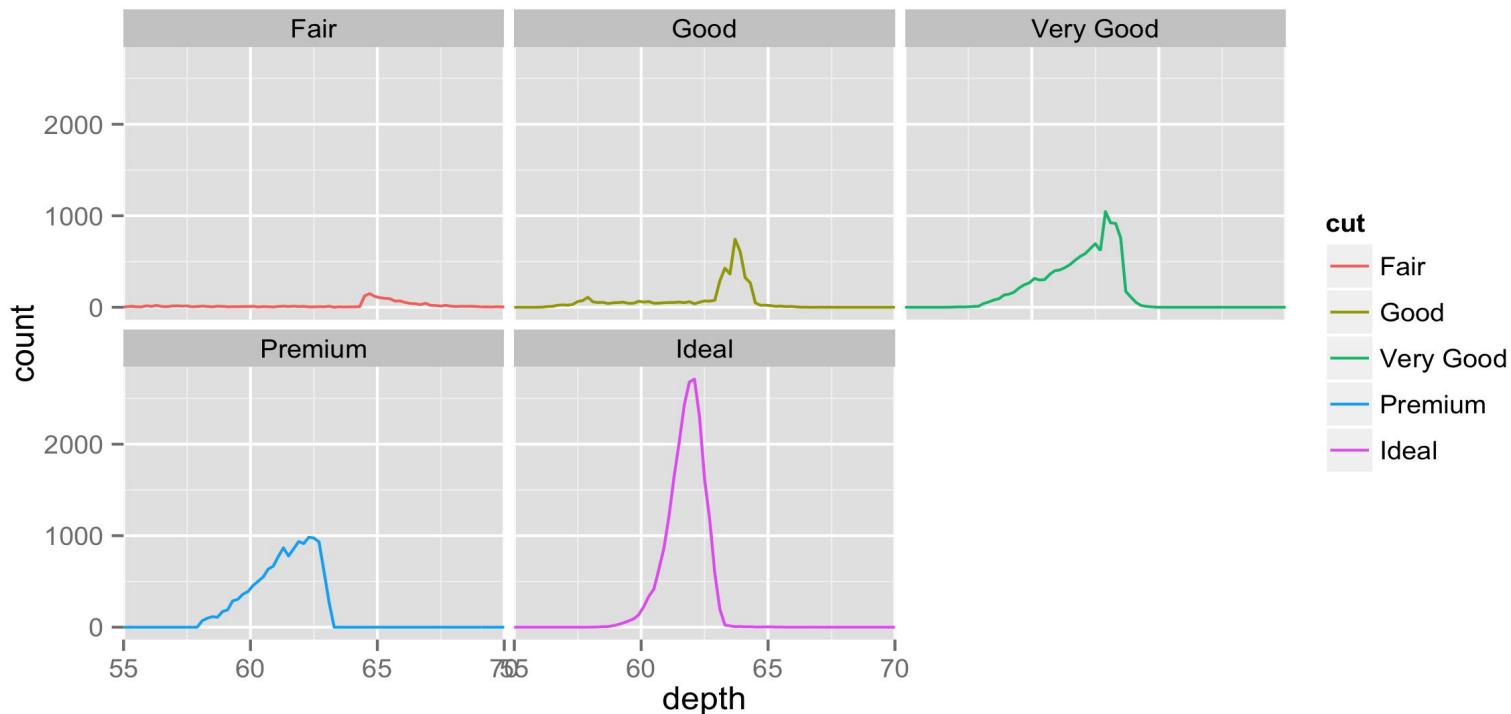
```
g + geom_histogram(binwidth = 0.2) + geom_freqpoly(binwidth=0.2)  
+ facet_wrap( ~ cut) + zoom
```



# Comparing Variables

What if we threw away the bars?

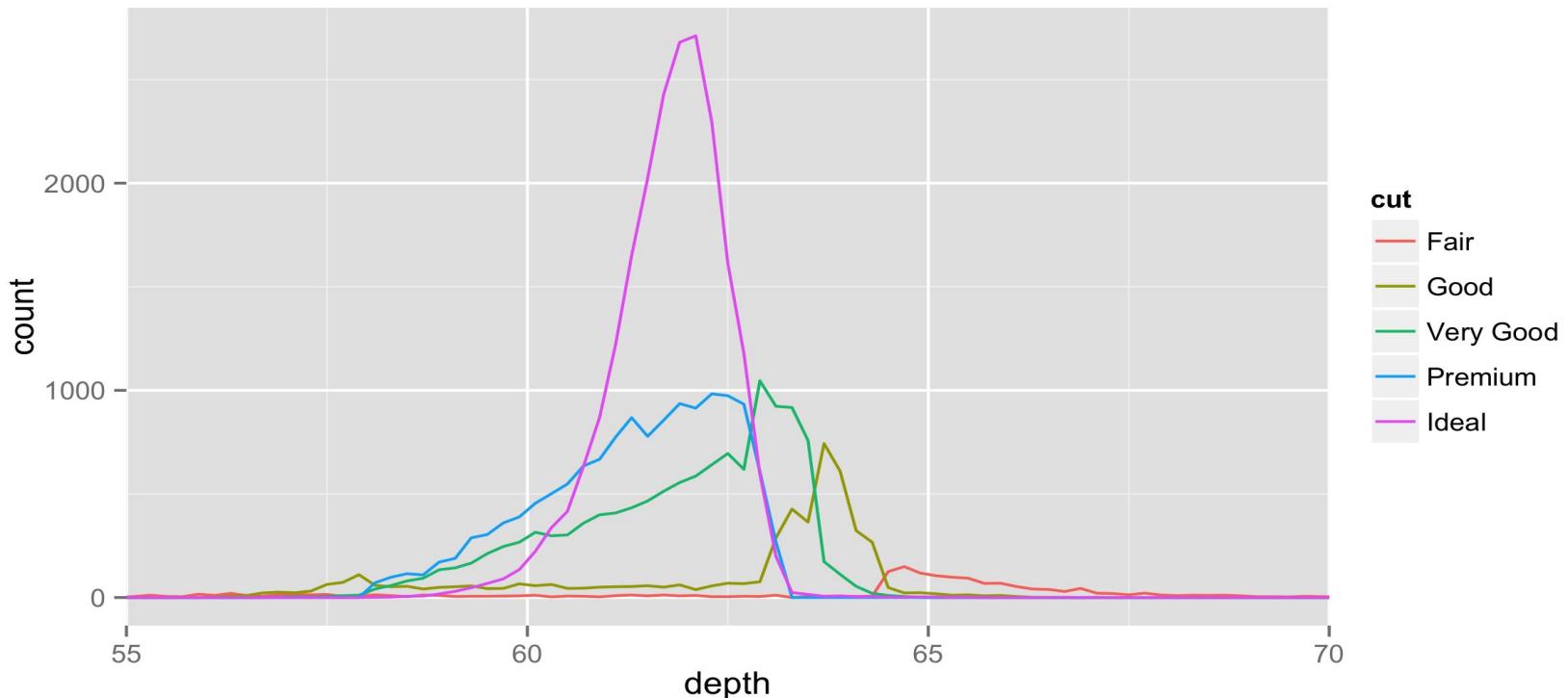
```
g + geom_freqpoly(aes(color = cut), binwidth = 0.2) +  
  facet_wrap( ~ cut) + zoom
```



# Comparing Variables

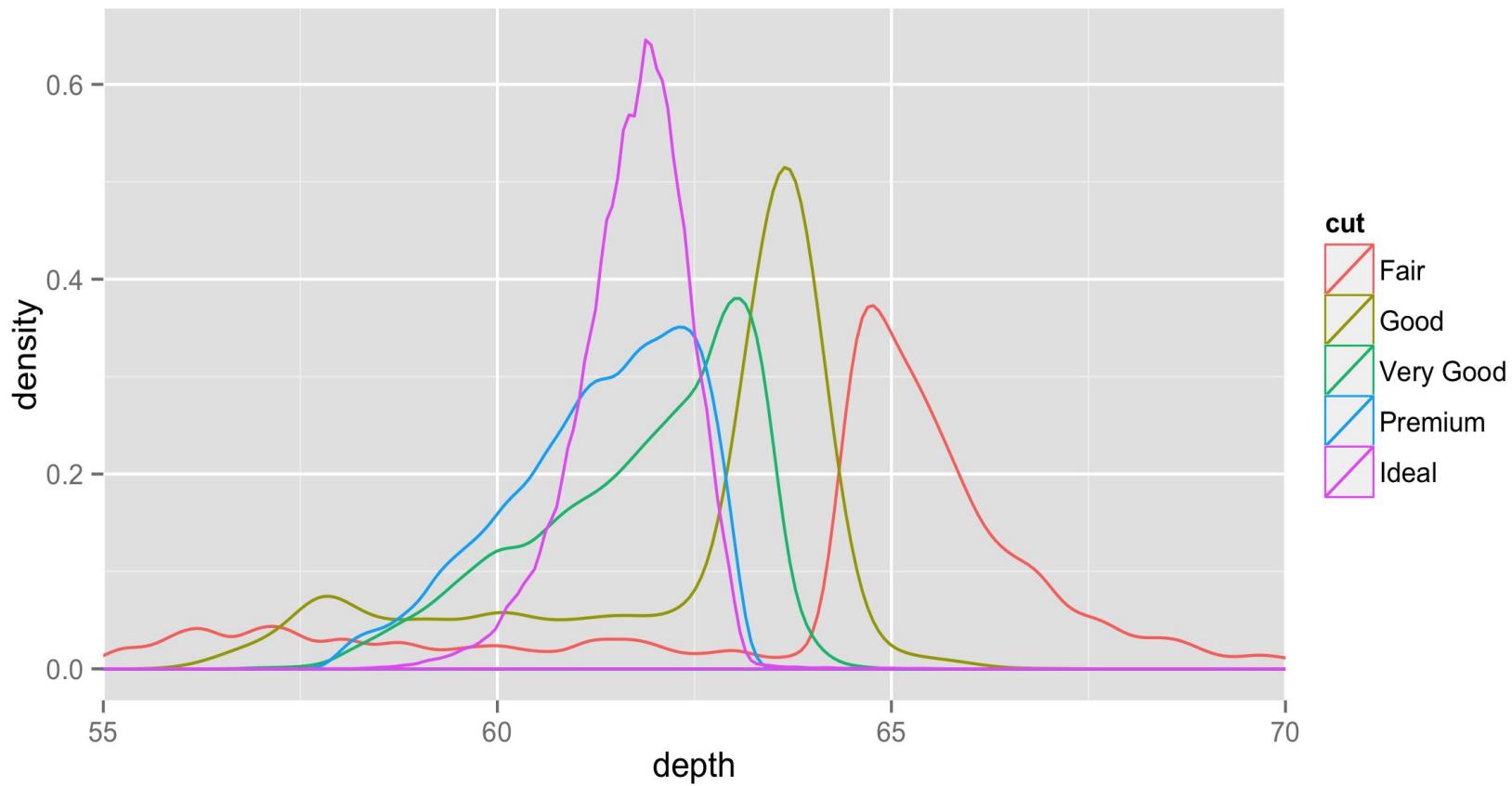
What if we just drew each on the same plot?

```
g + geom_freqpoly(aes(color = cut), binwidth = 0.2) + zoom
```



# Comparing Variables

```
g + geom_density(aes(color = cut)) + zoom
```



## Comparing Variables

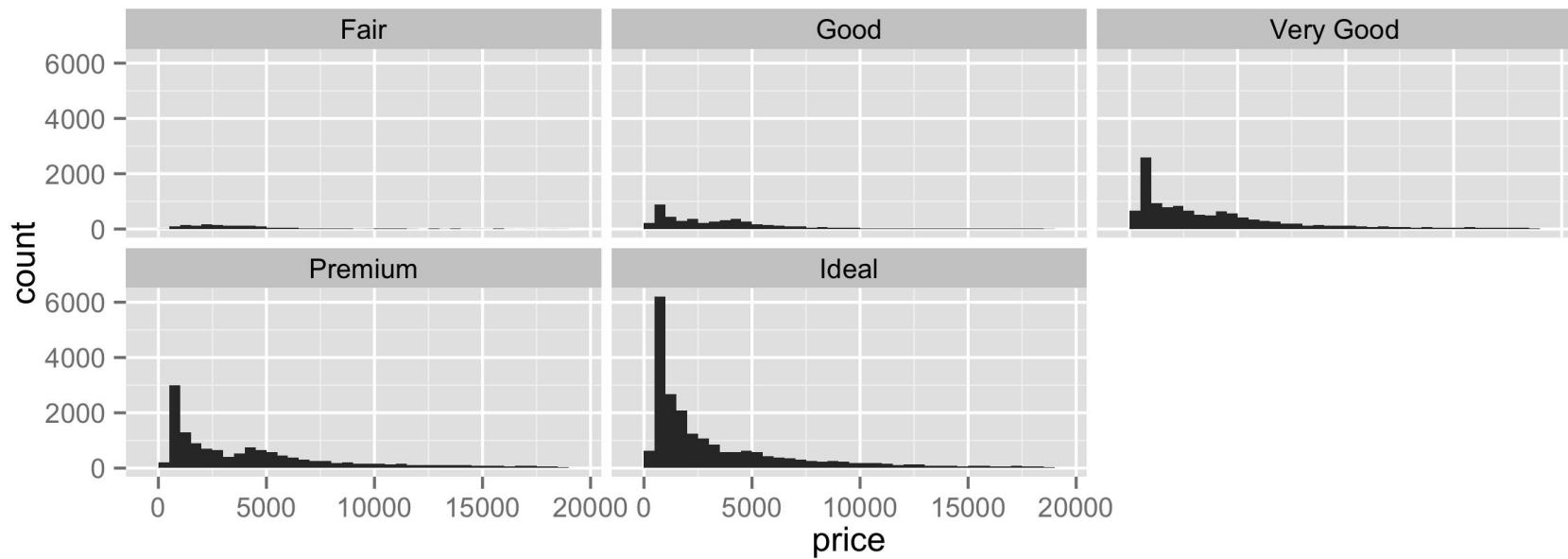
---

- ❖ Compare the distribution of prices for different cuts.
- ❖ Does anything seem unusual?

# Comparing Variables

Attempt #1: Faceting makes comparison hard in this case.

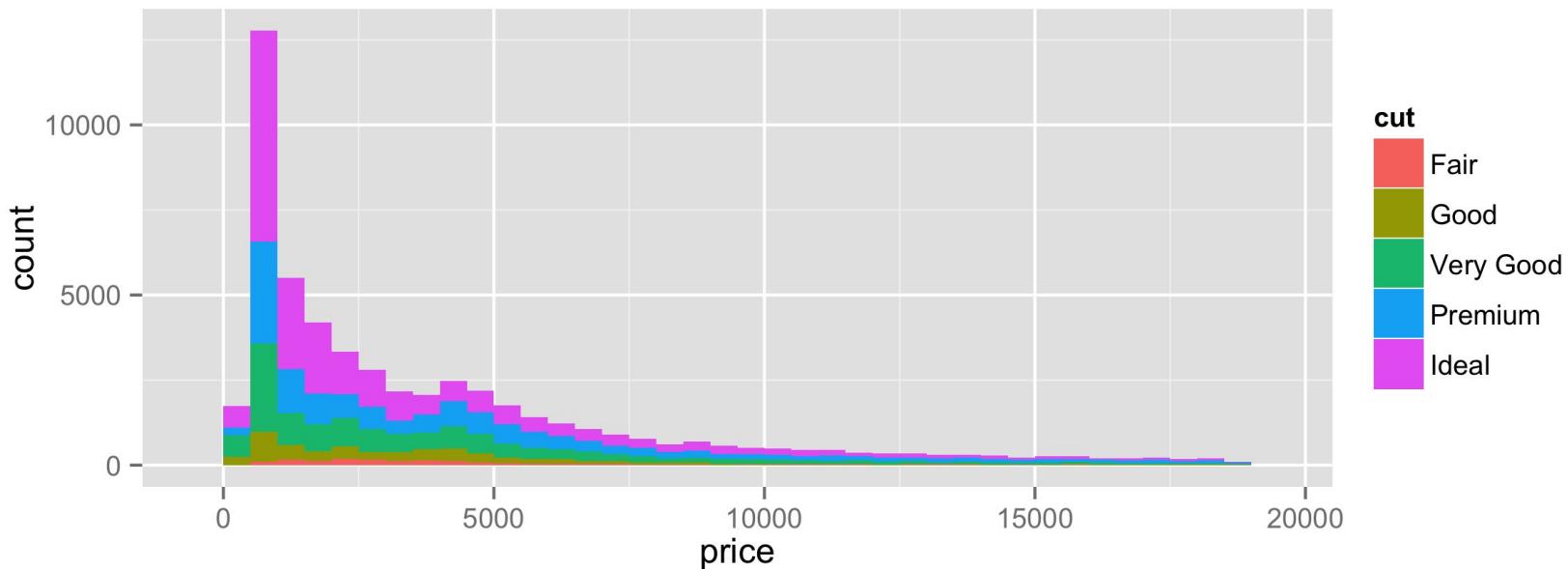
```
ggplot(data = diamonds, aes(x = price)) + geom_histogram  
(binwidth = 500) + facet_wrap(~ cut)
```



## Comparing Variables

Attempt #2: Stacked heights are also hard to compare.

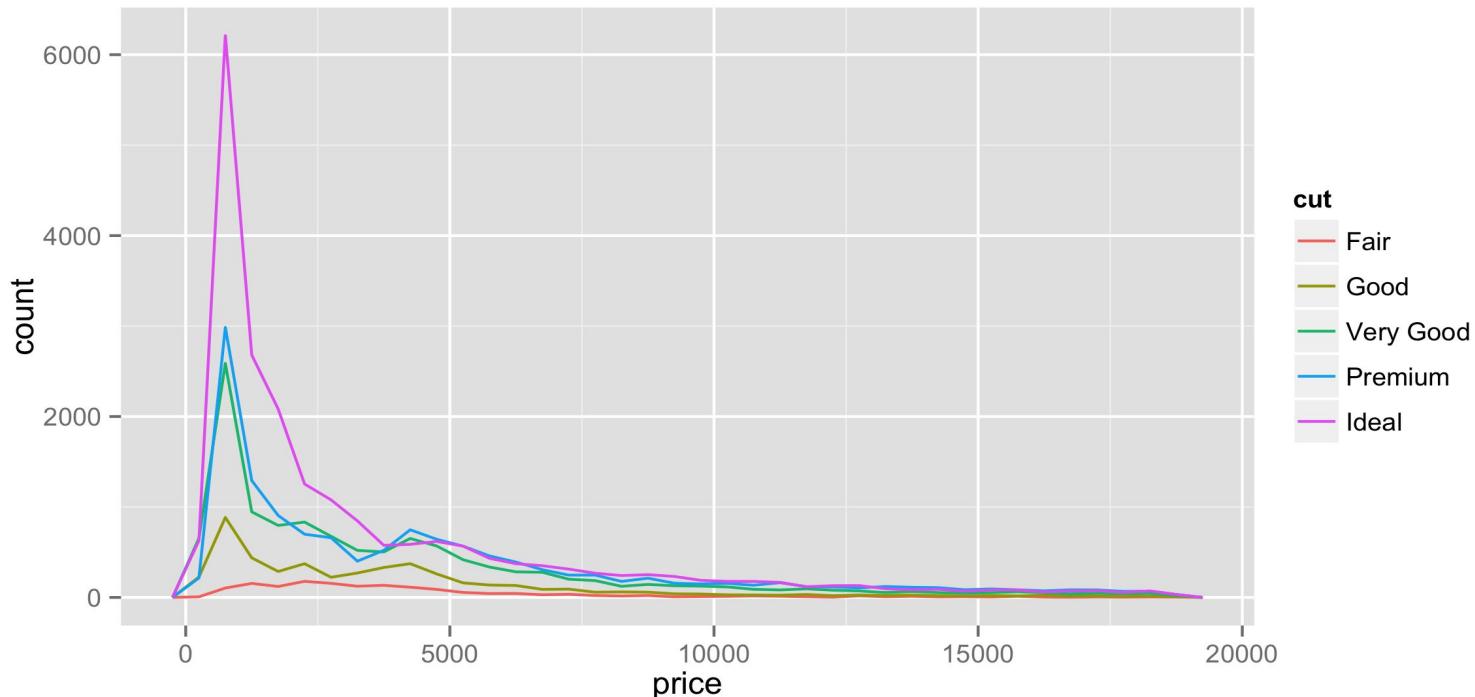
```
ggplot(data = diamonds, aes(x = price)) + geom_histogram(aes(fill = cut))
```



## Comparing Variables

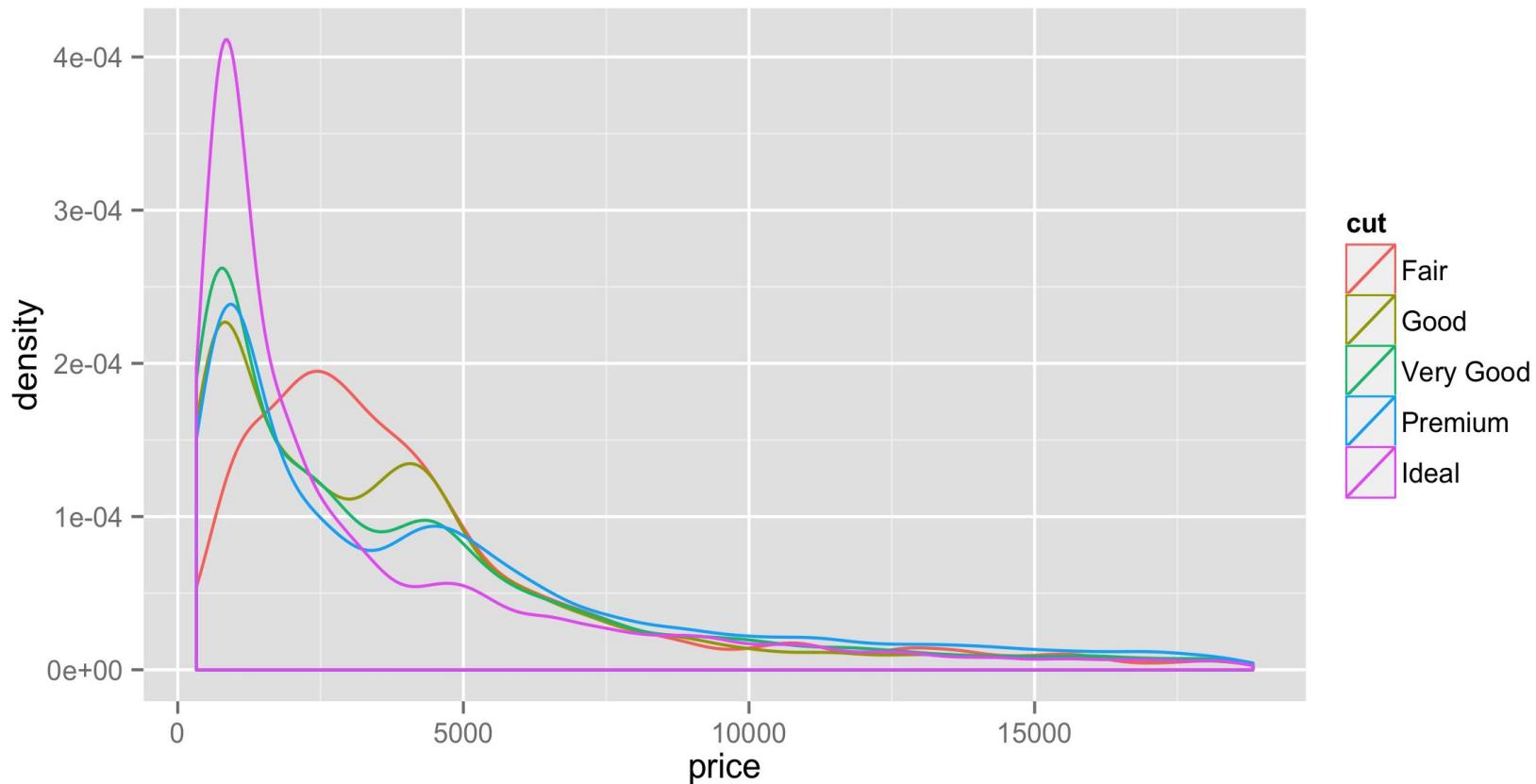
Attempt #3: Better, but categories still have differing relative frequencies.

```
ggplot(data = diamonds, aes(x = price)) + geom_freqpoly(aes  
(color = cut), binwidth = 500)
```



## Comparing Variables

```
ggplot(data = diamonds, aes(x = price)) + geom_density(aes(color = cut))
```



# Outline

---

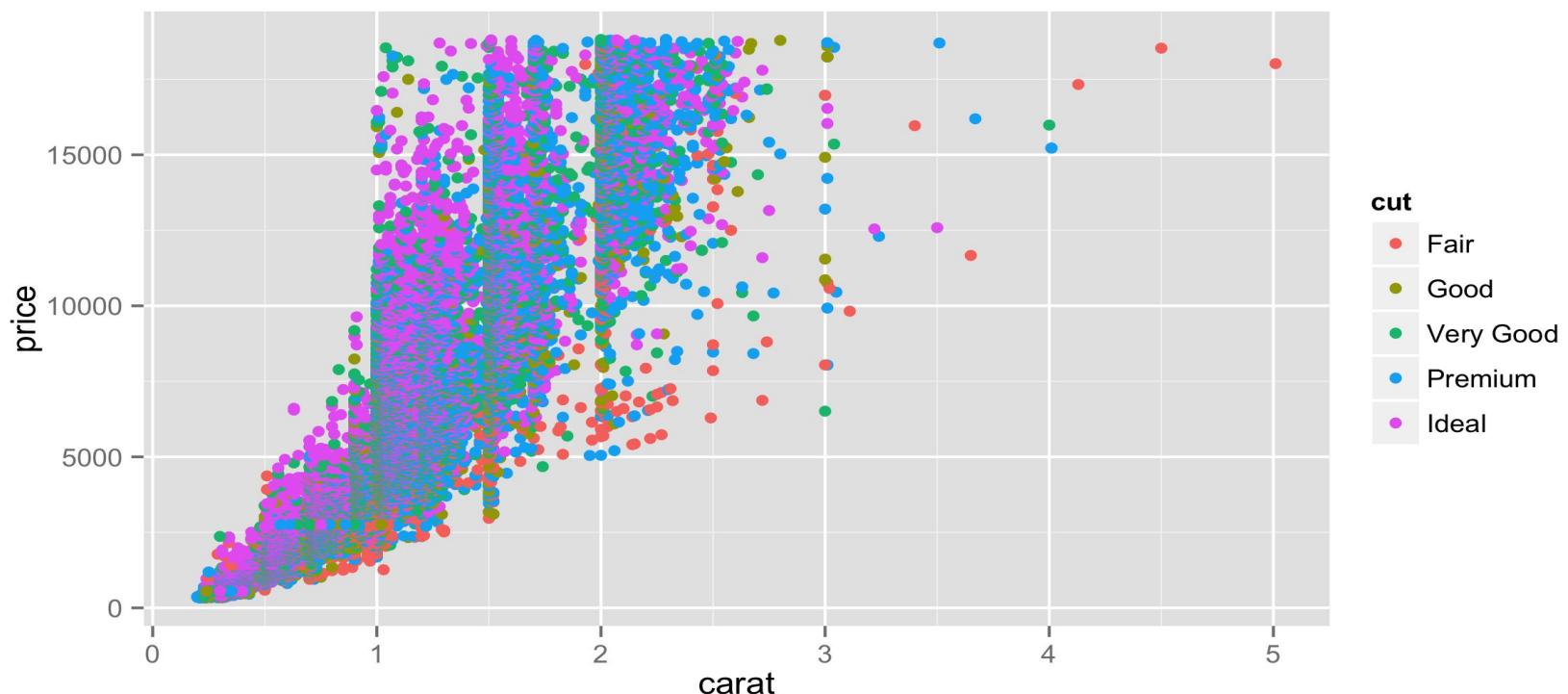
- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ Constructing a ggplot2 plot
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving Graphs

# Visualizing Big Data

# Geoms for Big Data

Is this helpful at all?

```
g <- ggplot(data = diamonds, aes(x = carat, y = price))  
g + geom_point(aes(color = cut))
```



## Geoms for Big Data

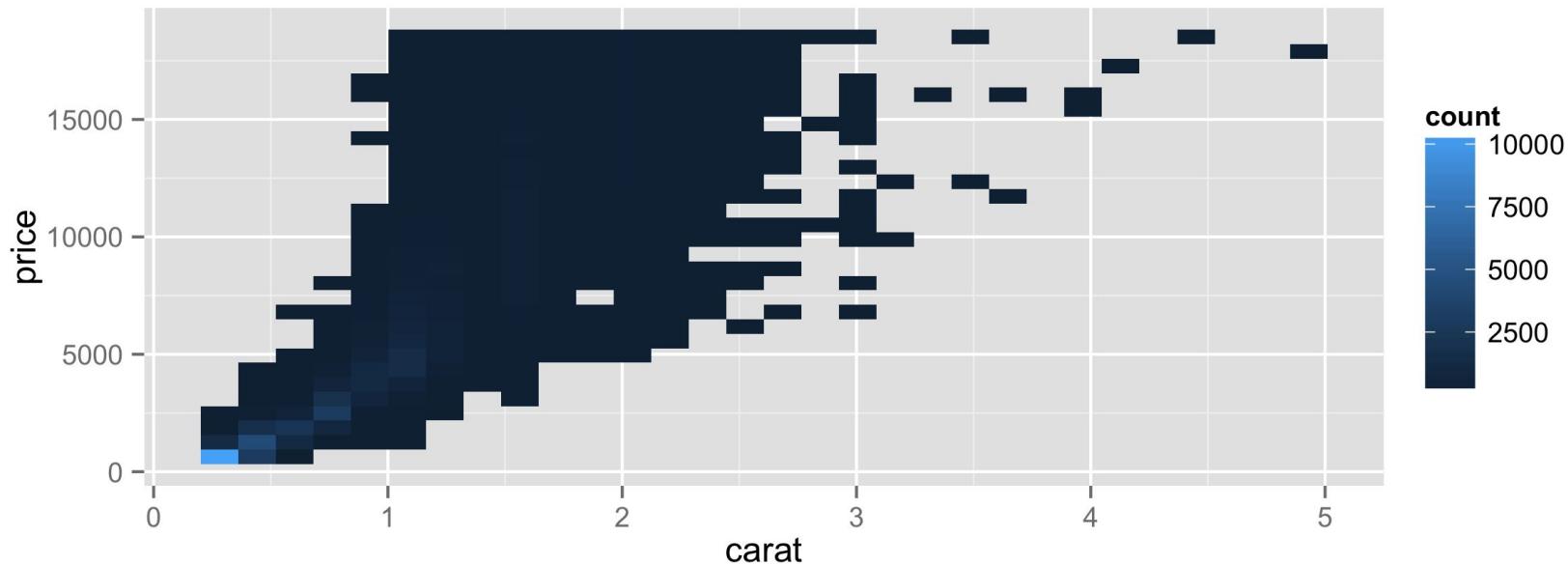
---

How do we work with visualizations when they become overwhelming due to the amount of data?

# Geoms for Big Data

Heat map of 2d bin counts

```
g + geom_bin2d()
```

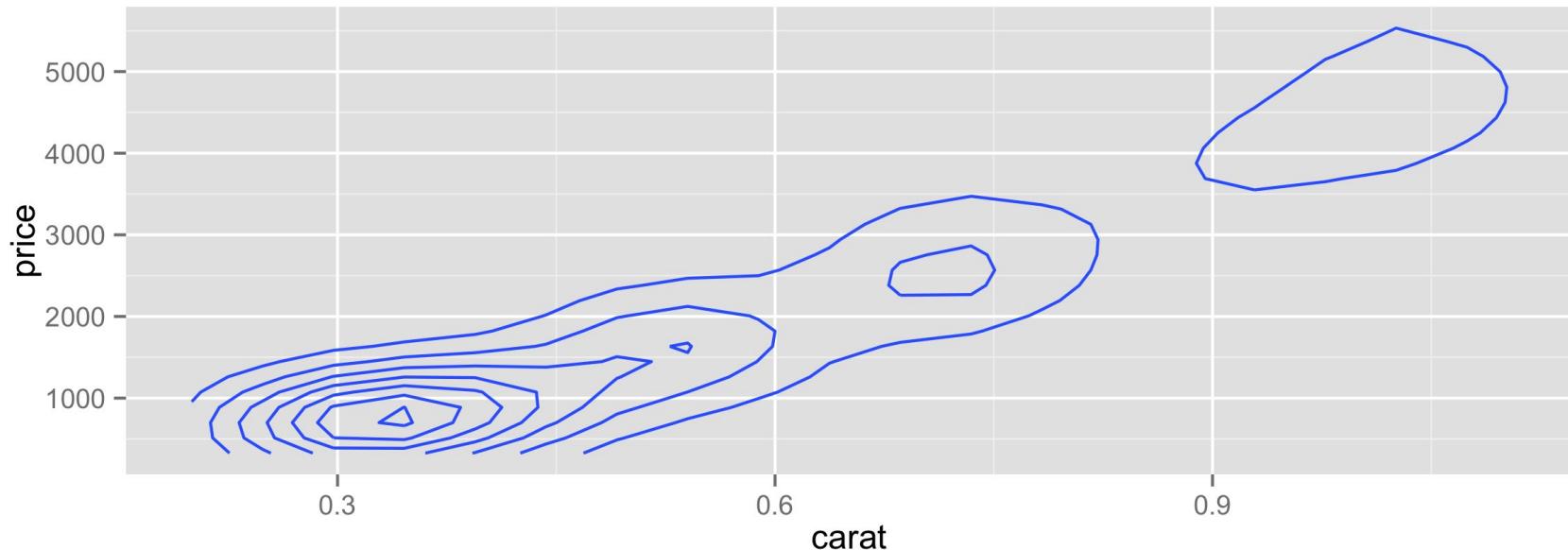


# Geoms for Big Data

---

2d density plot

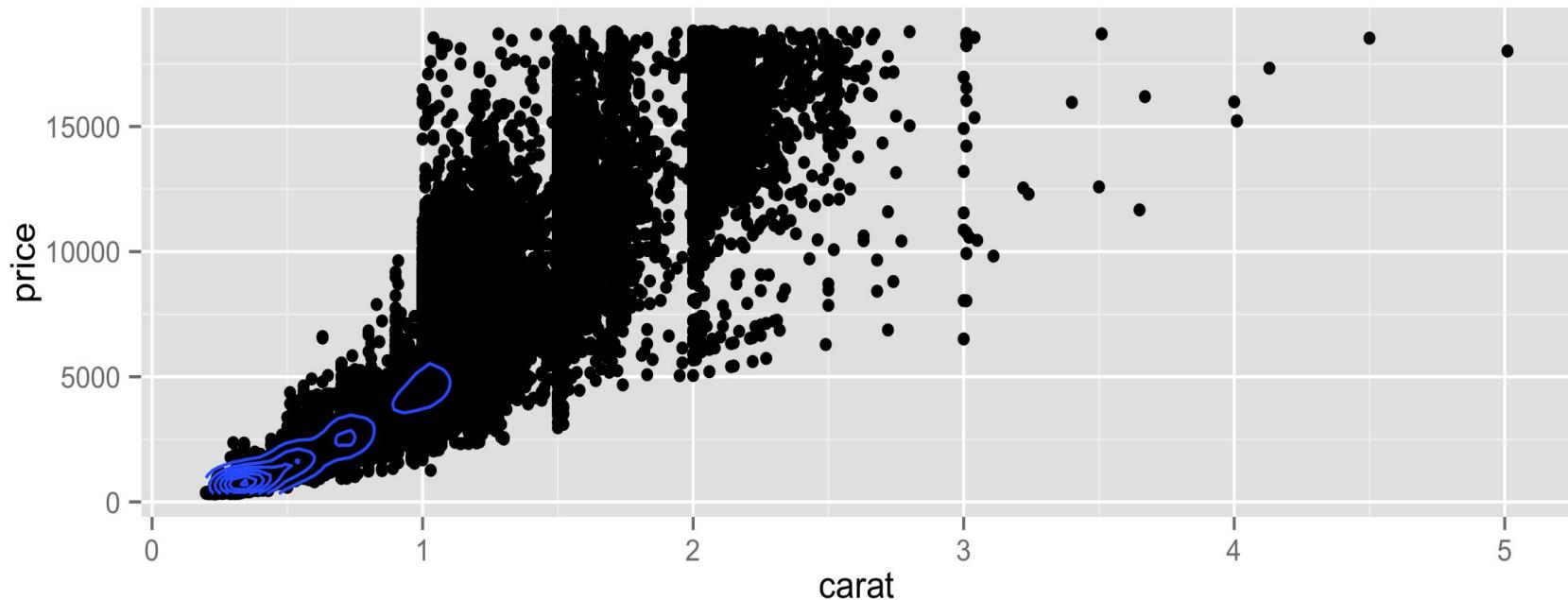
```
g + geom_density2d()
```



## Geoms for Big Data

2d density on top of a scatter plot

```
g + geom_point() + geom_density2d()
```

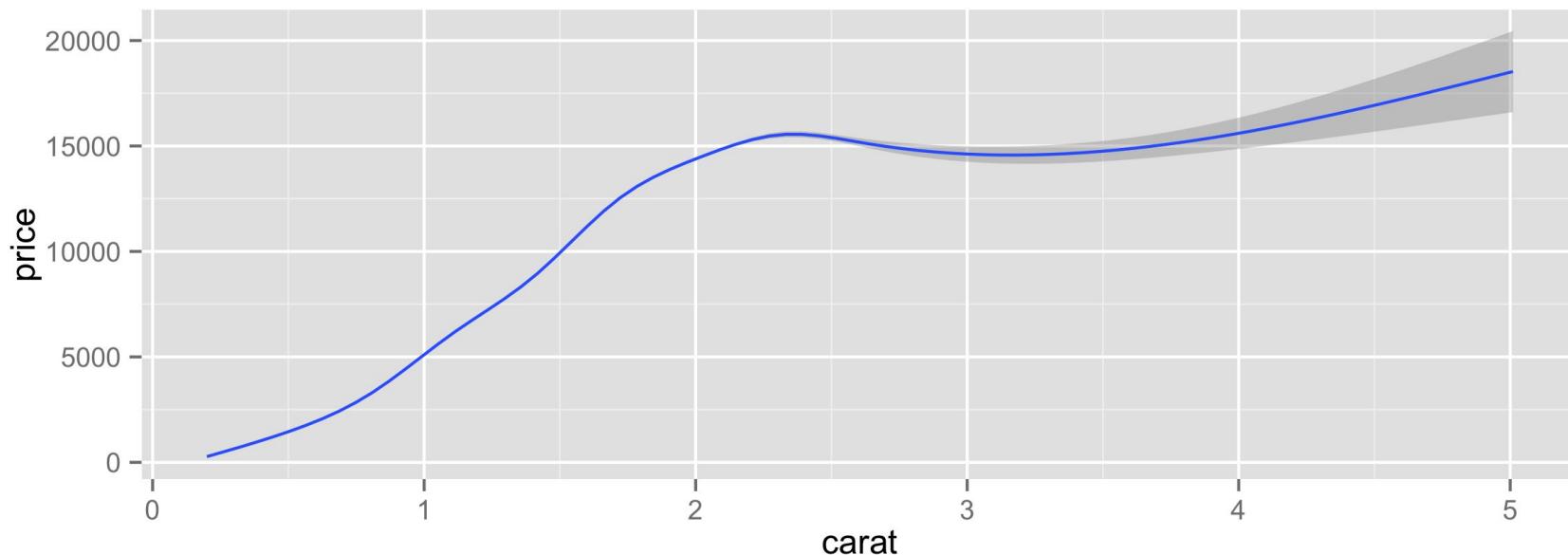


# Geoms for Big Data

---

Smoothed conditional mean

```
g + geom_smooth()
```

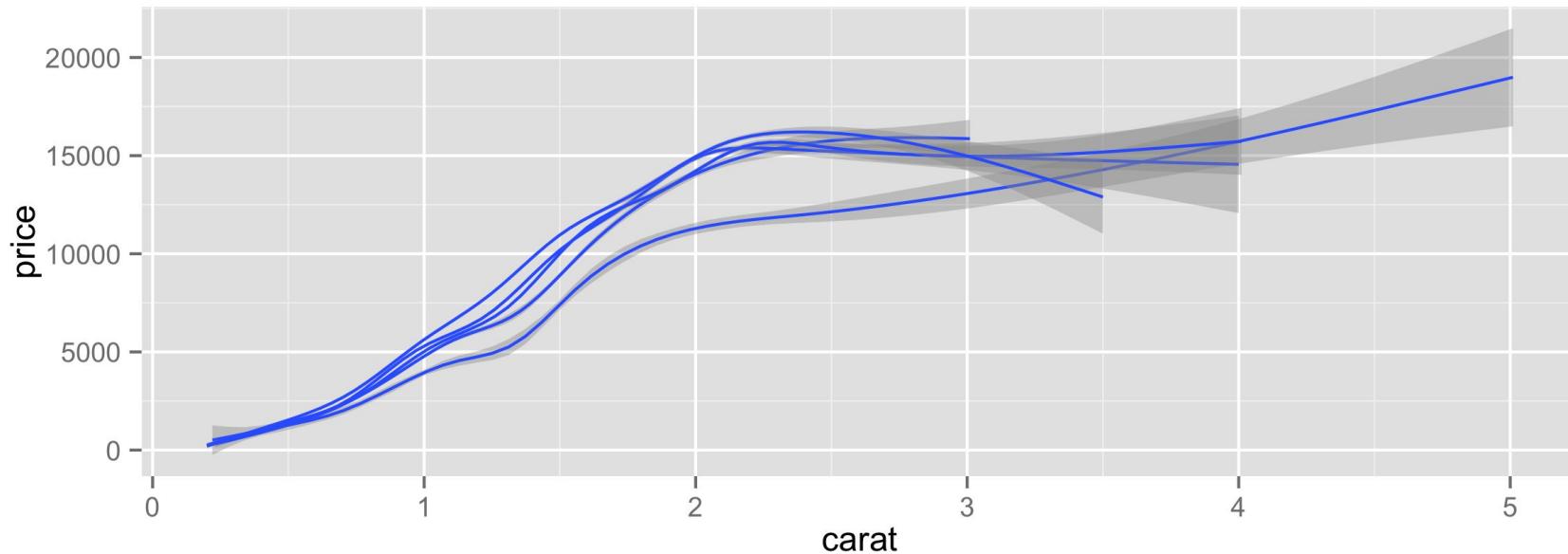


## Geoms for Big Data

---

Same, but for each level of cut

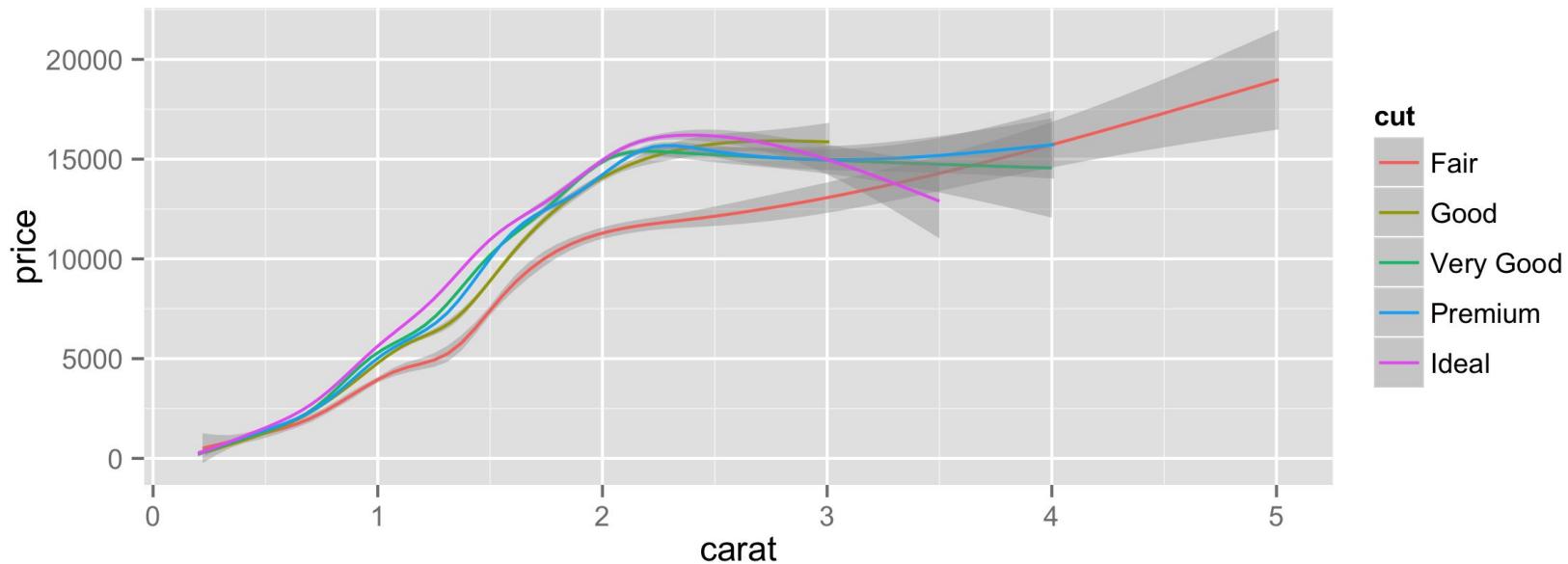
```
g + geom_smooth(aes(group = cut))
```



## Geoms for Big Data

Same, but each cut receives a different color

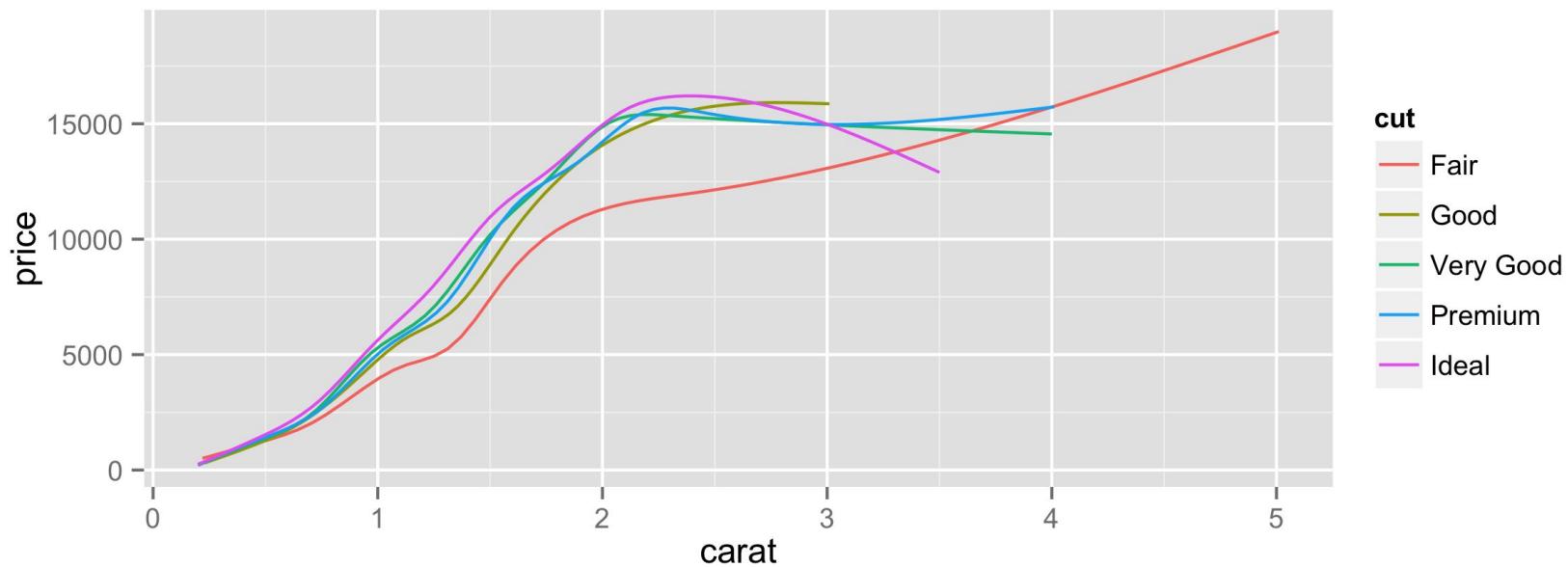
```
g + geom_smooth(aes(color = cut))
```



# Geoms for Big Data

Same, but without confidence bands

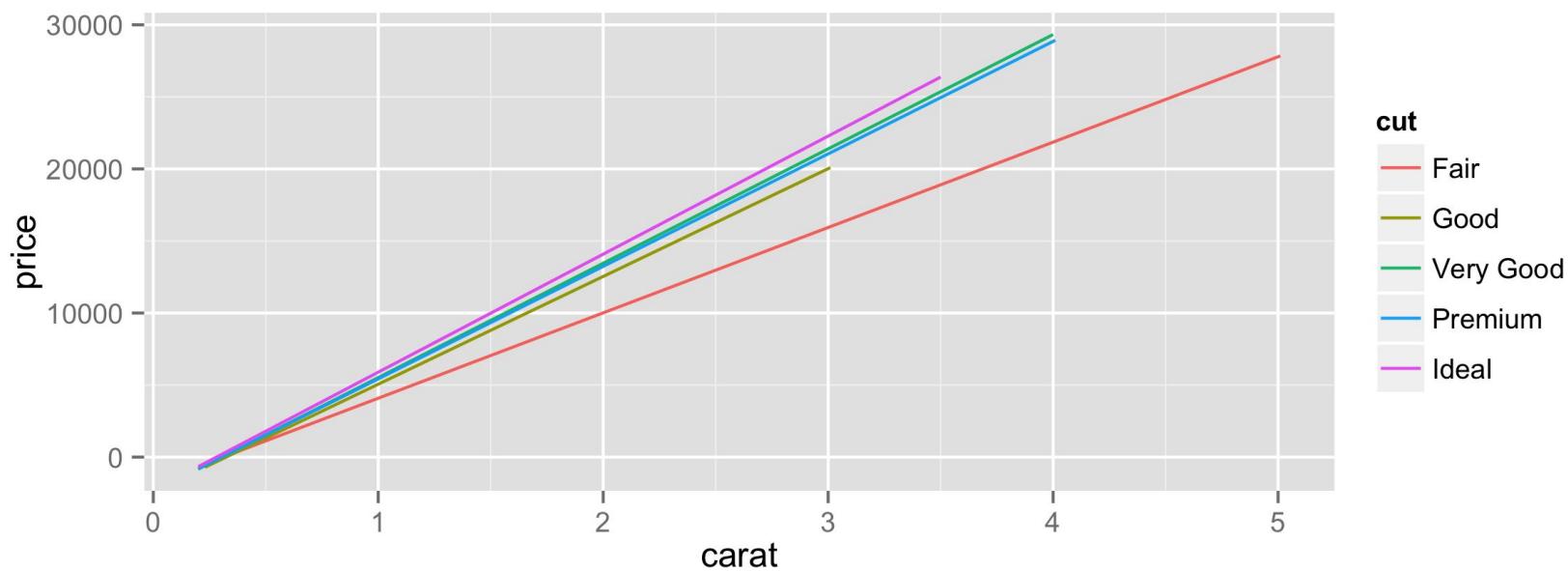
```
g + geom_smooth(aes(color = cut), se = FALSE)
```



# Geoms for Big Data

Regression lines

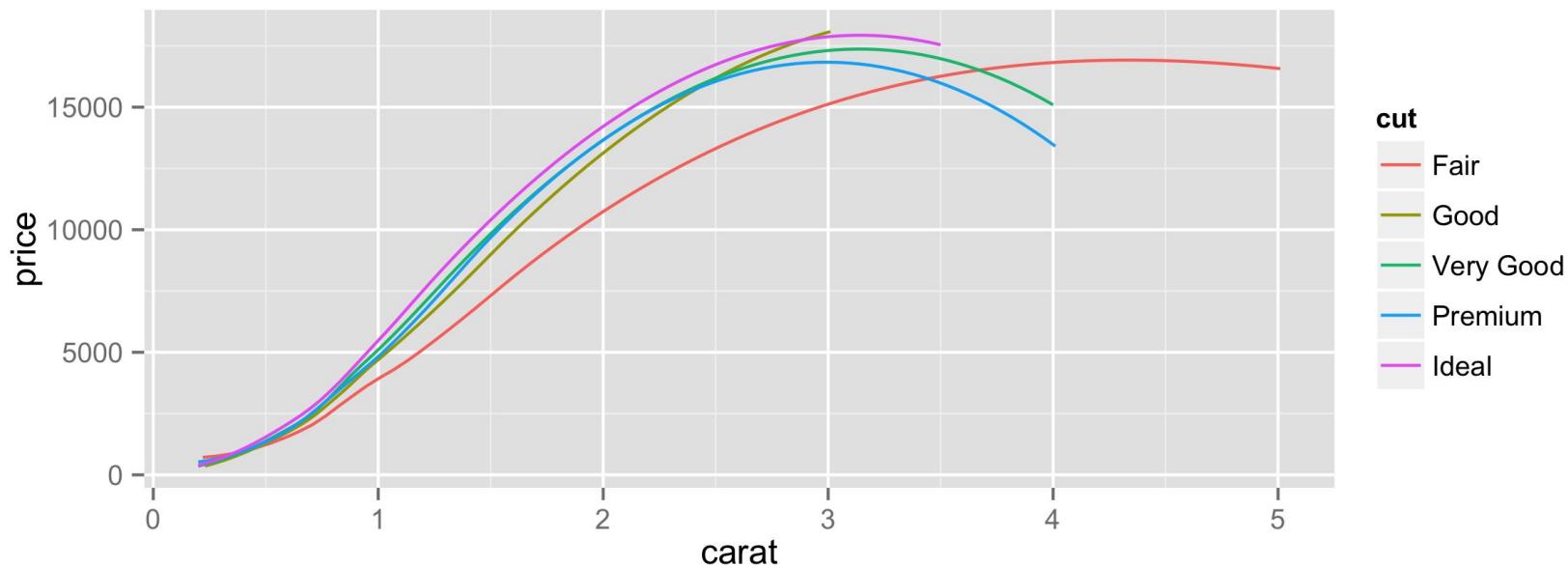
```
g + geom_smooth(aes(color = cut), method = "lm", se = FALSE)
```



# Geoms for Big Data

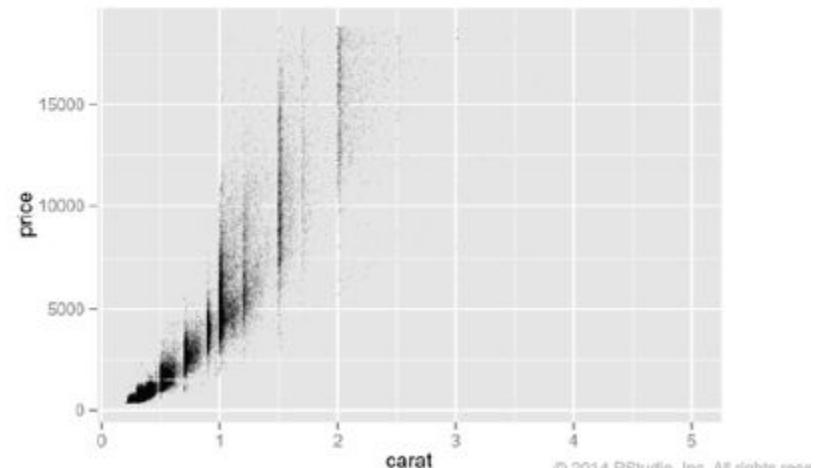
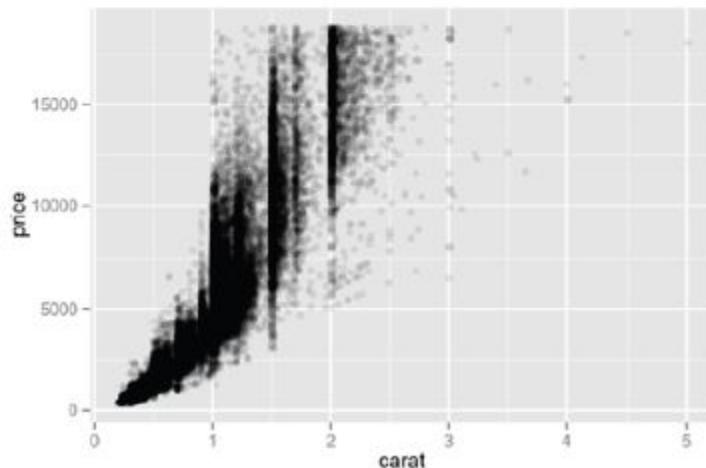
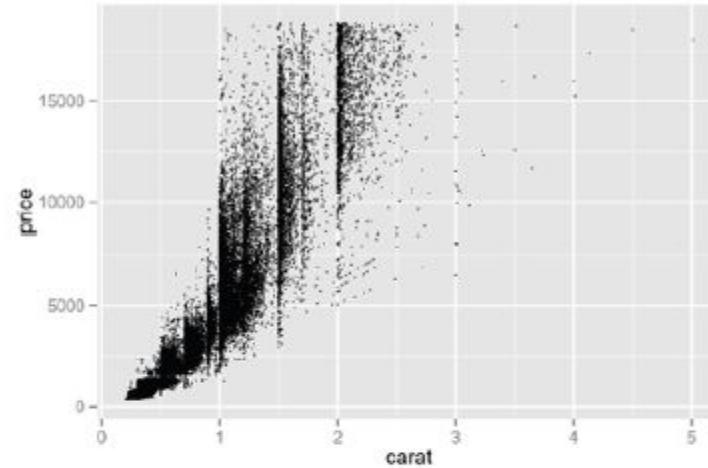
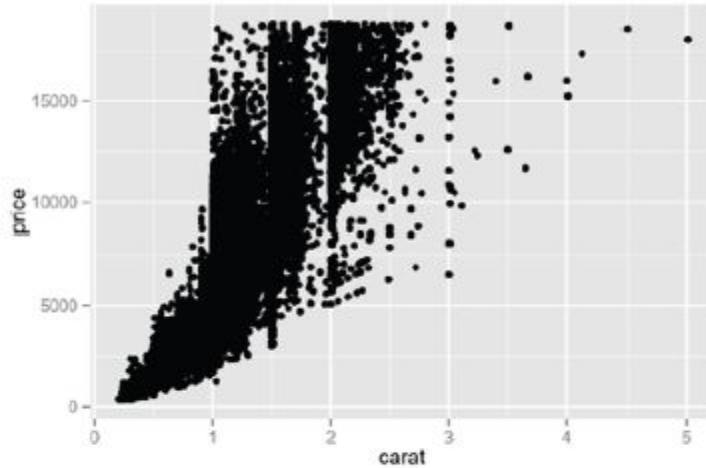
LOESS smoother

```
g + geom_smooth(aes(color = cut), method = "loess", se = FALSE)
```



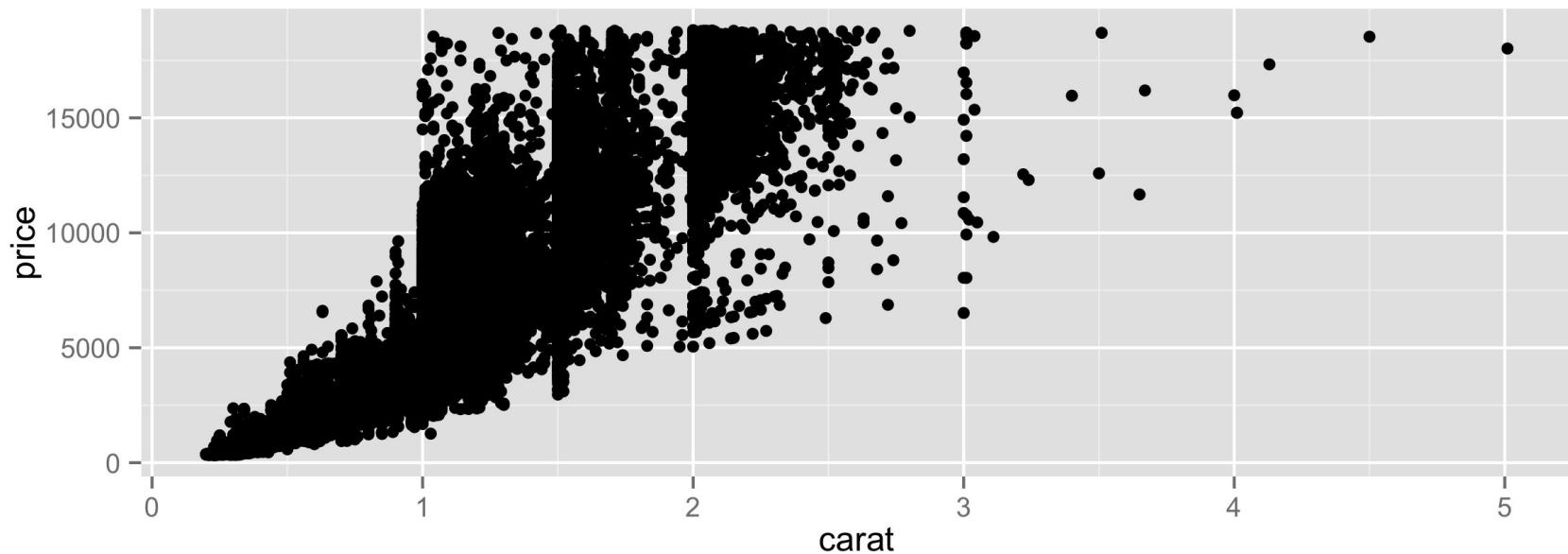
# Geoms for Big Data

We can also adjust the size and/or transparency of points.



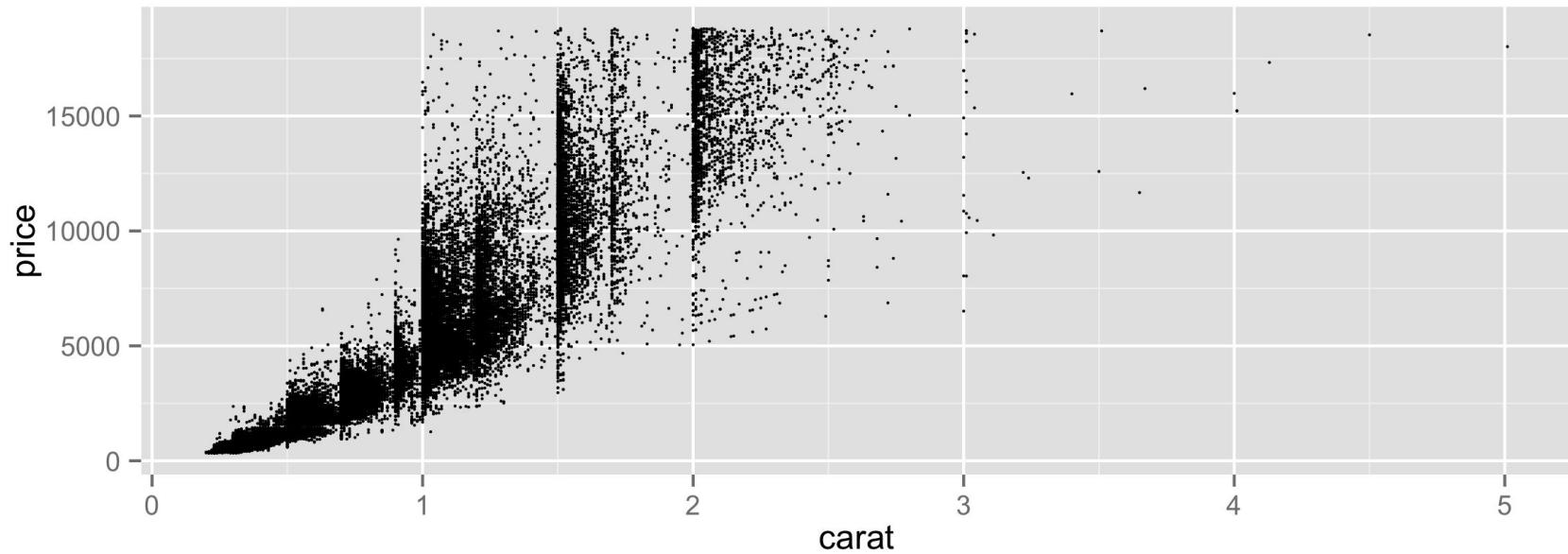
## Geoms for Big Data

```
g + geom_point()
```



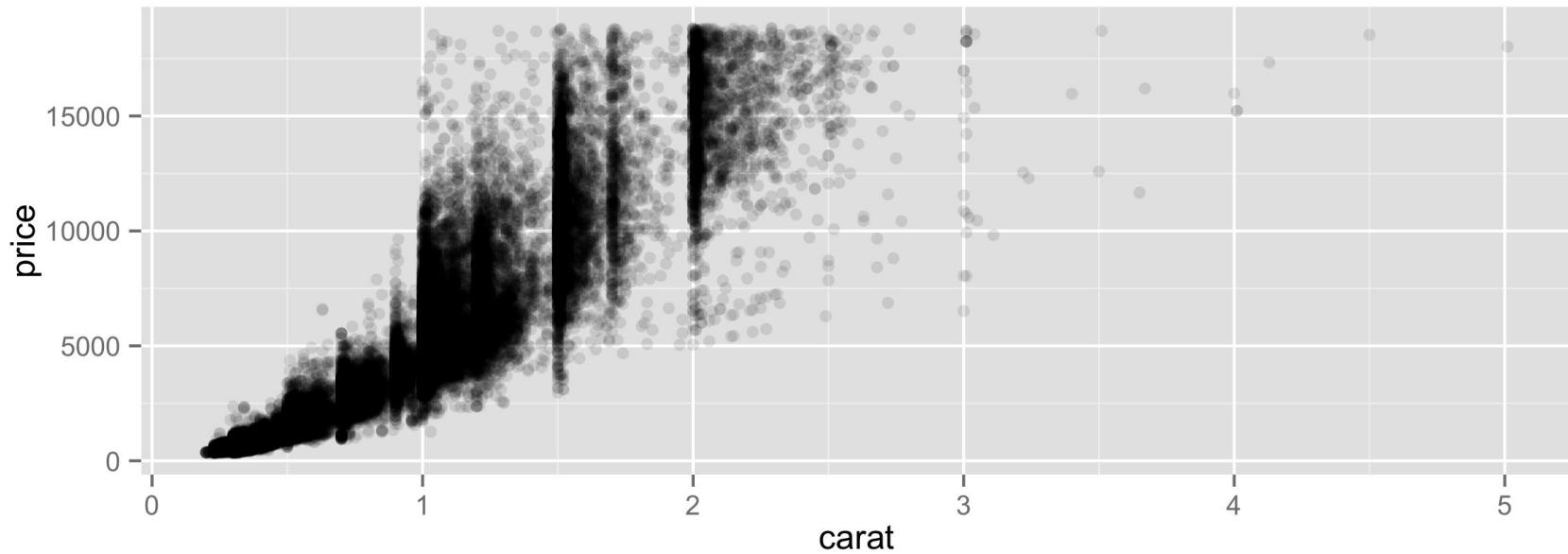
## Geoms for Big Data

```
g + geom_point(size = 0.5)
```



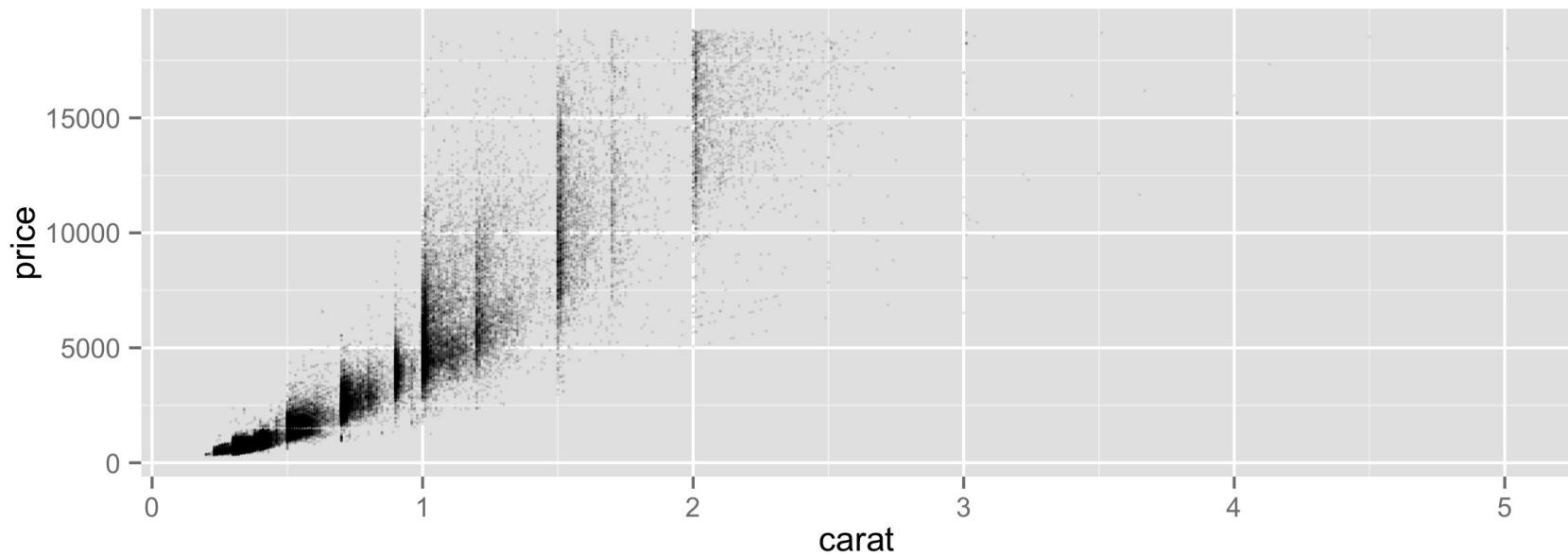
## Geoms for Big Data

```
g + geom_point(alpha = 0.1)
```



## Geoms for Big Data

```
g + geom_point(size = 0.5, alpha = 0.1)
```



# Outline

---

- ❖ Why ggplot2?
- ❖ The “Grammar of Graphics”
- ❖ Constructing a ggplot2 plot
- ❖ Scatterplots
- ❖ Bar charts
- ❖ Histograms
- ❖ Visualizing big data
- ❖ Saving Graphs

# Saving Graphs

# Saving Plots

---

Uses size on screen:

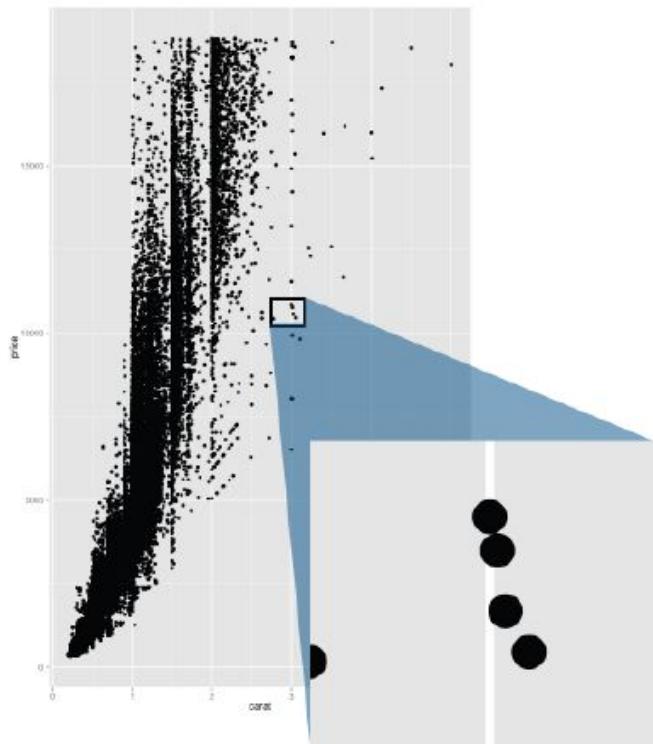
```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Uses specified size:

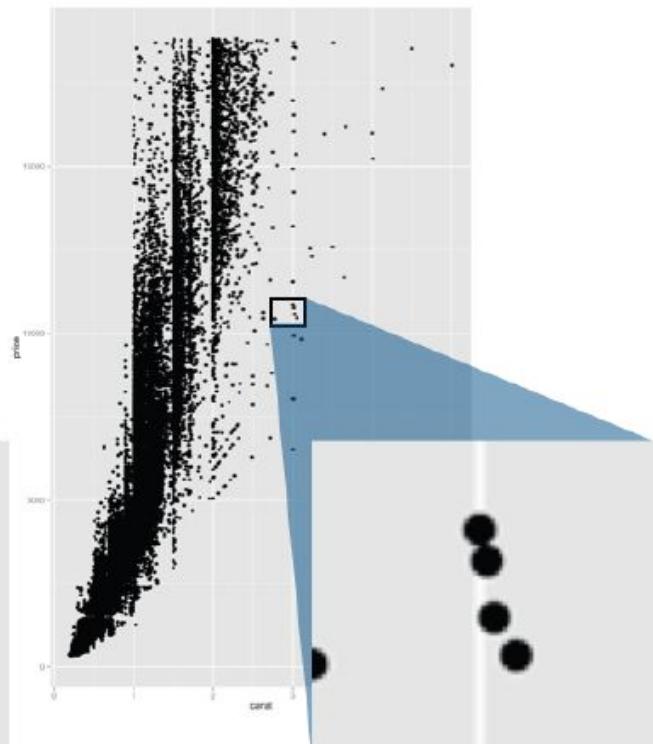
```
ggsave("my-plot.pdf", width = 6, height = 6)
ggsave("my-plot.png", width = 6, height = 6)
```

# Saving Plots

---



pdf



png

## Summary

---

- ❖ ggplot + aesthetics = hundreds of plots (e.g., color, size, shape, alpha, etc.)
- ❖ ggplot + geoms = hundreds of plots (e.g., point, boxplots, smooth, bar, etc.)
- ❖ ggplot + geoms + aesthetics = thousands of plots
- ❖ ggplot + geoms + aesthetics + parameters = hundreds of thousands of plots