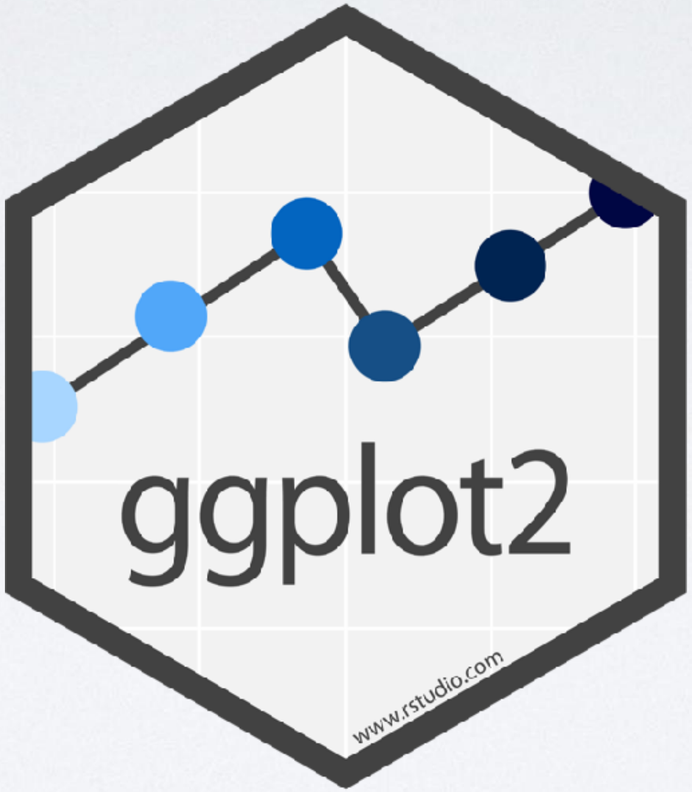
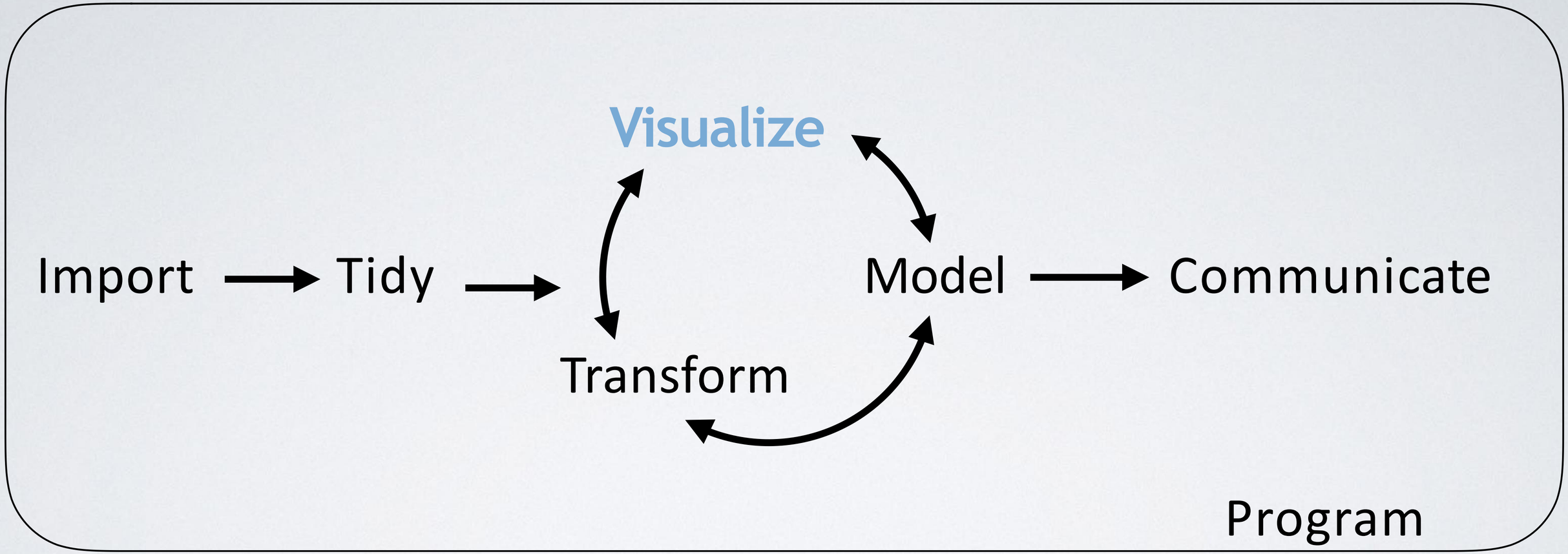
Visualize Data with



Slides CC BY-SARStudio

(Applied) Data Science



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

- John Tukey

mpg

Fuel economy data for 38 models of car.

```
mpg
```

manufacturer <chr>	displ <dbl>	year <int>	cyl <int>	trans <chr>	drv <chr>	cty <int>	hwy <int>	fl <chr>	class <chr>
audi	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	2.8	1999	6	manual(m5)	f	18	26	p	compact
audi	3.1	2008	6	auto(av)	f	18	27	p	compact

Quiz

Confer with your group.

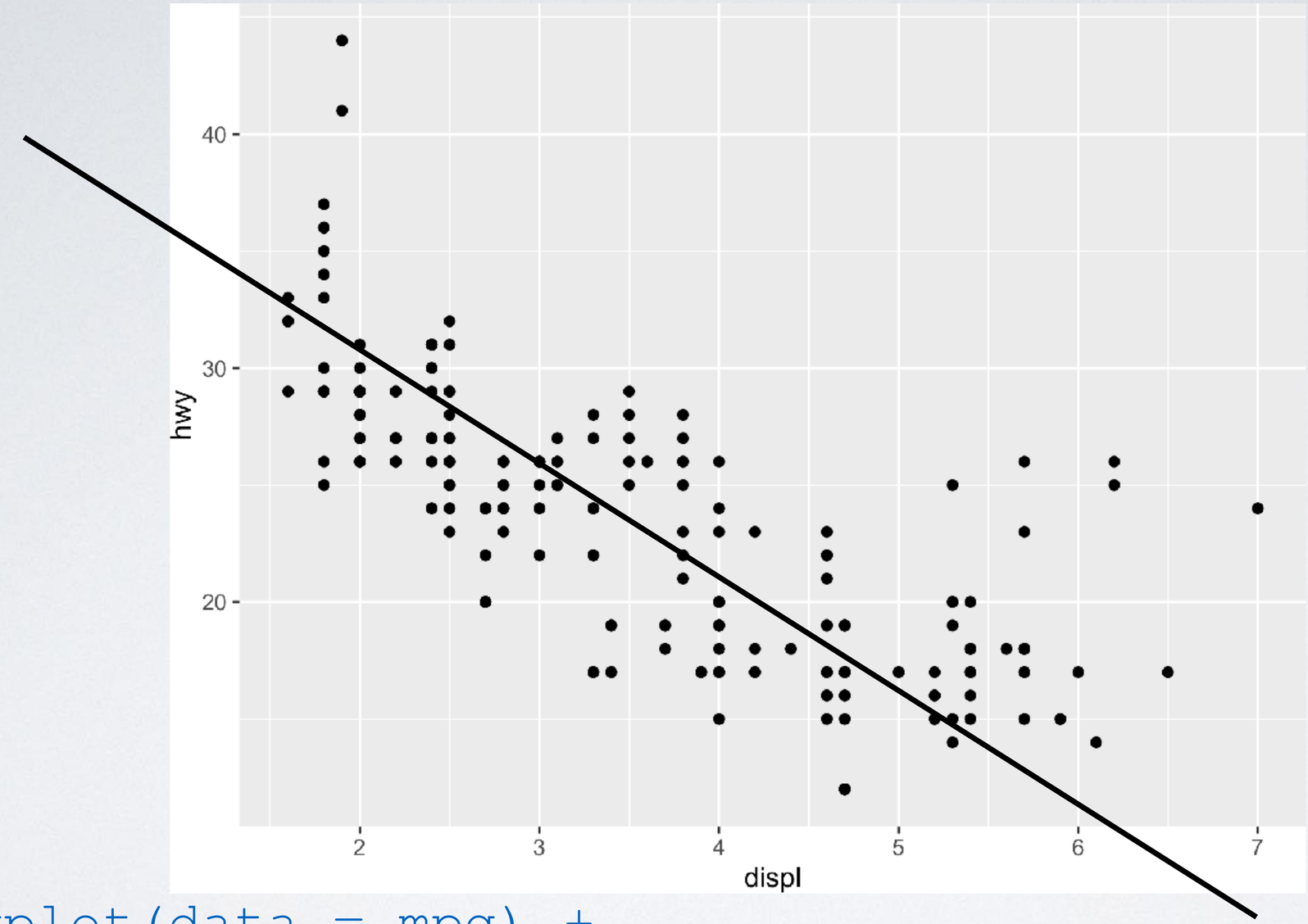
What relationship do you expect to see between engine size (displ) and mileage (hwy)?

No peeking ahead!

Your Turn 1

Run this code in **02-Visualize-Exercises.Rmd** to make a graph. Pay strict attention to spelling, capitalization, and parentheses!

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




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

1. "Initialize" a plot with ggplot()
2. Add layers with geom_functions

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

Pro tip: Always put the + at the end of a line, Never at the start

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

data

+ before new line

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

type of layer

aes()

x variable

y variable

A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

```
  geom_point(mapping = aes(x = displ, y = hwy))
```

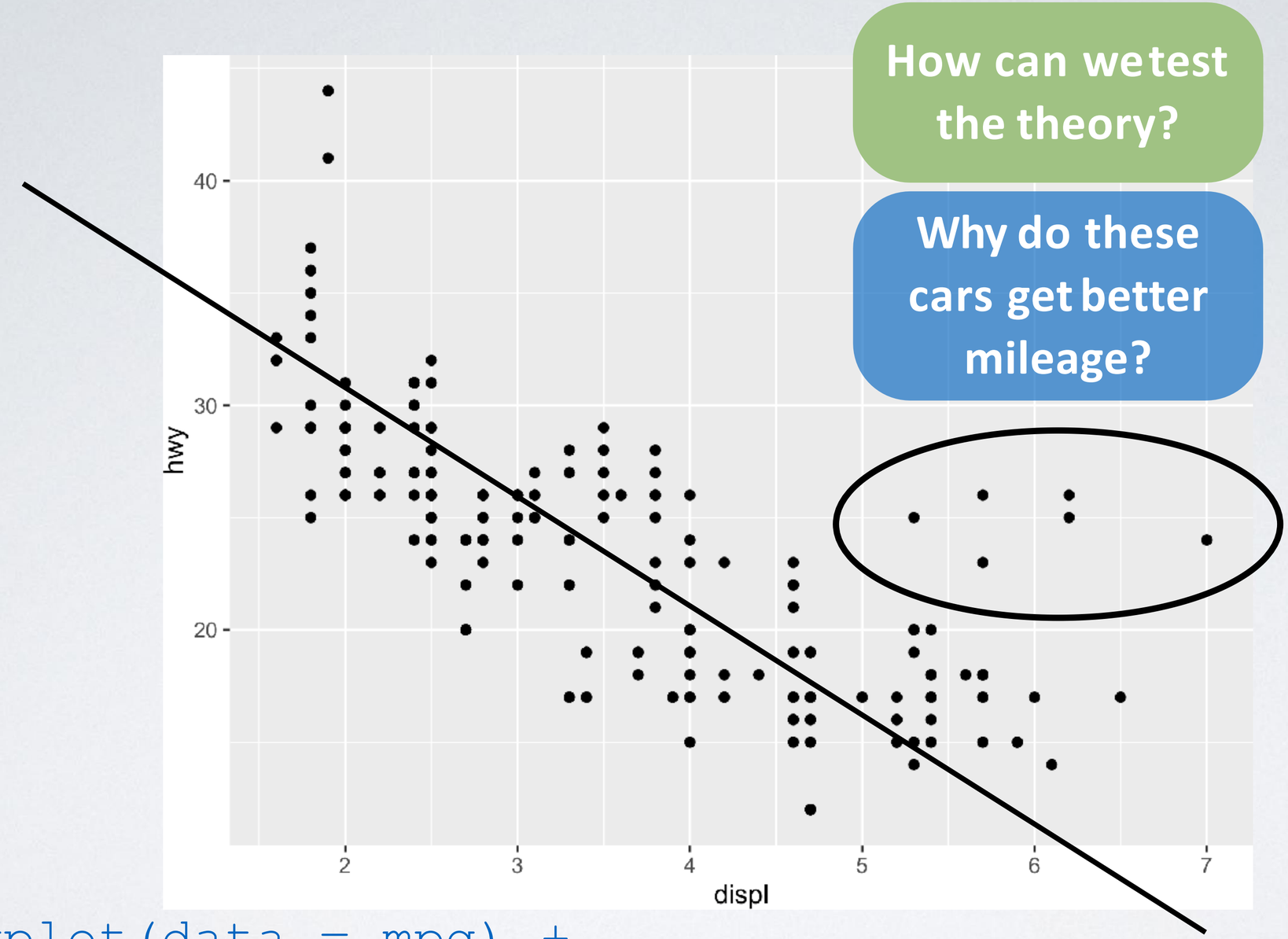
A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Mappings

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

- John Tukey

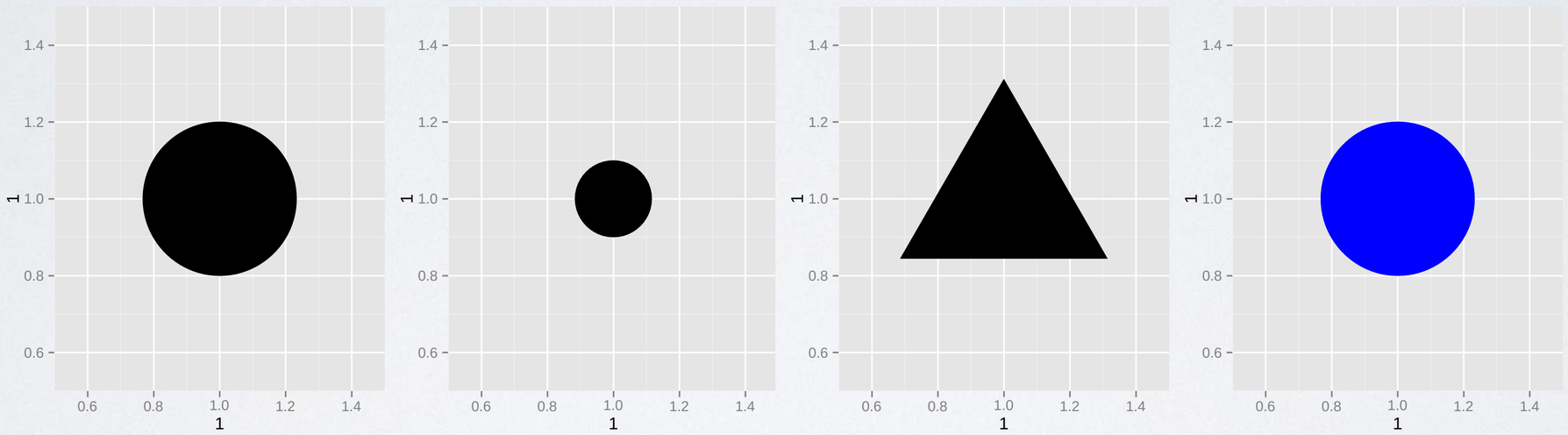


How can we test the theory?

Why do these cars get better mileage?

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

Aesthetics



Visual Space		Data Space
color	⟷	class
Red	⟷	2seater
Brown	⟷	compact
Green	⟷	midsize
Aqua	⟷	minivan
Blue	⟷	pickup
Violet	⟷	subcompact
Pink	⟷	suv

Aesthetics

aesthetic property

Variable to map it to

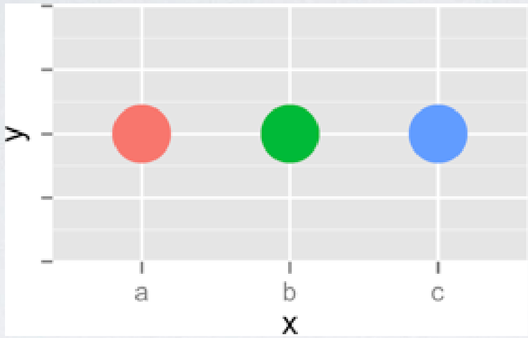
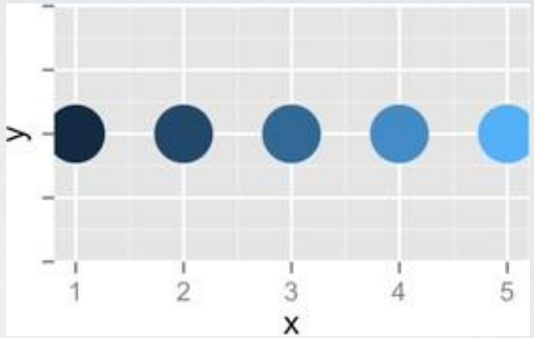
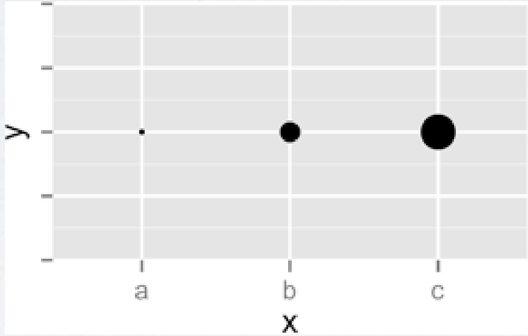
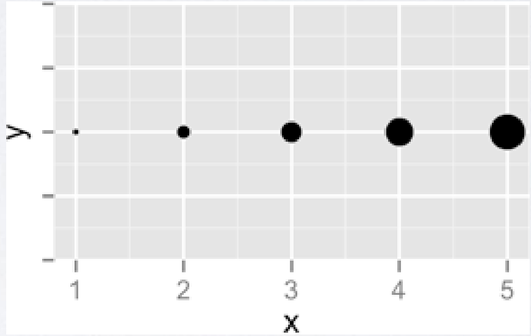
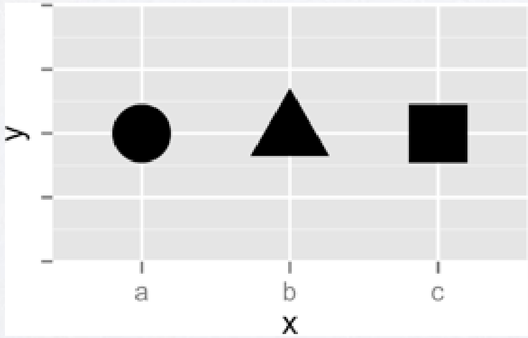
```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, color = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, size = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, shape = class))
ggplot(mpg) + geom_point(aes(x = displ, y = hwy, alpha = class))
```

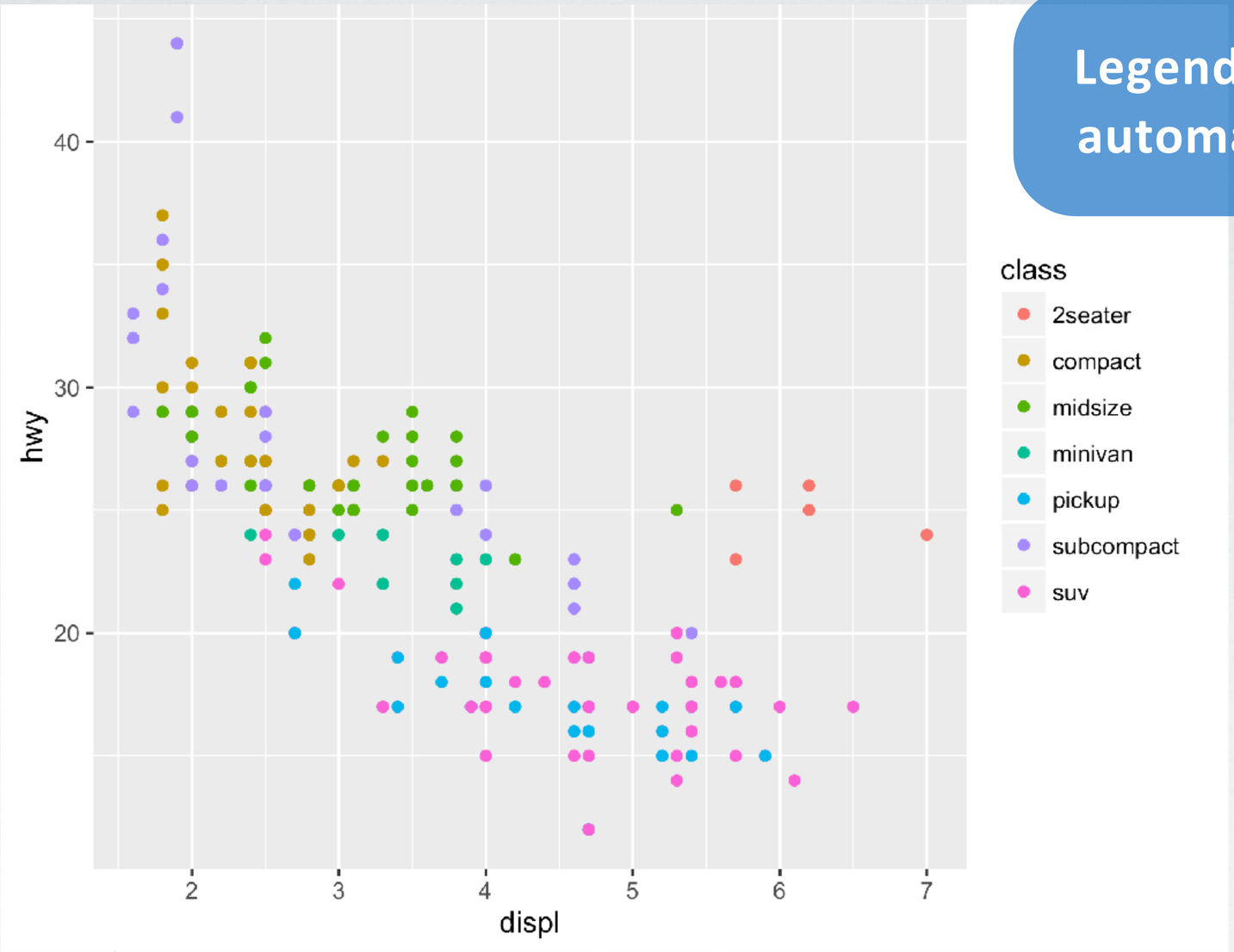
Your Turn 2

In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

Do different things happen when you map aesthetics to discrete and continuous variables?

What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		



Legend added automatically

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

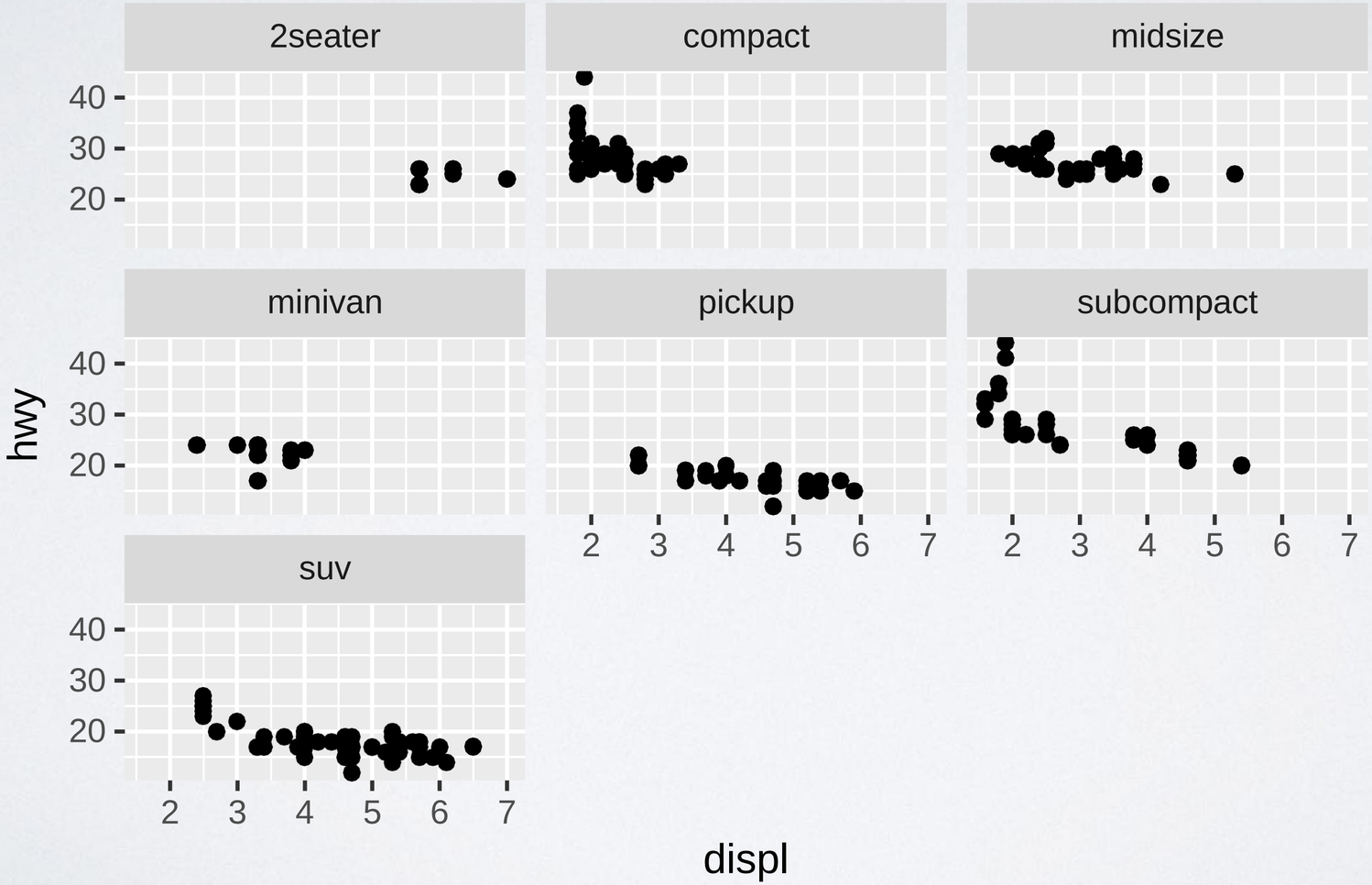
Error!

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

Facets

Facets

Subplots that display subsets of the data.



CC by RStudio

Help me

What do **facet_grid** and **facet_wrap** do?

```
q <- ggplot(mpg) + geom_point(aes(x = displ, y = hwy))
q + facet_grid(. ~ cyl)
q + facet_grid(drv ~ .)
q + facet_grid(drv ~ cyl)
q + facet_wrap(~ class)
```

CC by RStudio

summary

facet_grid() - 2D grid, rows ~ cols, . for no split
facet_wrap() - 1D ribbon wrapped into 2D

CC by RStudio

A ggplot2 template

Make any plot by filling in the parameters of this template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>)) +
  <FACET_FUNCTION>
```

CC by RStudio

Your Turn 3

Add the black code to your graph. What does it do?

```
ggplot(data = mpg) +
  geom_point(mapping = aes(displ, hwy, color = class)) +
  labs(title = "Fuel Efficiency by Engine Size",
       subtitle = "Data facetted by class",
       x = "Engine Size (displacement in liters)",
       y = "Fuel Efficiency (MPG)",
       color = "Class of\nAutomobile",
       caption = "Data from the EPA")
```

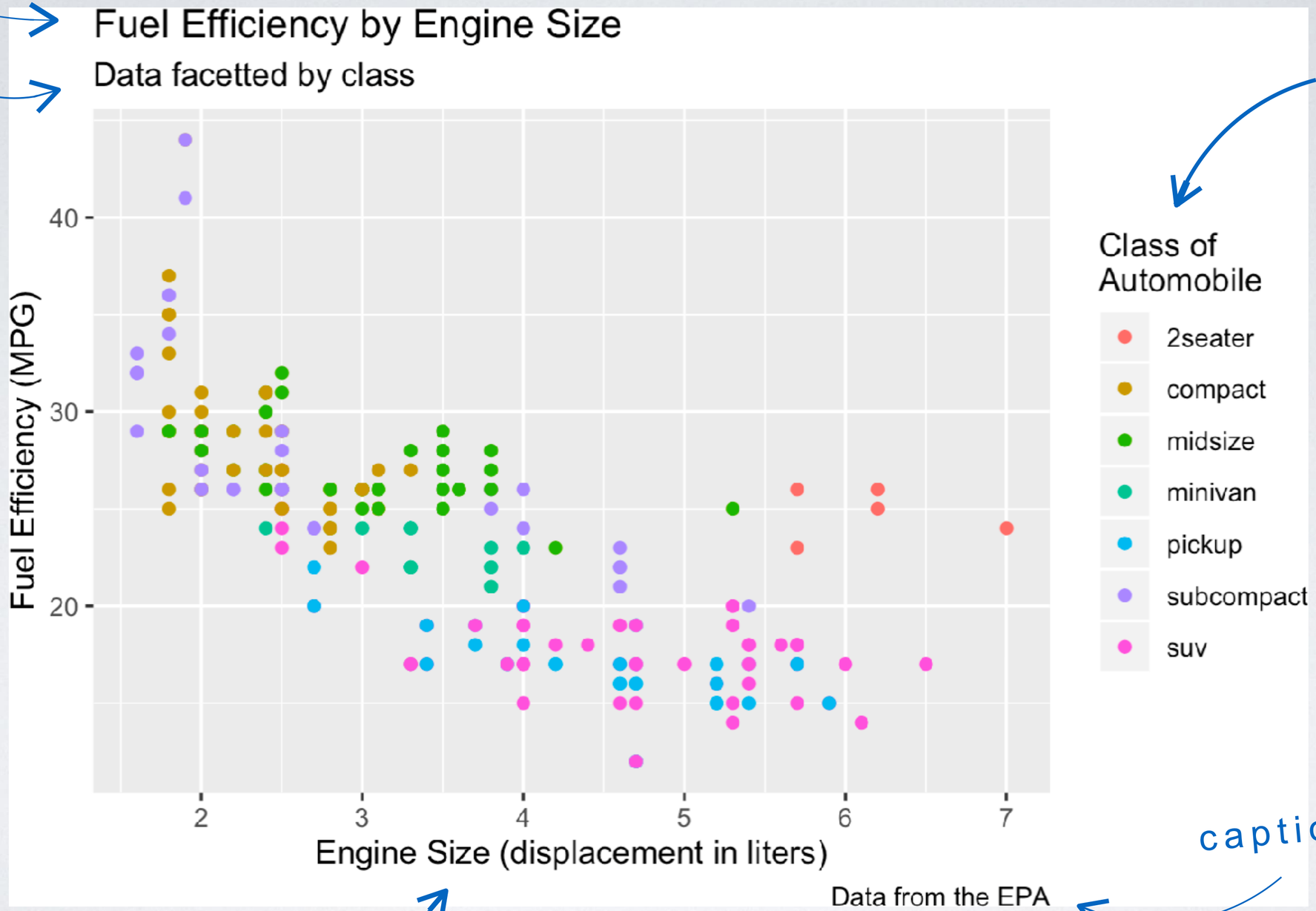
CC by RStudio

Title

SUBTitle

y

color



x

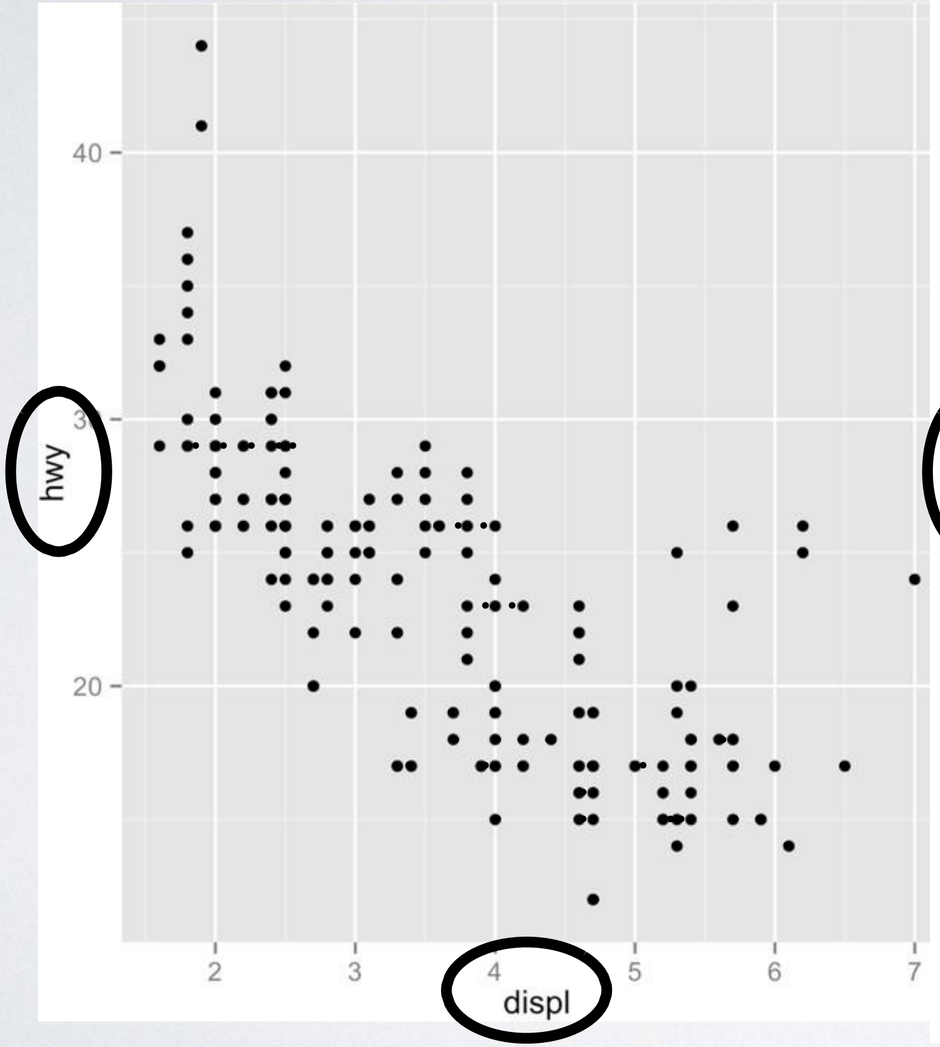
caption

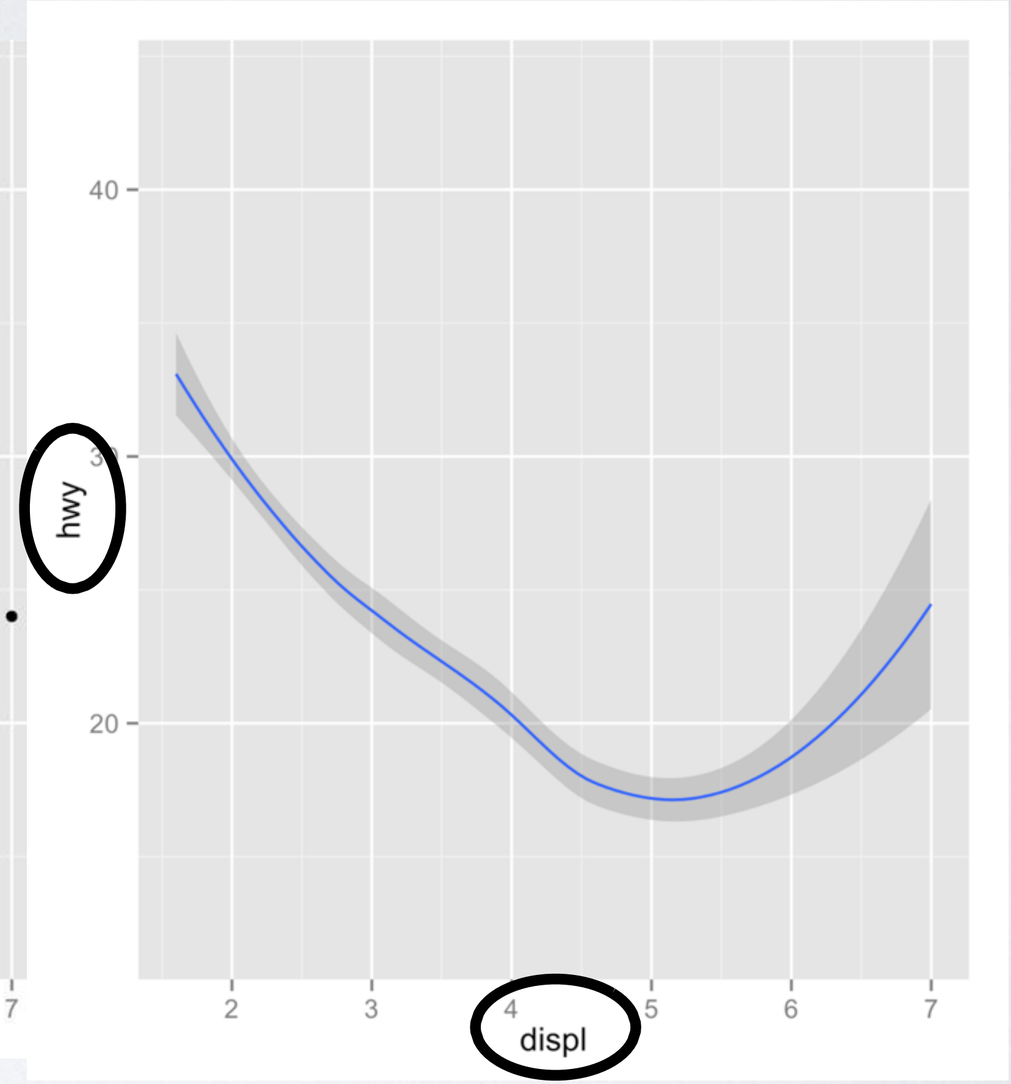
CC by RStudio

Geoms

How are these plots similar?

Same: x var , y var , data

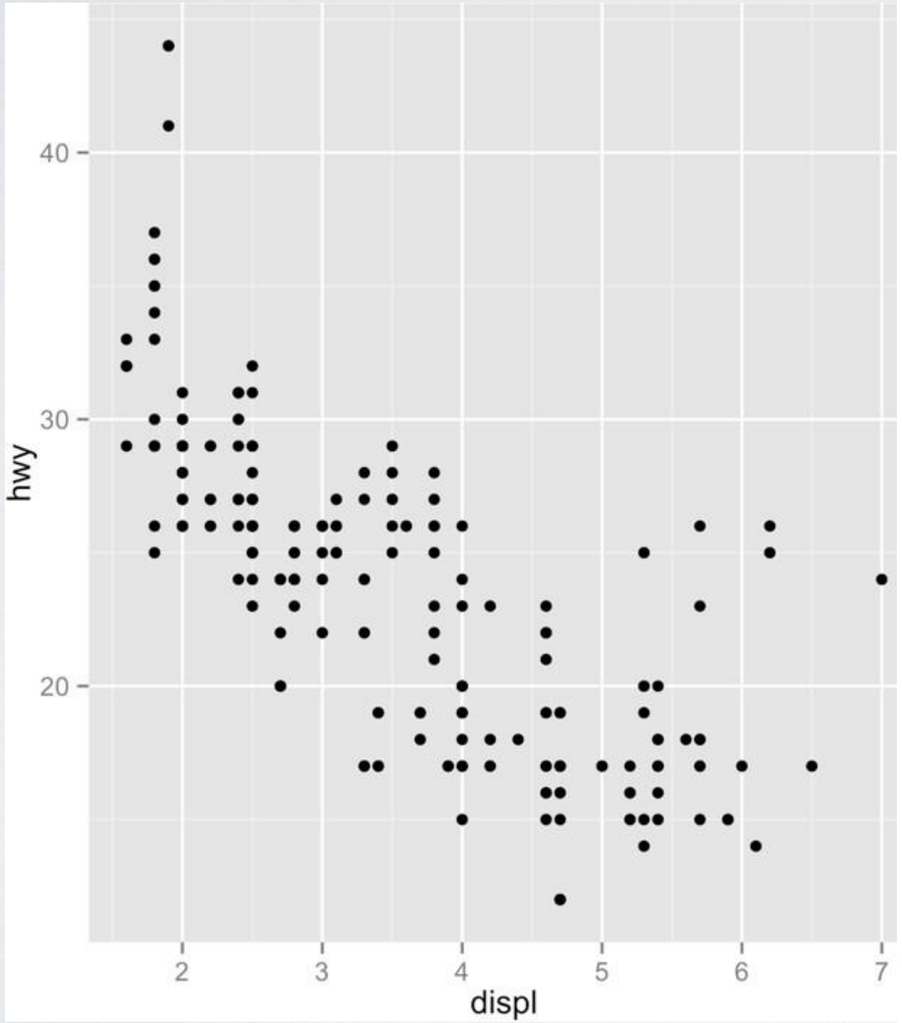


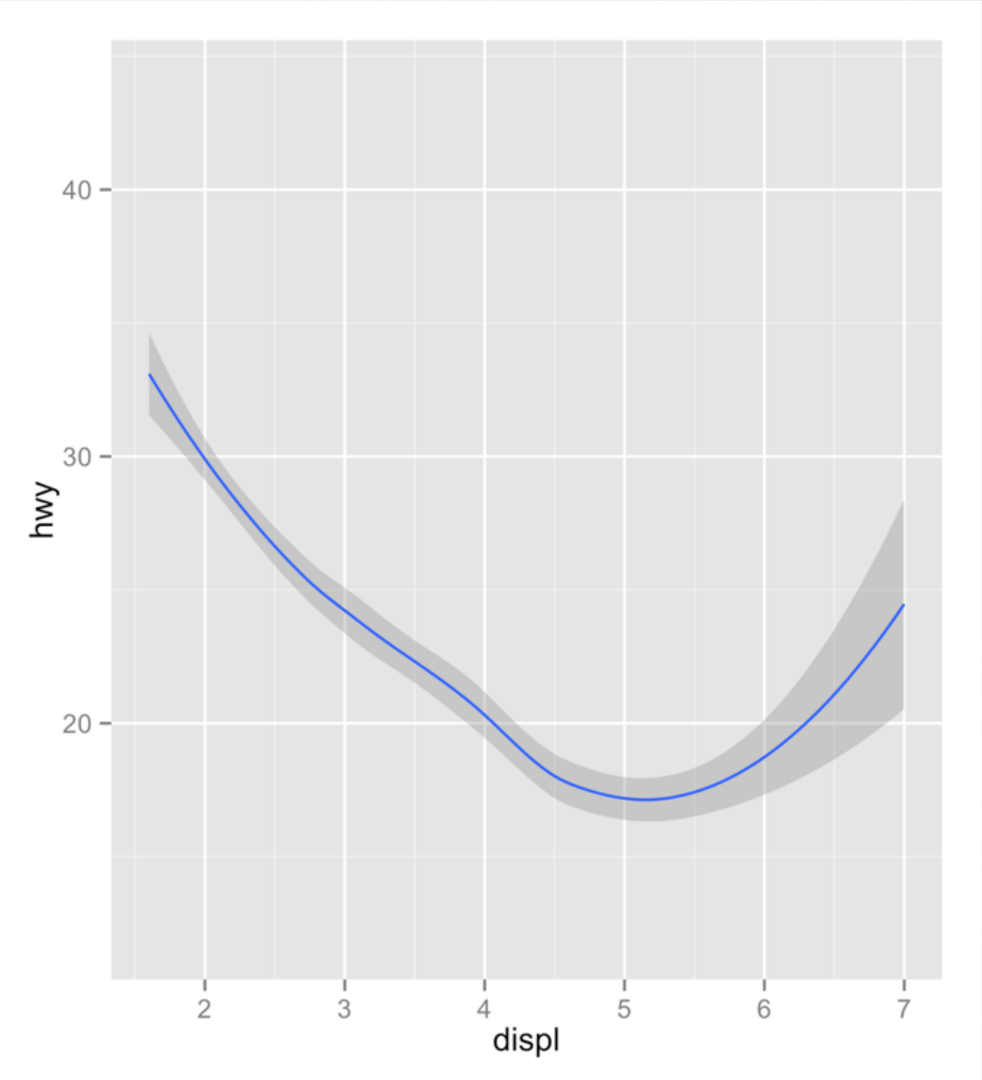


How are these plots different?

Different: geometric object (geom),

e.g. the visual object usedto represent the data





geoms

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

rstudio.cloud/learn/cheat-sheets

Click Cheat Sheets in the sidebar

RStudio Cloud

rstudio.cloud/learn/cheat-sheets

Studio Cloud

Learn | Guide | What's New | Primers | Cheat Sheets

Garrett Grolemund

Spaces
- Your Workspace
- R In 90 Minutes
- Welcome-to-the-Tidyverse
- New Space

Learn
- Guide
- What's New
- Primers
- Cheat Sheets
- Feedback and Questions

Info
- Terms and Conditions
- System Status

RStudio Cheat Sheets

Cheat sheets make it easy to learn and use your favorite R packages and the RStudio IDE.

Deep Learning with Keras

Download

Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. Keras supports both convolution based networks and recurrent networks (as well as combinations of the two), runs seamlessly on both CPU and GPU devices, and is capable of running on top of multiple back-ends including TensorFlow, CNTK, and Theano.

Updated December 2017

Dates and Times

Download

Lubridate makes it easier to work with dates and times in R. This lubridate cheatsheet covers how to round dates, work with time zones, extract elements of a date or time, parse dates into R and more. The back of the cheatsheet describes lubridate's three timespan classes: periods, durations, and intervals; and explains how to do math with date-times.

Updated December 2017

Work with Strings

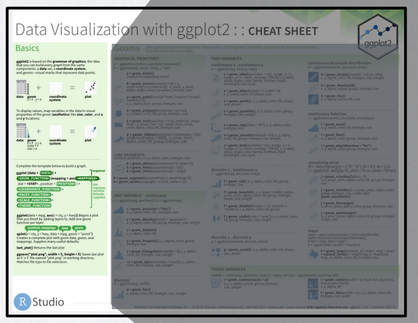
The stringr package provides an easy to use toolkit for working with strings, i.e. character data, in R. This cheatsheet guides you through stringr's functions for manipulating strings. The back

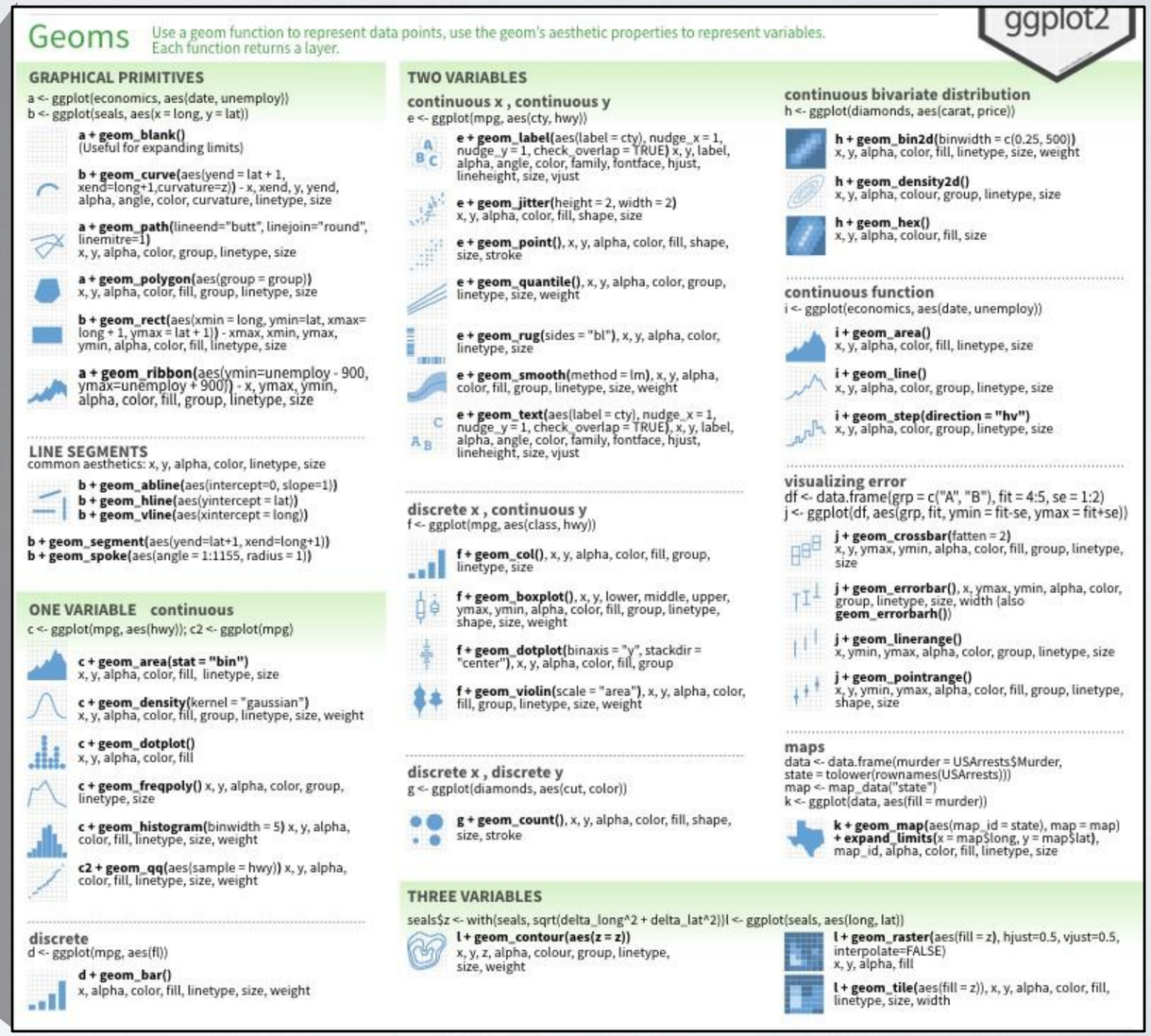
Apply Functions

The purrr package makes it easy to work with lists and functions. This cheatsheet will remind you how to manipulate lists with purrr as well as how to apply functions iteratively to each element of a list

geom_ functions

Each requires a mapping argument.





Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

ggplot2

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

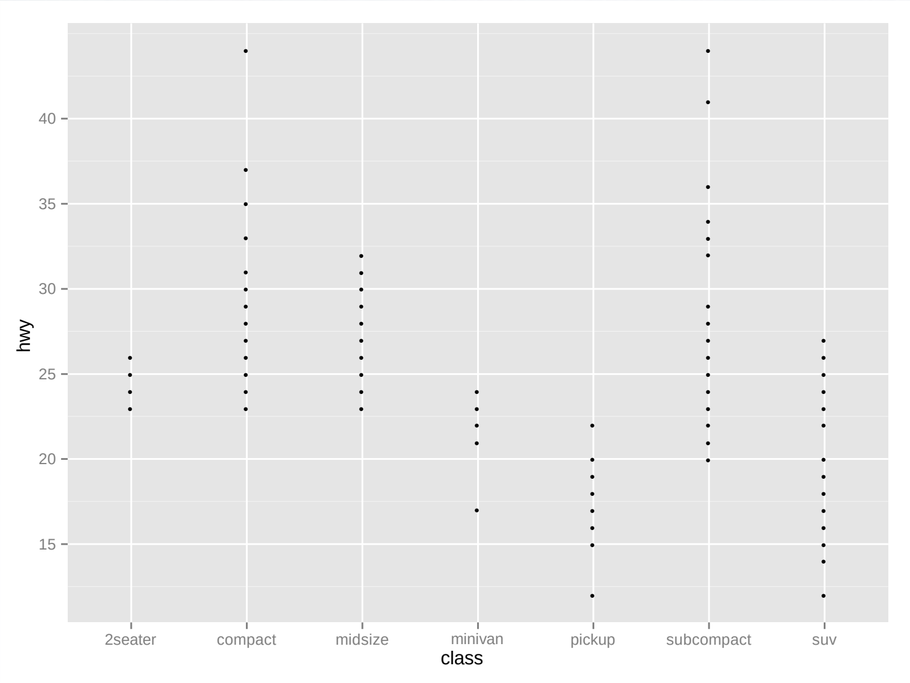
l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

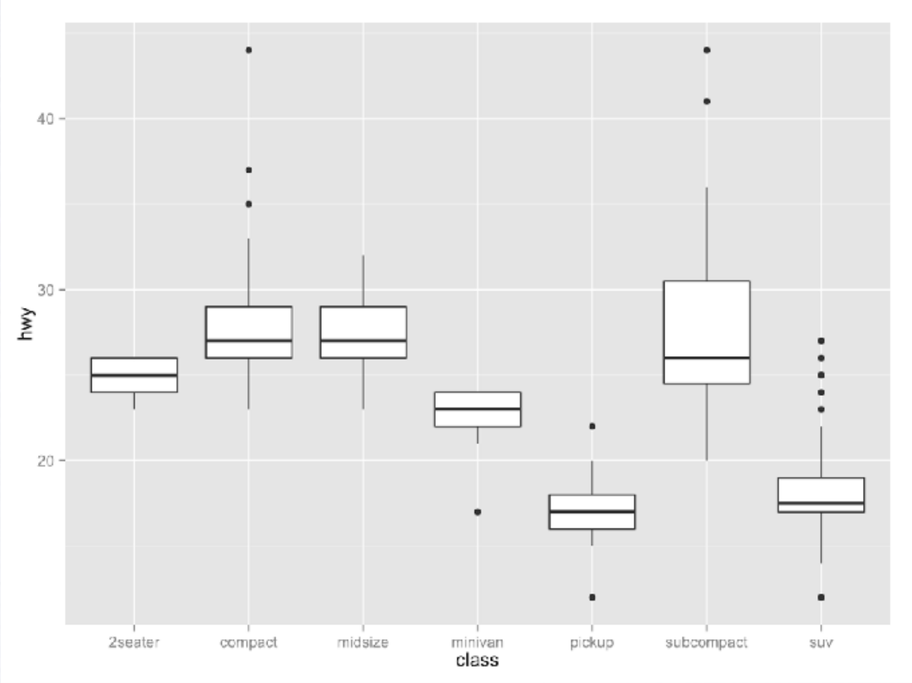
l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

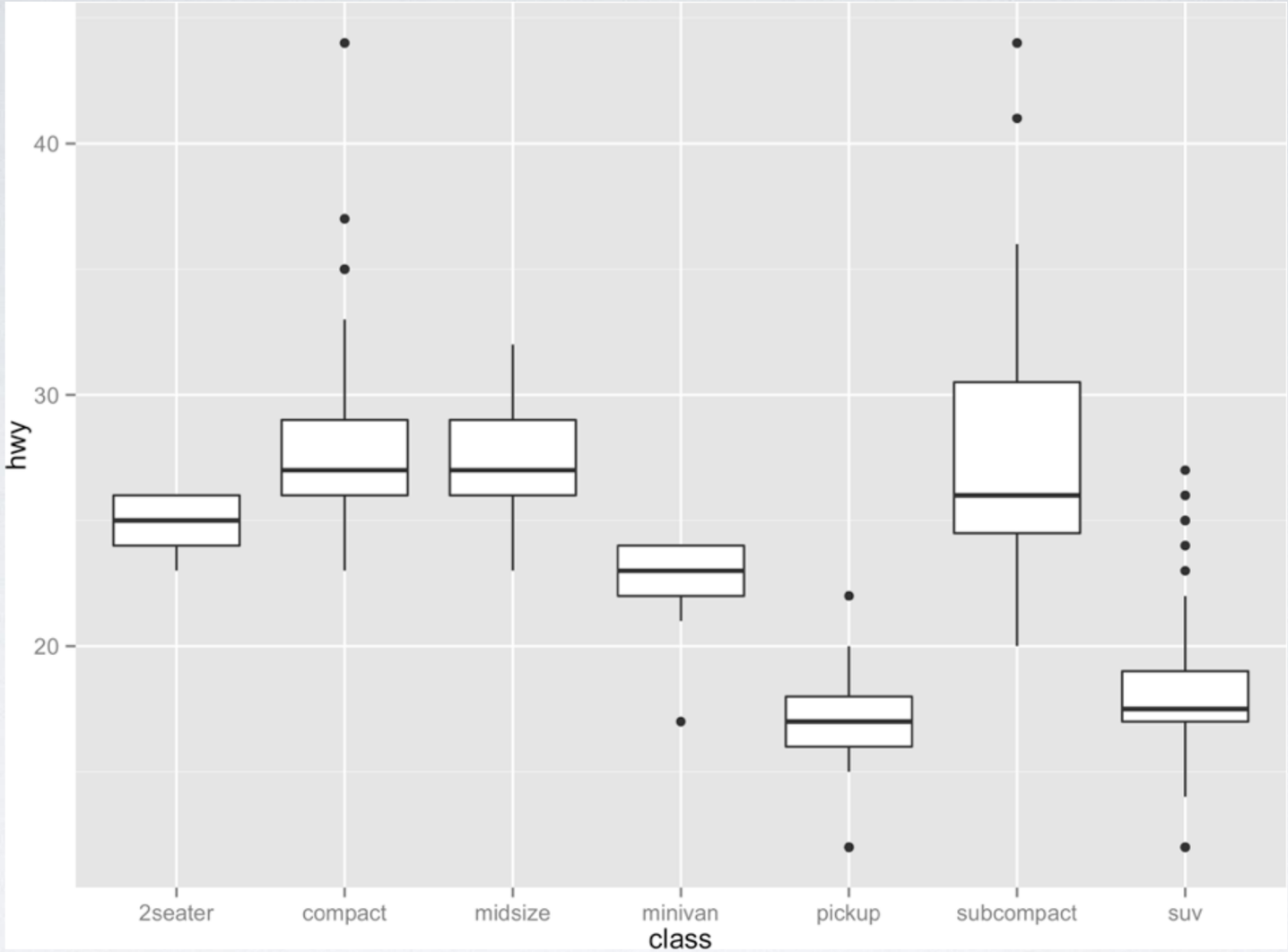
Your Turn

What is the best way to replace this scatterplot with one that draws boxplots. Use the cheatsheet. Try your best guess.






```
ggplot(mpg) + geom_point(aes(class, hwy))
```

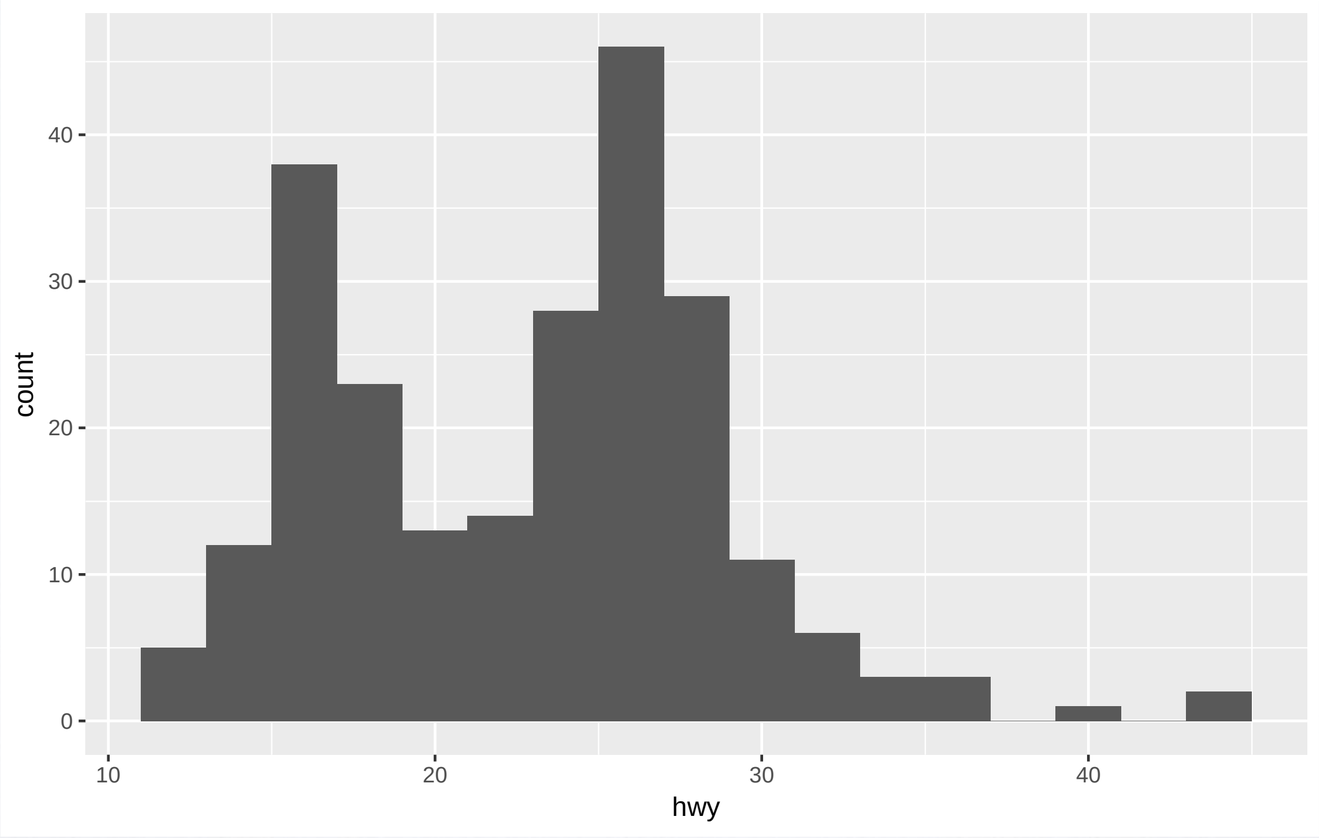


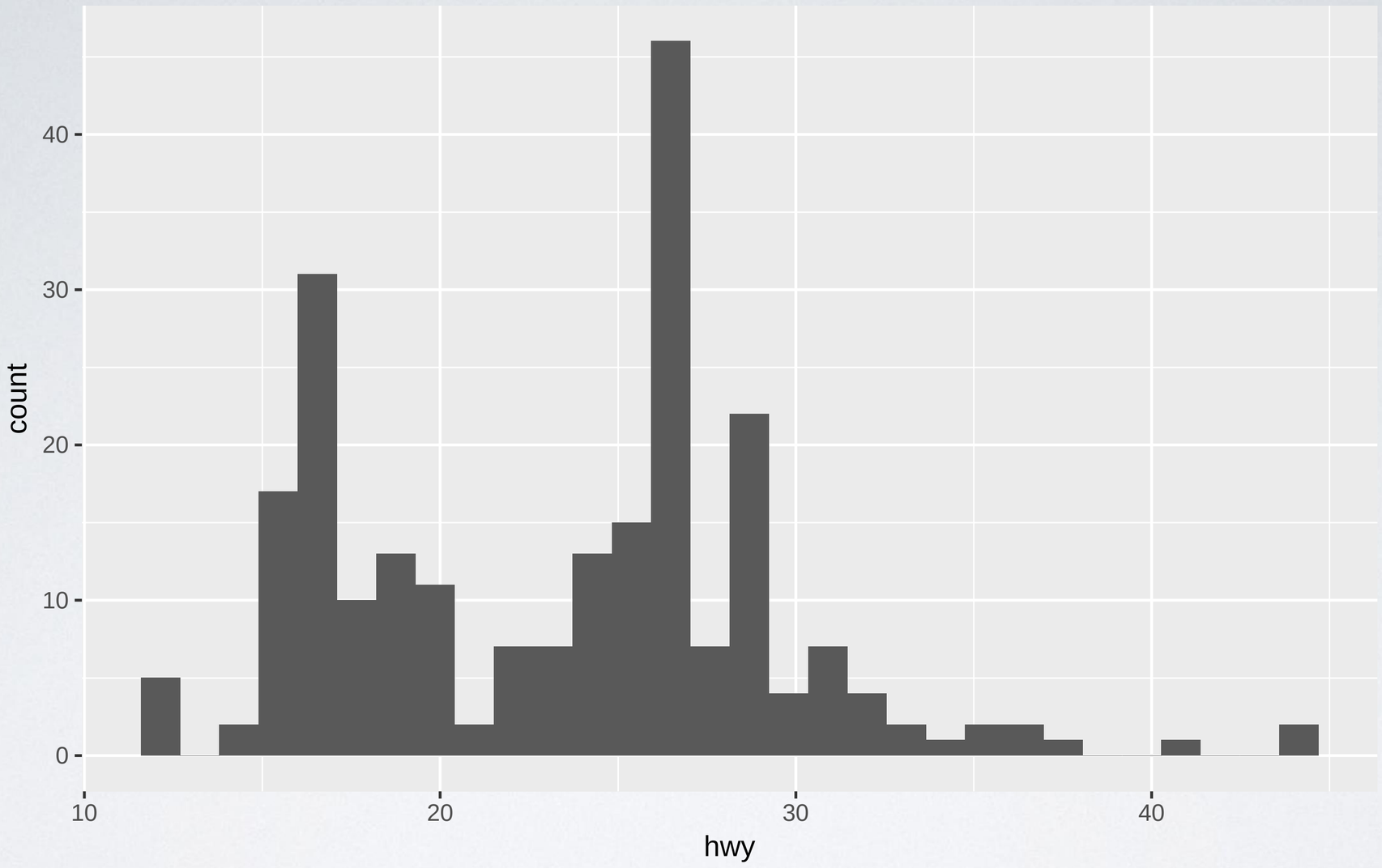

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = class, y = hwy))
```

Your Turn 5

Make the histogram of **hwy** below. Use the cheatsheet.

Hint: do not supply a **y** variable.

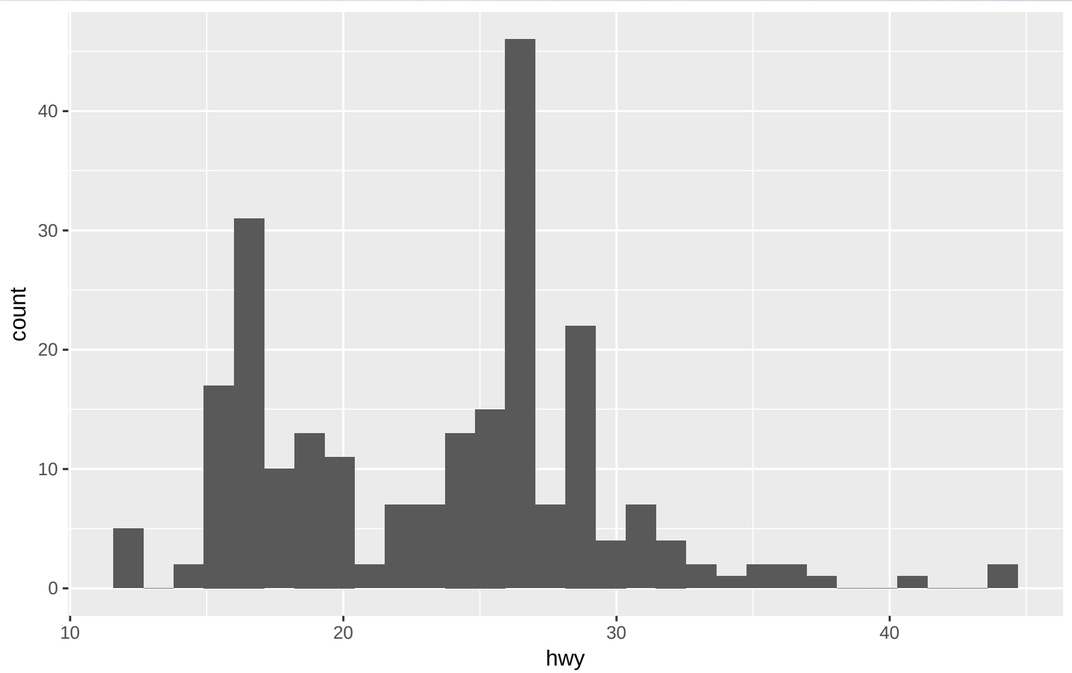


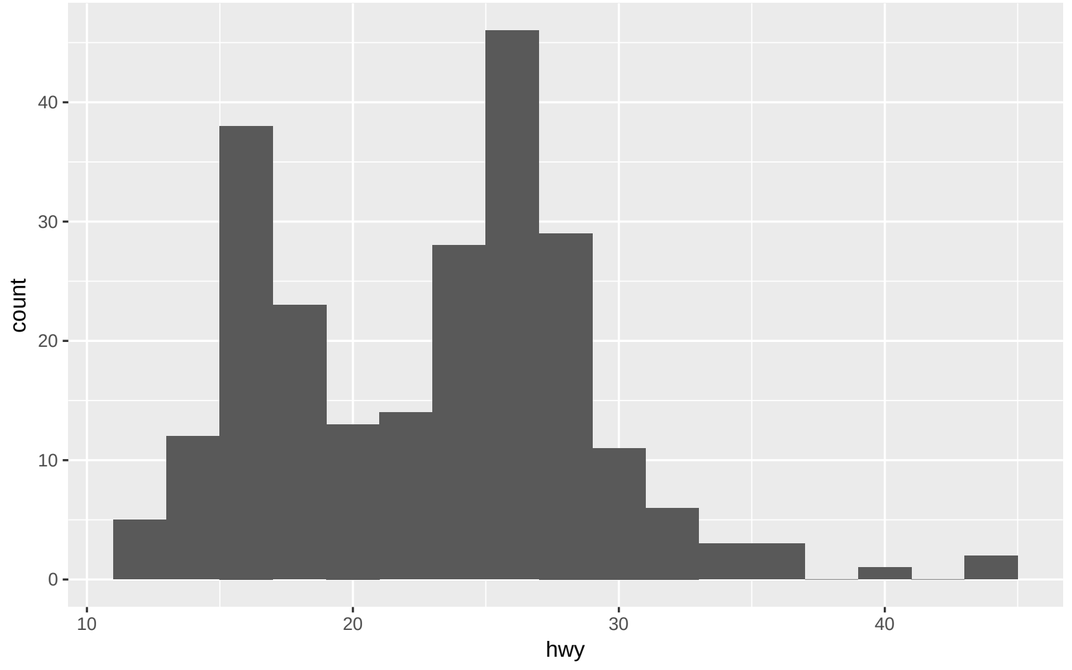



```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy))
```

Quiz

What is the difference?





"Help" pages

To open the documentation for a function, type

```
?geom_histogram
```

?

function name (no parentheses)

geom_freqpoly {ggplot2} R Documentation

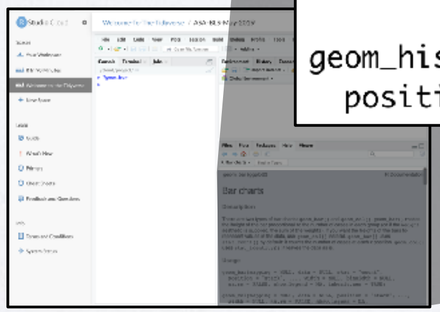
Histograms and frequency polygons

Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons (`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

Usage

```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)

geom_histogram(mapping = NULL, data = NULL, stat = "bin",
  position = "stack", ..., binwidth = NULL, bins = NULL,
```



Tips

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

geom_freqpoly {ggplot2} R Documentation

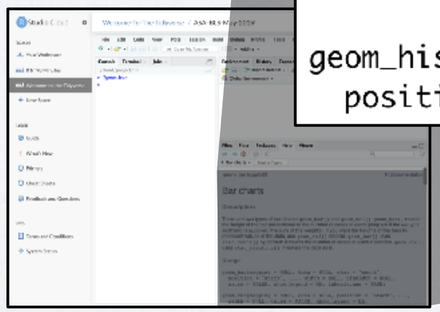
Histograms and frequency polygons

Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (geom_histogram()) display the counts with bars; frequency polygons (geom_freqpoly()) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

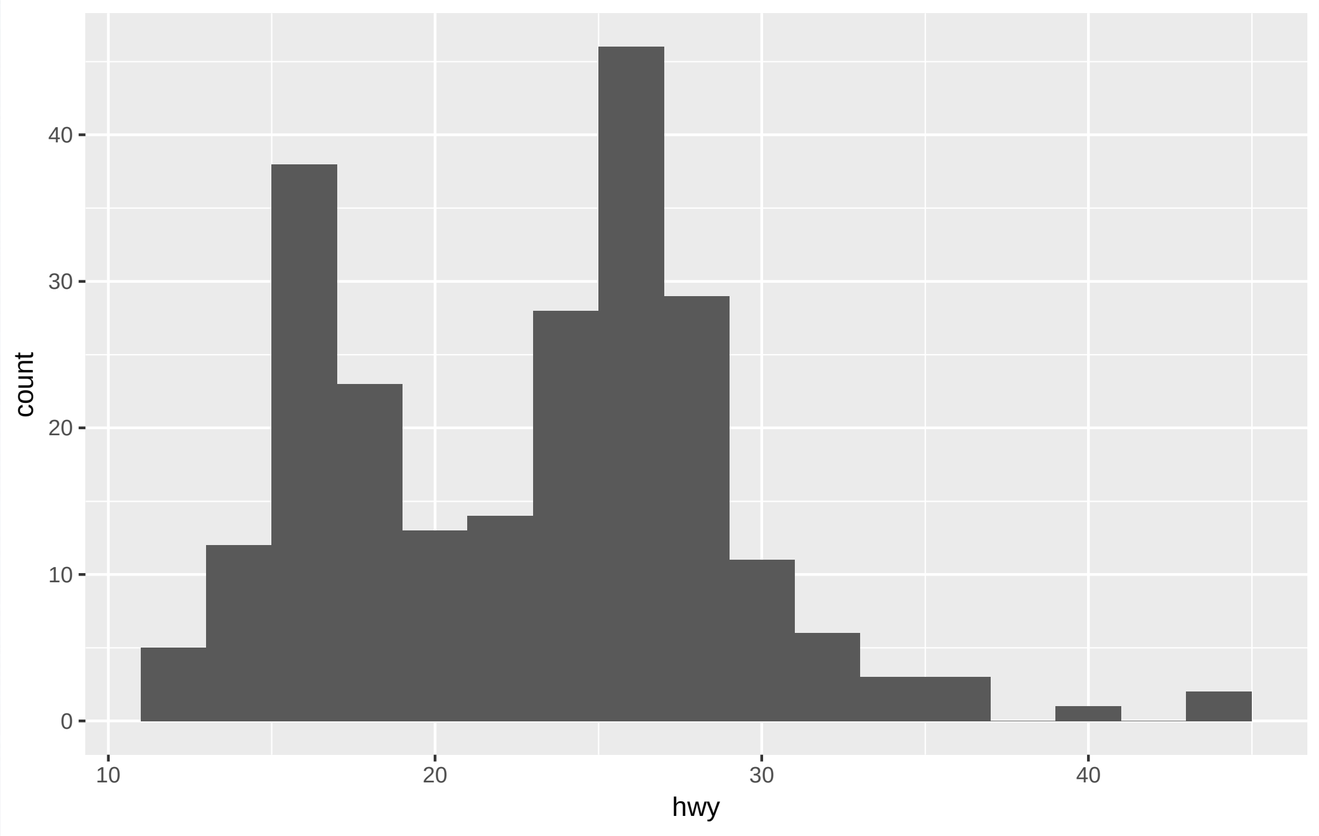
Usage

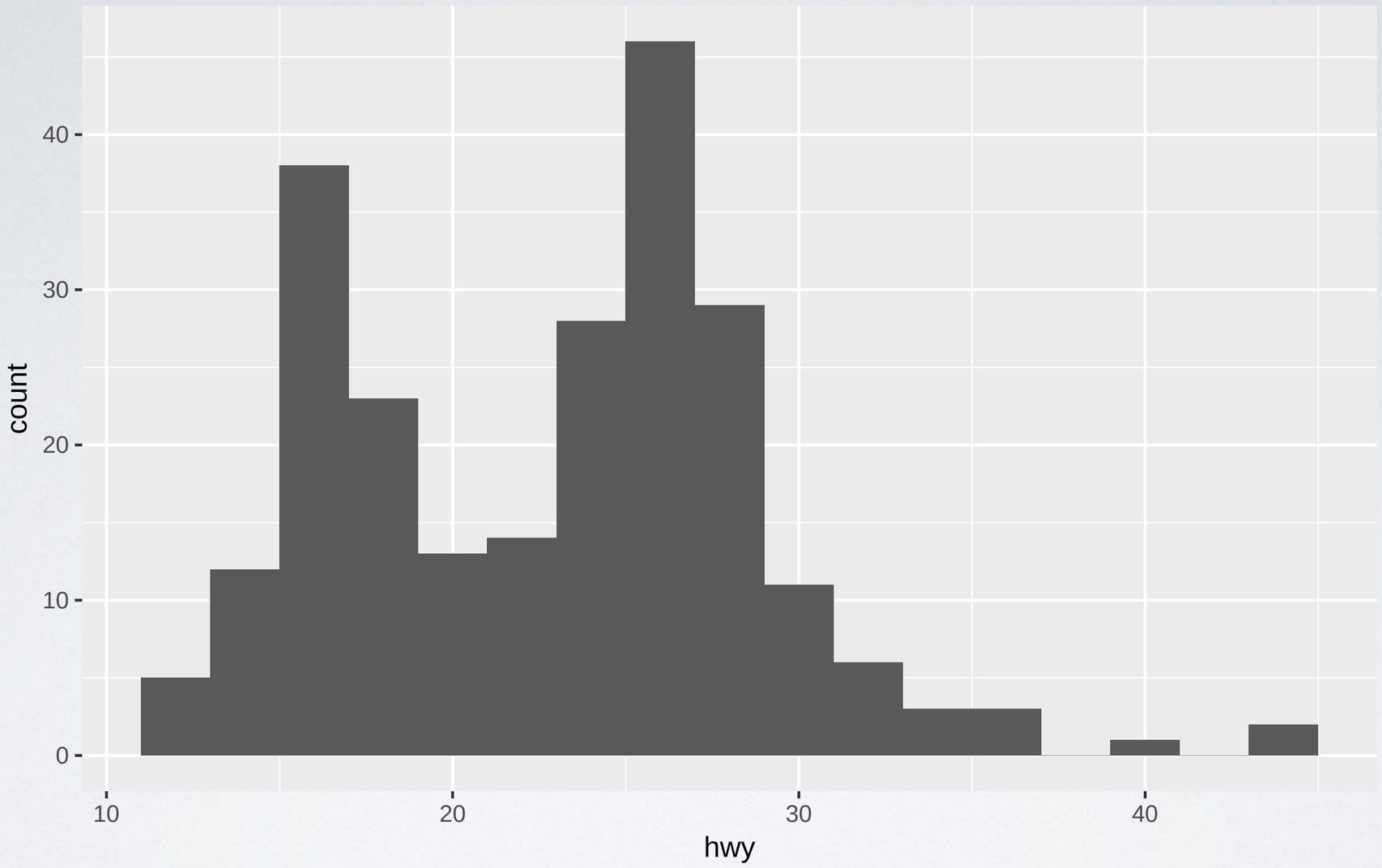
```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)

geom_histogram(mapping = NULL, data = NULL, stat = "bin",
  position = "stack", ..., binwidth = NULL, bins = NULL,
```



Your Turn 6

Use the help page for **geom_histogram** to make the bins **2** mpg wide.






```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy), binwidth = 2)
```

ggplot2.tidyverse.org/

Create Elegant Data Visualisati ×

Garrett

ggplot2.tidyverse.org

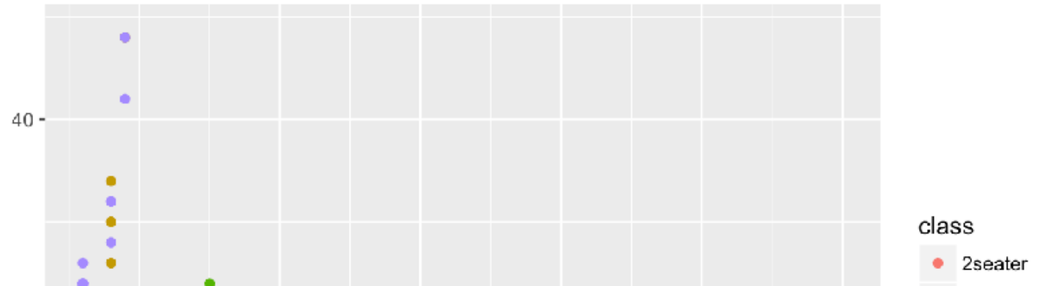
ggplot2 part of the tidyverse

Reference Articles News

Usage

It's hard to succinctly describe how ggplot2 works because it embodies a deep philosophy of visualisation. However, in most cases you start with `ggplot()`, supply a dataset and aesthetic mapping (with `aes()`). You then add on layers (like `geom_point()` or `geom_histogram()`), scales (like `scale_colour_brewer()`), faceting specifications (like `facet_wrap()`) and coordinate systems (like `coord_flip()`).

```
library(ggplot2)

ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
```



Links

Download from CRAN at https://cran.r-project.org/package=ggplot2

Browse source code at https://github.com/tidyverse/ggplot2

Report a bug at https://github.com/tidyverse/ggplot2/issues

Learn more at http://r4ds.had.co.nz/data-visualisation.html

License

GPL-2 | file LICENSE

Developers

Hadley Wickham

Author, maintainer

r-statistics.co

Top 50 ggplot2 Visualizations -

r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-

r-statistics.co by Selva Prabhakaran

Search..

- **Tutorial**

R Tutorial

ggplot2

ggplot2 Short Tutorial
ggplot2 Tutorial 1 - Intro
ggplot2 Tutorial 2 - Theme
ggplot2 Tutorial 3 - Masterlist
ggplot2 Quickref

Foundations

Linear Regression
Statistical Tests
Missing Value Treatment
Outlier Analysis
Feature Selection
Model Selection
Logistic Regression
Advanced Linear Regression

Advanced Regression Models

Advanced Regression Models

Time Series

Time Series Analysis
Time Series Forecasting
More Time Series Forecasting

Top 50 ggplot2 Visualizations - The Master List (With Full R Code)

> What type of visualization to use for what sort of problem? This tutorial helps you choose the right type of chart for your specific objectives and how to implement it in R using ggplot2.

This is part 3 of a three part tutorial on ggplot2, an aesthetically pleasing (and very popular) graphics framework in R. This tutorial is primarily geared towards those having some basic knowledge of the R programming language and want to make complex and nice looking charts with R ggplot2.

- Part 1: Introduction to ggplot2, covers the basic knowledge about constructing simple ggplots and modifying the components and aesthetics.
- Part 2: Customizing the Look and Feel, is about more advanced customization like manipulating legend, annotations, multiplots with faceting and custom layouts
- Part 3: Top 50 ggplot2 Visualizations - The Master List, applies what was learnt in part 1 and 2 to construct other types of ggplots such as bar charts, boxplots etc.

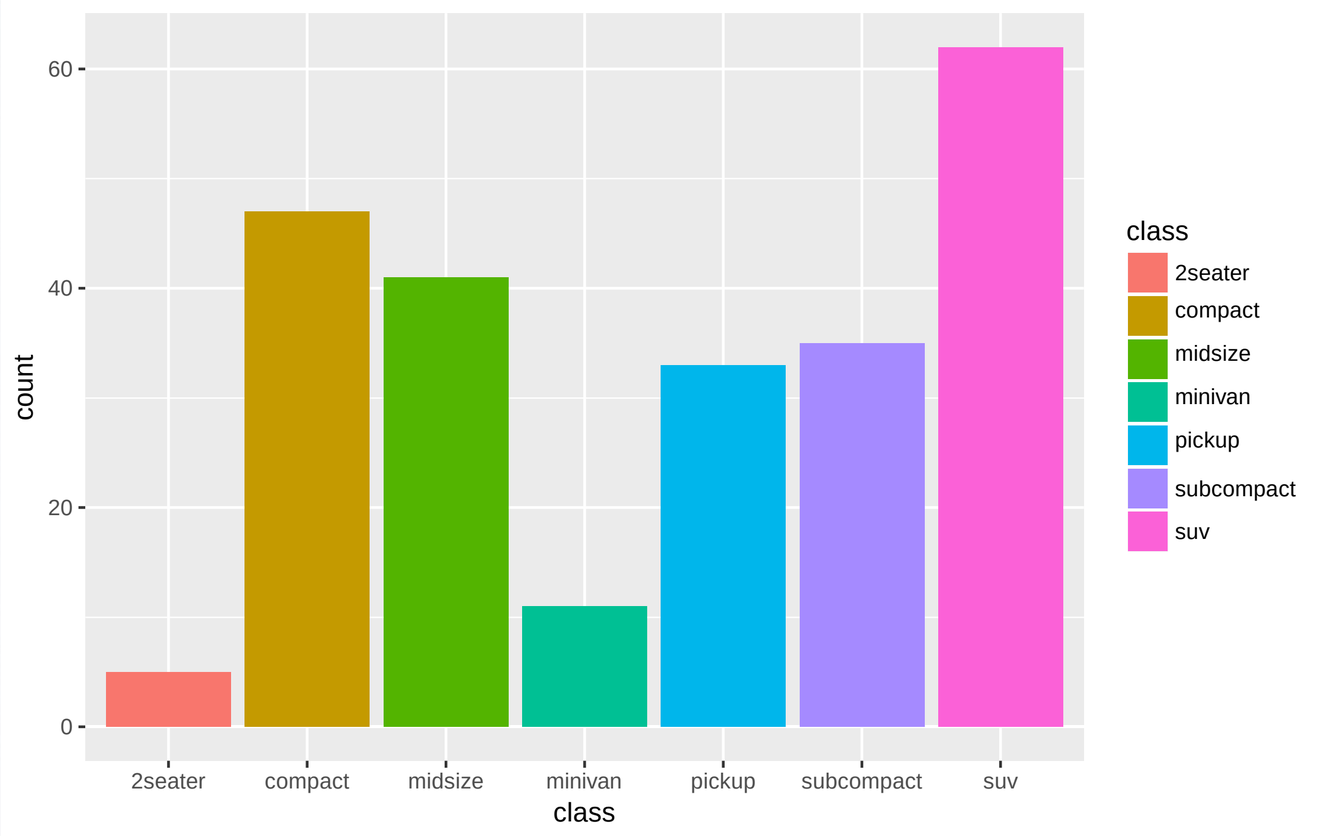
Top 50 ggplot2 Visualizations - The Master List

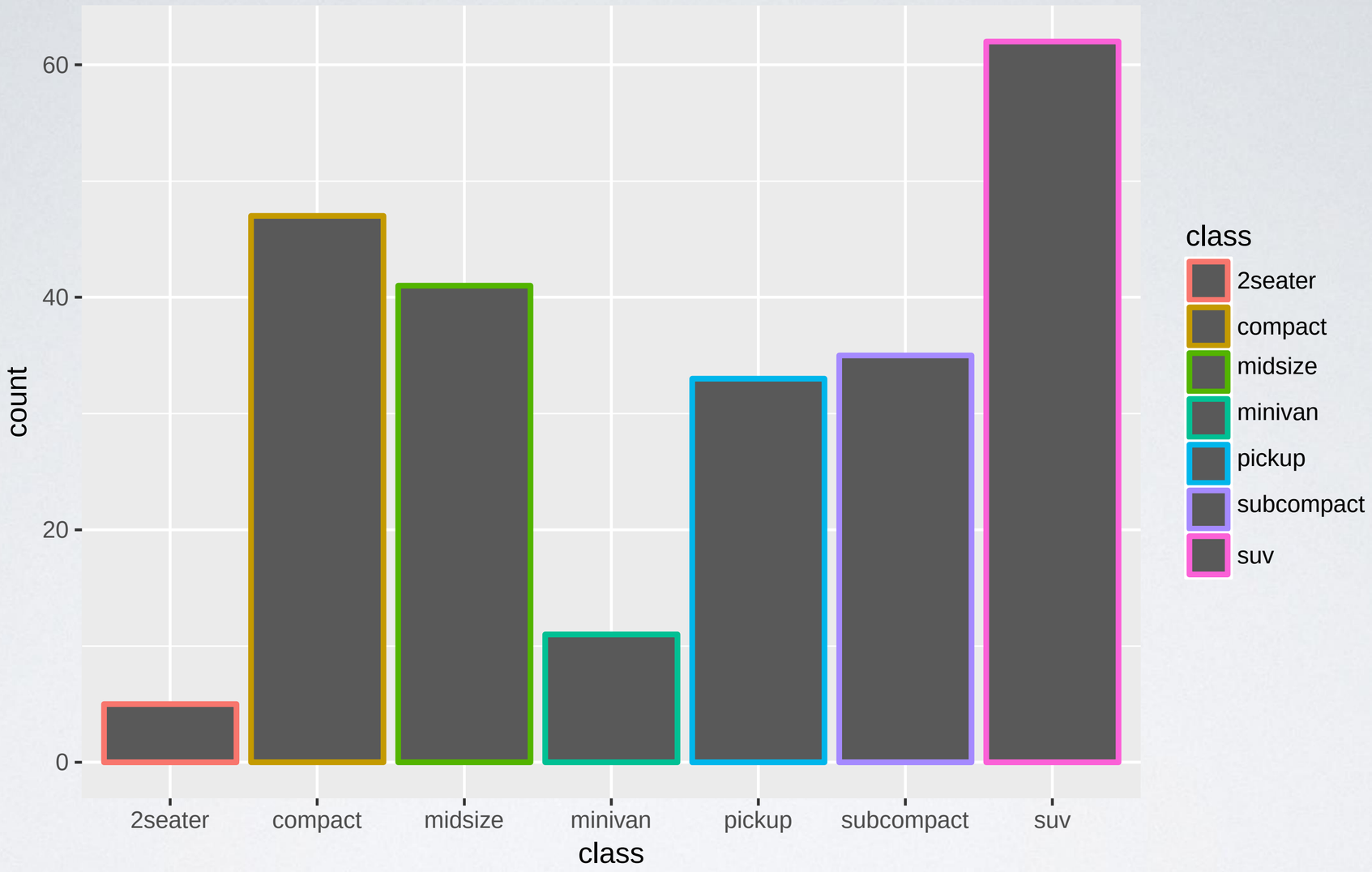
An effective chart is one that:

1. Conveys the right information without distorting facts.
2. Is simple but elegant. It should not force you to think much in order to get it.
3. Aesthetics supports information rather that overshadow it.
4. Is not overloaded with information.

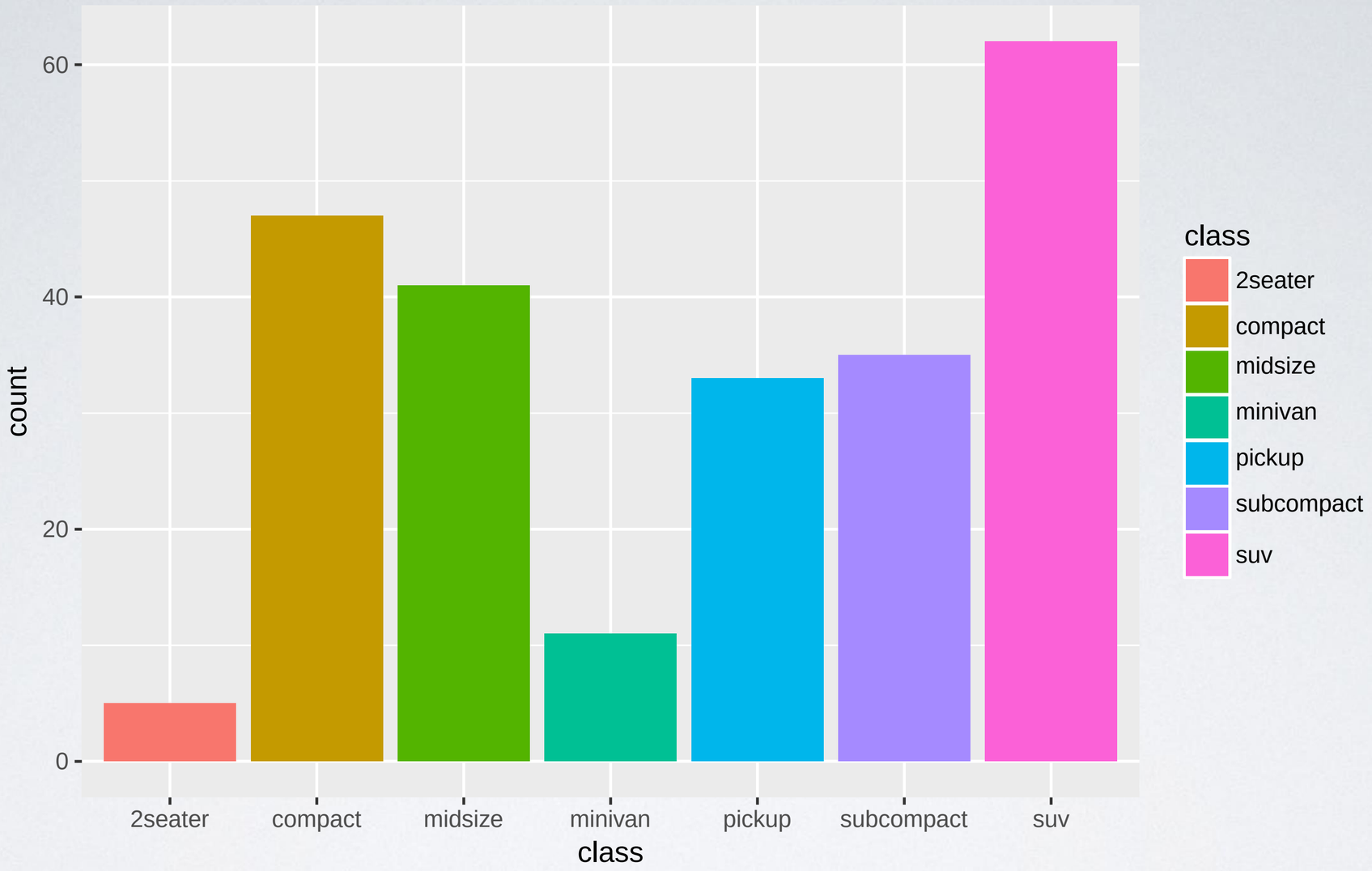
Your Turn 7

With a partner, make the bar chart of **class** below. Use the cheatsheet. Hint: do not supply a **y** variable.

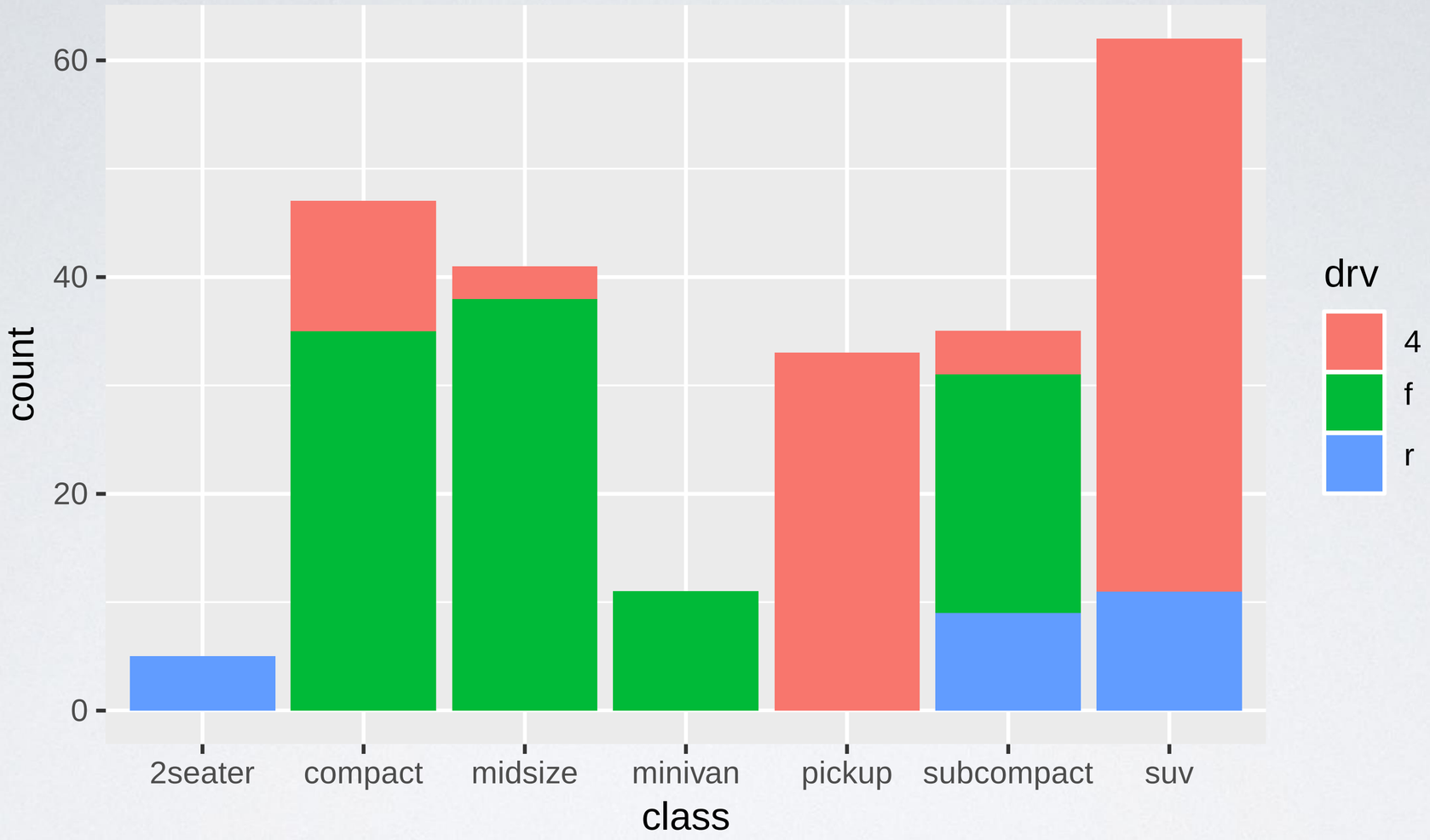





```
ggplot(data = mpg) +
  geom_bar(mapping = aes(x = class, color = class))
```




```
ggplot(data = mpg) +
  geom_bar(mapping = aes(x = class, fill = class))
```

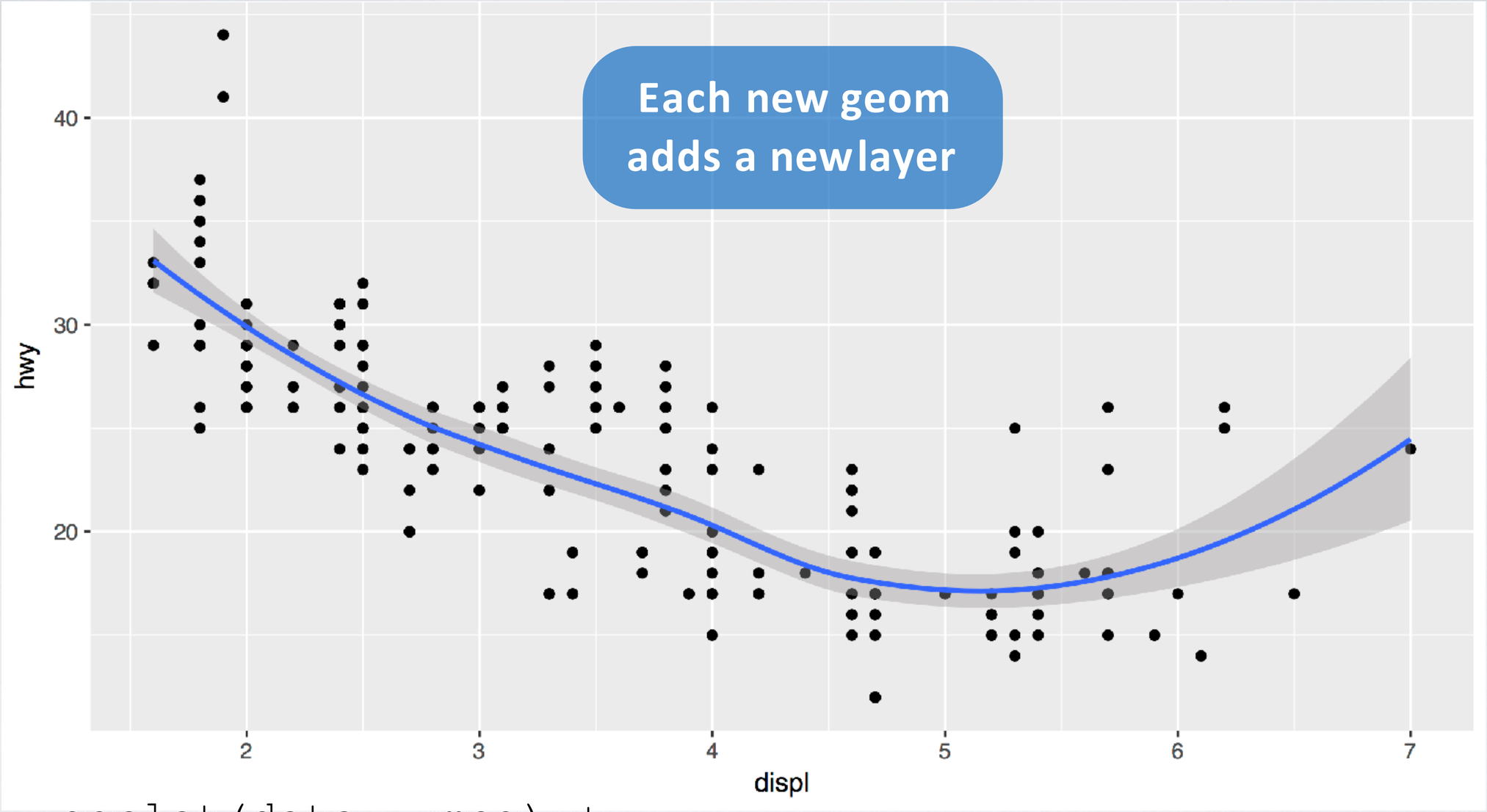



```
ggplot(data = mpg) +
  geom_bar(mapping = aes(x = class, fill = drv))
```

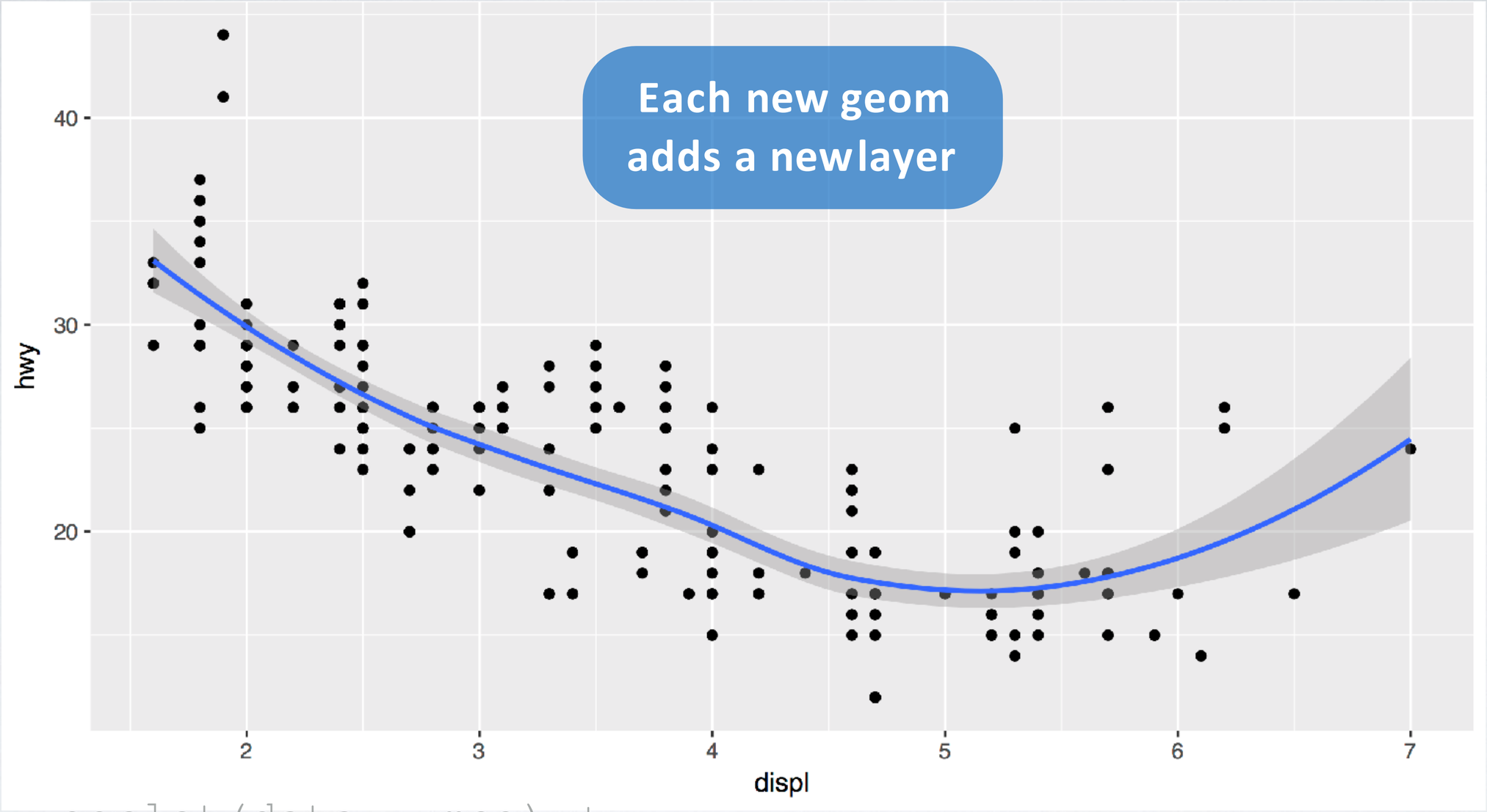
Quiz

What will this code do?

```
ggplot(mpg) +
  geom_point(aes(displ, hwy)) +
  geom_smooth(aes(displ, hwy))
```

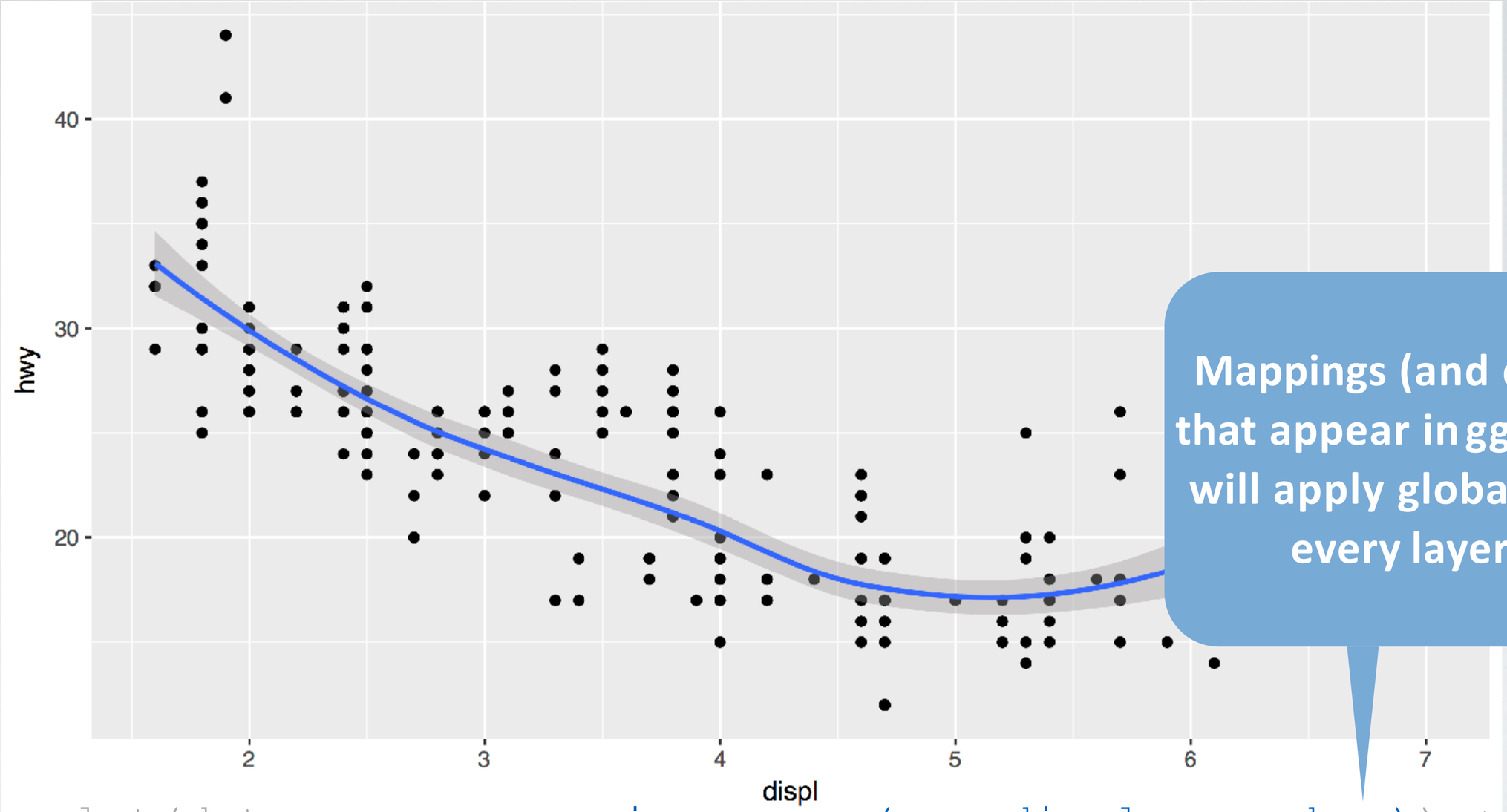



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



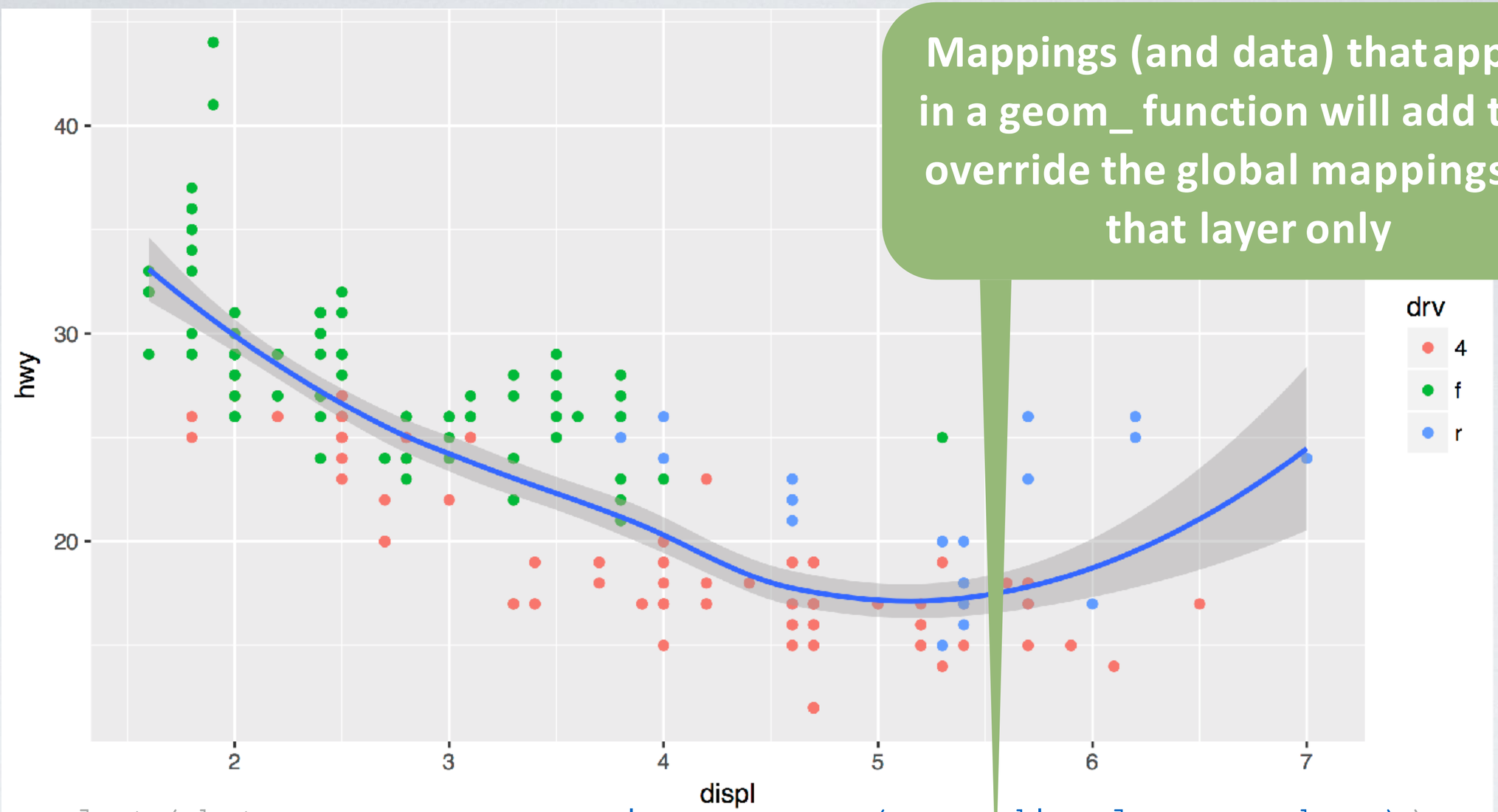

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

global vs. local



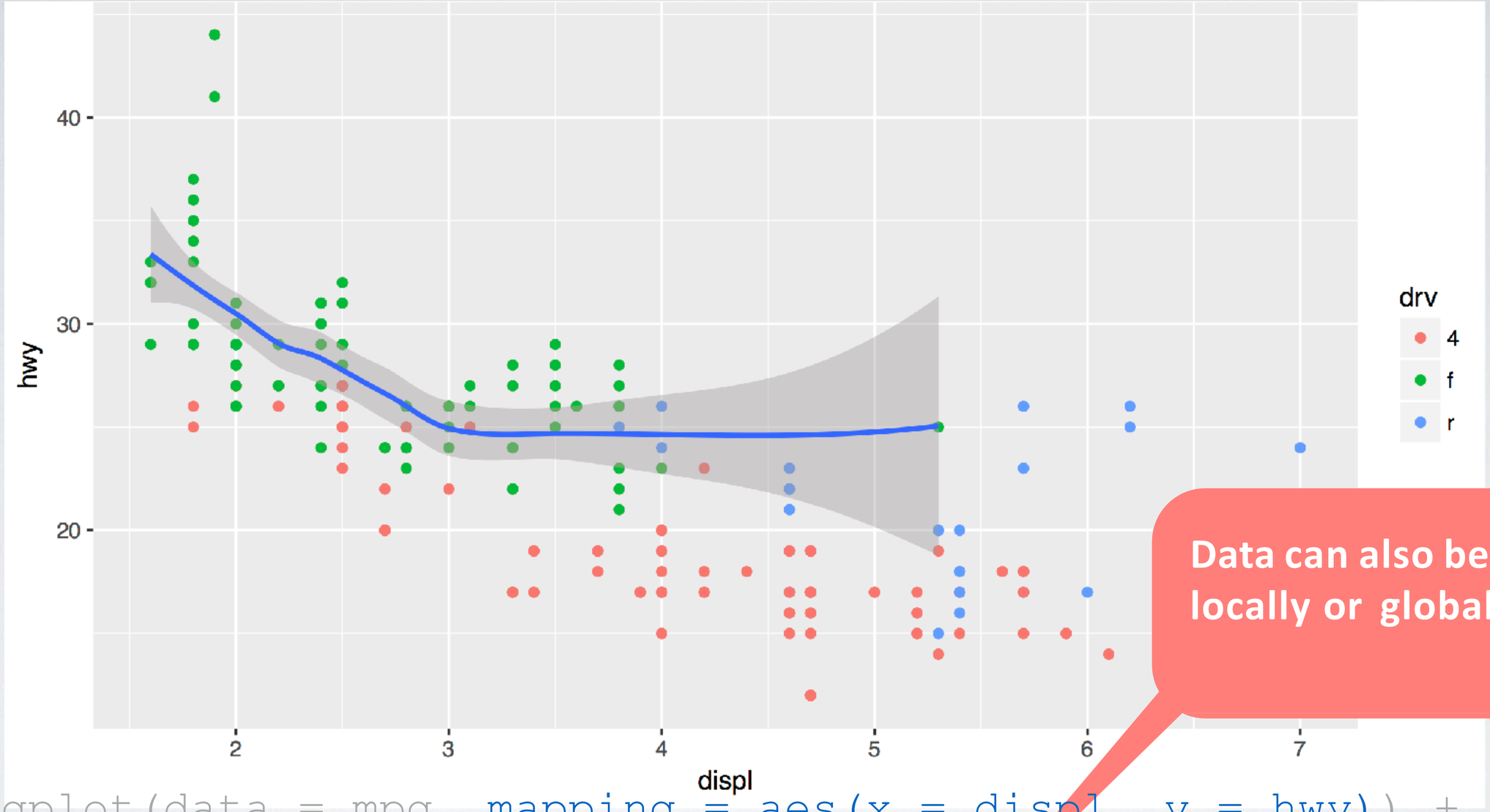
Mappings (and data) that appear in ggplot() will apply globally to every layer

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```



Mappings (and data) that appear in a geom_ function will add to or override the global mappings for that layer only

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth()
```



Data can also be set locally or globally

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth(data = filter(mpg, drv == "f"))
```

Quiz

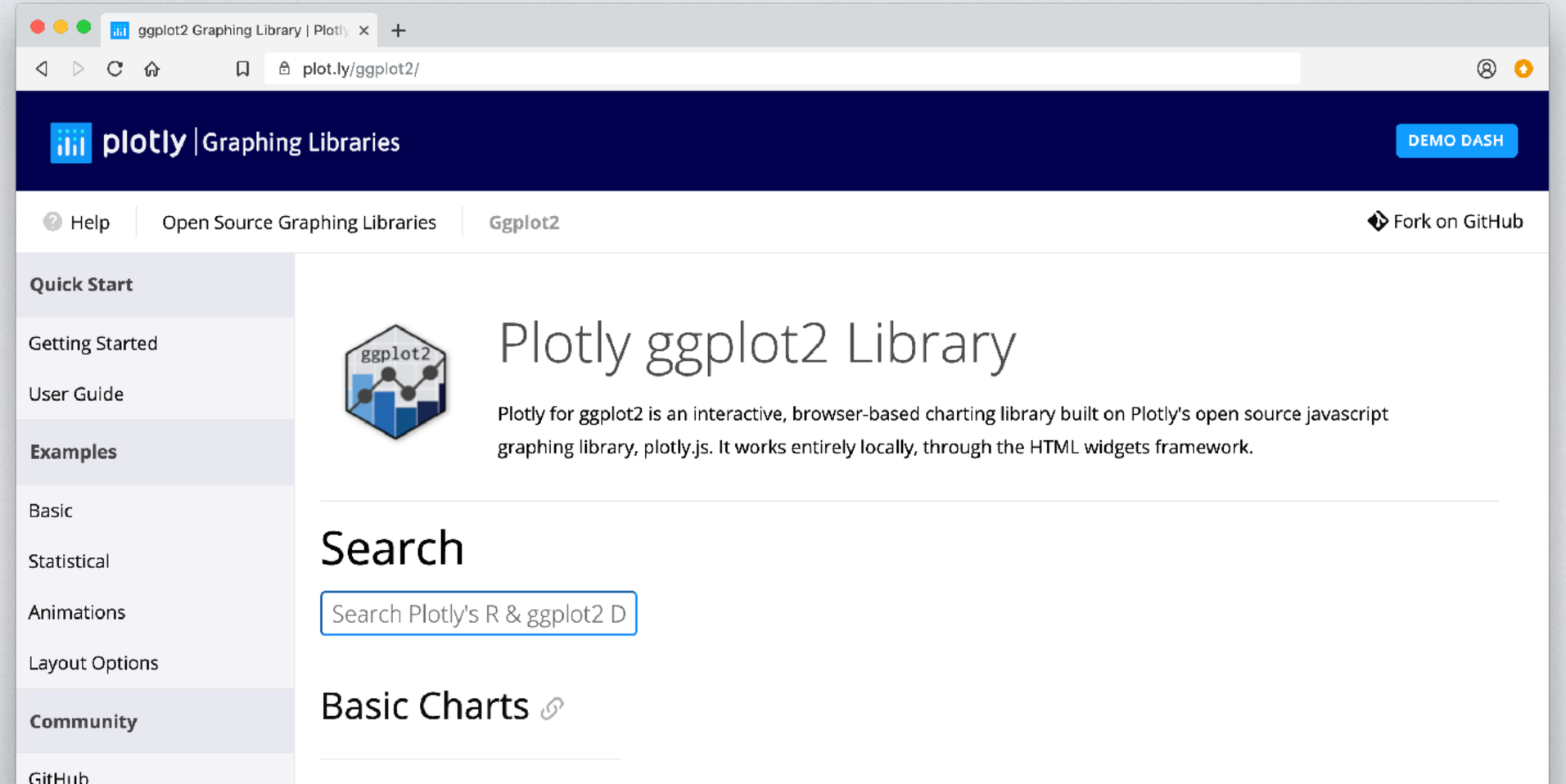
What is different about this plot? Run the code!

```
p <- ggplot(mpg) +
  geom_point(aes(displ, hwy)) +
  geom_smooth(aes(displ, hwy))

library(plotly)
ggplotly(p)
```

interactivity

Plotly

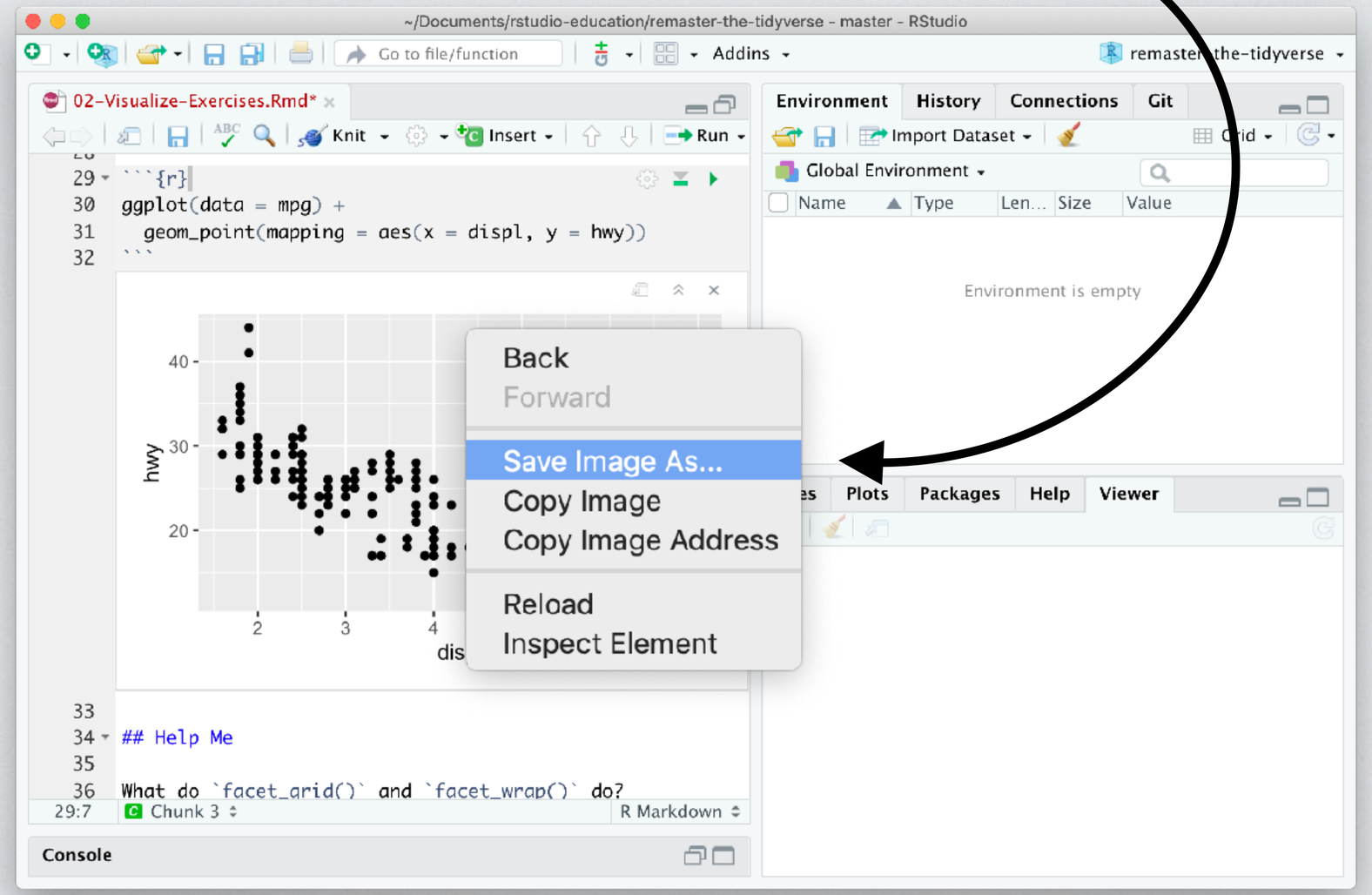
Tools for making interactive plots. plot.ly/ggplot2/



Saving graphs

GUI method

Right click on the plot



Code method

ggsave() saves the last plot.

Uses size on screen:

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

Specify size in inches

```
ggsave("my-plot.pdf", width =
```

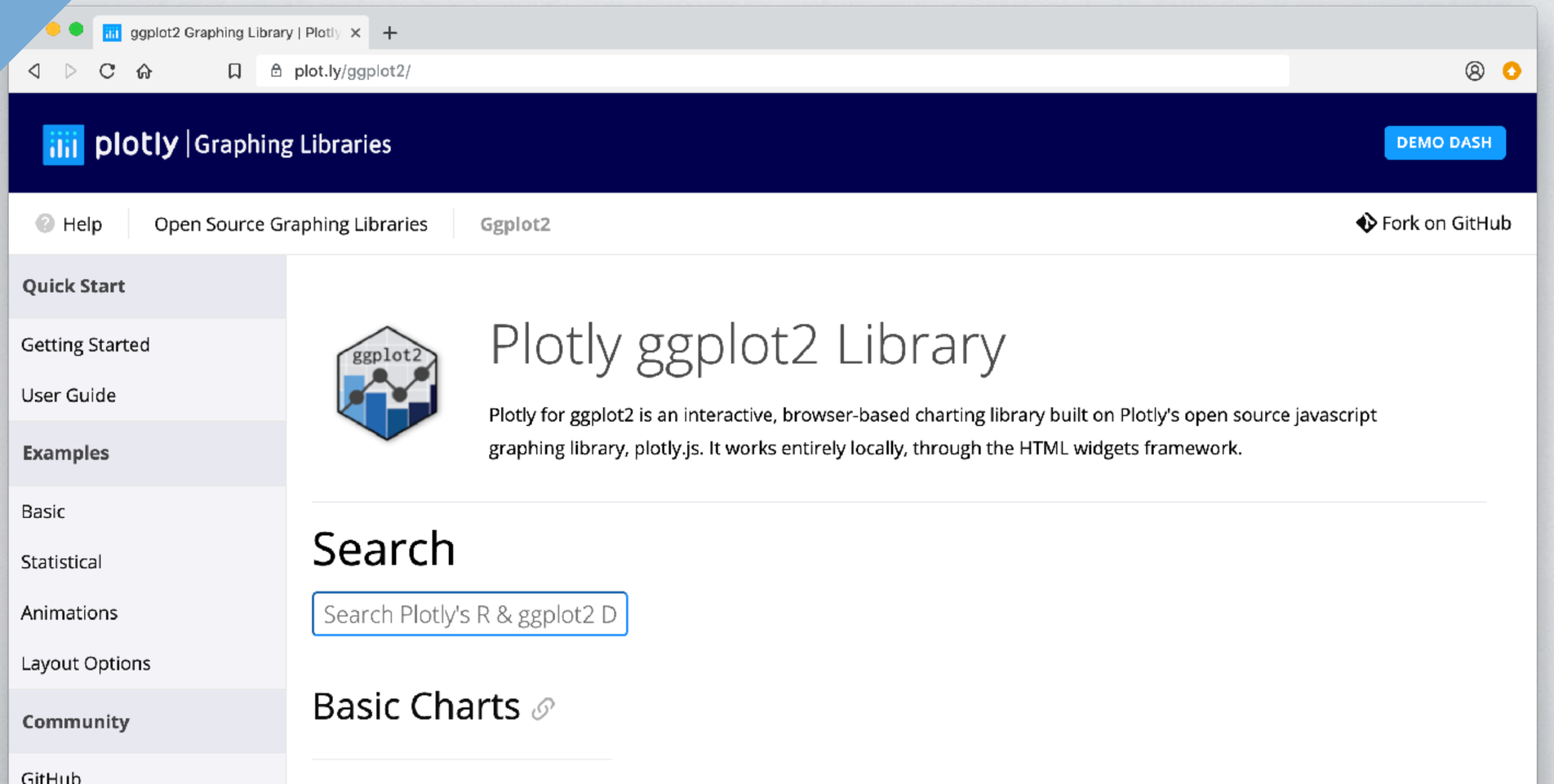
Q: But where will it save it?

A: Alongside your .Rmd

Save with htmlwidgets::saveWidget()

Plotly

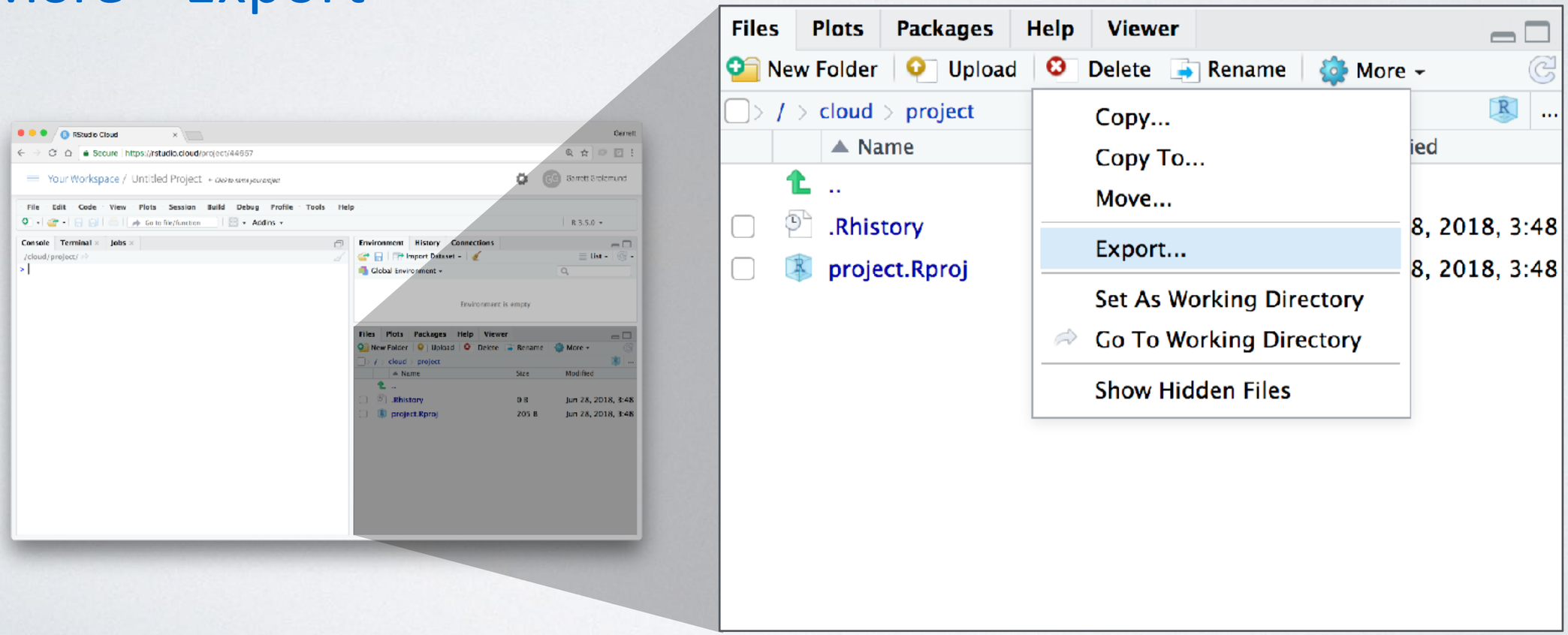
s for making interactive plots. plot.ly/ggplot2/



Download files

In the files pane, check next to the file(s) to download

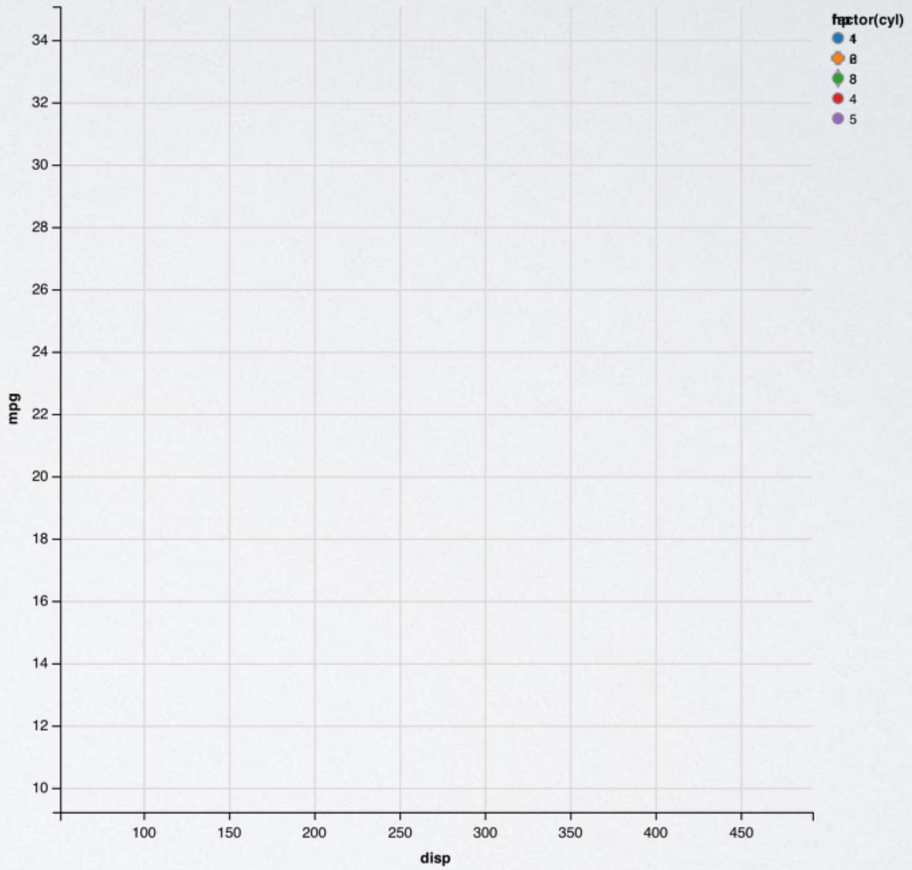
More > Export



Grammar of Graphics

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

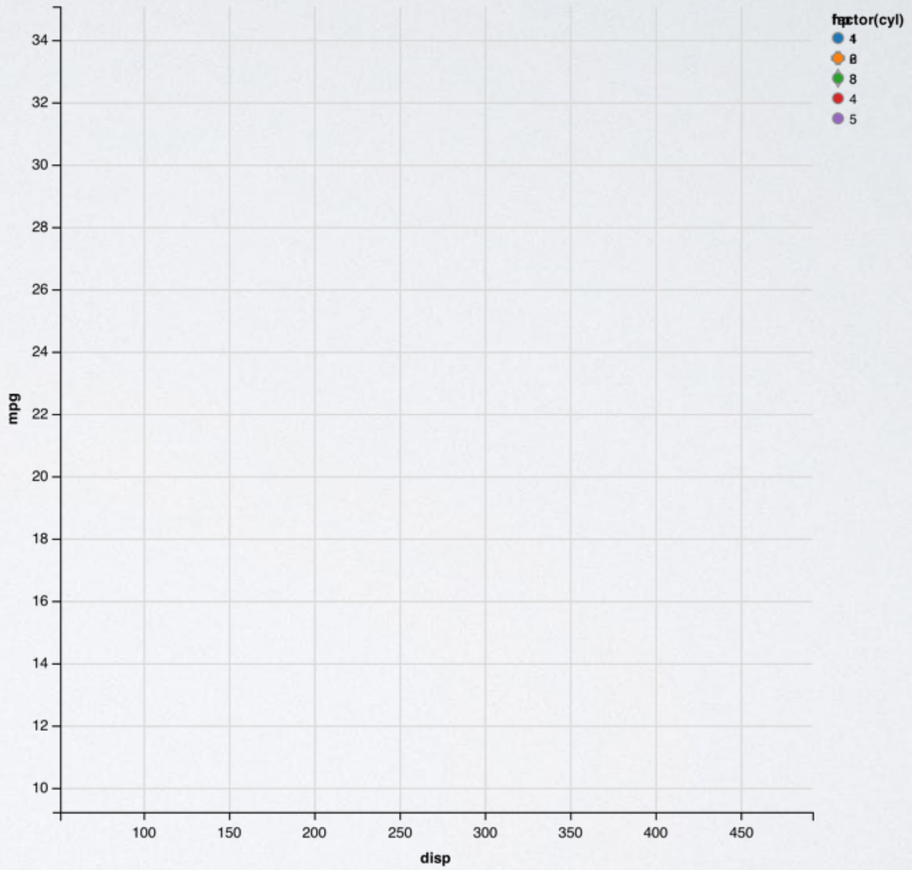
data **geom**



mappings

fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

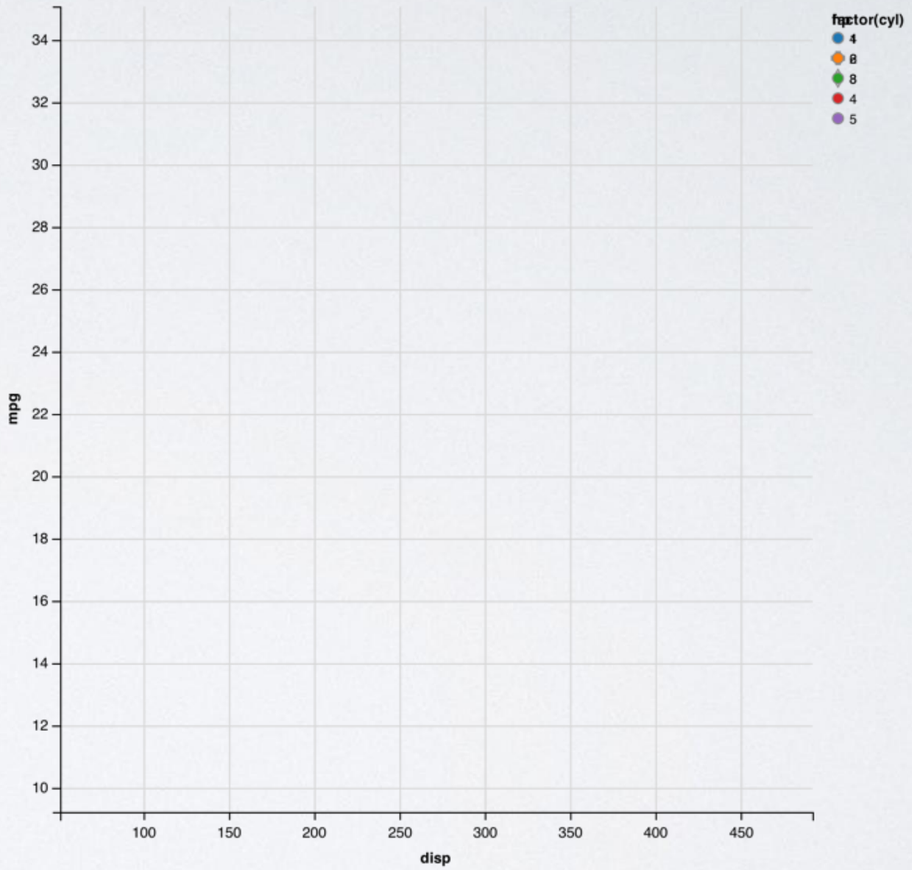


data **geom**

mappings

shape fill

mpg	cyl	disp	hp
21.0	6 ✚	160.0	2
21.0	6 ✚	160.0	2
22.8	4 ●	108.0	1
21.4	6 ✚	258.0	2
18.7	8 ♦	360.0	3
18.1	6 ✚	225.0	2
14.3	8 ♦	360.0	5
24.4	4 ●	146.7	1
22.8	4 ●	140.8	1
19.2	6 ✚	167.6	2
17.8	6 ✚	167.6	2
16.4	8 ♦	275.8	3
17.3	8 ♦	275.8	3
15.2	8 ♦	275.8	3
10.4	8 ♦	472.0	4
10.4	8 ♦	460.0	4
14.7	8 ♦	440.0	4
32.4	4 ●	78.7	1
30.4	4 ●	75.7	1
33.9	4 ●	71.1	1

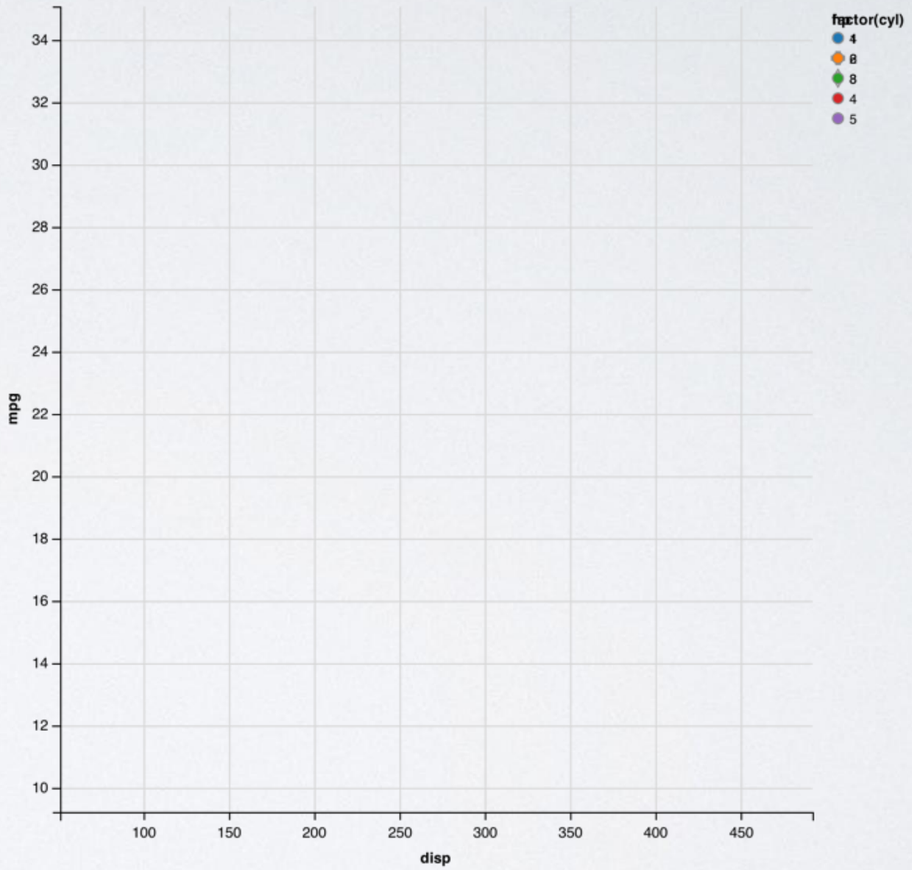


data **geom**

mappings

shape x fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1



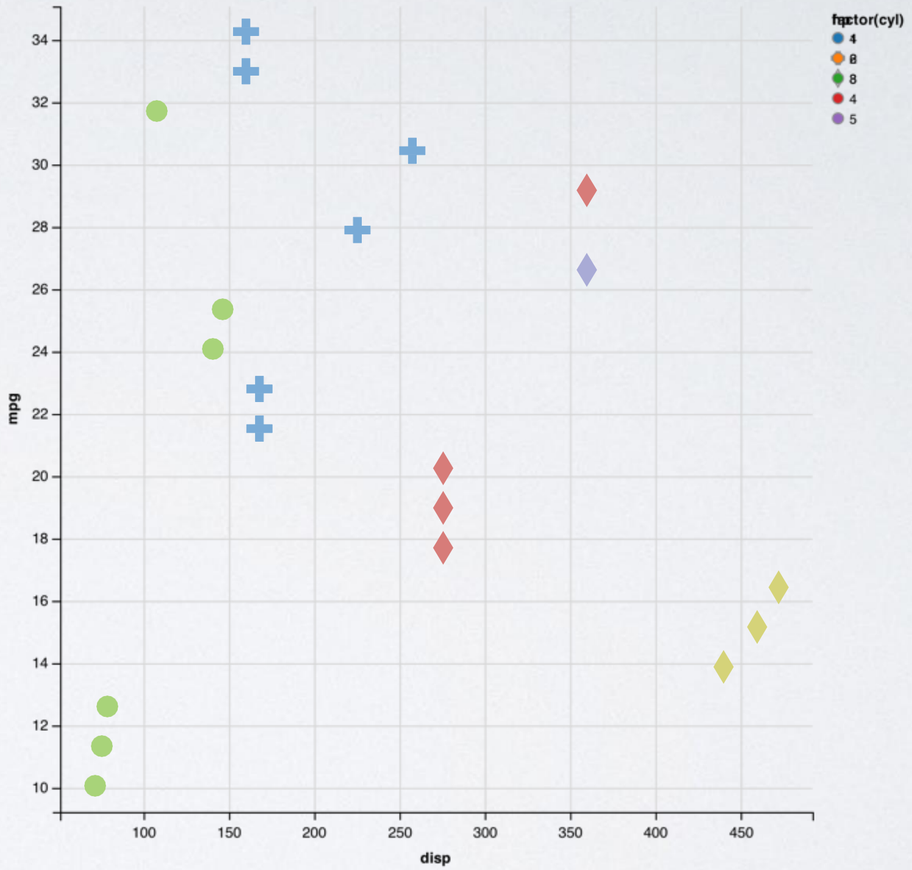
data **geom**

mappings

y	shape	x	fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data **geom**



mappings

y shape x fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data

geom

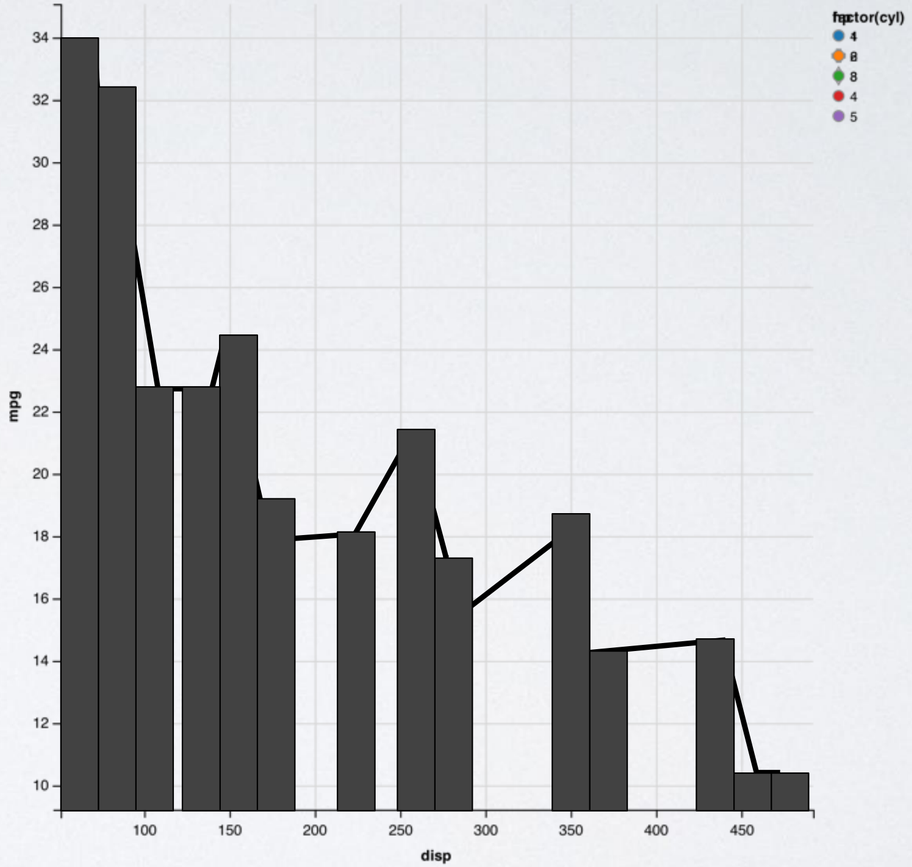
points

lines

mappings

y x

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1



data

geom

points

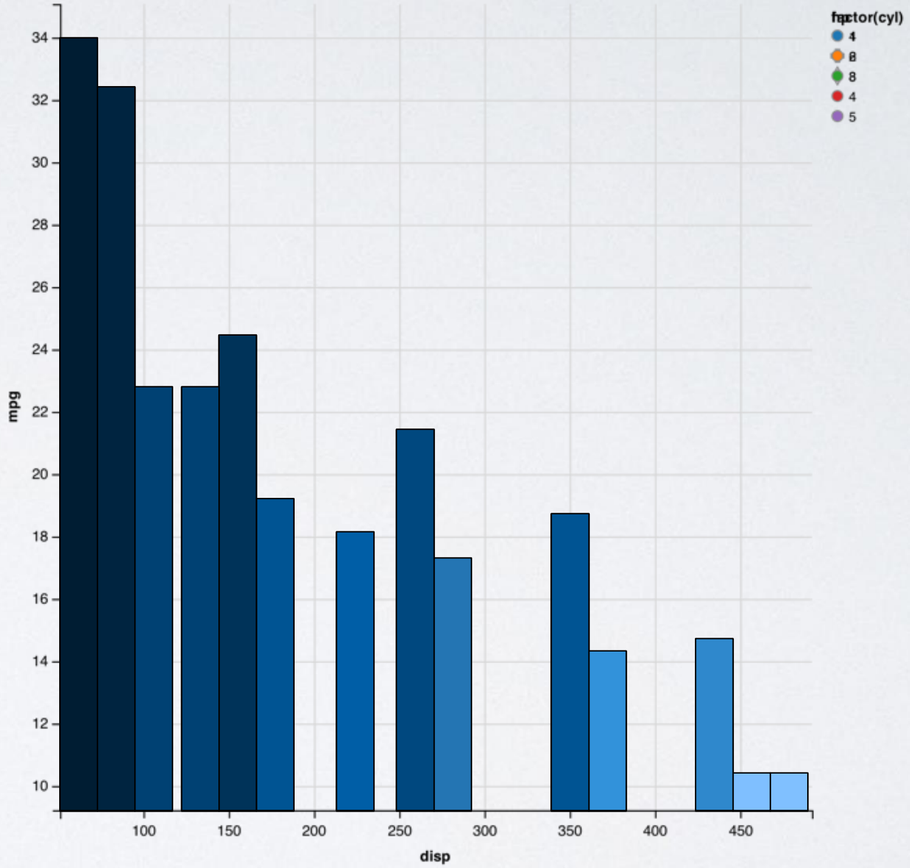
lines

bars

mappings

y fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1



data

geom

points

lines

bars

To make a graph

[template]

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

To make a graph

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data

1. Pick a **data**set

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

To make a graph

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data **geom**

1. Pick a **data**set

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a **geom** to display cases

To make a graph

mappings

fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data **geom**

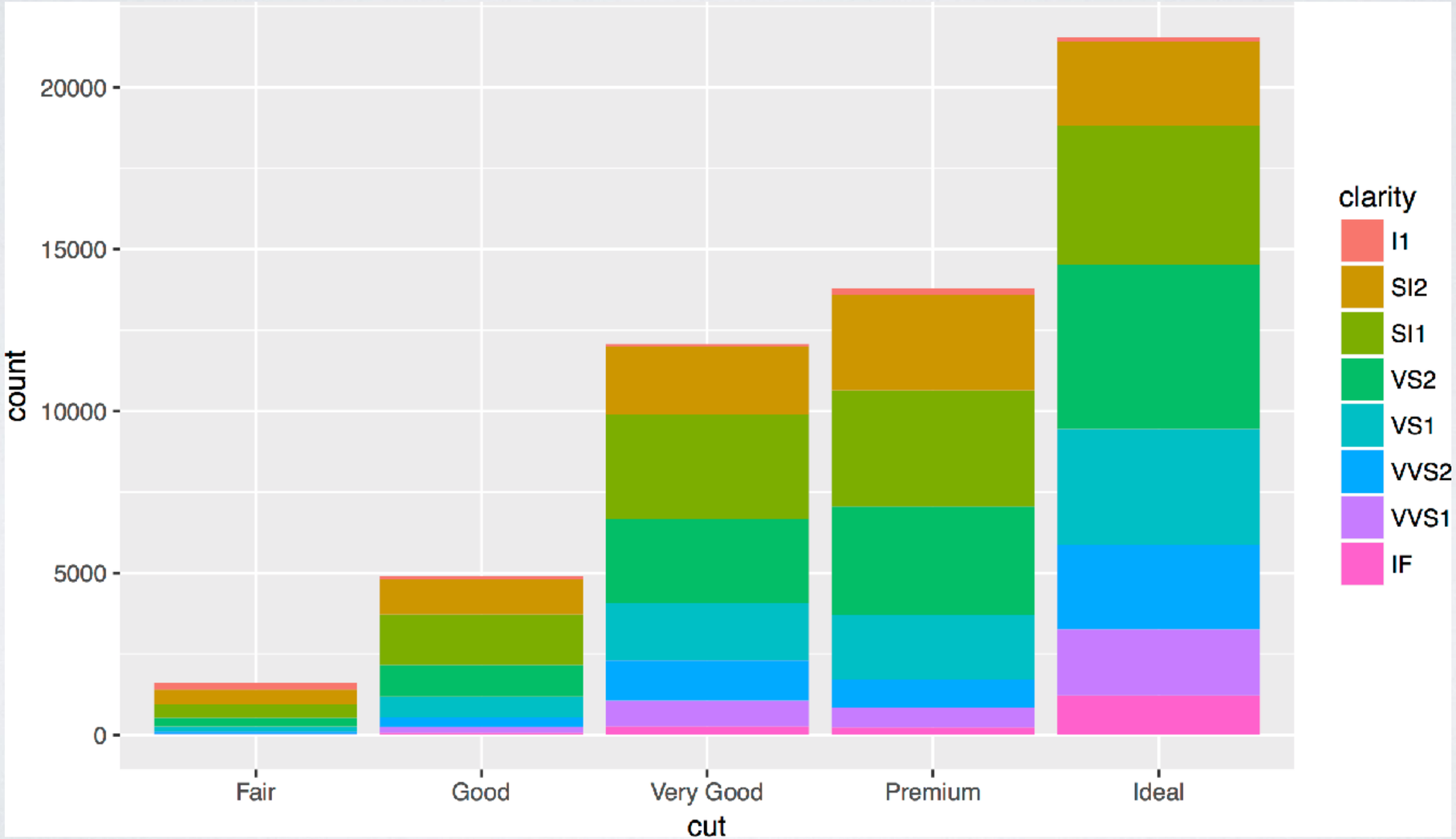
1. Pick a **data**set

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a **geom** to display cases

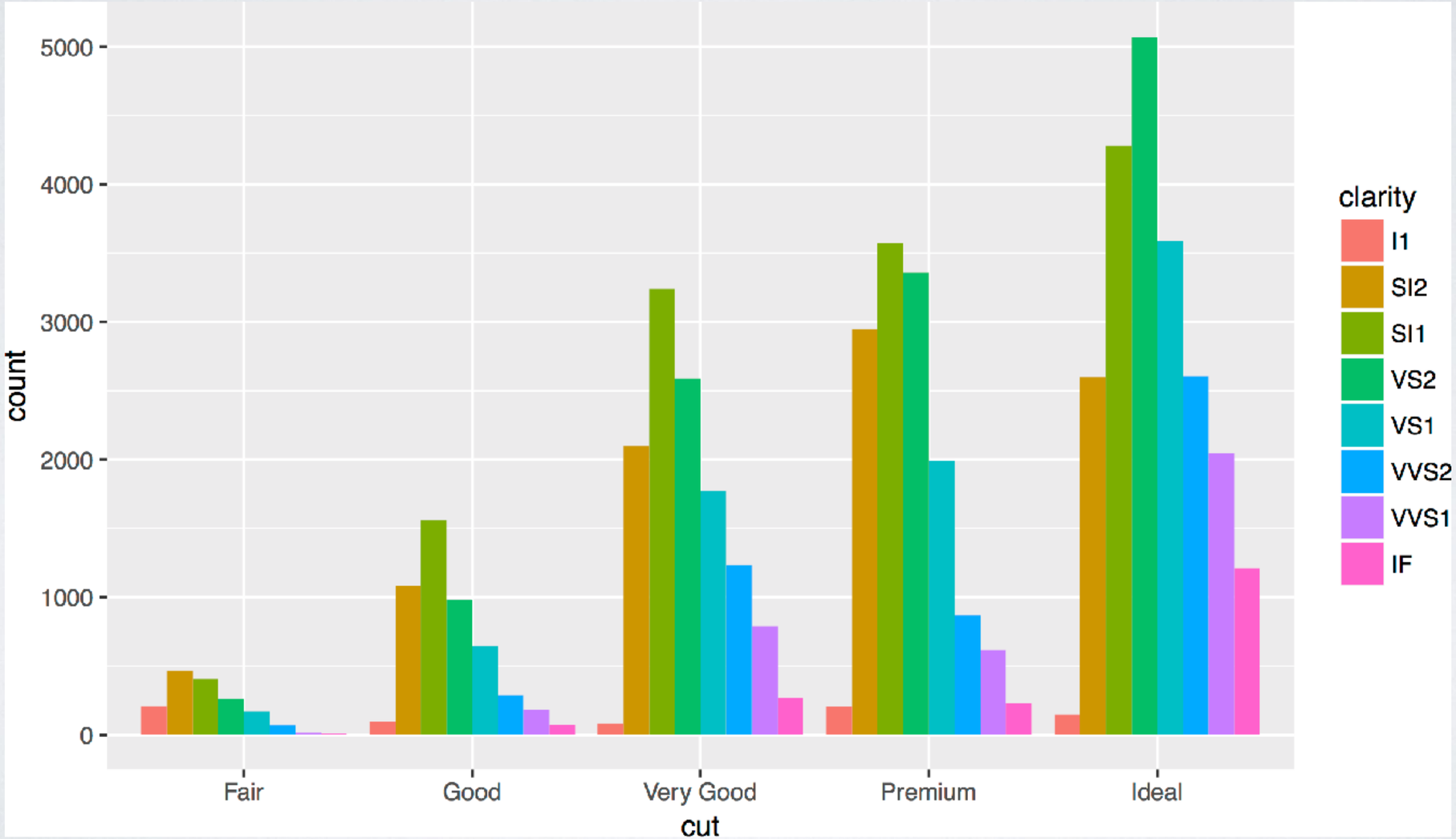
3. **Map** aesthetic properties to variables

What else?


20000
15000
10000
5000
0
count
Fair
Good
Very Good
Premium
Ideal
cut
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

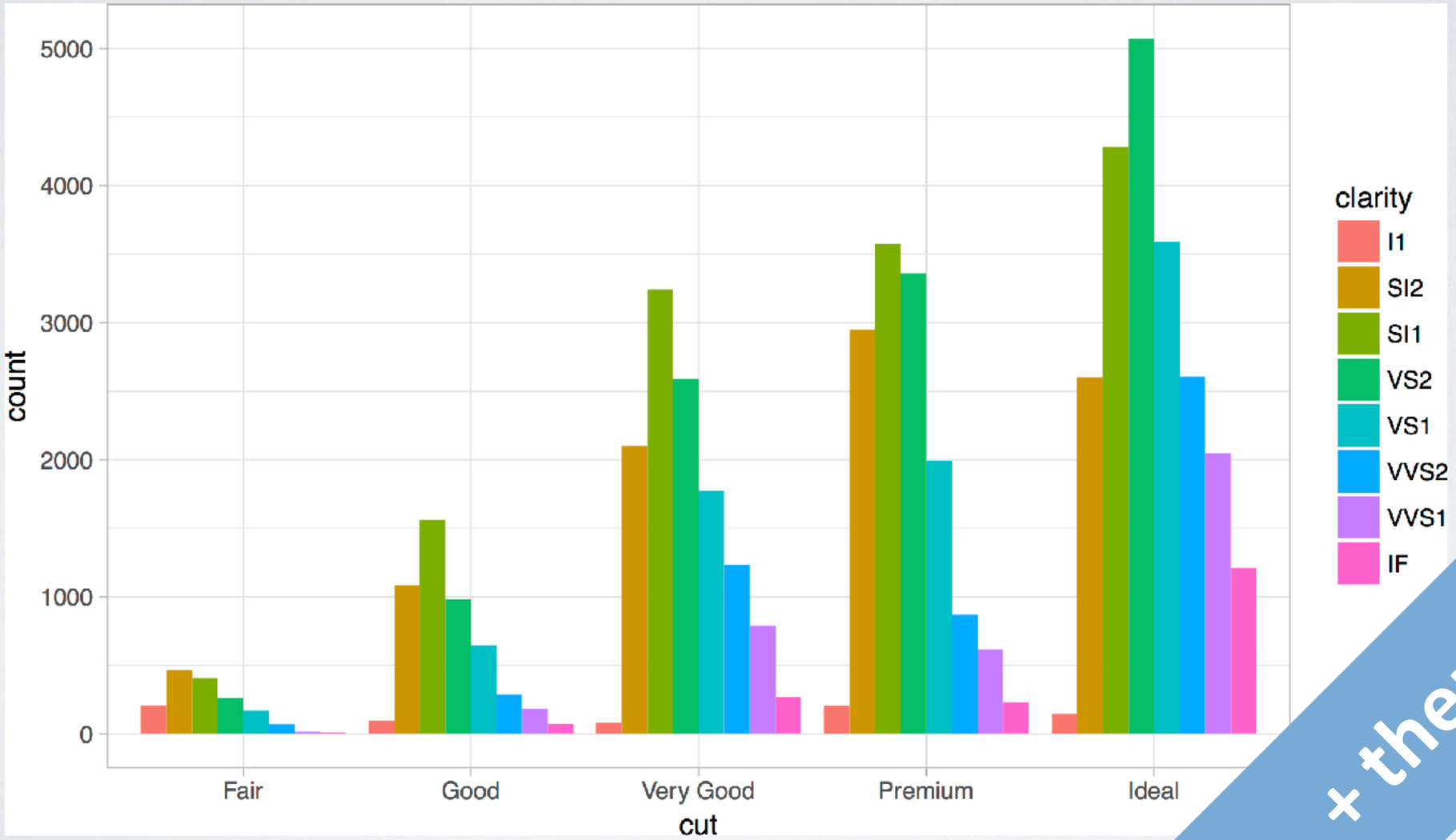
Position Adjustments

How overlapping objects are arranged



Themes

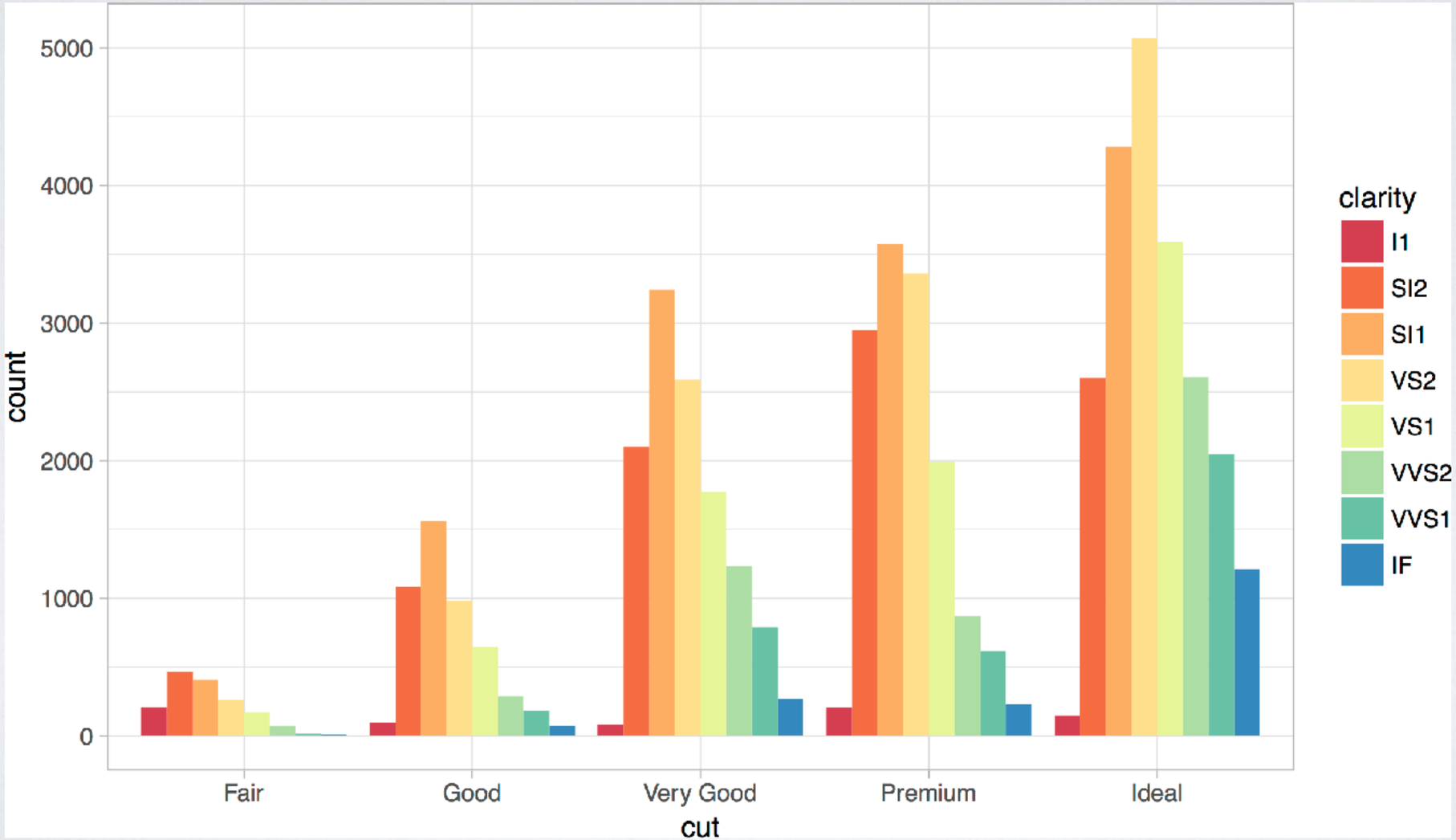
Visual appearance of non-data elements



+ theme_bw()

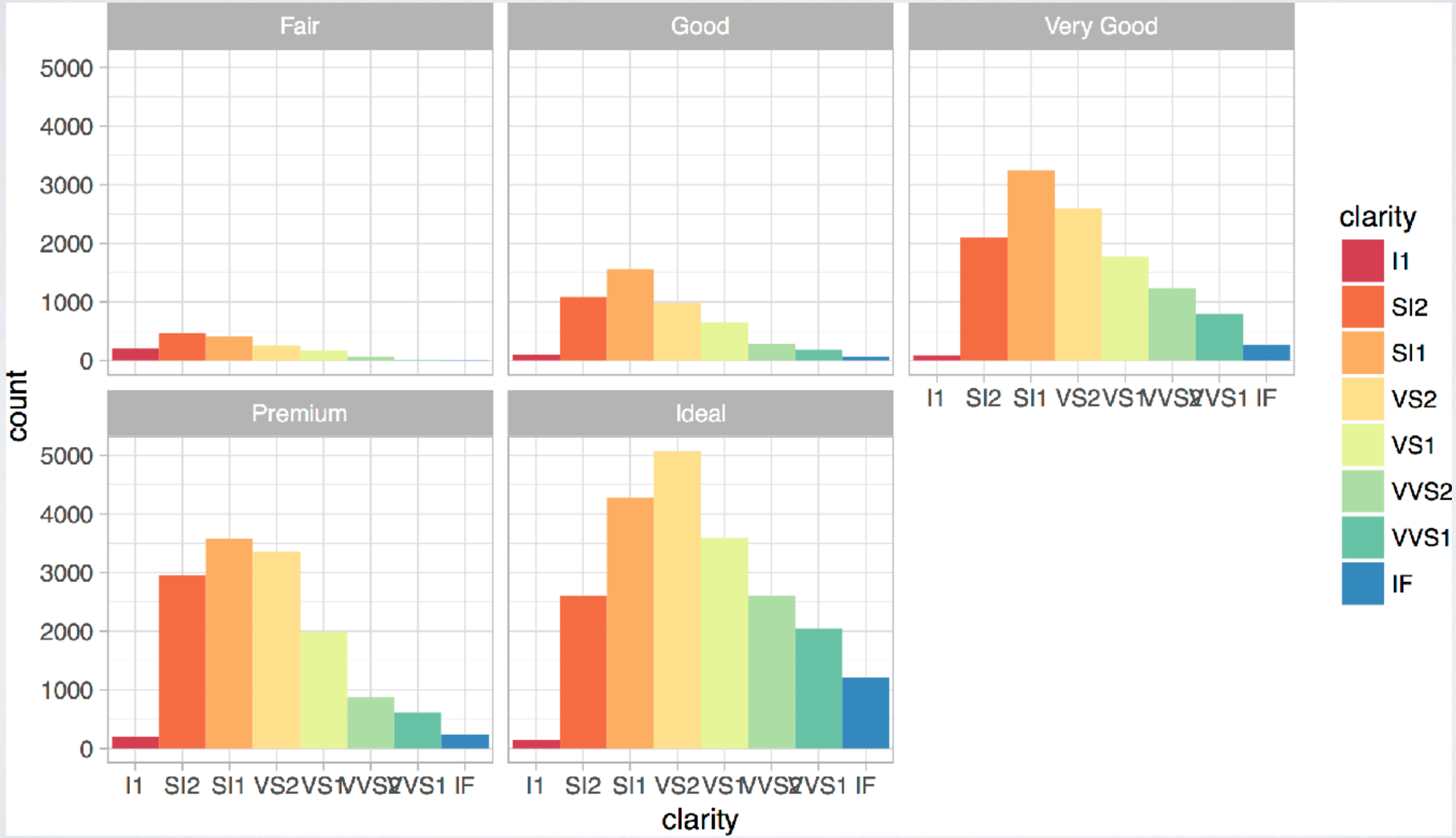
Scales

Customize color scales, other mappings



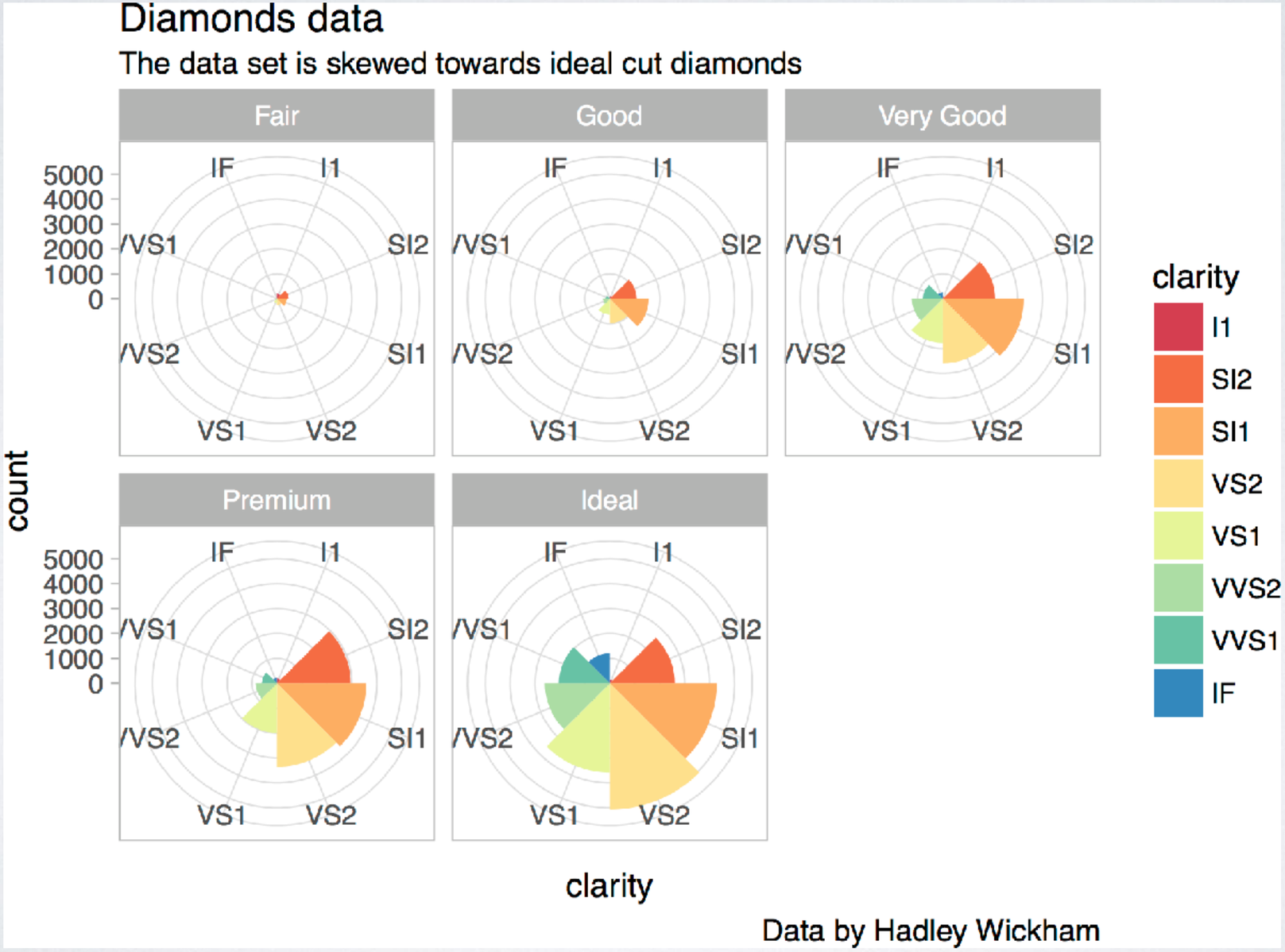
Facets

Subplots that display subsets of the data.



Coordinate systems

Titles and captions



A ggplot2 template

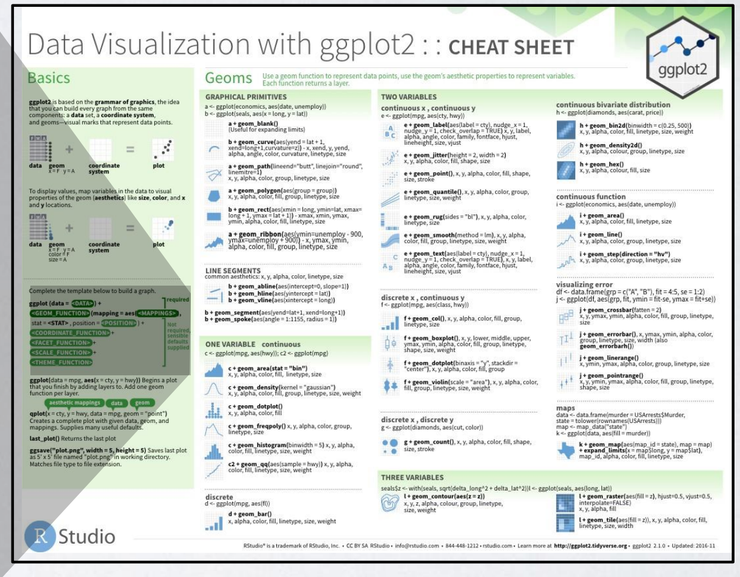
Make any plot by filling in the parameters of this template

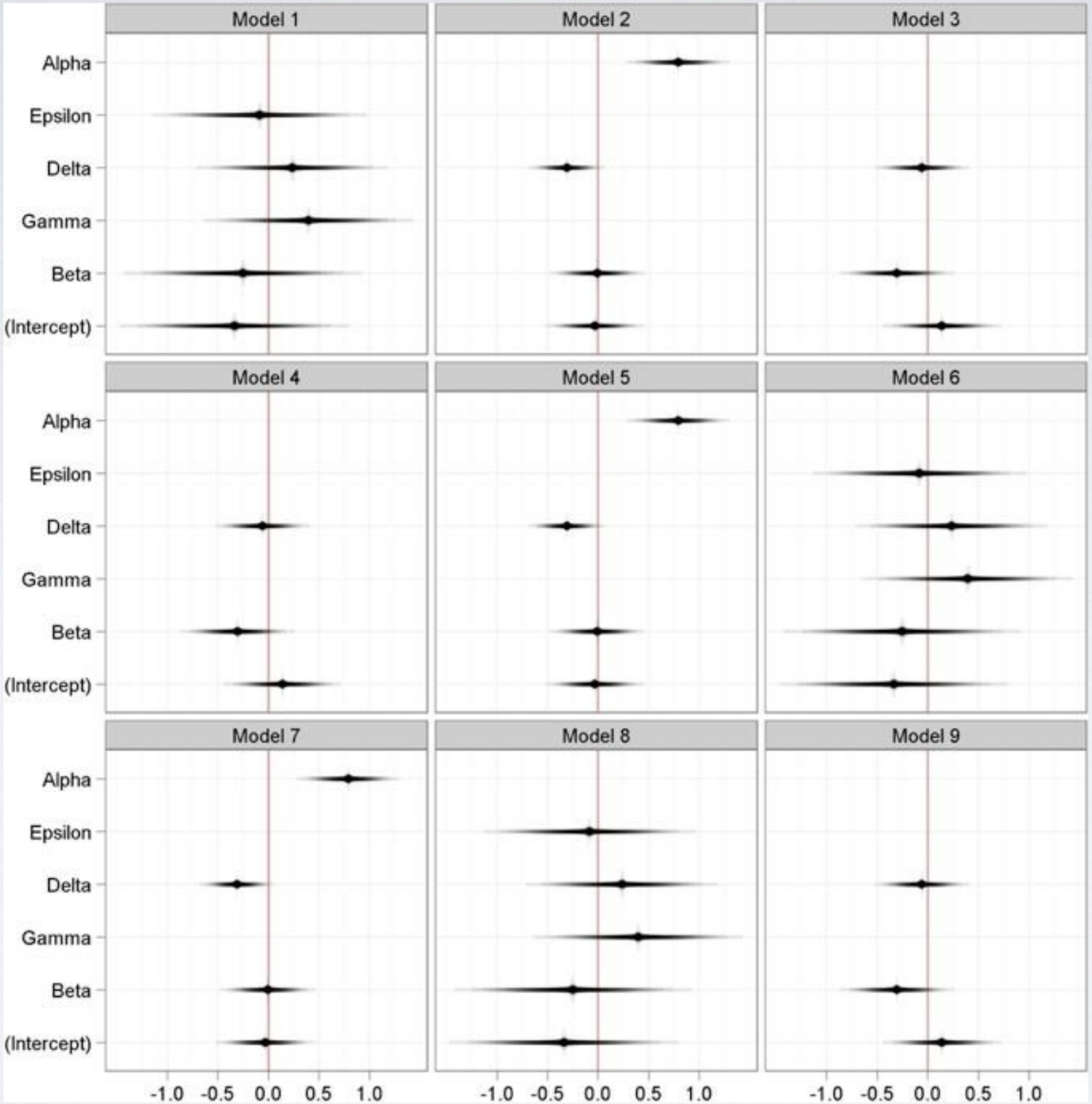
Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
 stat = <STAT> , position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

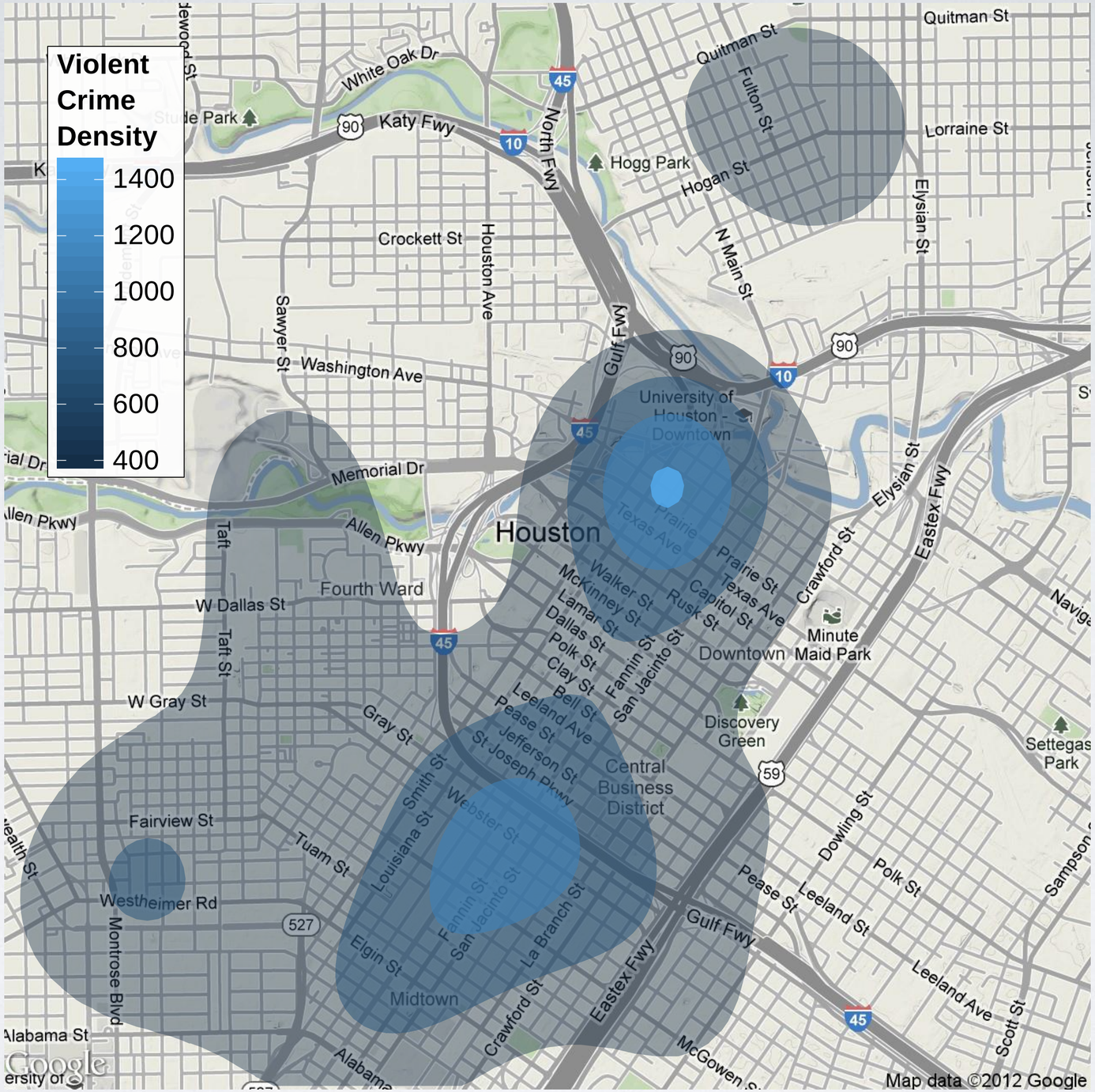
required

Not required, sensible defaults supplied





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



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

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

r4ds.had.co.nz

Complete and free online

RStudio Cloud | R for Data Science

r4ds.had.co.nz

Welcome

1 Introduction

I Explore

2 Introduction

3 Data visualisation

4 Workflow: basics

5 Data transformation

6 Workflow: scripts

7 Exploratory Data Analysis

8 Workflow: projects

II Wrangle

9 Introduction

10 Tibbles

11 Data import

12 Tidy data

13 Relational data

14 Strings

15 Factors

R for Data Science

The ultimate resource for the training

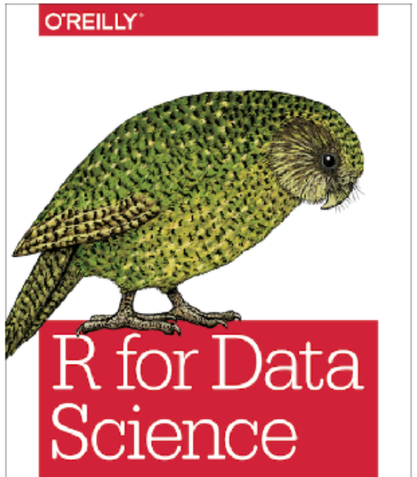
R for Data Science

Garrett Grolemund

Hadley Wickham

Welcome

This is the website for **"R for Data Science".** This book will teach you how to do data science with R: You'll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it. In this book, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you'll learn how to clean data and draw plots—and many other things besides. These are the skills that allow data science to happen, and here you will find the best practices for doing each of these things with R. You'll learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You'll also learn how to



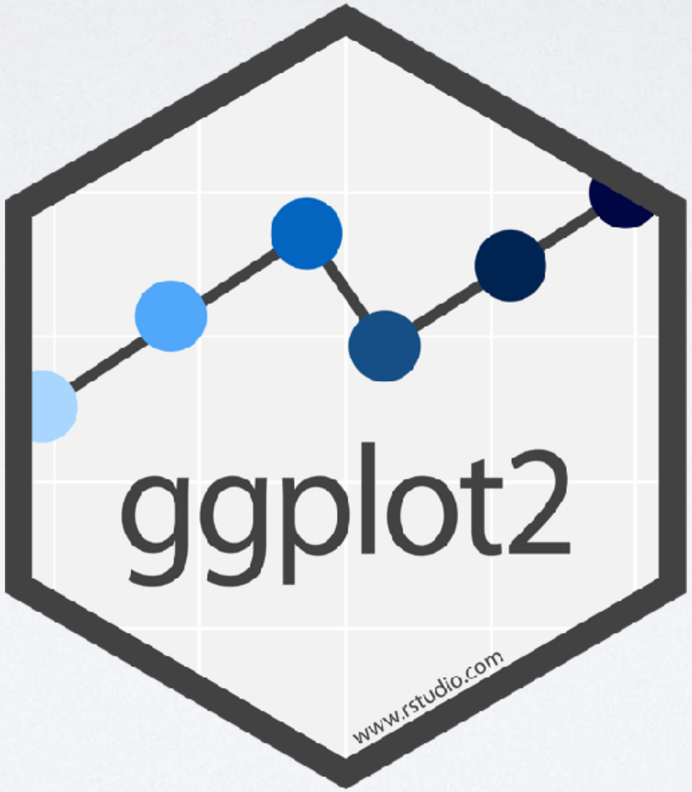
ggplot2

Your Turn

Navigate up to the **03-Transform** folder.

Open **03-Transform-Exercises.Rmd**

Visualize Data with



Slides CC BY-SARStudio