Strata+ Hadoop WORLD

PRESENTED BY

O'REILLY cloudera

RStudio

strataconf.com

#StrataHadoop

R QuickStart

Transform and visualize data

Garrett Grolemund

Data Scientist and Master Instructor

March 2016

Email: garrett@rstudio.com

@StatGarrett

Warm up

Introduce yourself to your table

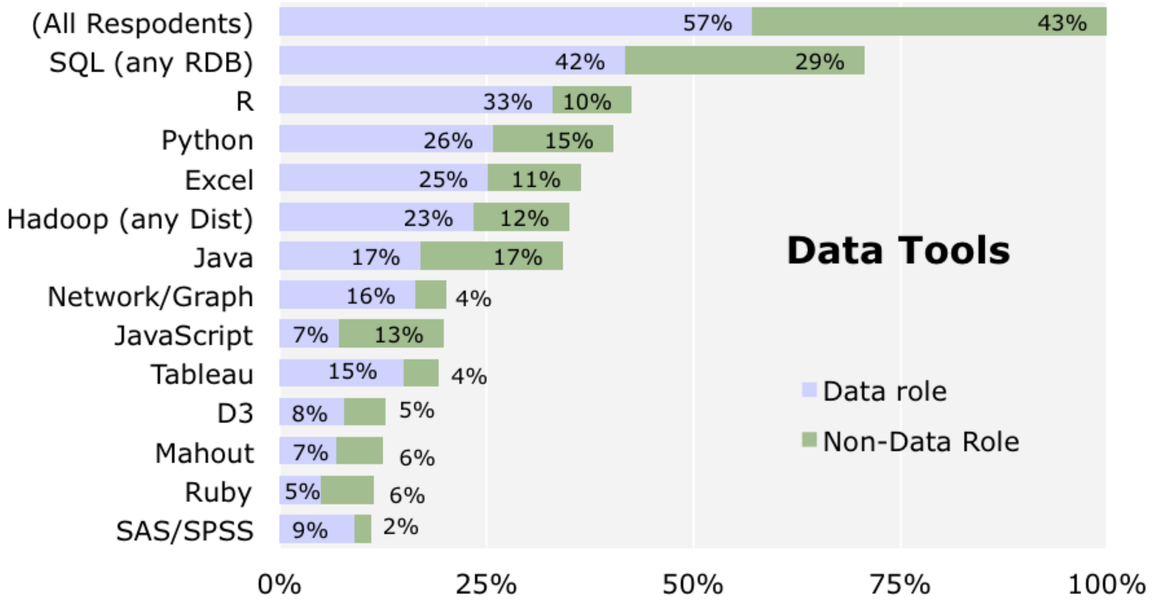
- Name
- What do you do with data?
- For how long have you been using R?

R

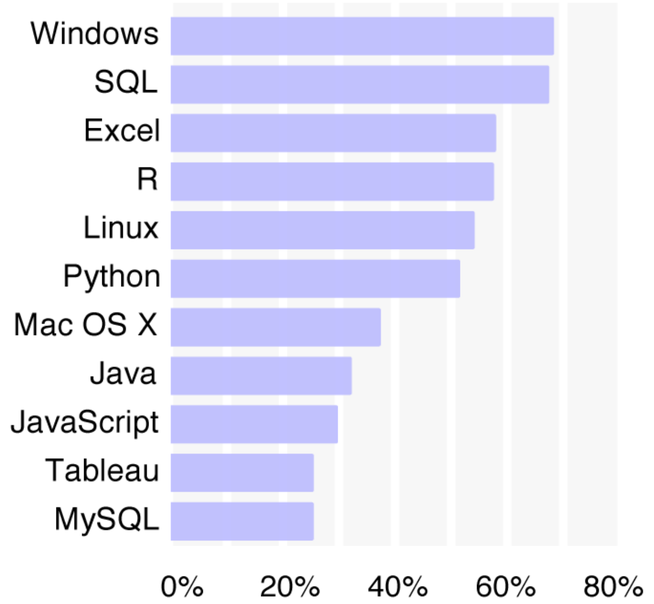
1. A full scripting language
2. Descends from S, Bell Laboratories
3. Evolved in university environment
4. Built-in data science/statistics toolkit

Data Science Salary Survey

2012 & 2013 (combined)



2014



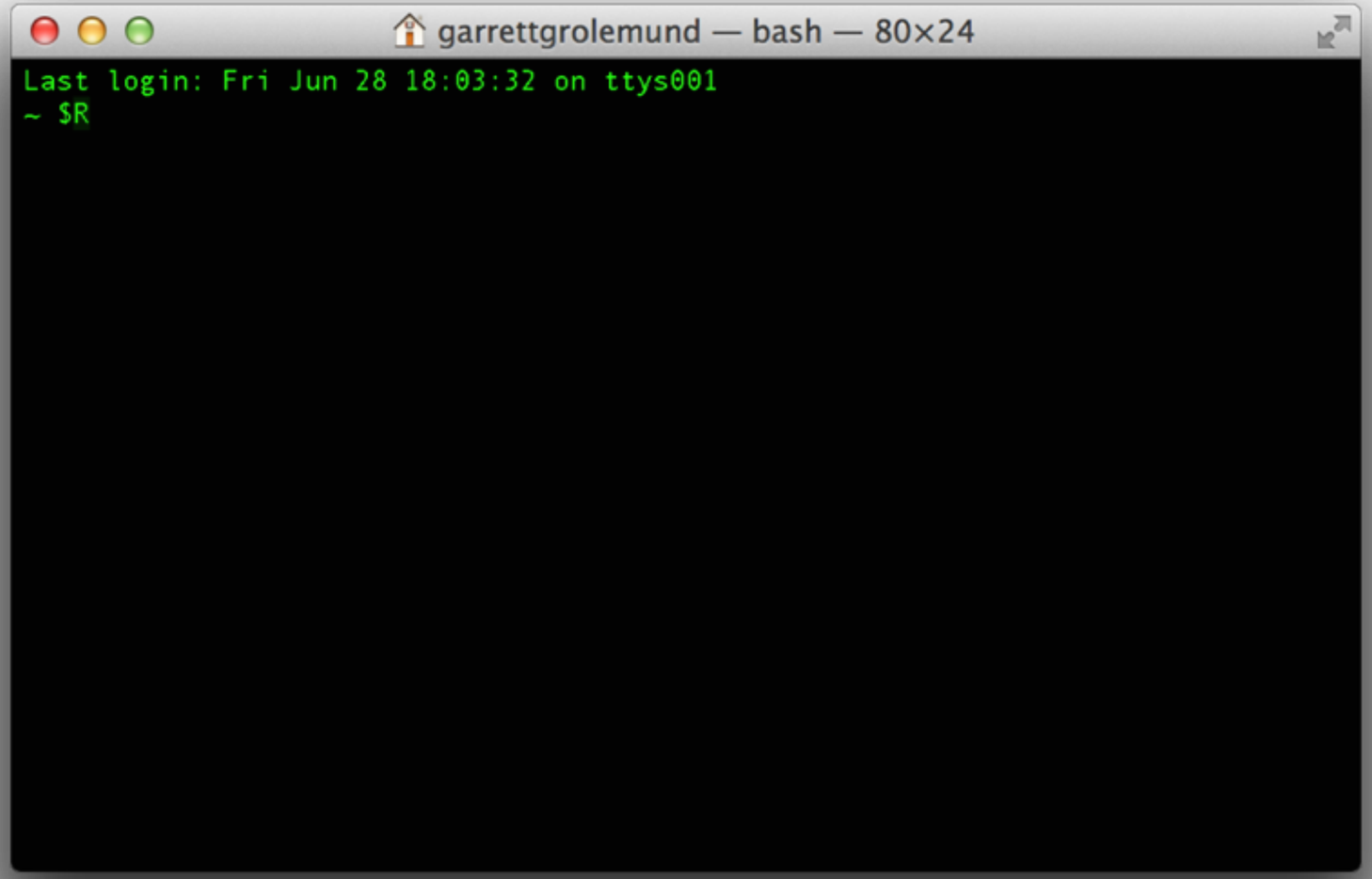
Source: http://r4stats.com/articles/popularity/

1. A full scripting language
2. Descends from S, Bell Laboratories
3. Evolved in university environment
4. Built-in data science/statistics toolkit

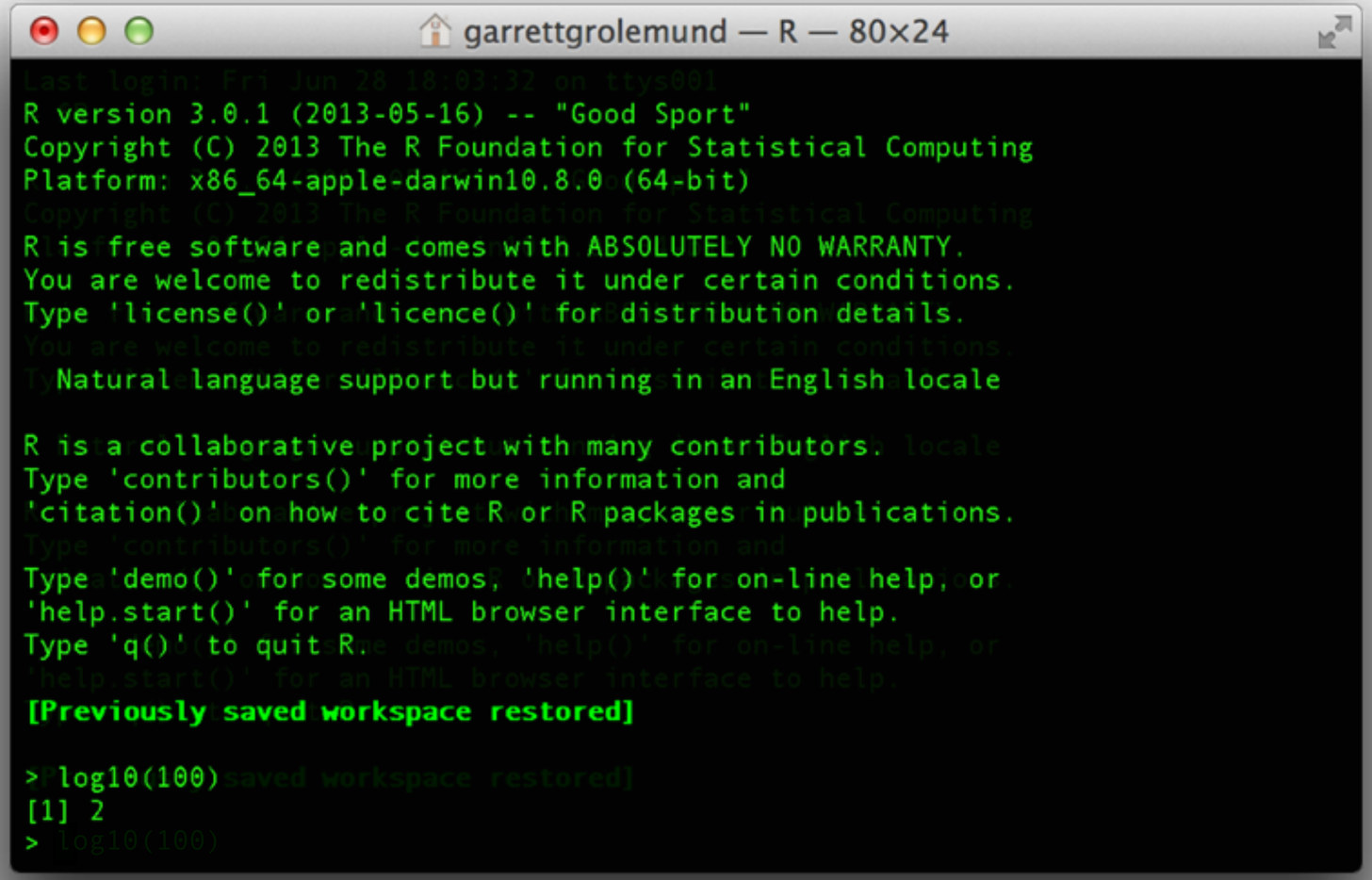
1. An integrated development environment (IDE) for R

2. Software that helps you write, run, debug, and manage code written in R

R: a computer programming language



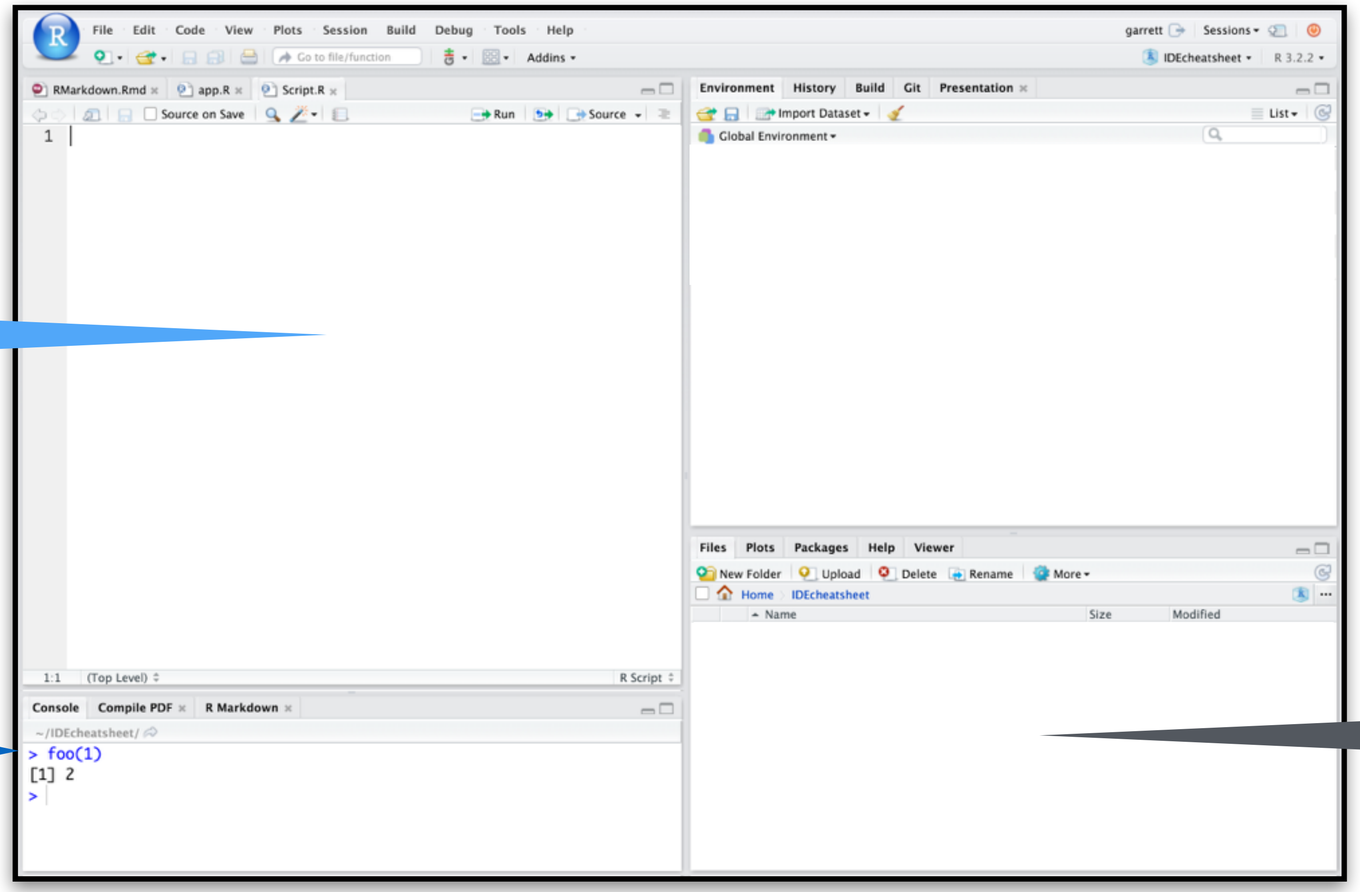
R: a computer programming language



RStudio: Software for R

Scripts Pane
to compose
R code

Command line
to run R code



File Browser
to manage
scripts

RStudio IDE Cheat Sheet

learn more at www.rstudio.com

RStudio

The RStudio IDE is an Integrated Development Environment in R that comes in three versions

Desktop IDE
A local version of the IDE for your desktop

Open Source Server
for larger compute resources and remote access

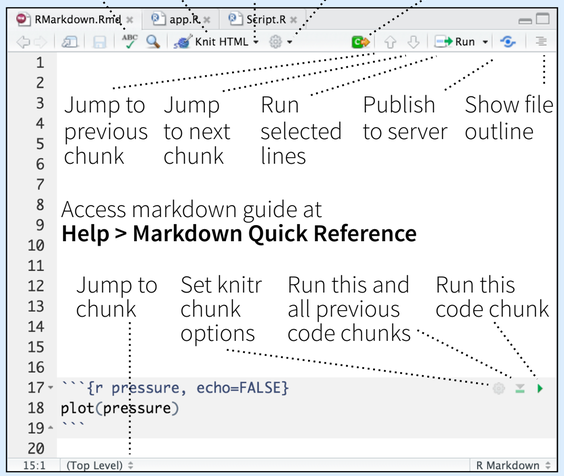
Professional Server
for teams that share large compute resources, large data, and uniform environments for collaboration

Download all at **www.rstudio.com**. Each provides the same useful interface:

Documents and Apps

Open Shiny, R Markdown, knitr, Sweave, LaTeX, .Rd files and more in Source Pane

Check spelling | Render output | Choose output format | Choose output location | Insert code chunk



Jump to previous chunk | Jump to next chunk | Run selected lines | Publish to server | Show file outline

Access markdown guide at **Help > Markdown Quick Reference**

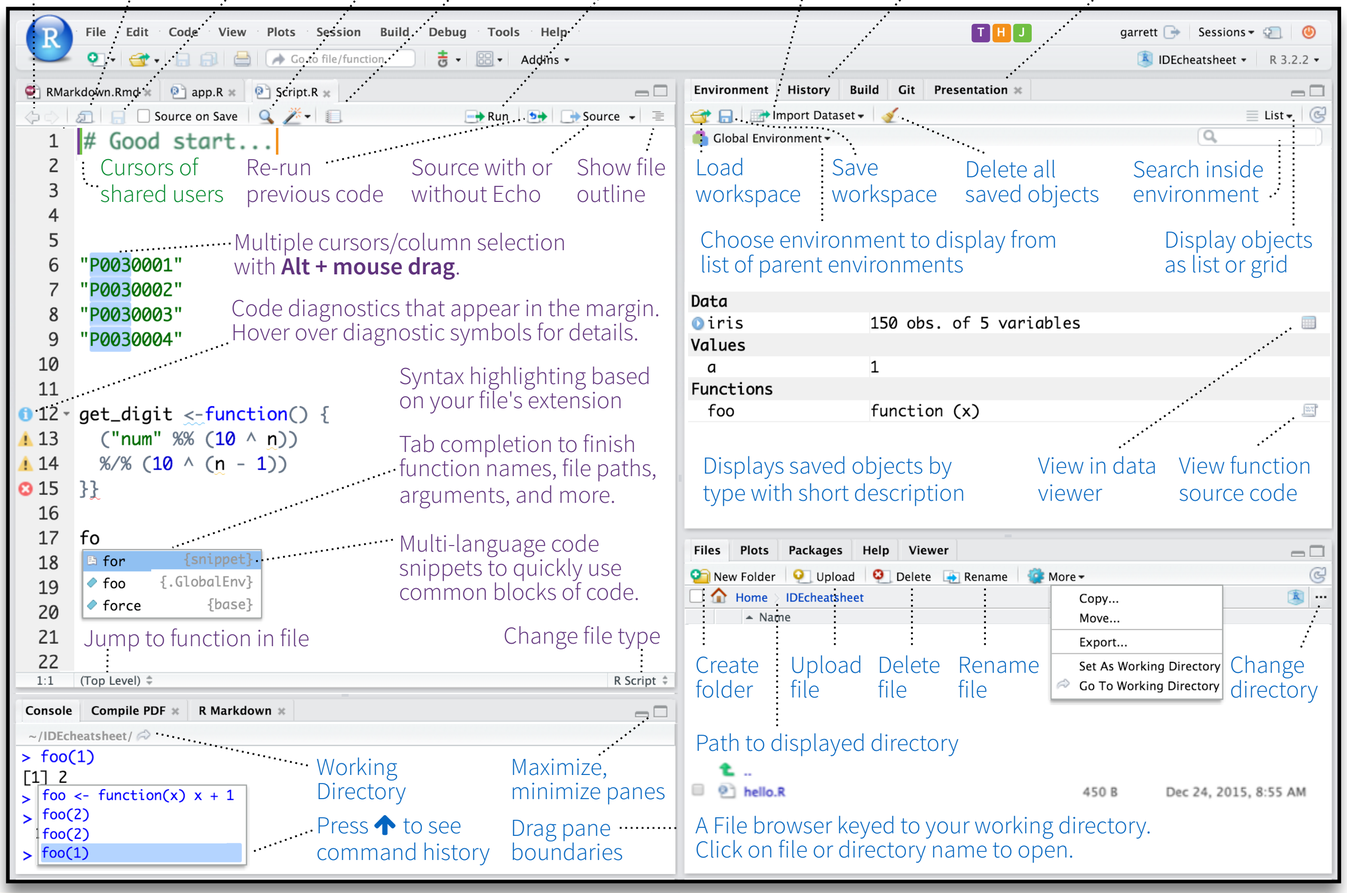
Jump to chunk | Set knitr chunk options | Run this and all previous code chunks | Run this code chunk

RStudio recognizes that files named **app.R**, **server.R**, **ui.R**, and **global.R** belong to a shiny app

Run app | Choose location to view app | Publish to shinyapps.io or server | Manage publish accounts

Write Code

Navigate tabs | Open in new window | Save | Find and replace | Compile as notebook | Run selected code



Cursors of shared users | Re-run previous code | Source with or without Echo | Show file outline

Multiple cursors/column selection with **Alt + mouse drag**.

Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.

Syntax highlighting based on your file's extension

Tab completion to finish function names, file paths, arguments, and more.

Multi-language code snippets to quickly use common blocks of code.

Jump to function in file | Change file type

Working Directory | Maximize, minimize panes

Press ↑ to see command history | Drag pane boundaries

R Support

Import data file with wizard | History of past commands to run/add to source | Display .RPres slideshows **File > New File > R Presentation**

Load workspace | Save workspace | Delete all saved objects | Search inside environment

Choose environment to display from list of parent environments | Display objects as list or grid

Displays saved objects by type with short description | View in data viewer | View function source code

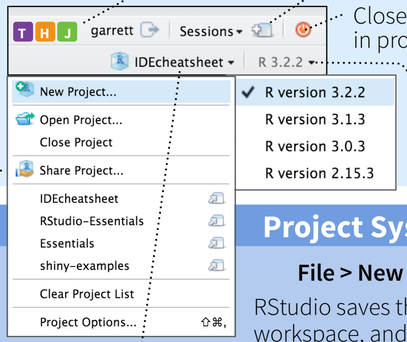
Create folder | Upload file | Delete file | Rename file | Change directory

Path to displayed directory

A File browser keyed to your working directory. Click on file or directory name to open.

RStudio Pro Features

Share Project with Collaborators | Active shared collaborators | Start **new R Session** in current project



Close R Session in project

Select R Version

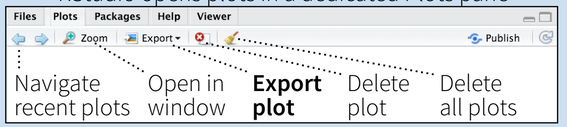
Name of current project

Project System

File > New Project

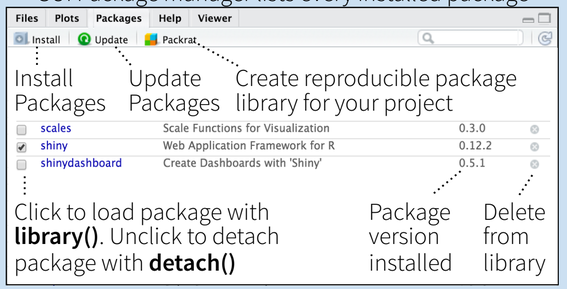
RStudio saves the call history, workspace, and working directory associated with a project. It reloads each when you re-open a project.

RStudio opens plots in a dedicated Plots pane



Navigate recent plots | Open in window | **Export plot** | Delete plot | Delete all plots

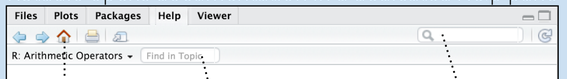
GUI Package manager lists every installed package



Install Packages | Update Packages | Create reproducible package library for your project

Click to load package with **library()**. Unclick to detach package with **detach()** | Package version installed | Delete from library

RStudio opens documentation in a dedicated Help pane



Debug Mode

Open with **debug()**, **browse()**, or a breakpoint. RStudio will open the debugger mode when it encounters a breakpoint while executing code

Launch debugger mode from origin of error

Open traceback to examine the functions that R called before the error occurred

Version Control with Git or SVN

Turn on at **Tools > Project Options > Git/SVN**

Stage | Show file | Commit | Push/Pull | View

Your Turn

Use your username and password to log into the class RStudio Server Pro at

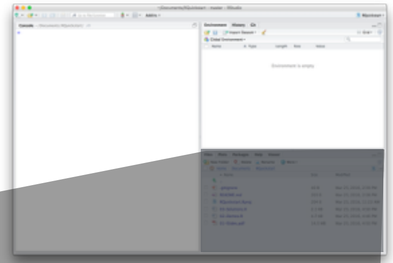
http://harpers-ferry.rstudio.com/

Open the project titled 01-Quickstart.

Class Materials

In the **RQuickstart** project, you will find

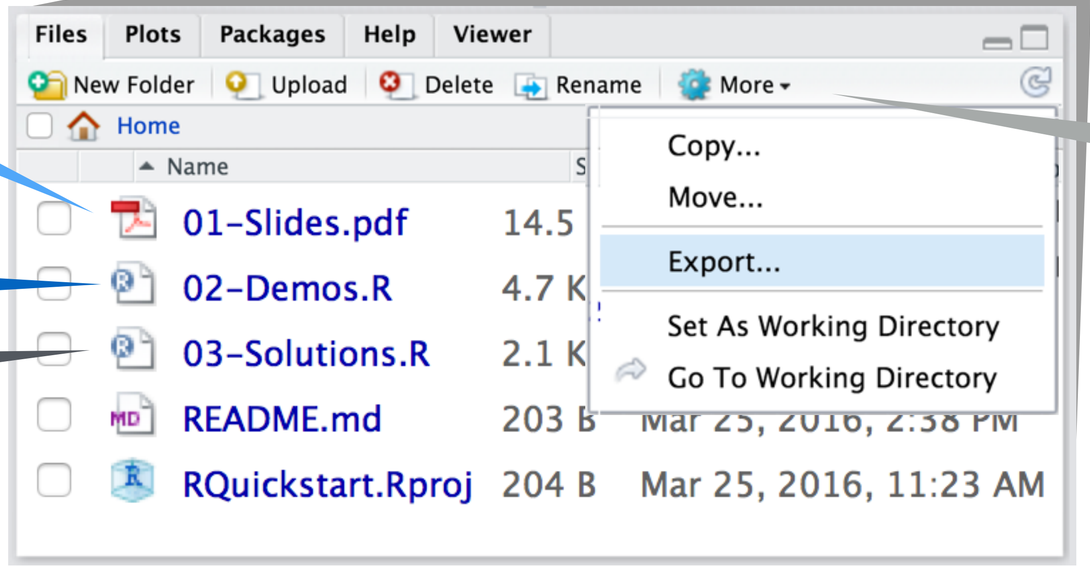
`01-Slides.pdf`
`02-Demos.R`
`03-Solutions.R`



Slides

Demos

Solutions



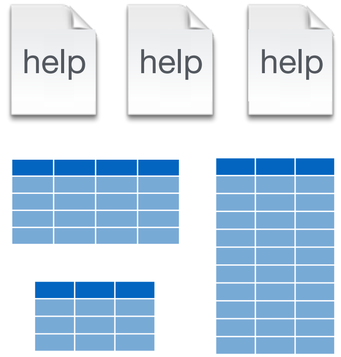
To Download

Packages

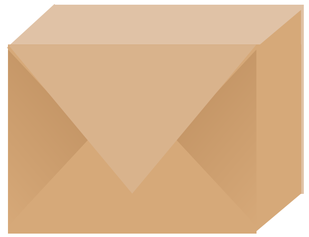
A bundle of functions, data sets, and help pages that you install in addition to the base R installation, e.g.

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

To use a package, you must load it with `library()`



=

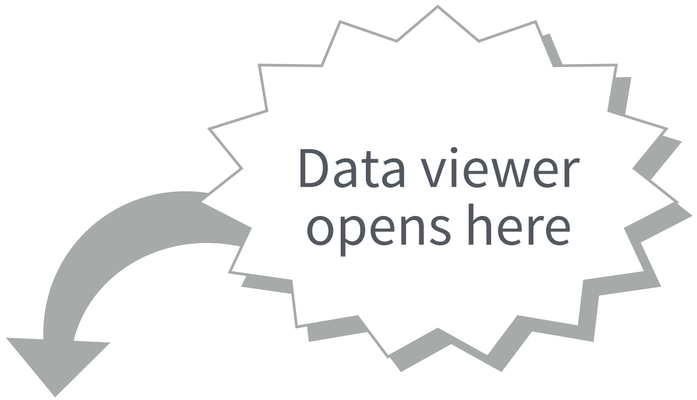


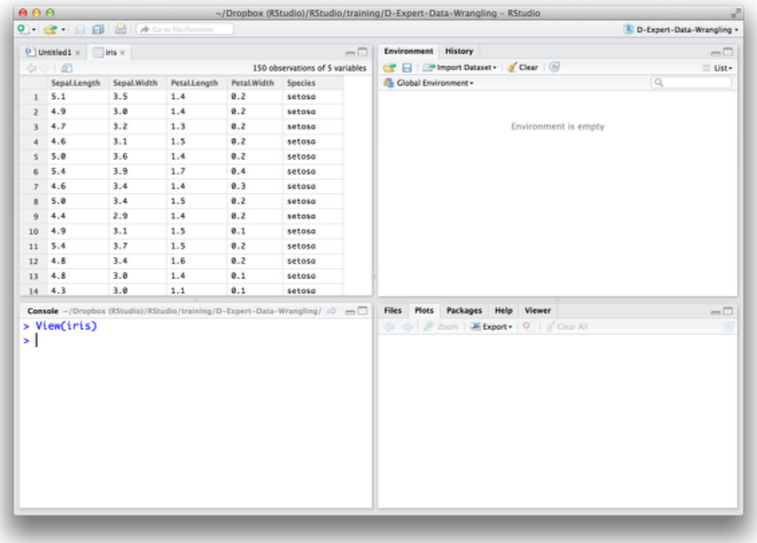
Run `library(ggplot2)`

View()

Examine any data set with the View() command (Capital V)

```
View(mpg)
View(iris)
View(mtcars)
```

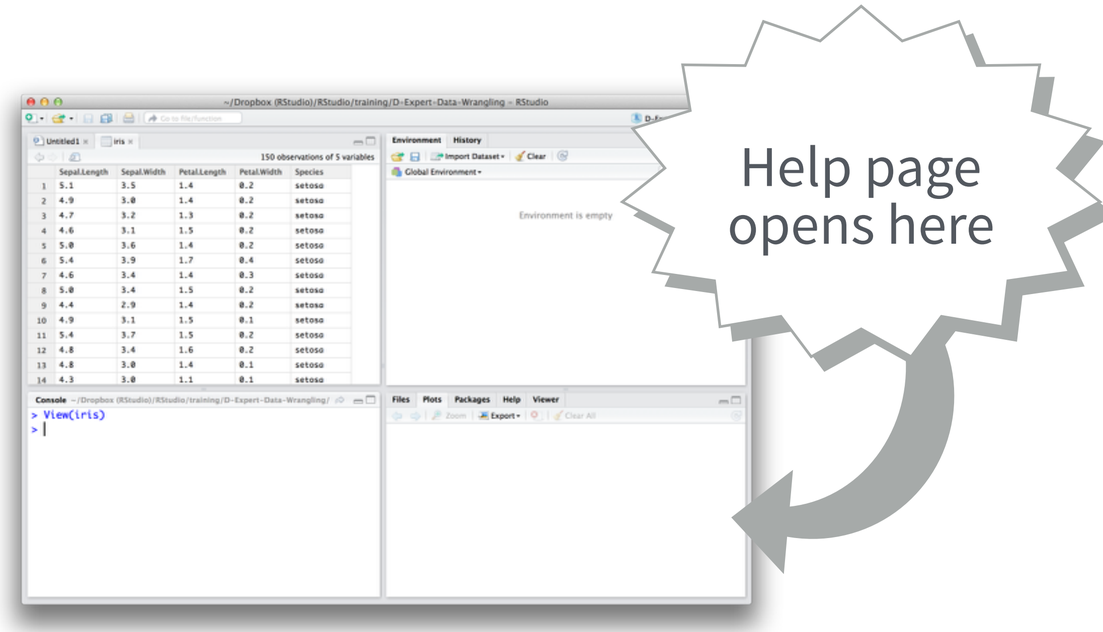




Help

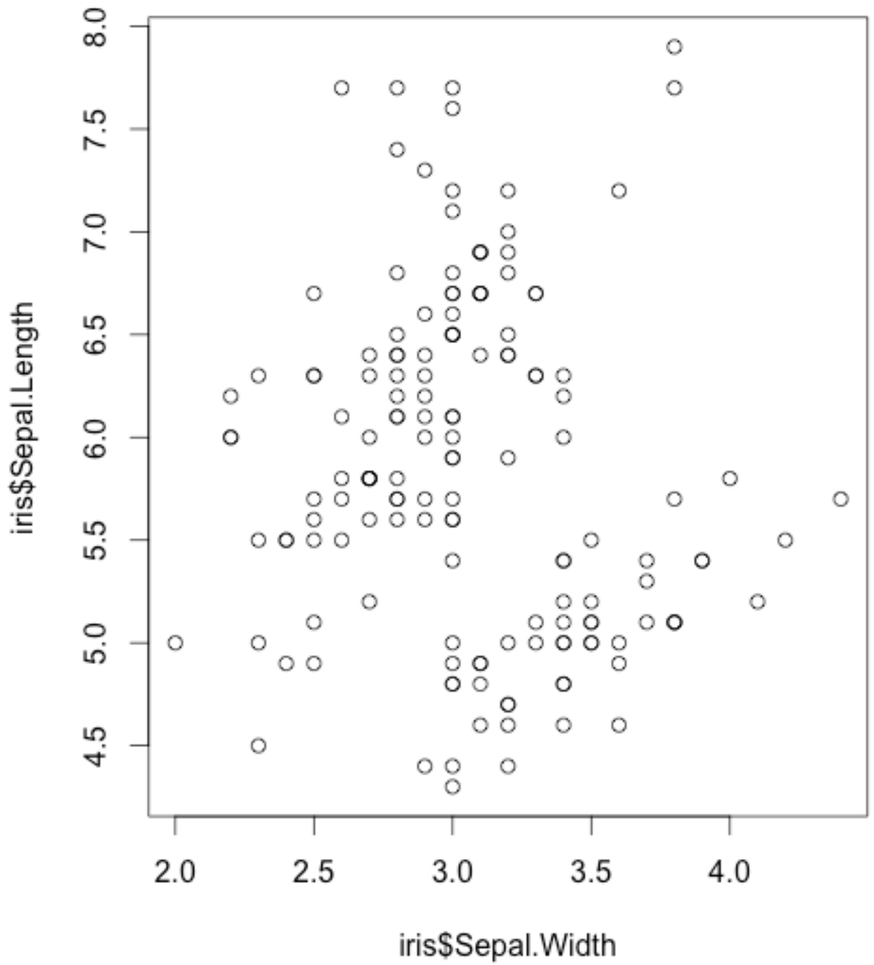
Open the help page for any function or data set by running the name with a **?** in front of it

```
?mpg
?ggplot
?geom_point
```



Visualizations with ggplot2

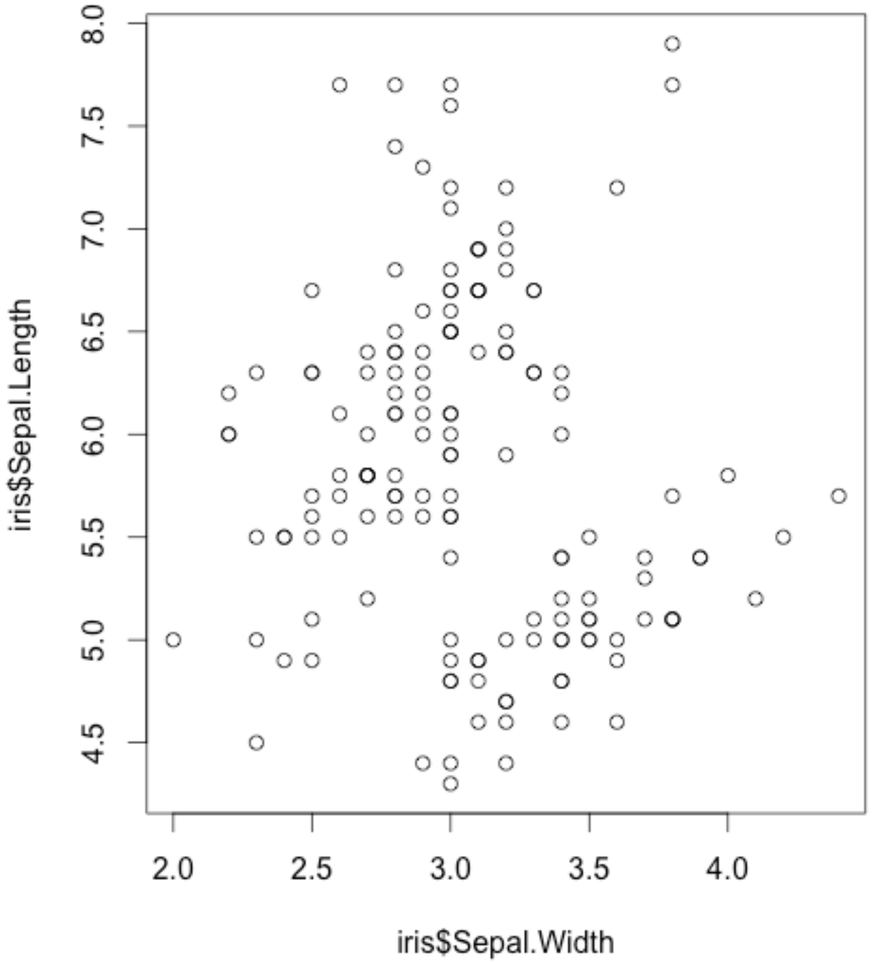
plot

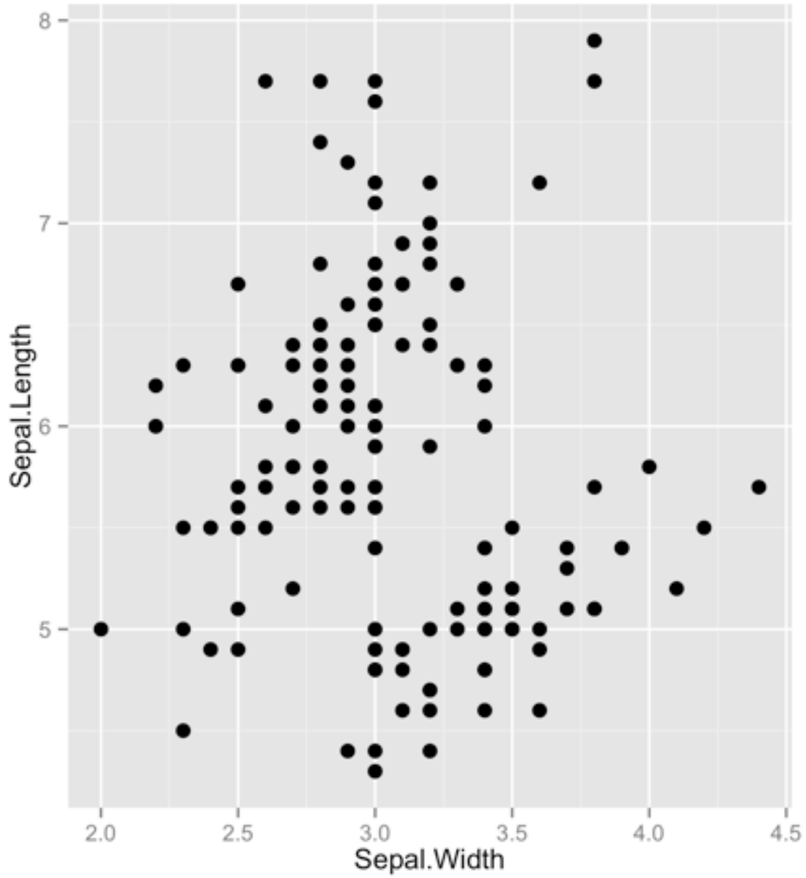



```
plot(iris$Sepal.Width,
     iris$Sepal.Length)
```

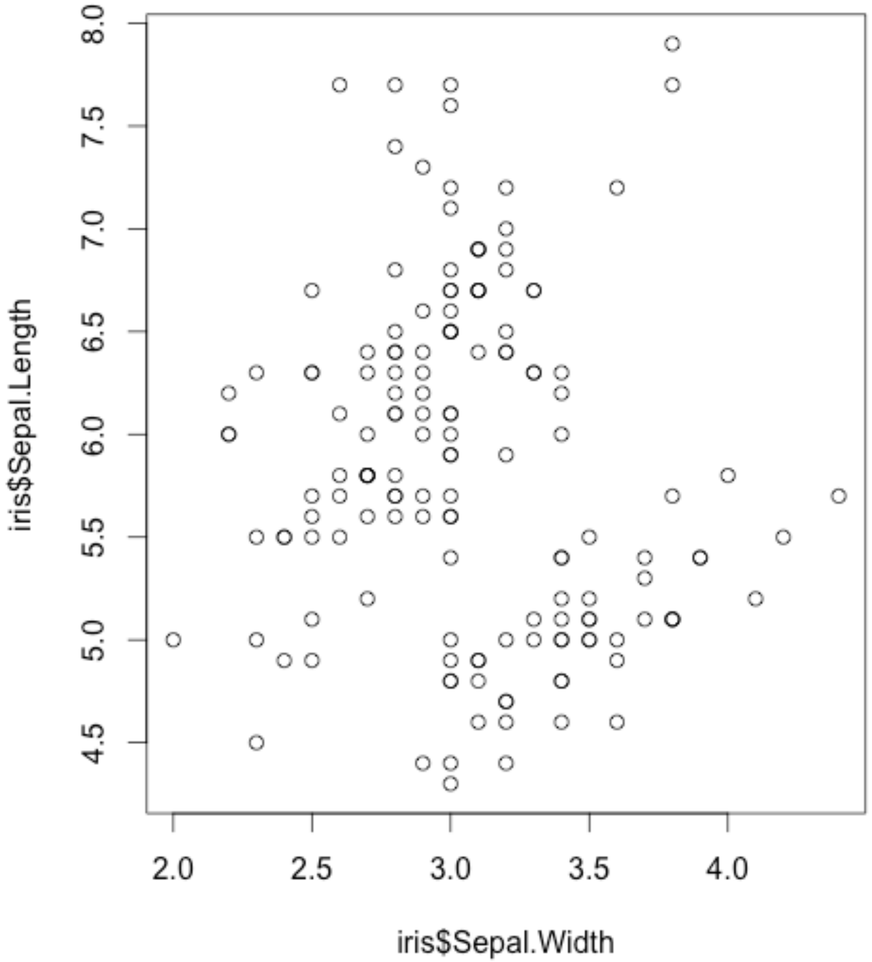
- R's basic plot method
- simple
- does different things in different contexts (usually in a helpful way)
- difficult to customize

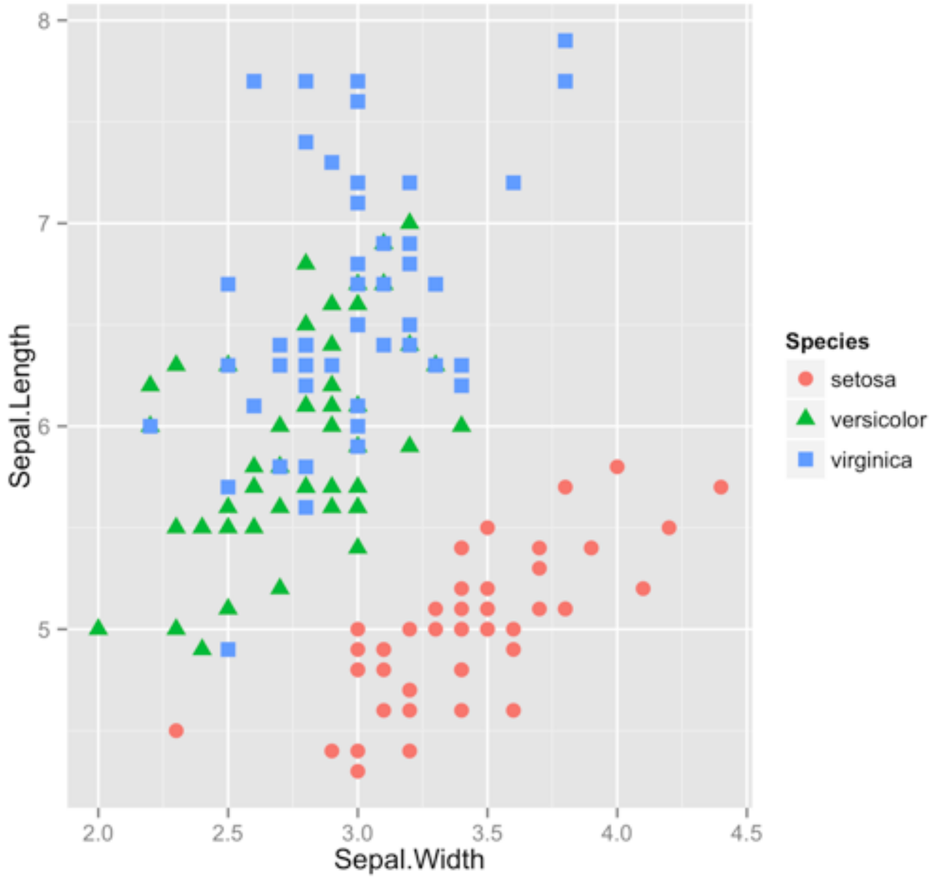
ggplot2



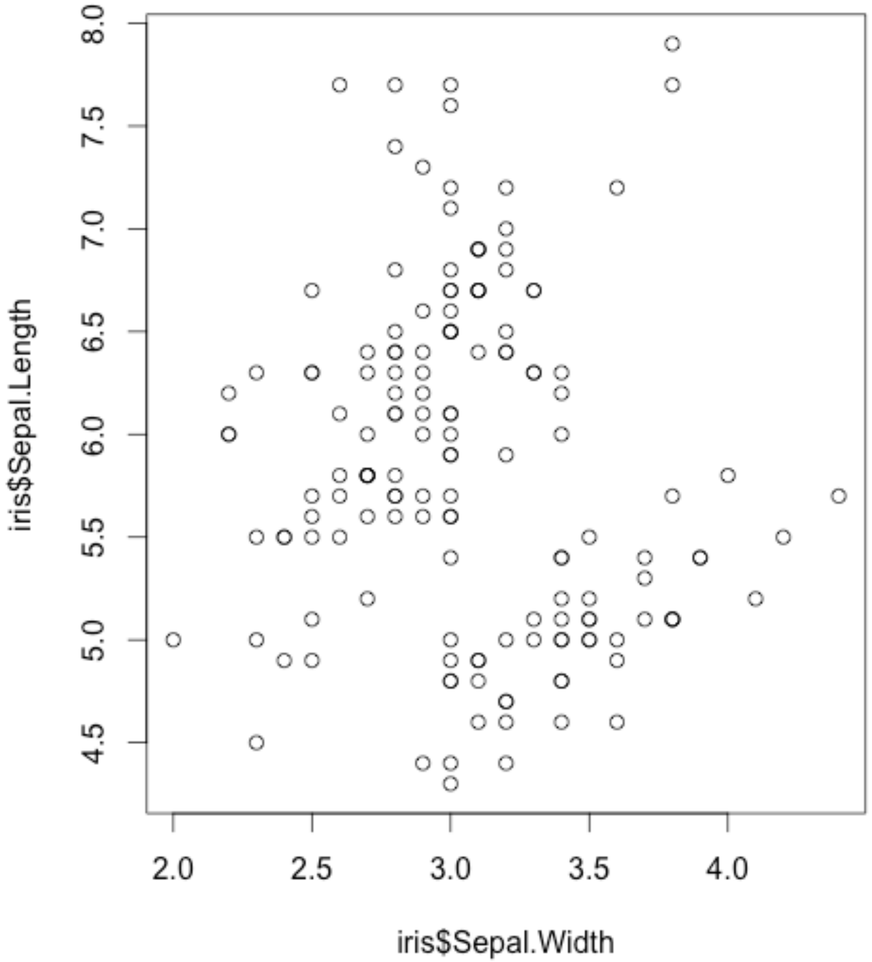


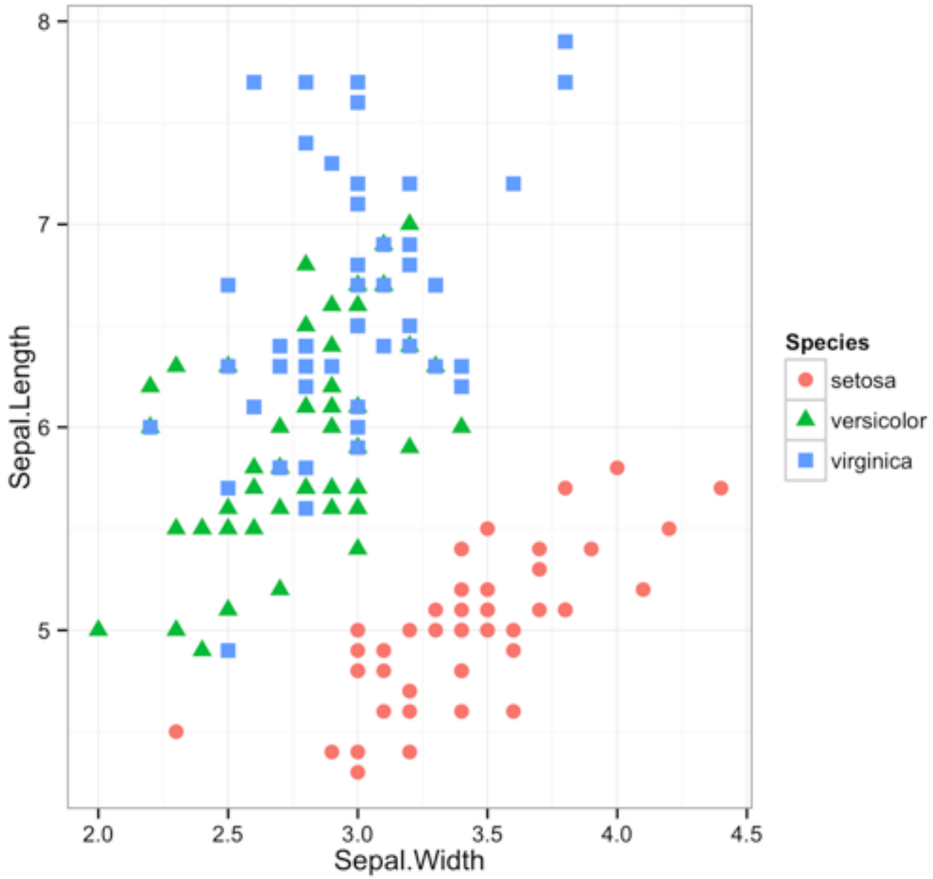
ggplot2



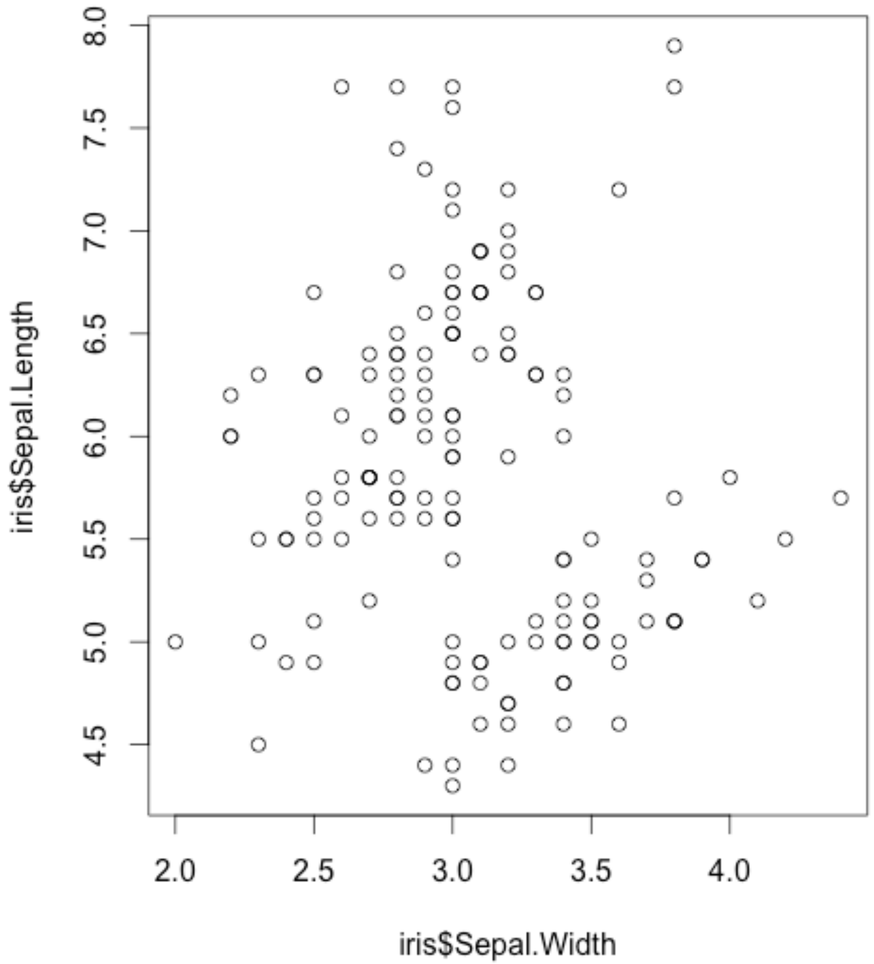


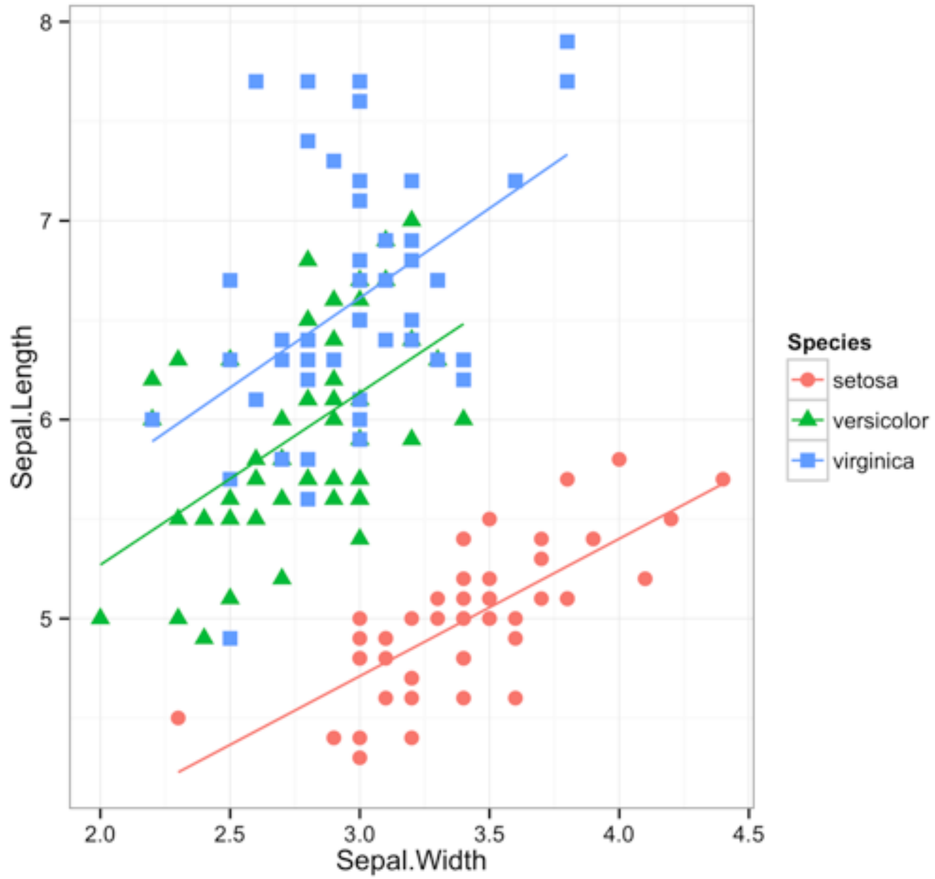
ggplot2



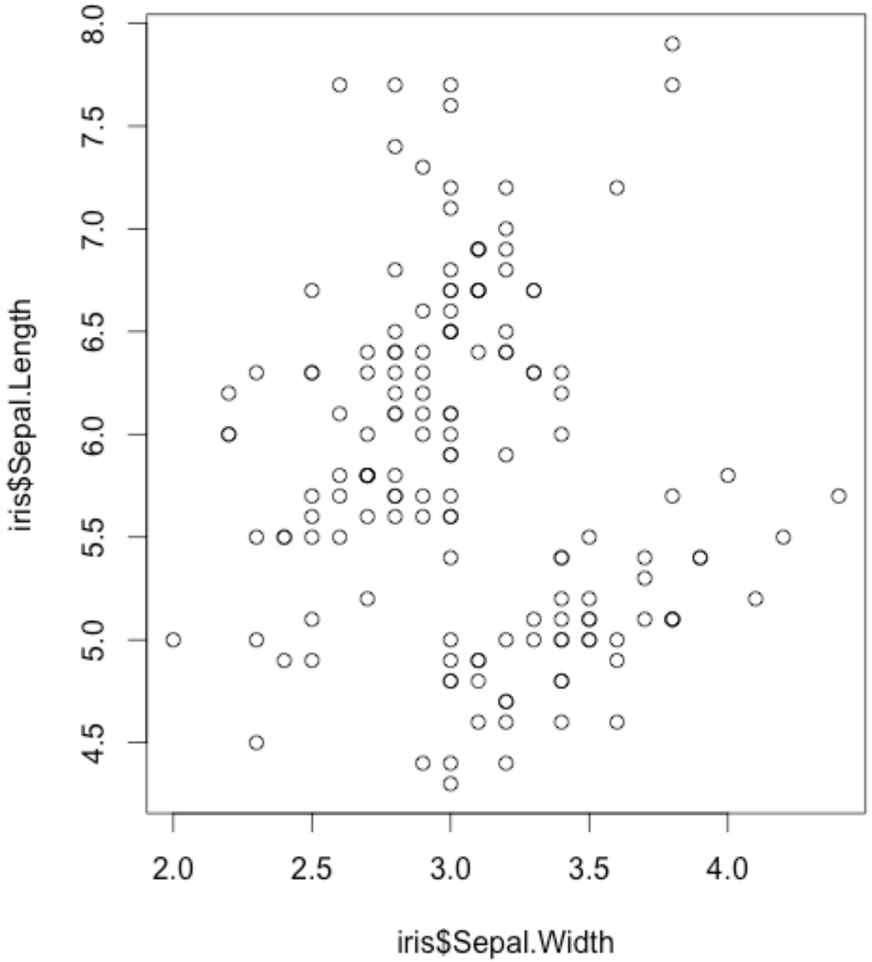


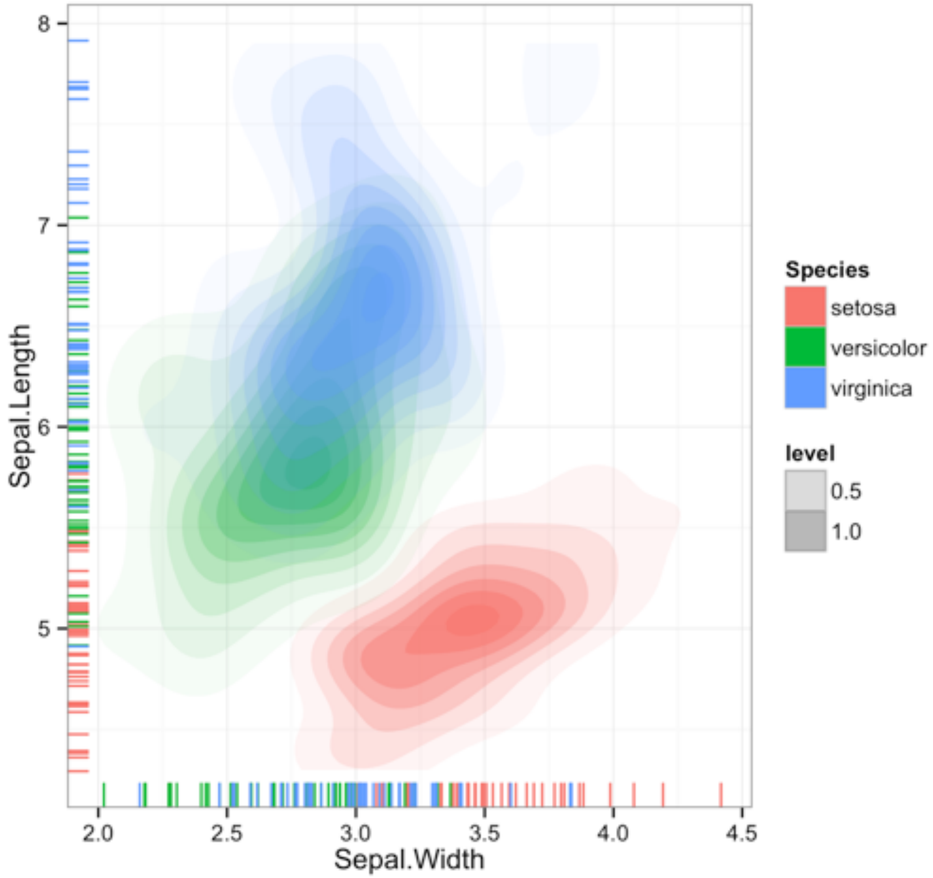
ggplot2



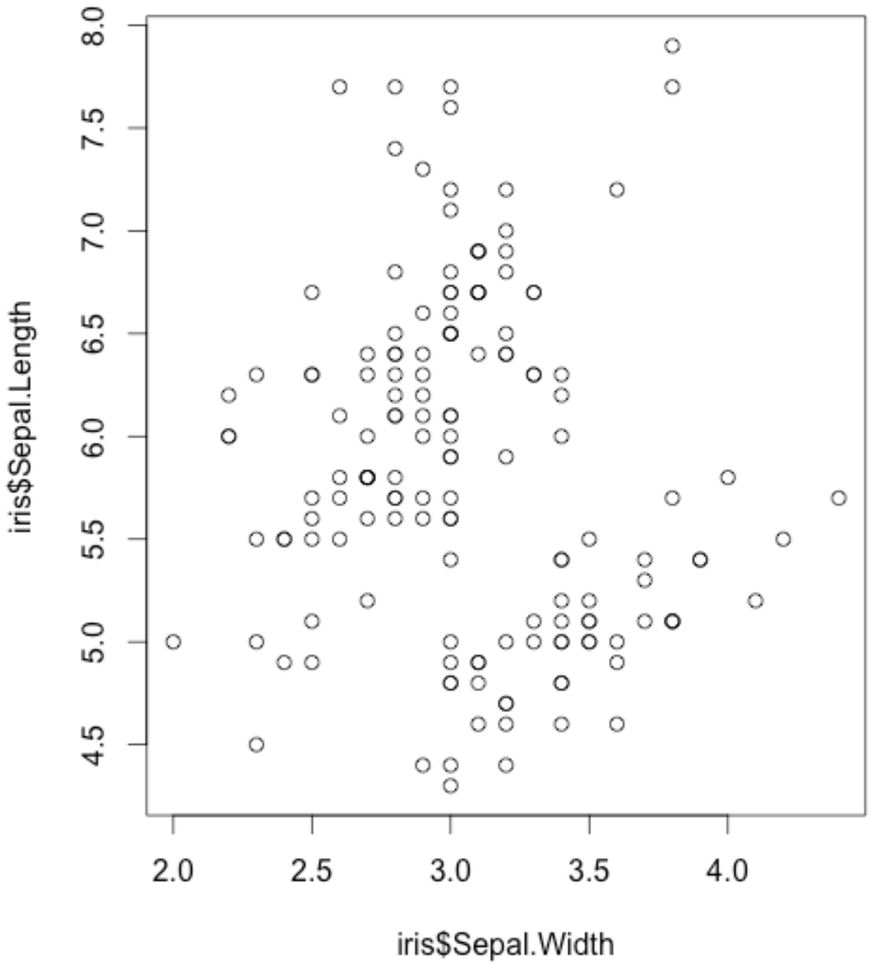


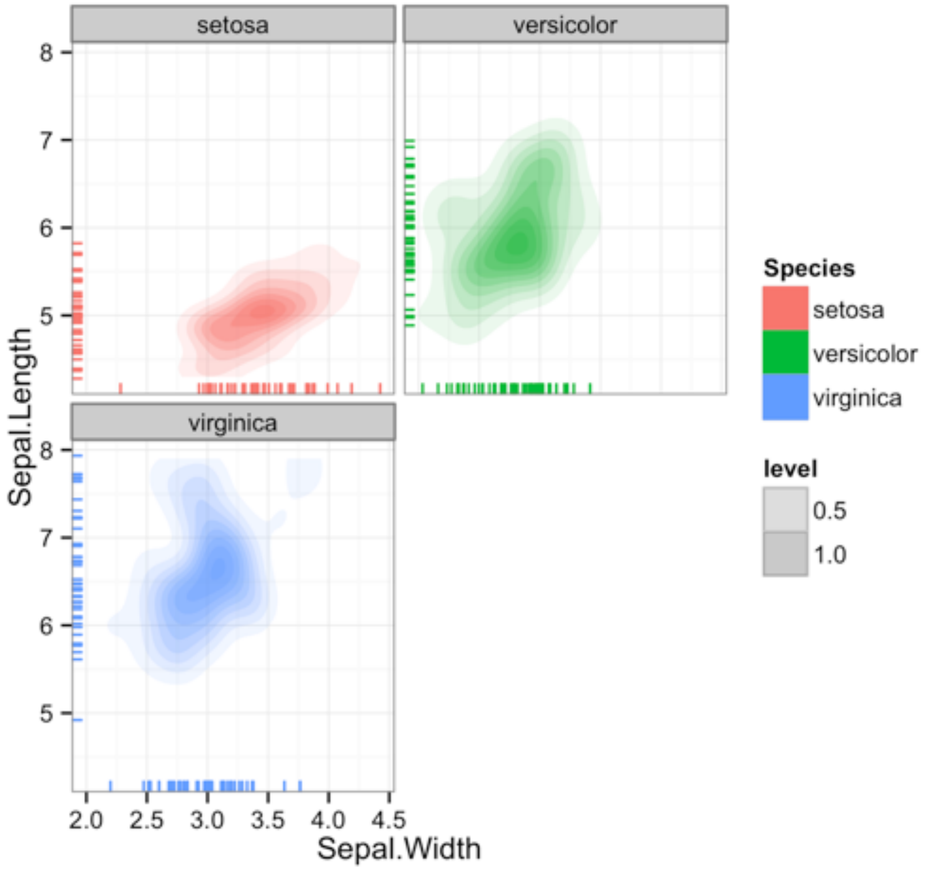
ggplot2

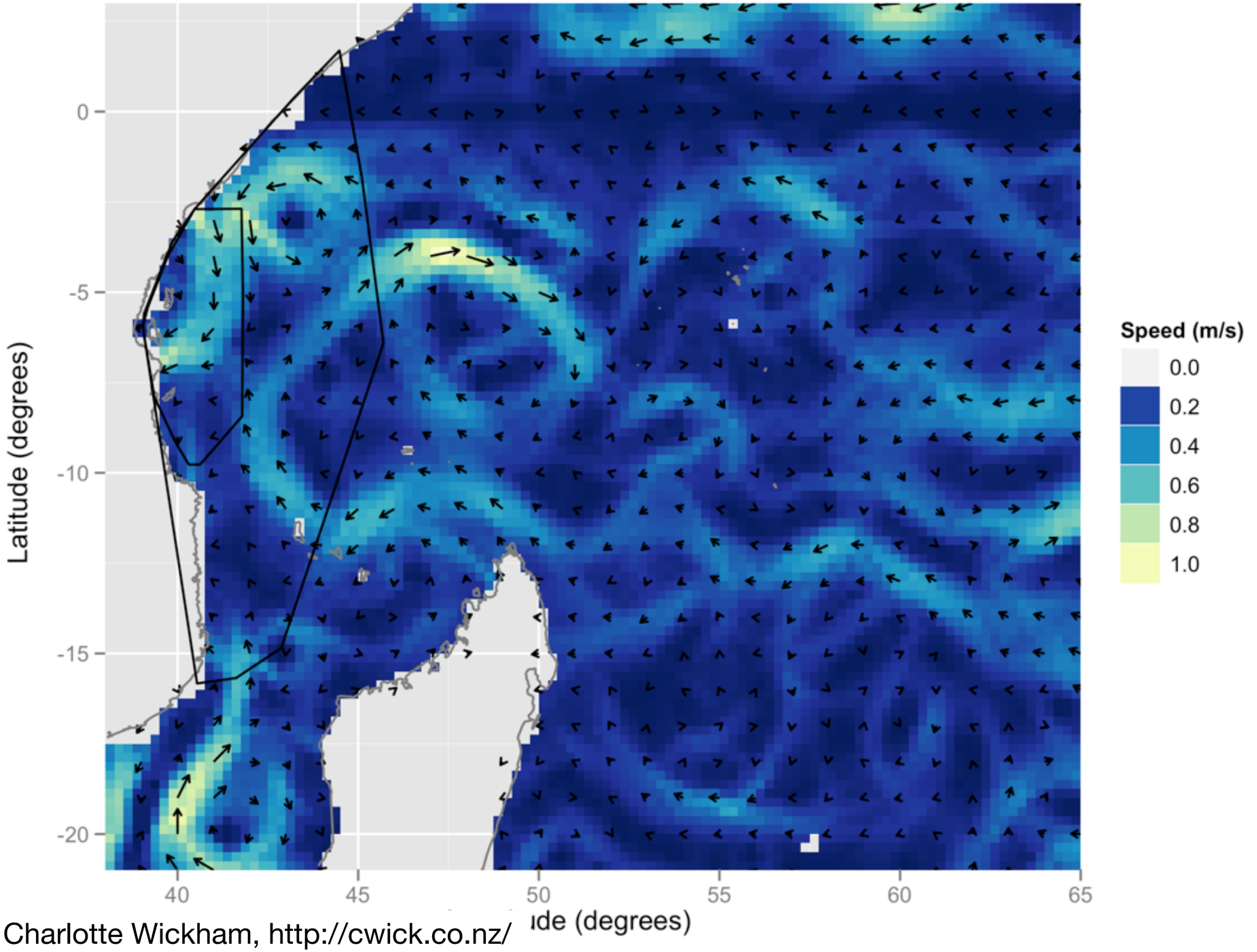




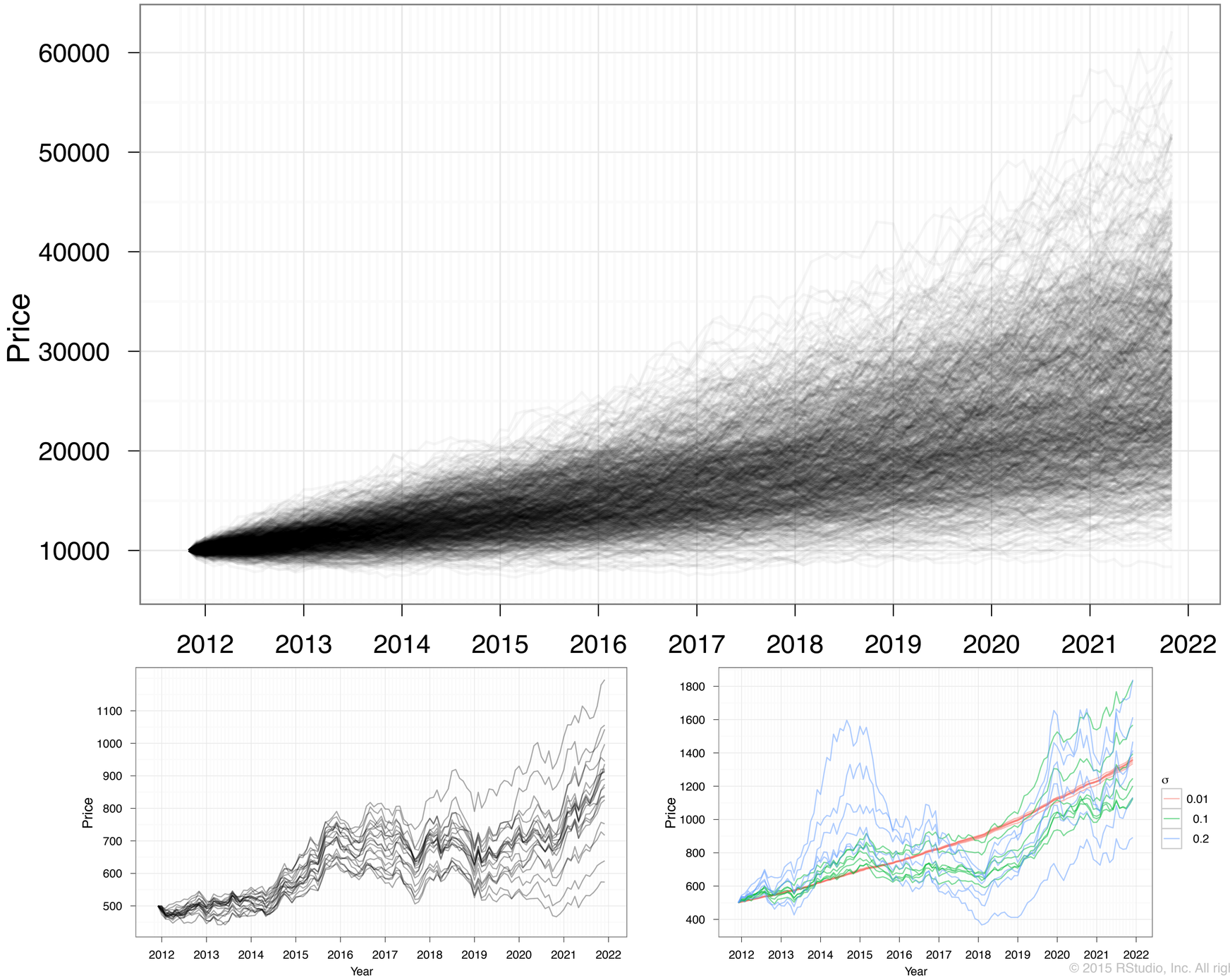
ggplot2

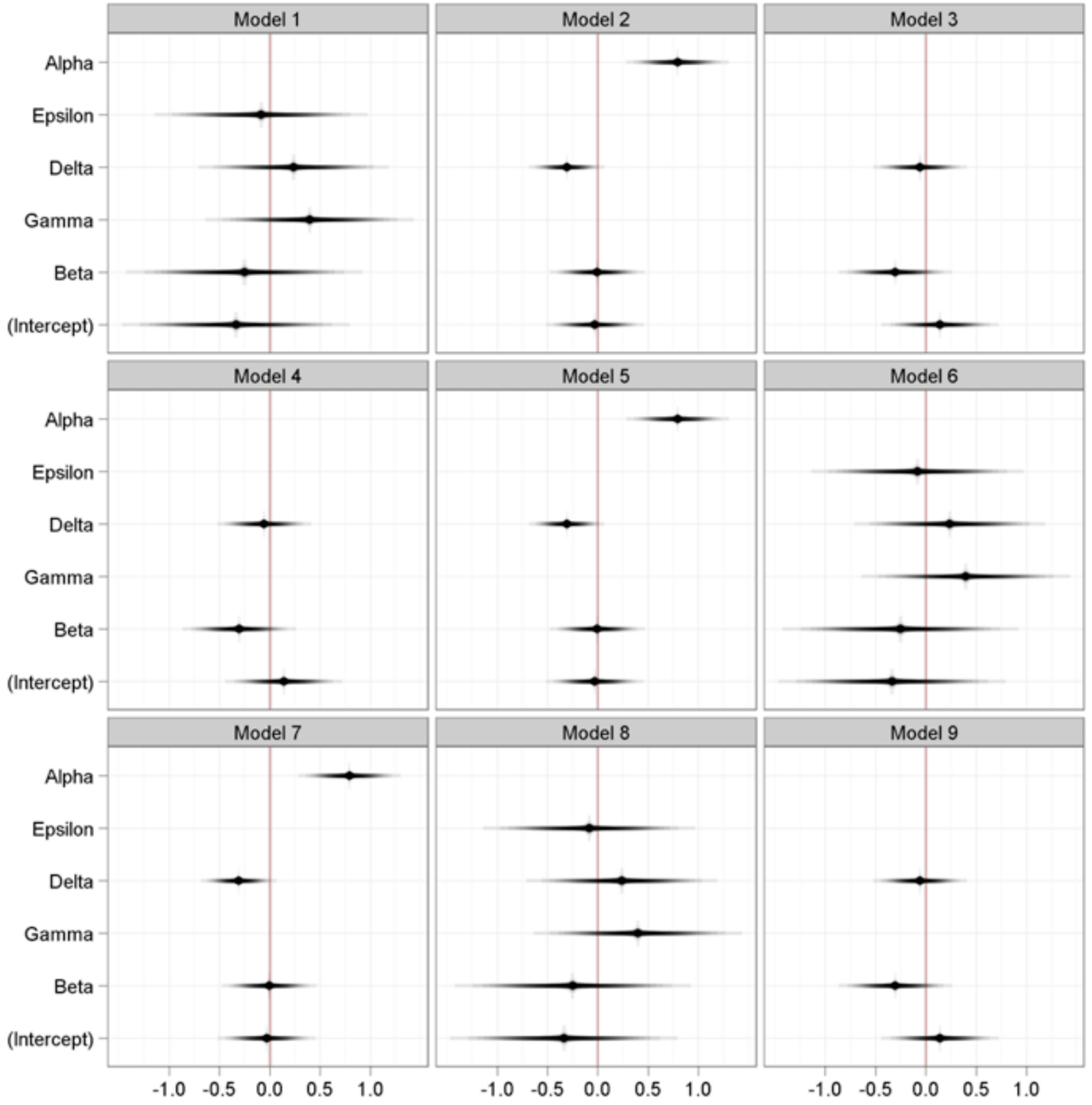




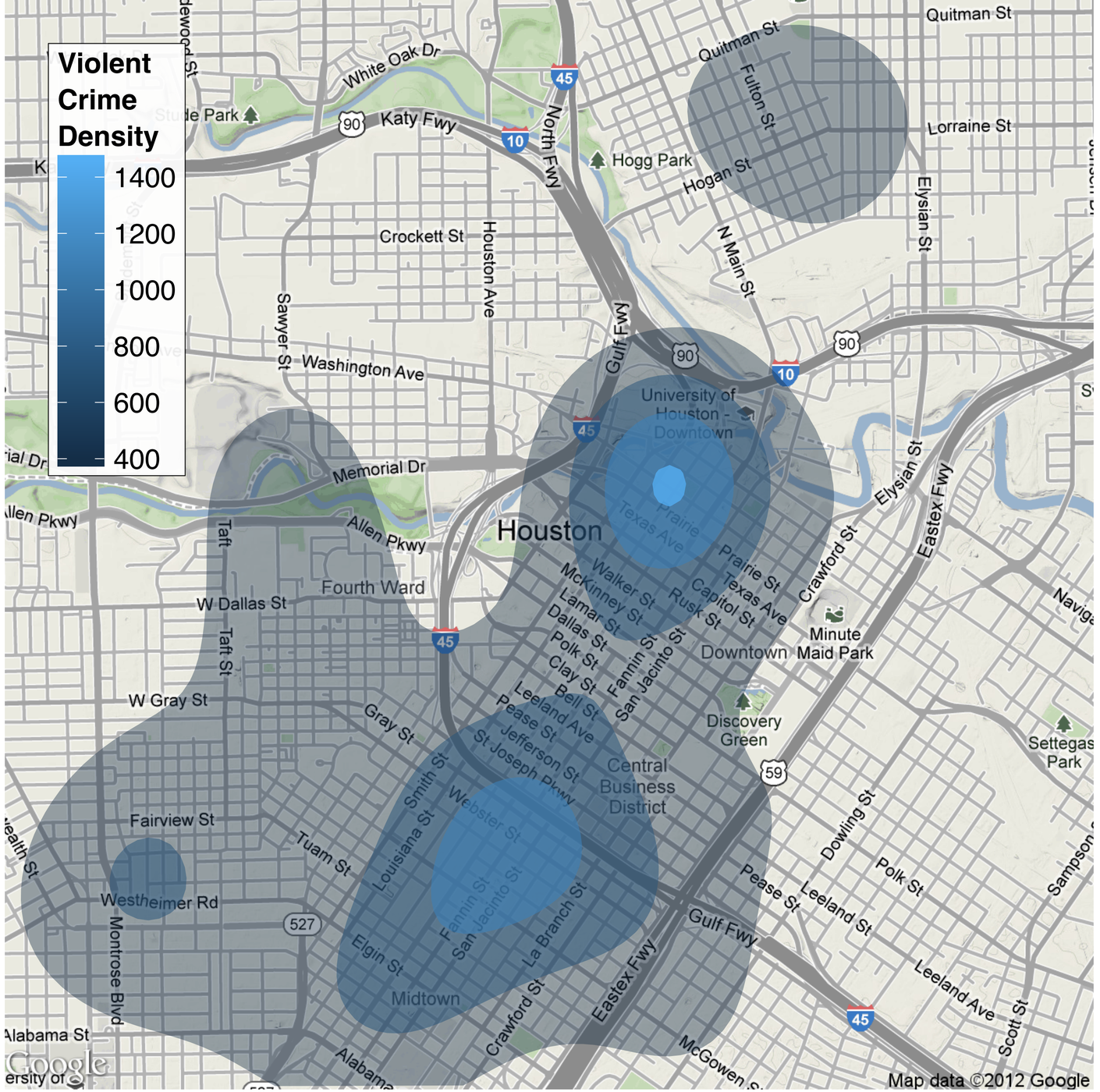


Charlotte Wickham, http://cwick.co.nz/

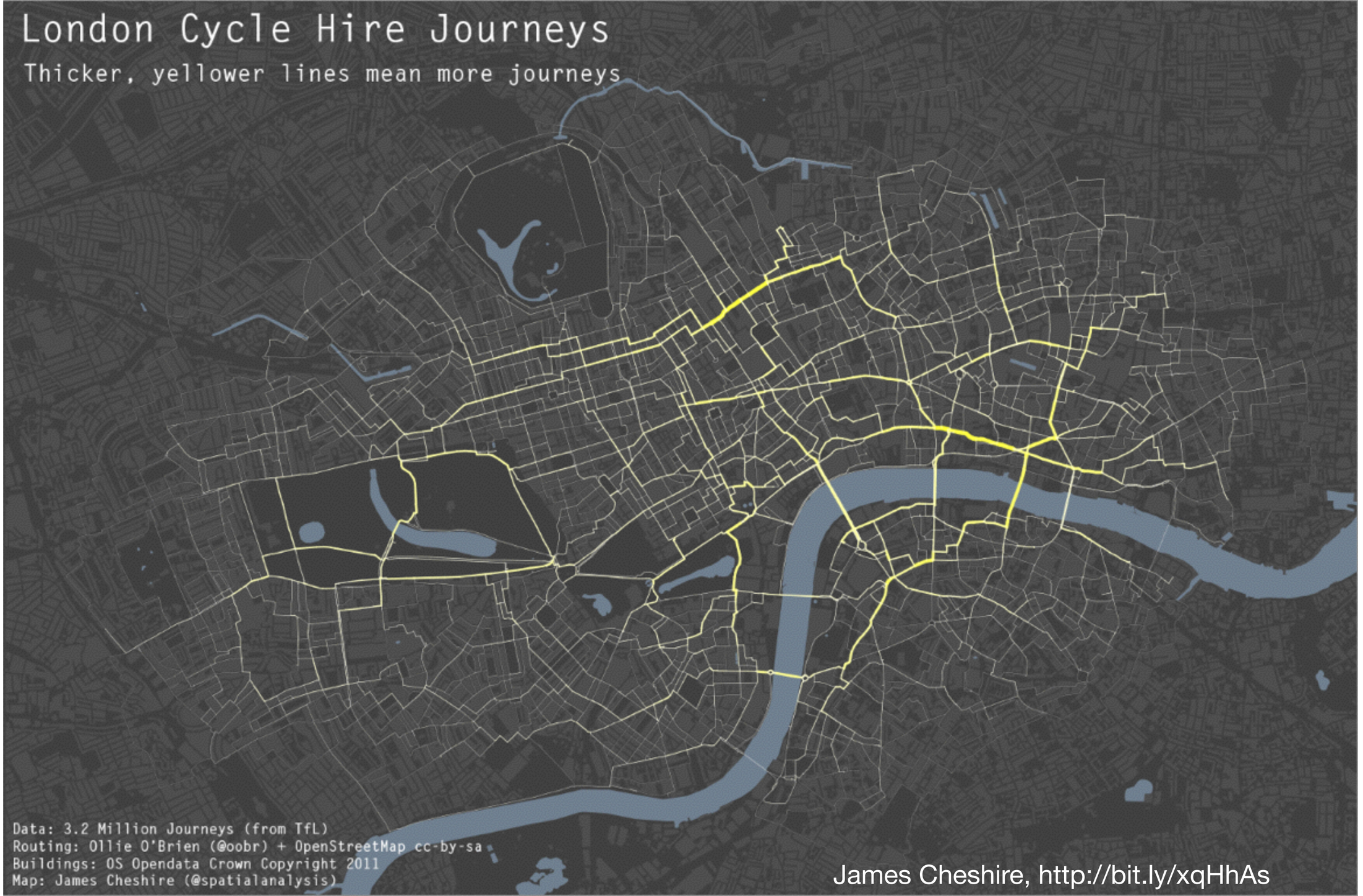




David B Sparks, http://bit.ly/hn54NW



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



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

Your turn

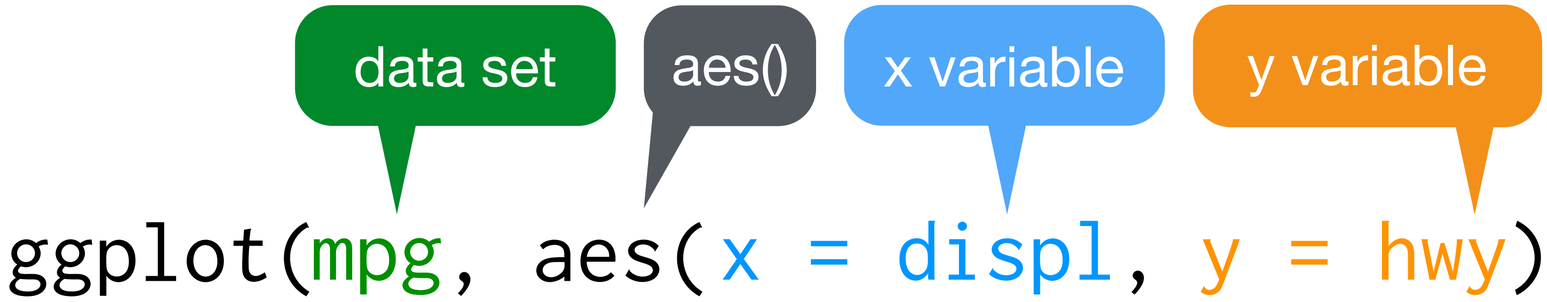
Confer with your neighbor.
What relationship do you expect to see between engine size (displ) and mileage (hwy)?

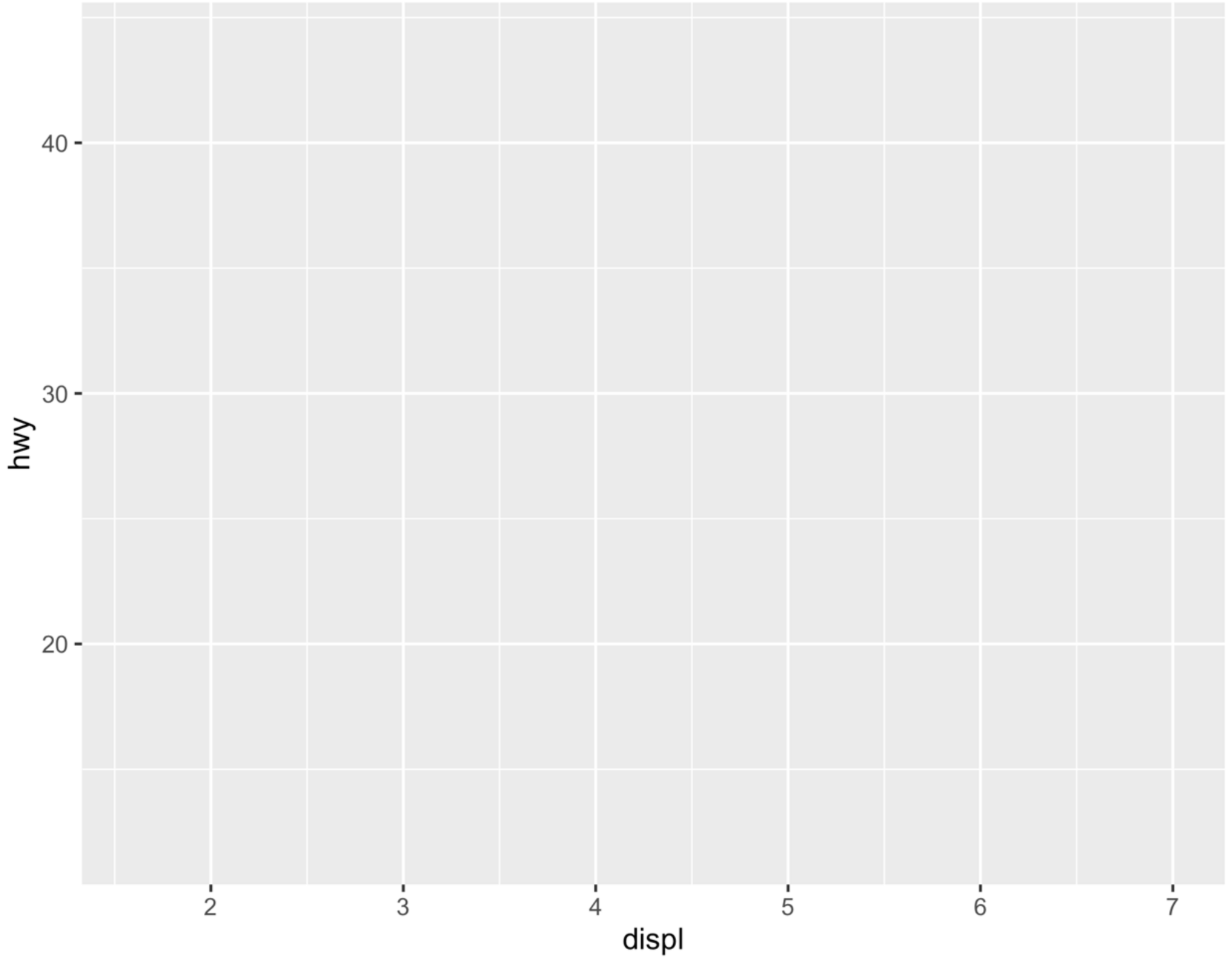
No peeking ahead!

How can we explore this?

ggplot

1. "Initialize" a plot with ggplot()

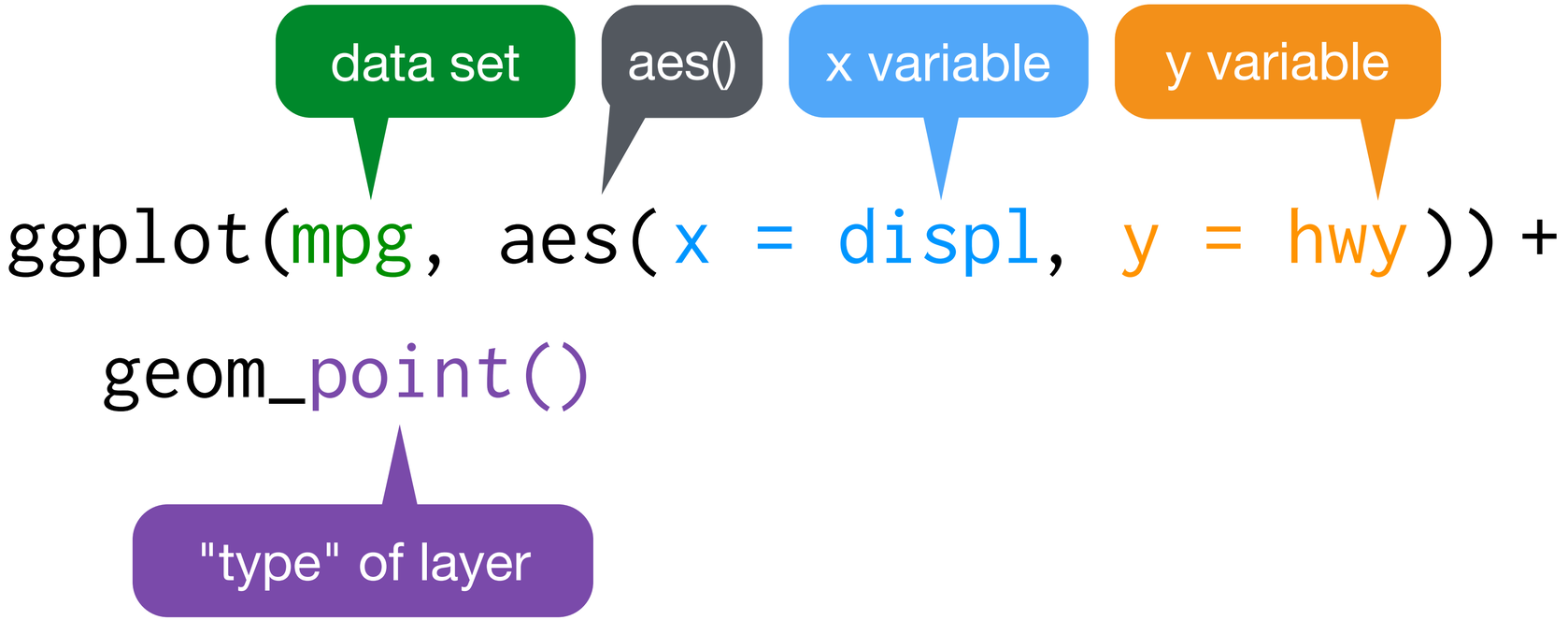


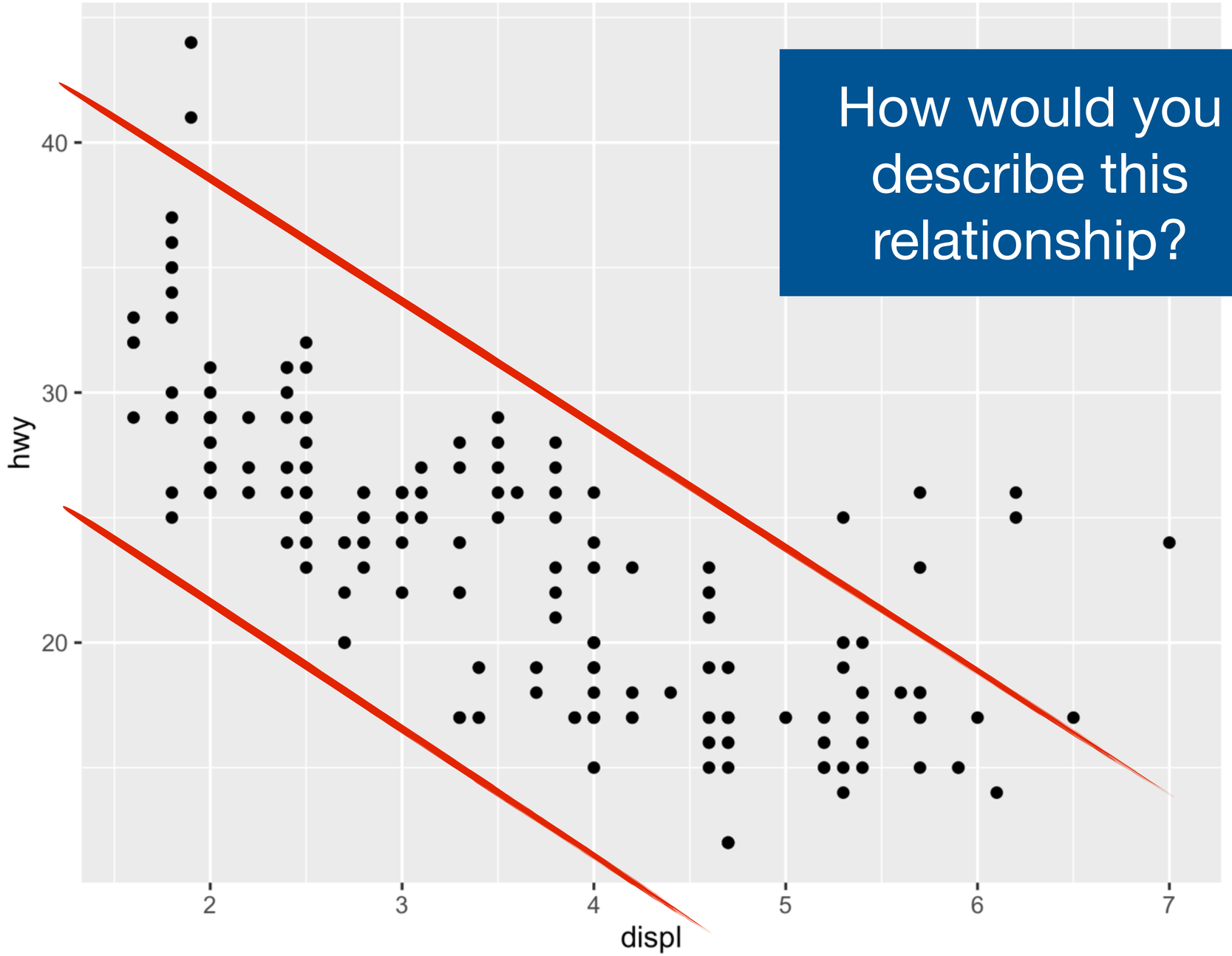



```
ggplot(mpg, aes(displ, hwy))
```

ggplot

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

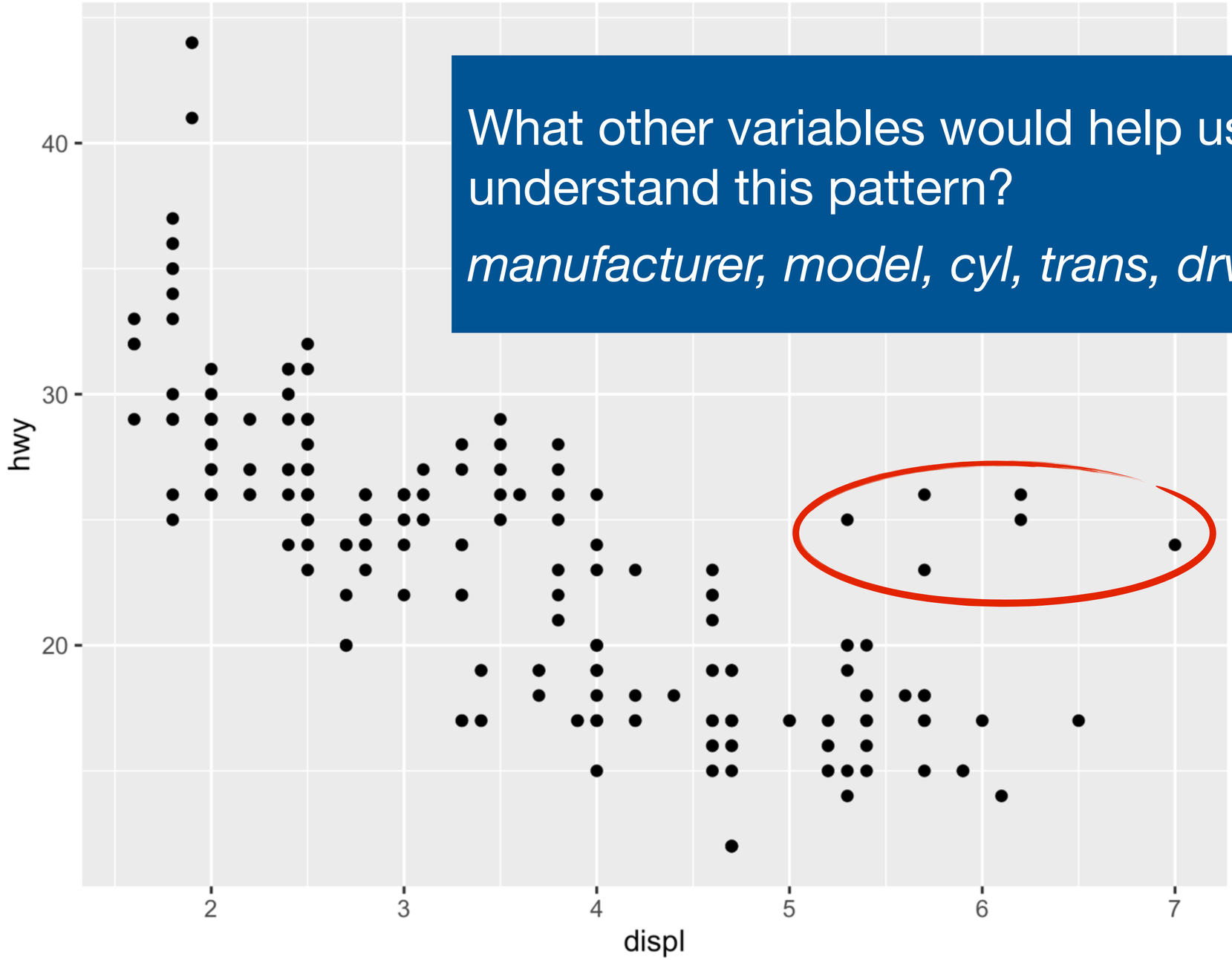





```
ggplot(mpg, aes(displ, hwy)) + geom_point()
```

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

- John Tukey



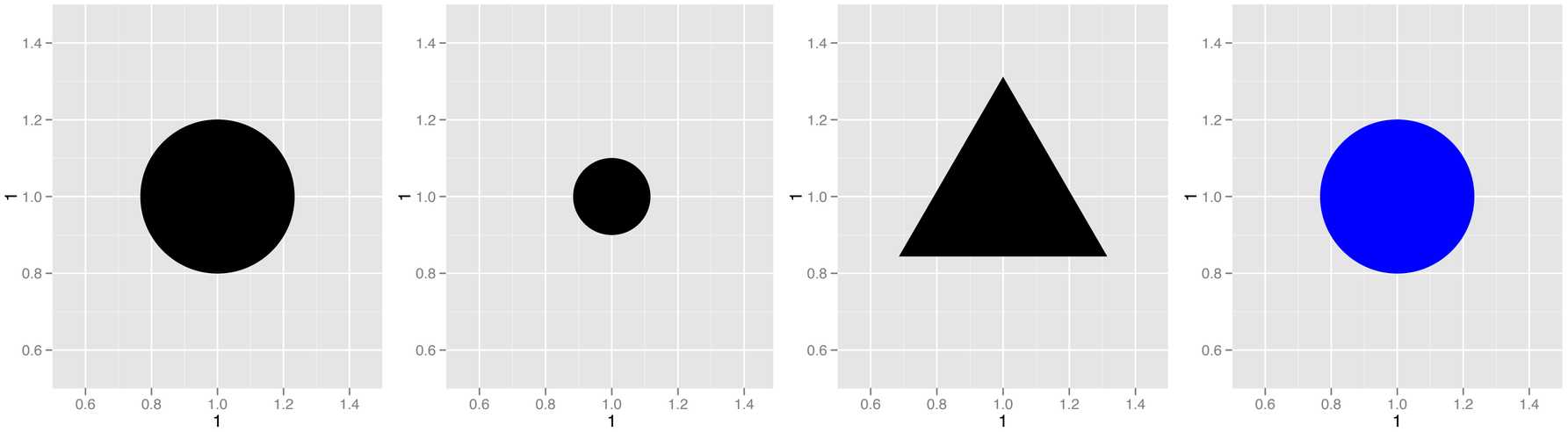
What other variables would help us understand this pattern?

manufacturer, model, cyl, trans, drv, class

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

Aesthetics

Aesthetics

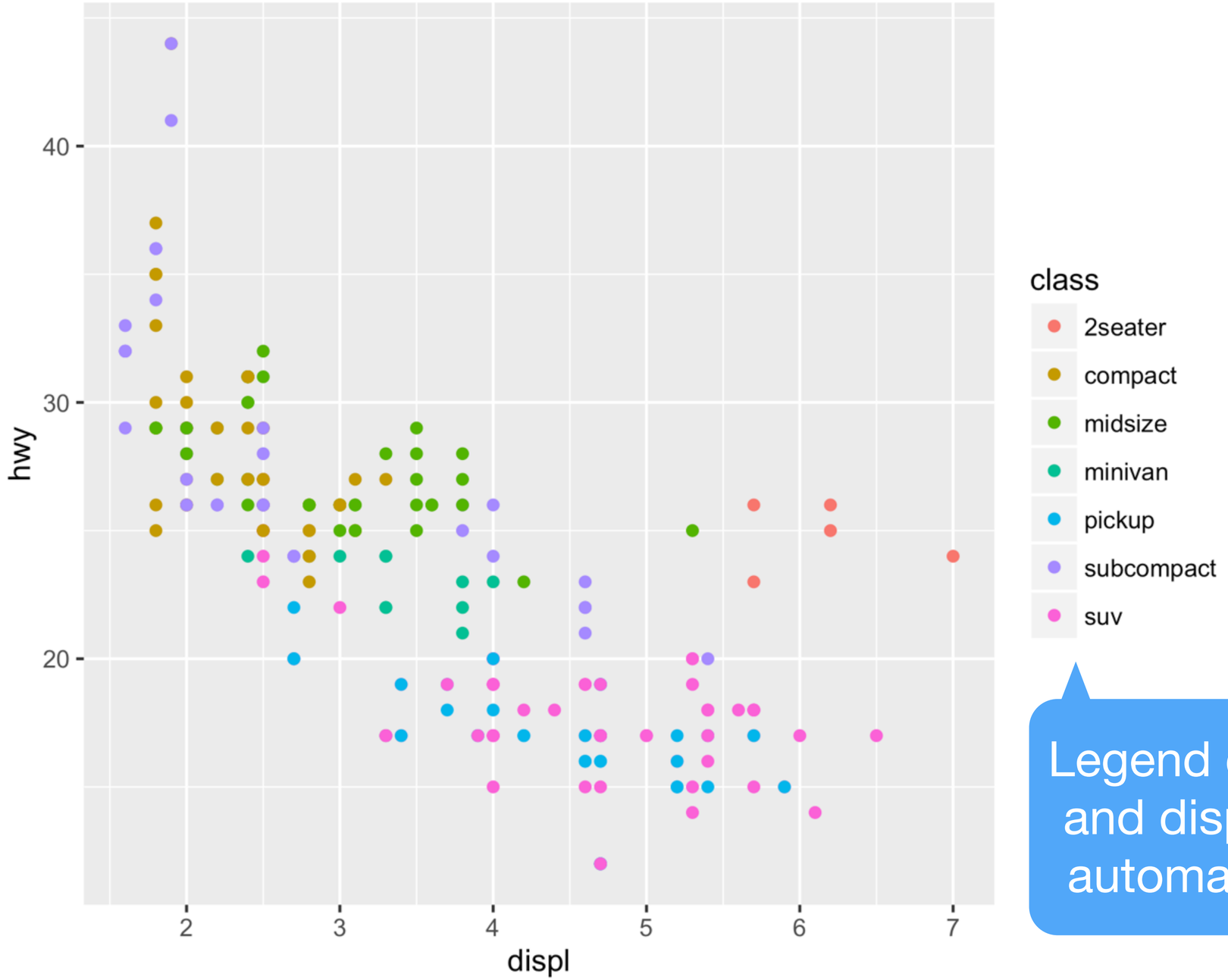


Aesthetics

aesthetic feature

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))
```



Legend chosen and displayed automatically.

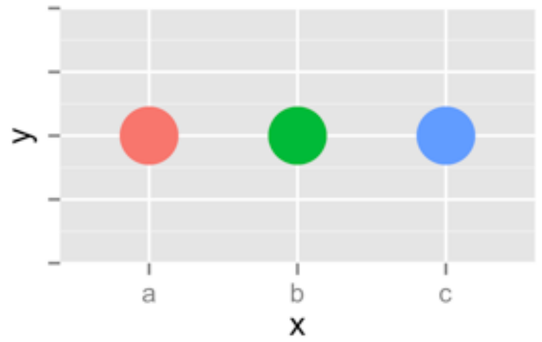
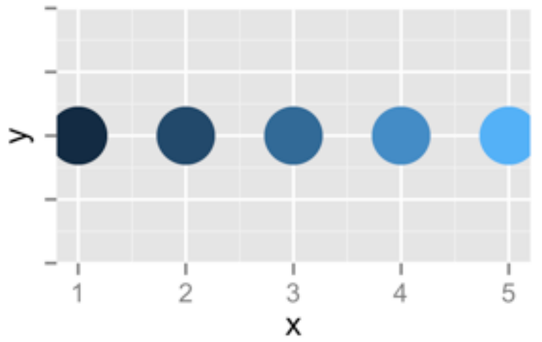
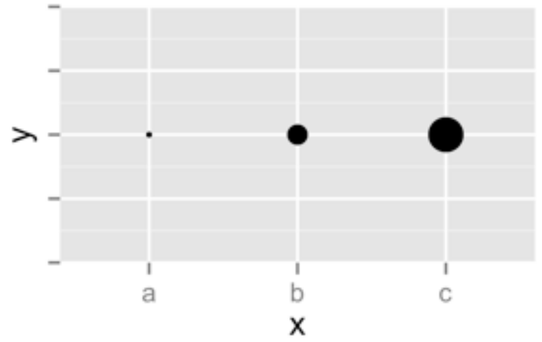
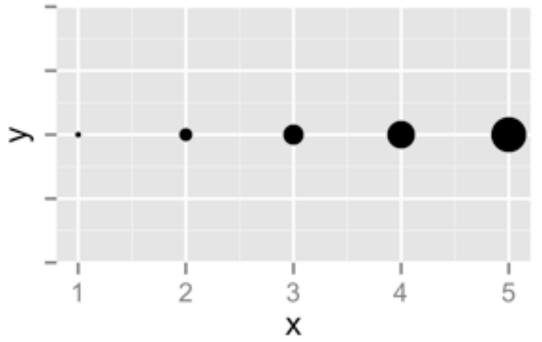
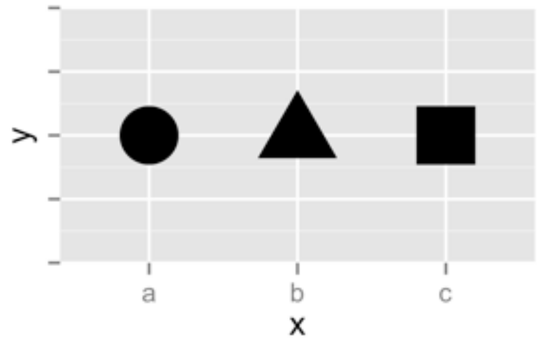
```
ggplot(mpg, aes(displ, hwy)) + geom_point(aes(color = class))
```

Your Turn

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

Do different things happen for discrete and continuous variables?

What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		

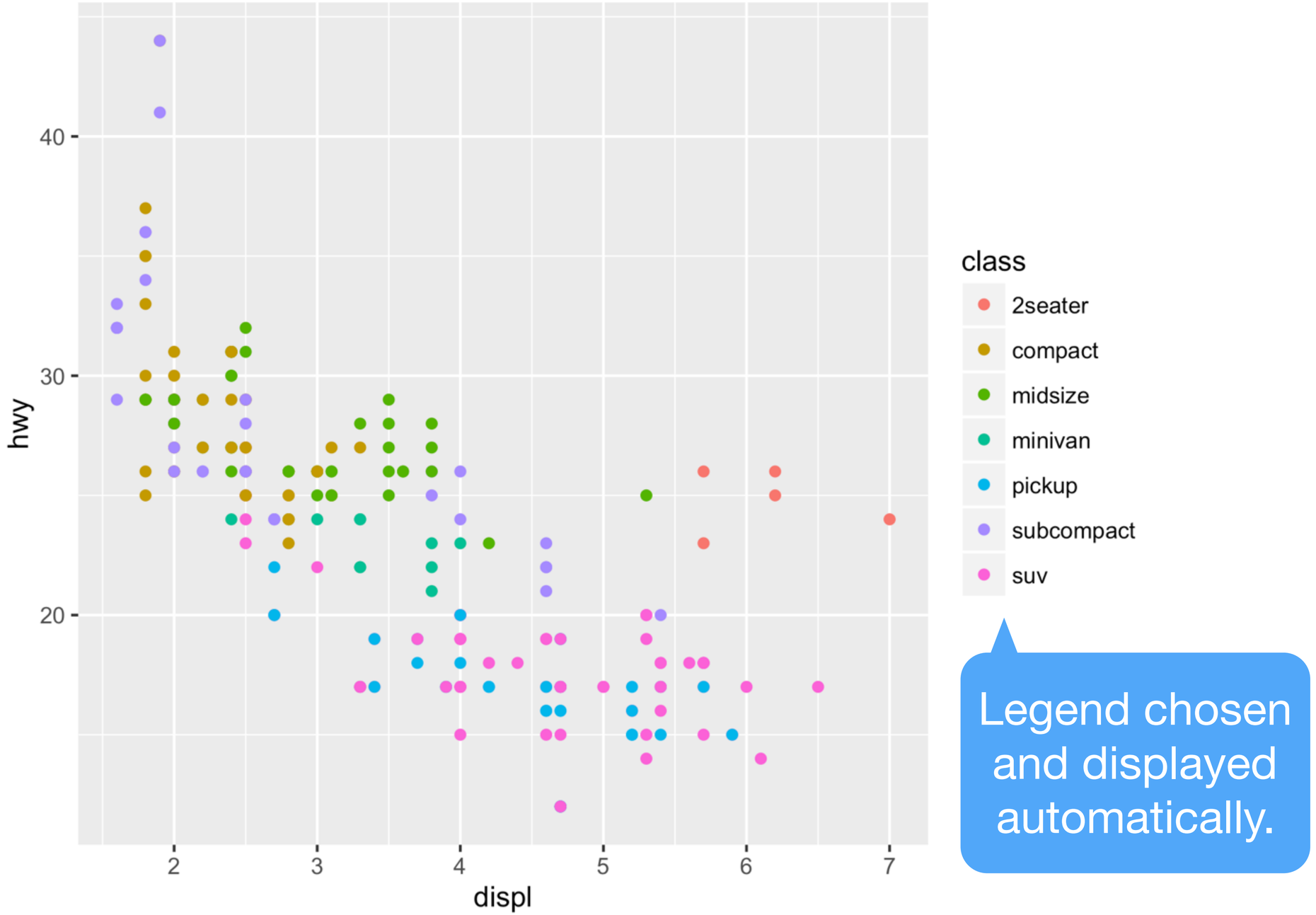
Mapping vs. setting

Use aes() (and the mapping argument) to build a mapping between data and aesthetic

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

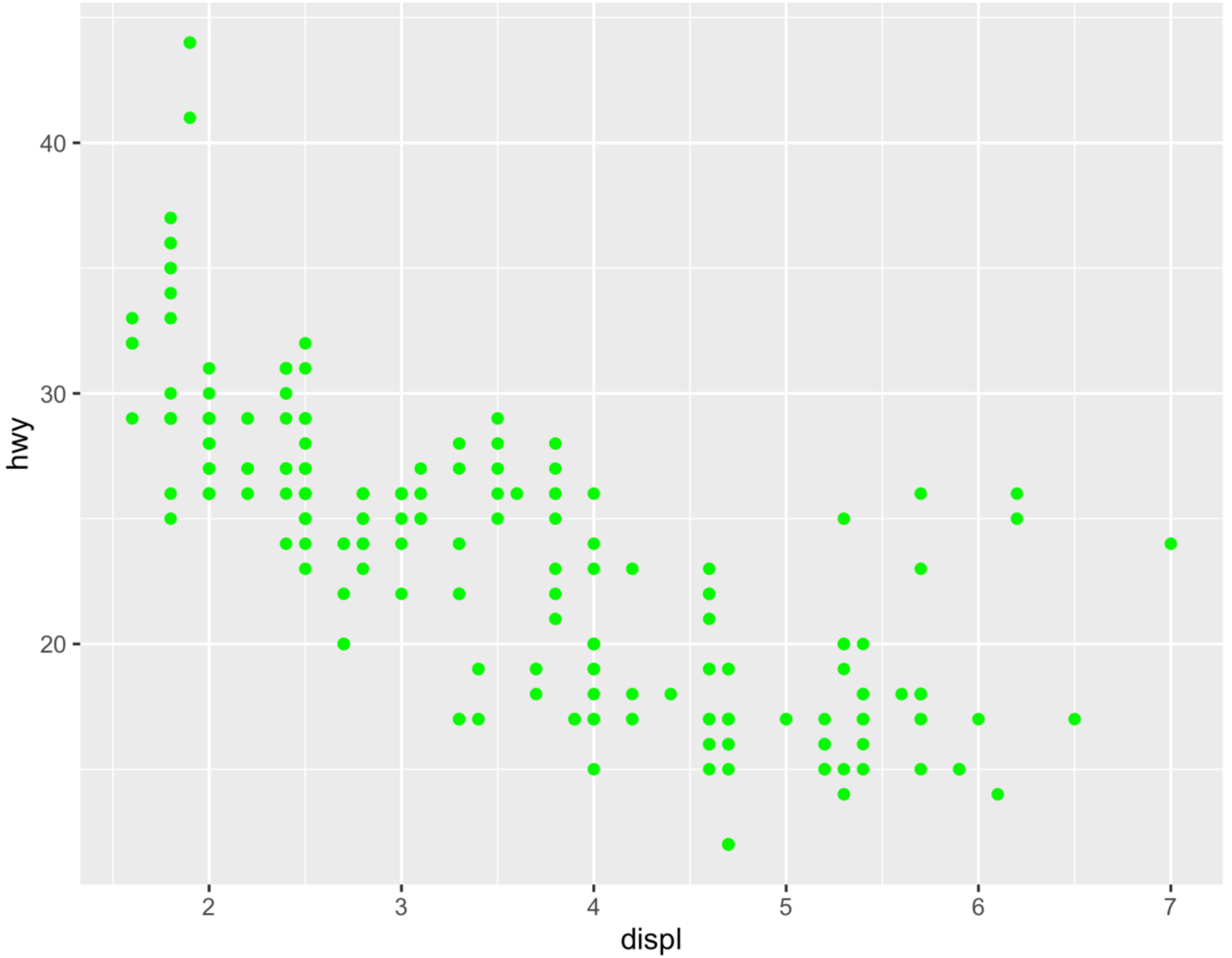
Manually set an aesthetic without aes()


```
ggplot(mpg, aes(displ, hwy)) + geom_point(color = "green")
ggplot(mpg, aes(displ, hwy)) + geom_point(size = 5)
ggplot(mpg, aes(displ, hwy)) + geom_point(shape = 3)
ggplot(mpg, aes(displ, hwy)) + geom_point(alpha = 0.5)
```



Legend chosen and displayed automatically.

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

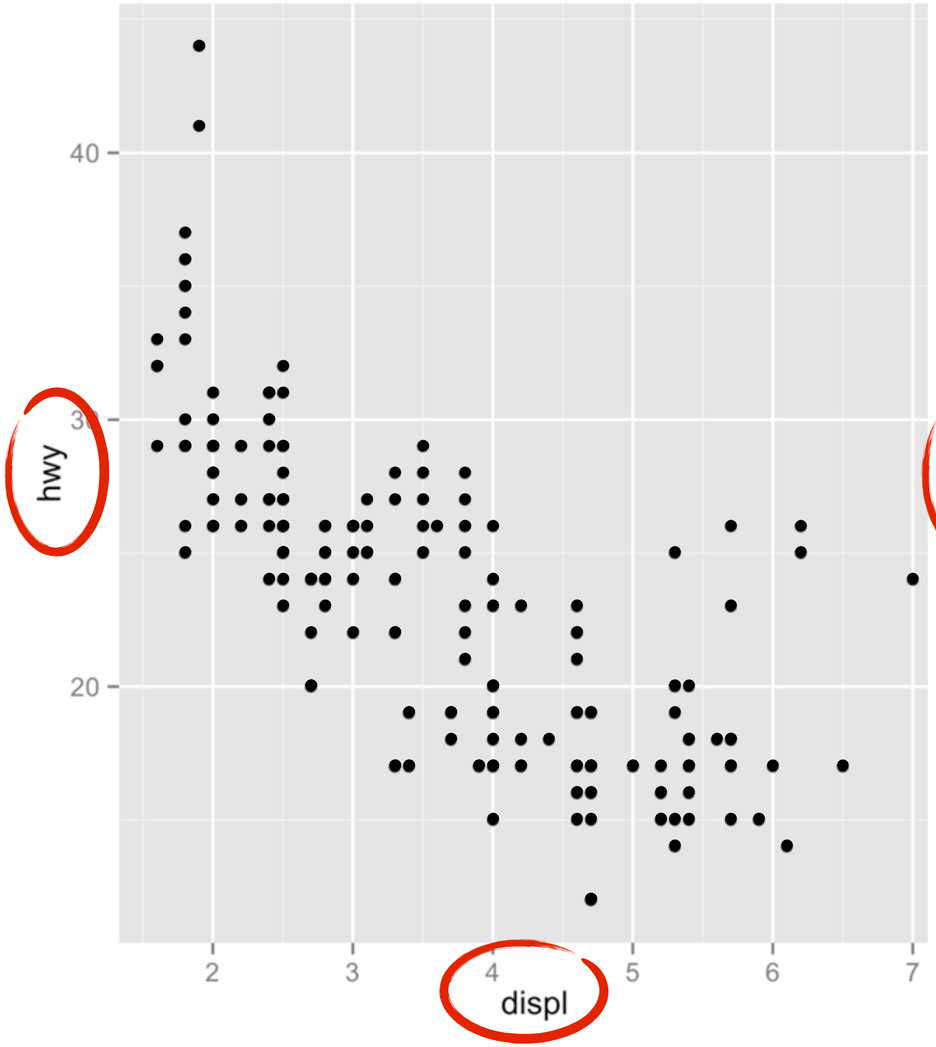


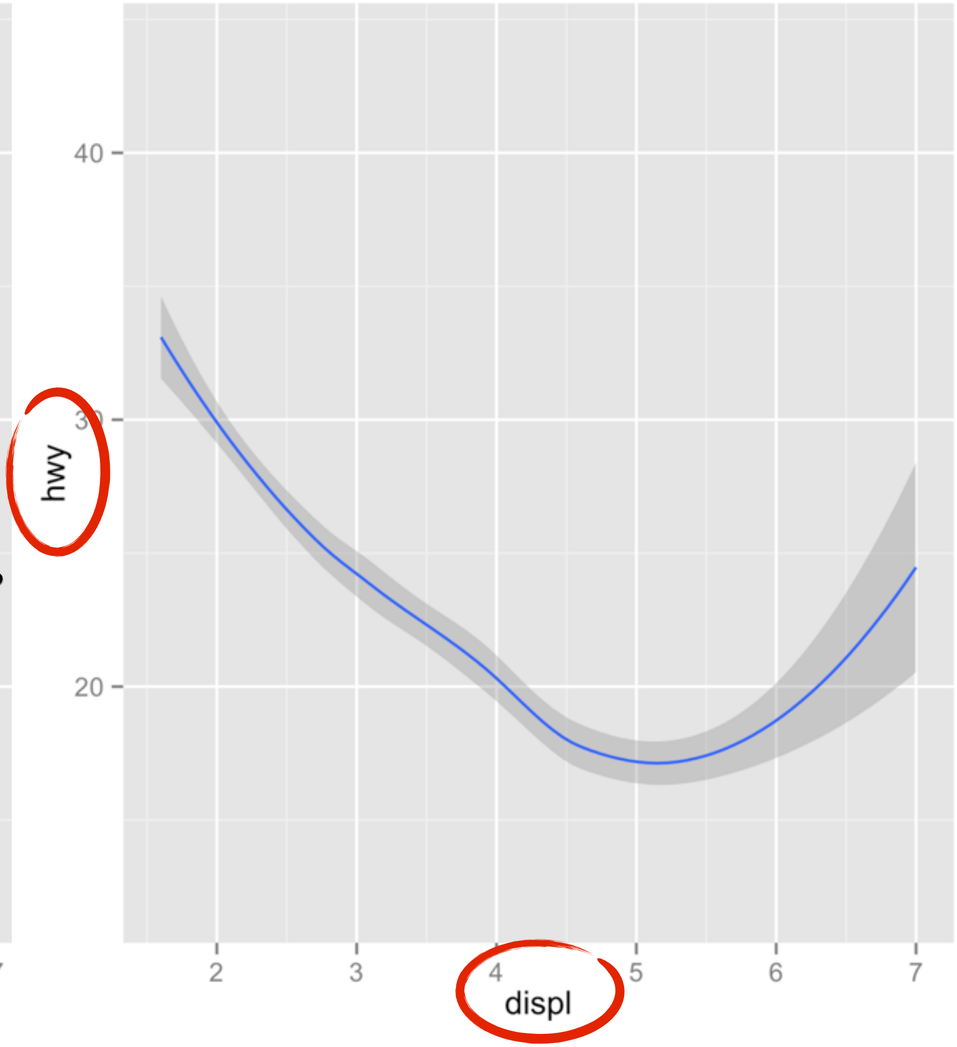

```
ggplot(mpg, aes(displ, hwy)) + geom_point(color = "green")
```

Geoms

How are these plots similar?

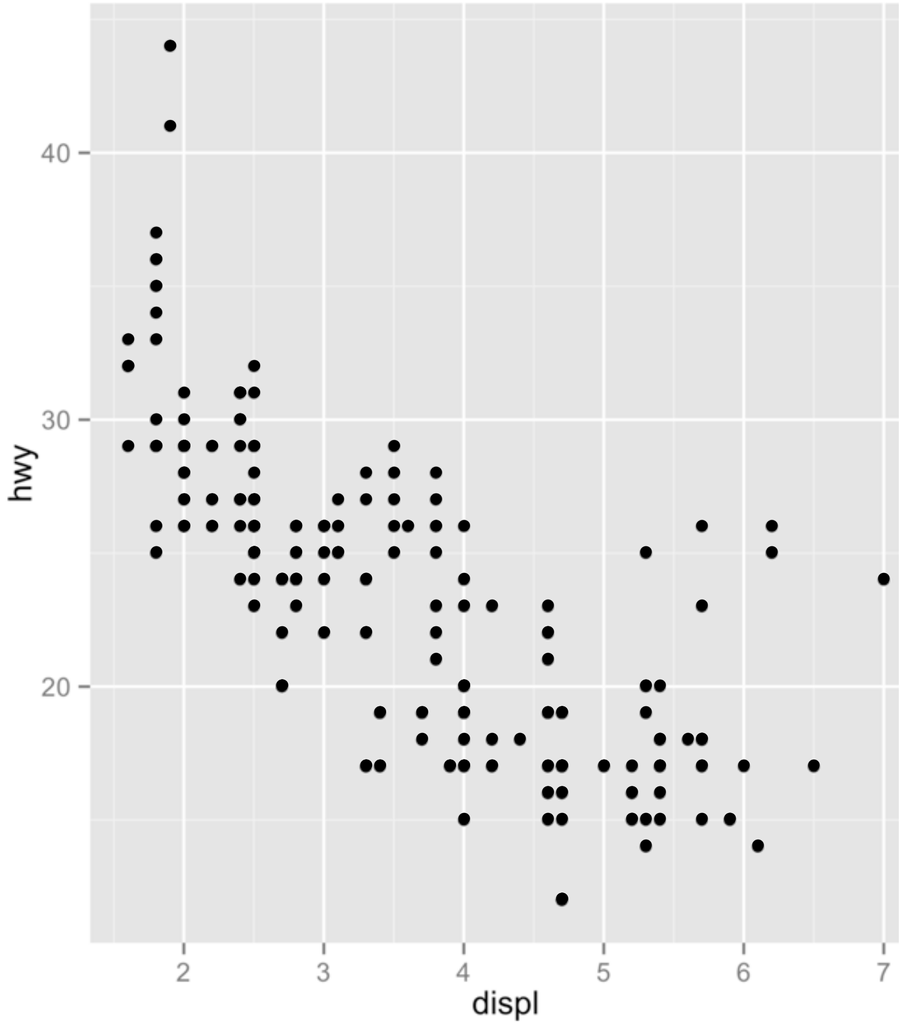
Same: x var, y var, data

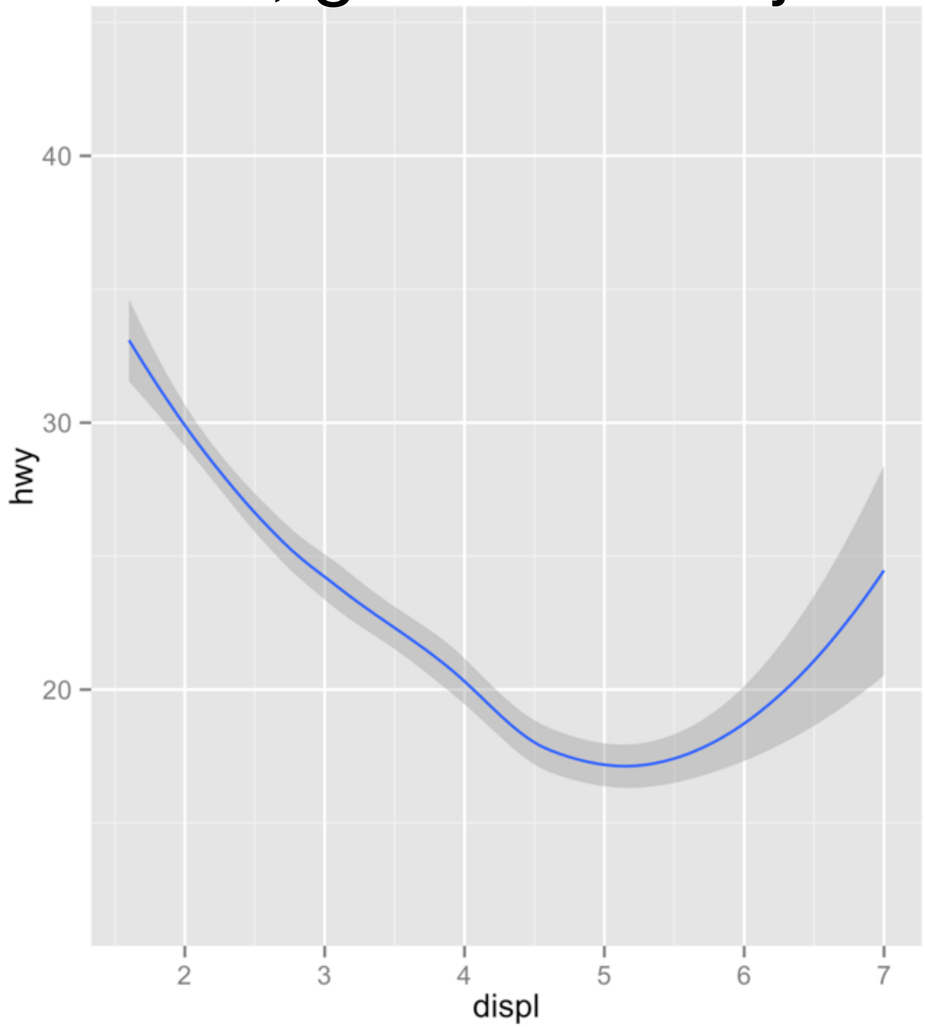




How are these plots different?

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





Geometric object

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

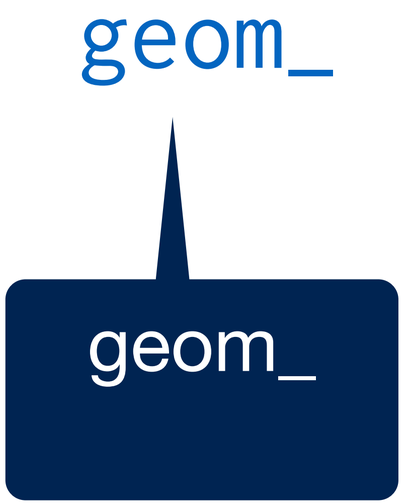
```
ggplot(data = mpg) +
  geom_point(aes(x = displ, y = hwy))
```

A Geom

Geoms add the layers to the graph.

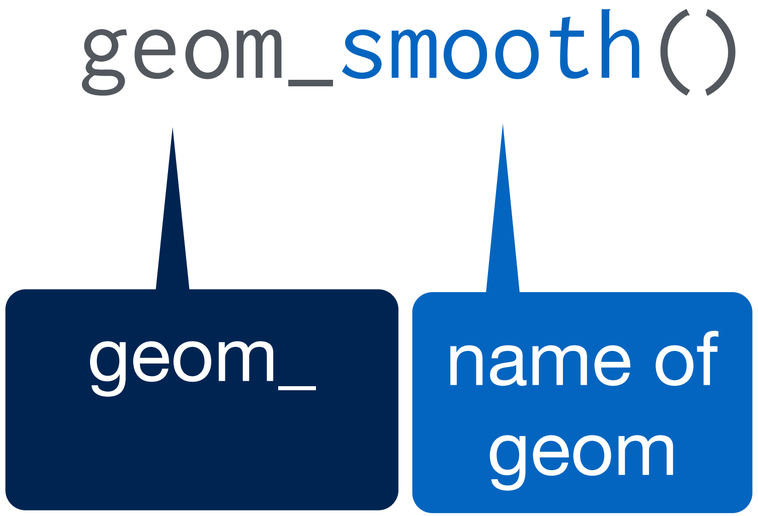
Geometric object

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



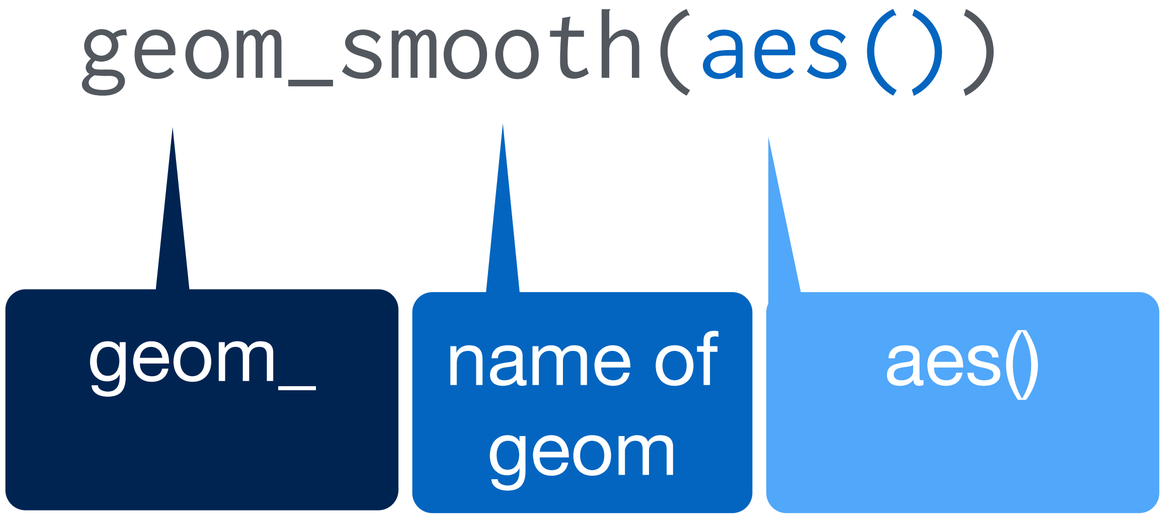
Geometric object

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



Geometric object

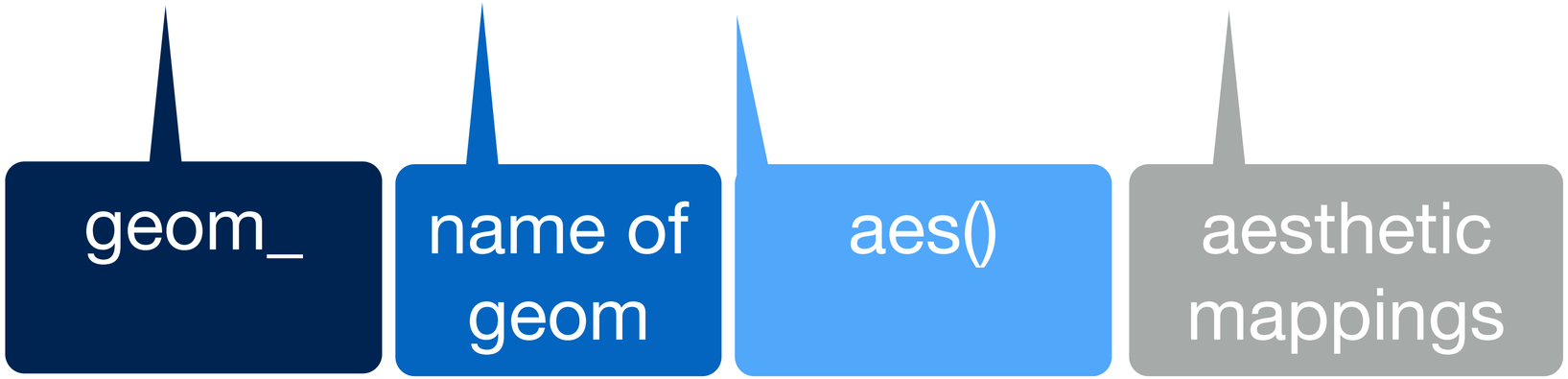
The "type" of graph, or what the graph draws



Geometric object

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

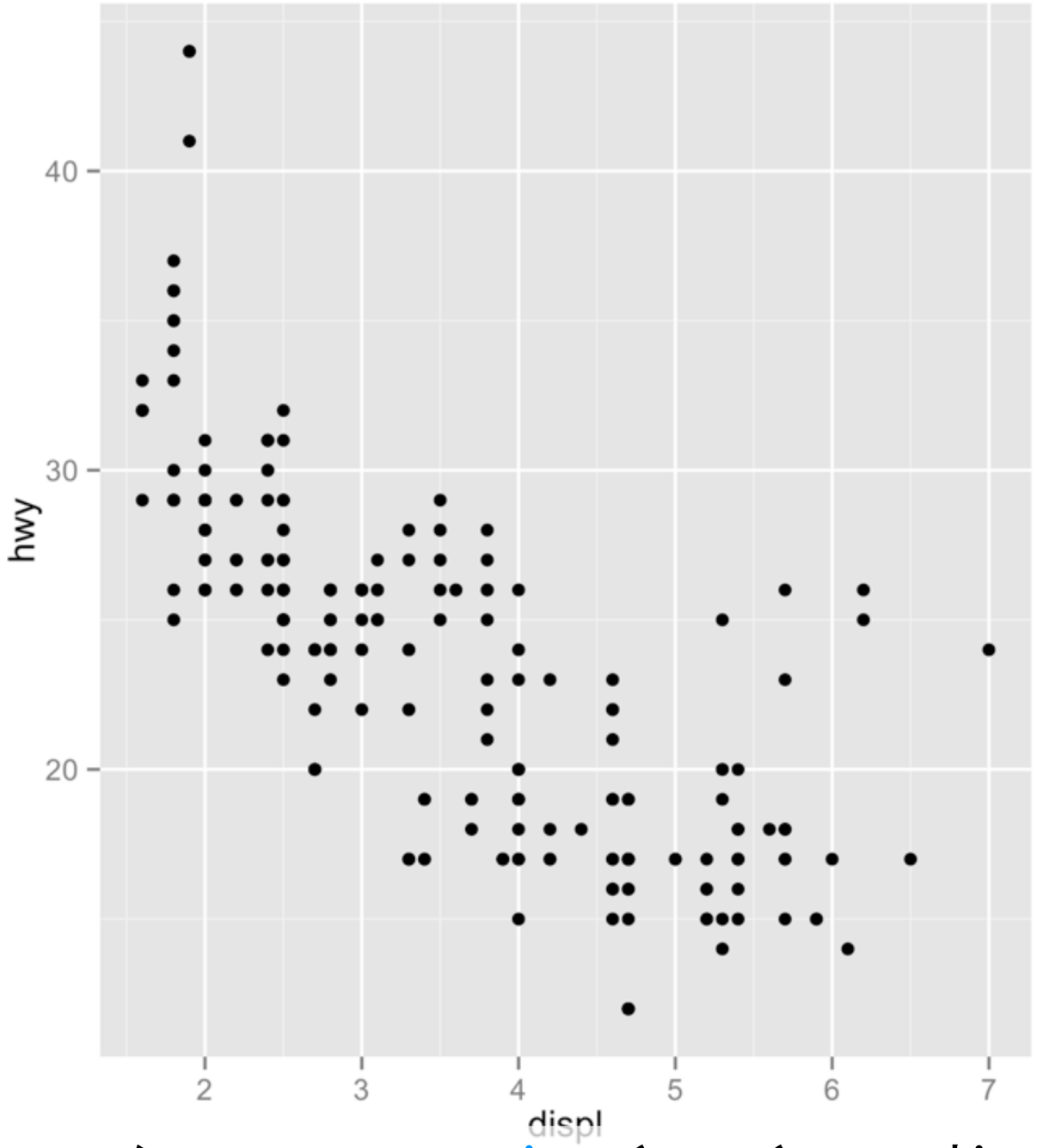
```
geom_smooth(aes(x = displ, y = hwy))
```



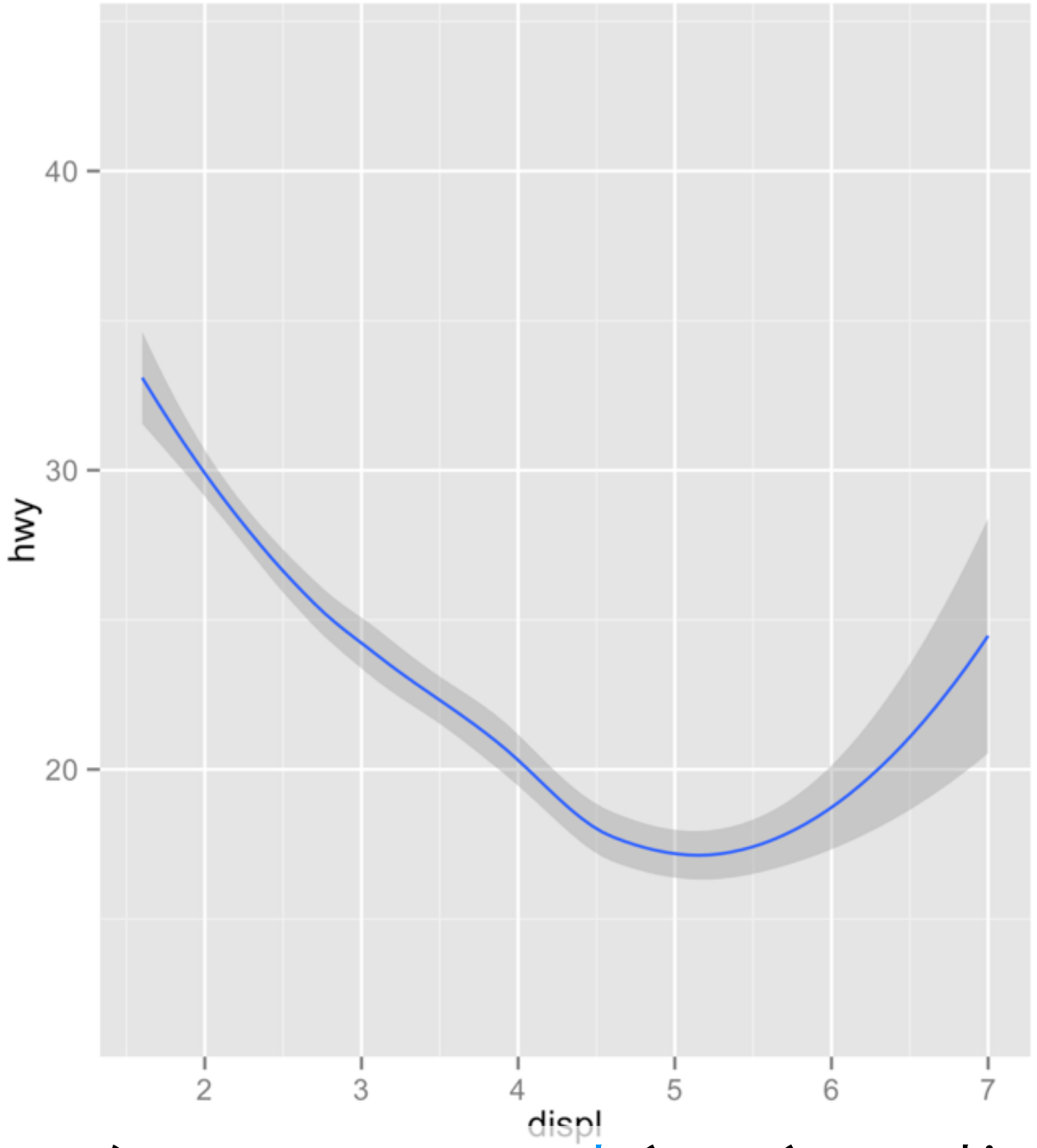
Geometric object

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

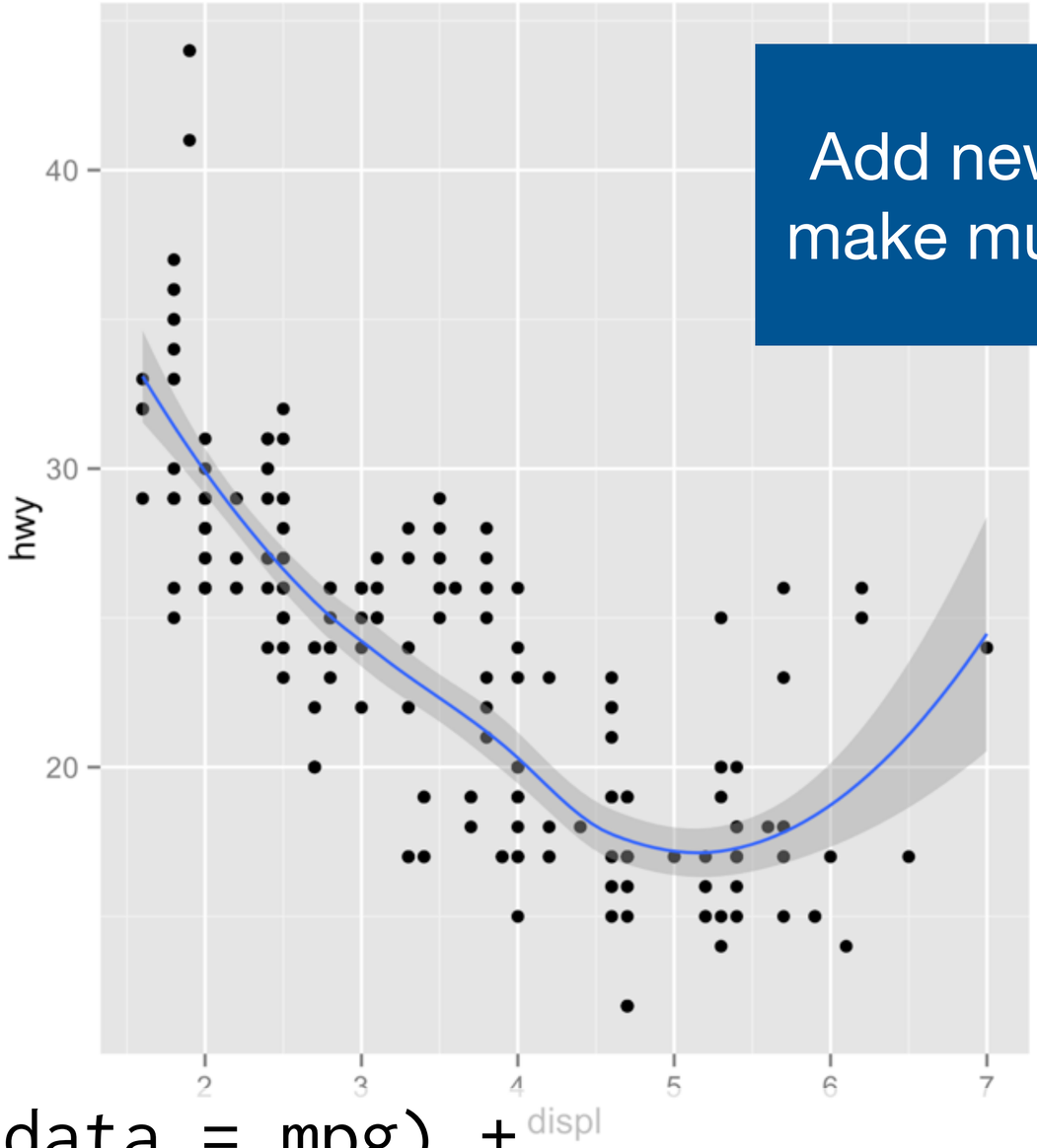
```
geom_smooth(aes(x = displ, y = hwy), …)
```




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




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



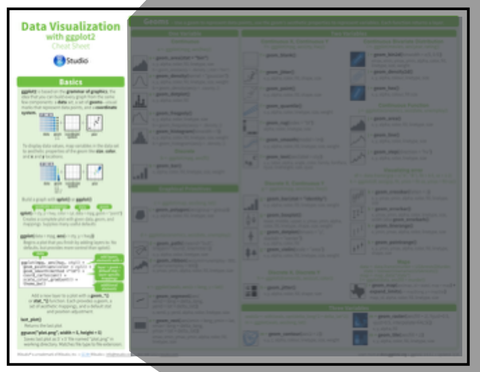
Add new geoms to make multiple layers

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

RStudio

Geoms

Help ▶
Cheatsheets ▶
Data Visualization with ggplot2



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

One Variable

Continuous

```
a <- ggplot(mpg, aes(hwy))
```

a + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

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

a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + geom_histogram(binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

```
b <- ggplot(mpg, aes(fl))
```

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

Graphical Primitives

```
map <- map_data("state")
c <- ggplot(map, aes(long, lat))
```

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

```
d <- ggplot(economics, aes(date, unemploy))
```

d + geom_path(lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, linetype, size

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

```
e <- ggplot(seals, aes(x = long, y = lat))
```

e + geom_segment(aes(
xend = long + delta_long,
yend = lat + delta_lat)**)**
x, xend, y, yend, alpha, color, linetype, size

e + geom_rect(aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

```
f <- ggplot(mpg, aes(cty, hwy))
```

f + geom_blank()
(Useful for expanding limits)

f + geom_jitter()
x, y, alpha, color, fill, shape, size

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

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

f + geom_rug(sides = "bl"**)**
alpha, color, linetype, size

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

f + geom_text(aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

```
g <- ggplot(mpg, aes(class, hwy))
```

g + geom_bar(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

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

g + geom_dotplot(binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

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

Discrete X, Discrete Y

```
h <- ggplot(diamonds, aes(cut, color))
```

h + geom_jitter()
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

```
i <- ggplot(movies, aes(year, rating))
```

i + geom_bin2d(binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + geom_density2d()
x, y, alpha, colour, linetype, size

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

Continuous Function

```
j <- ggplot(economics, aes(date, unemploy))
```

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

j + geom_line()
x, y, alpha, color, linetype, size

j + geom_step(direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error

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

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

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

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

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

Maps

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

l + 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))
m <- ggplot(seals, aes(long, lat))
```

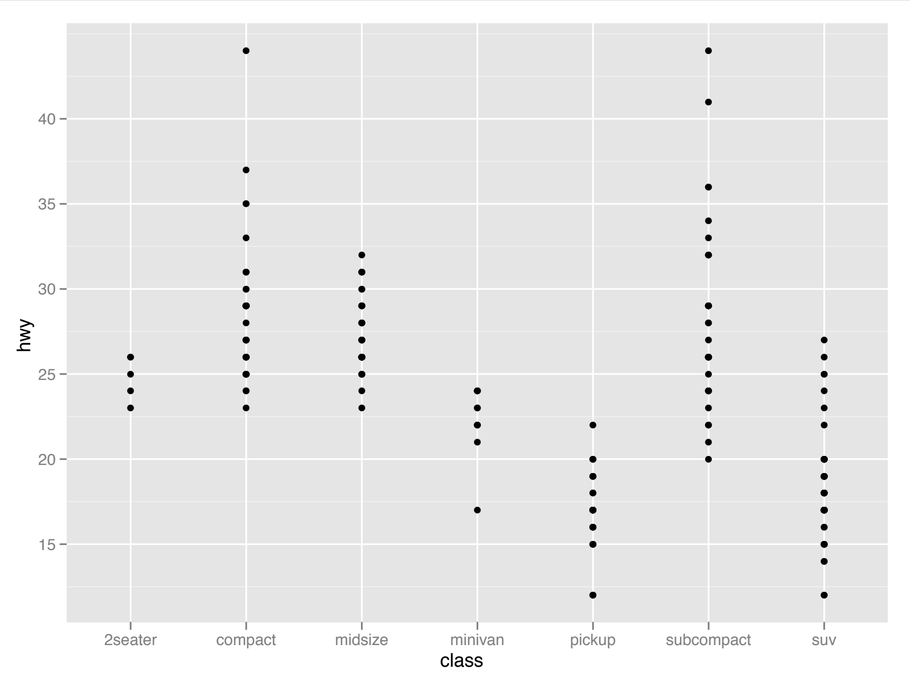
m + geom_contour(aes(z = z)**)**
x, y, z, alpha, colour, linetype, size, weight

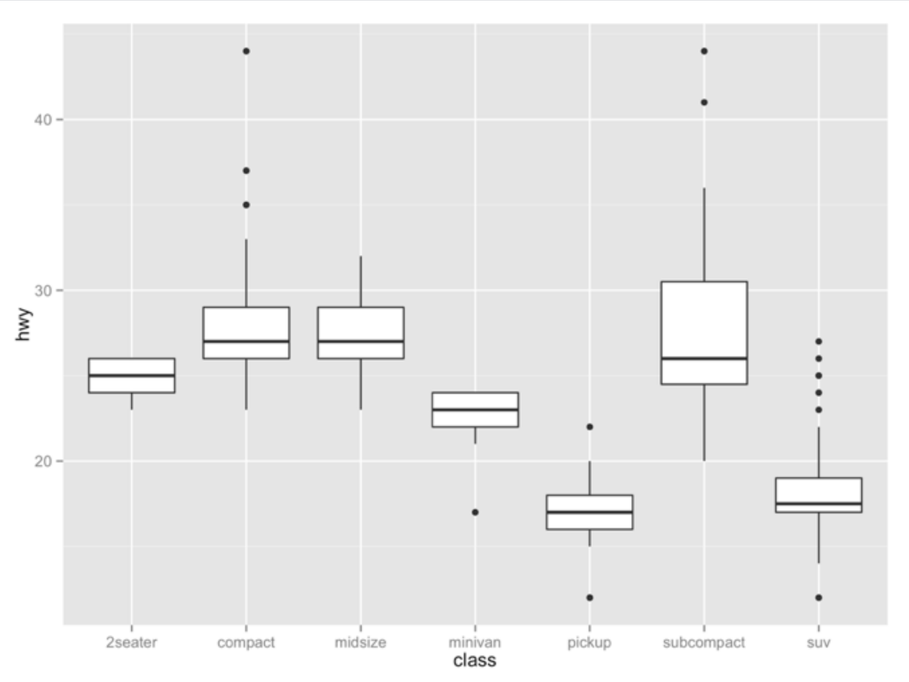
m + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill (fast)

m + geom_tile(aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size (slow)

Your Turn

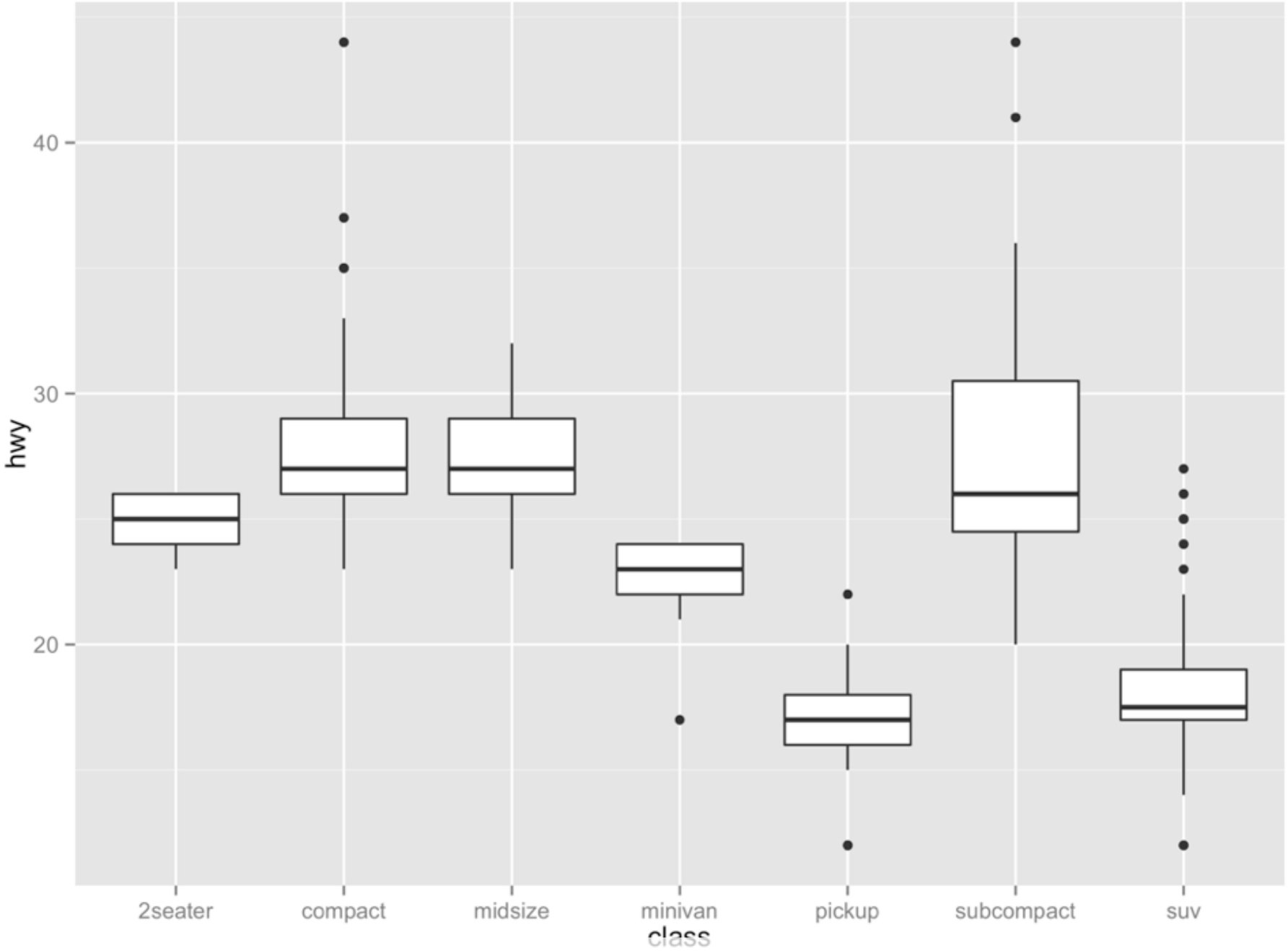
How would you replace this scatterplot with one that draws boxplots? Try out your best guess.






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

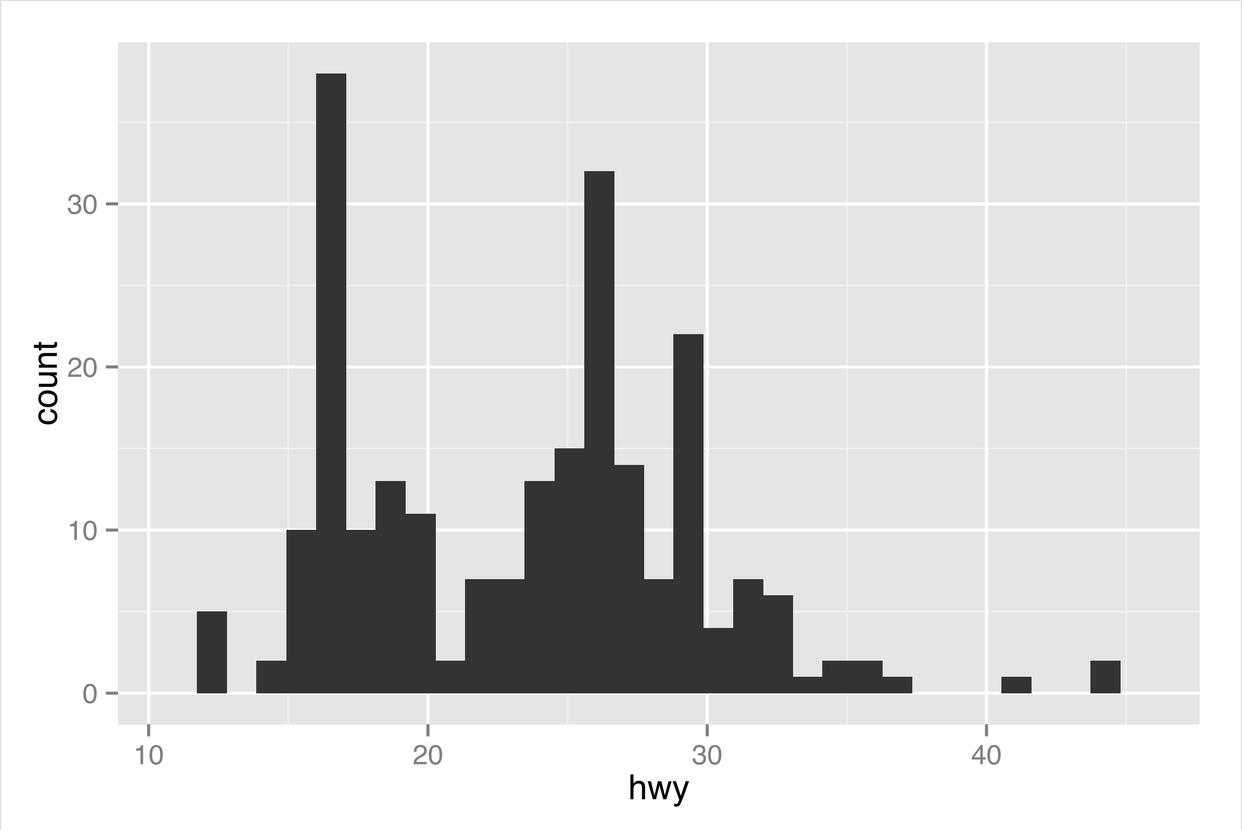
02:00

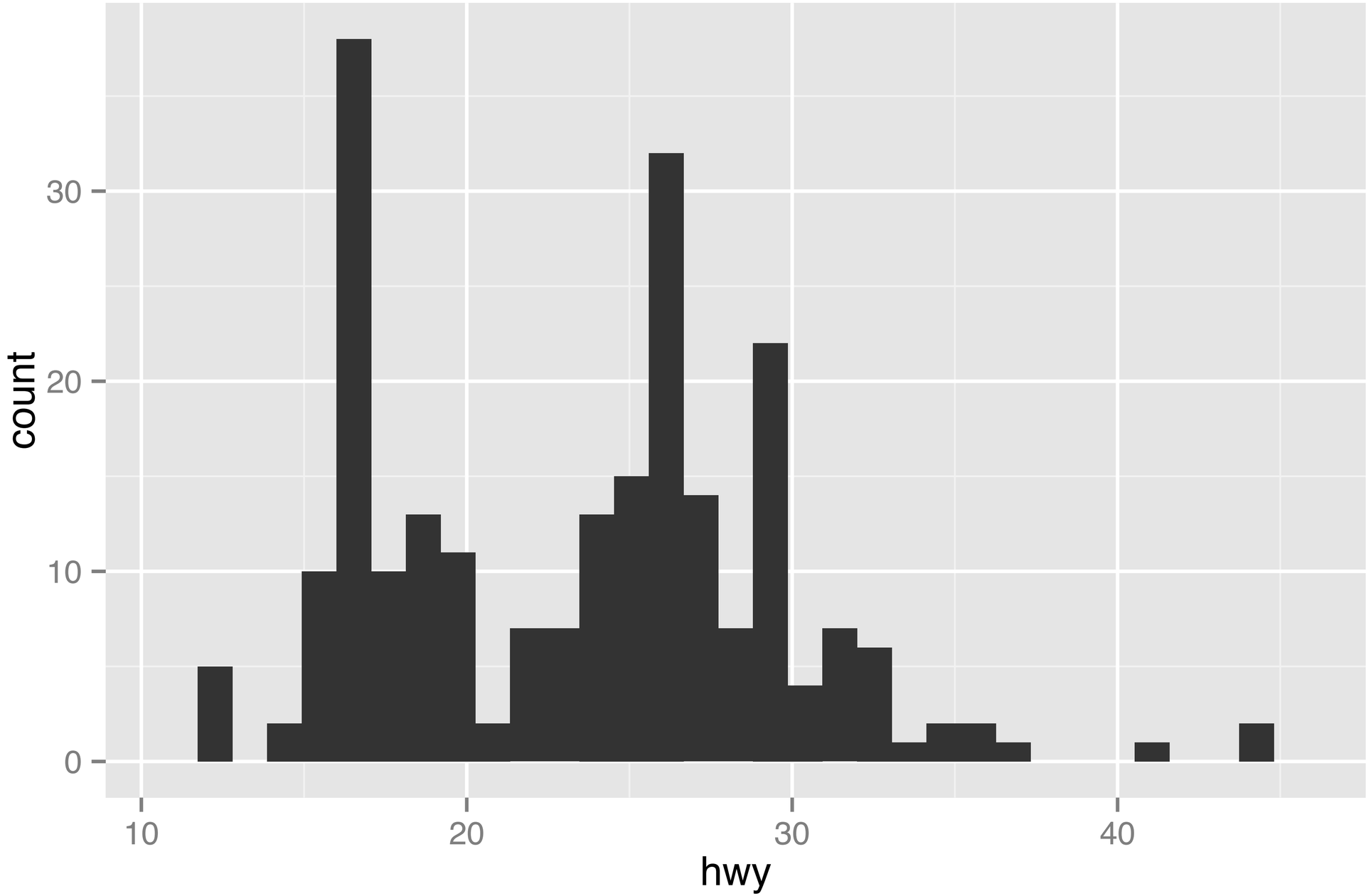



```
ggplot(data = mpg) + geom_boxplot(aes(x = displ, y = hwy))
```

Your Turn

How would you create this plot?
Hint: histograms do not require a y aesthetic.






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

Global vs. Local

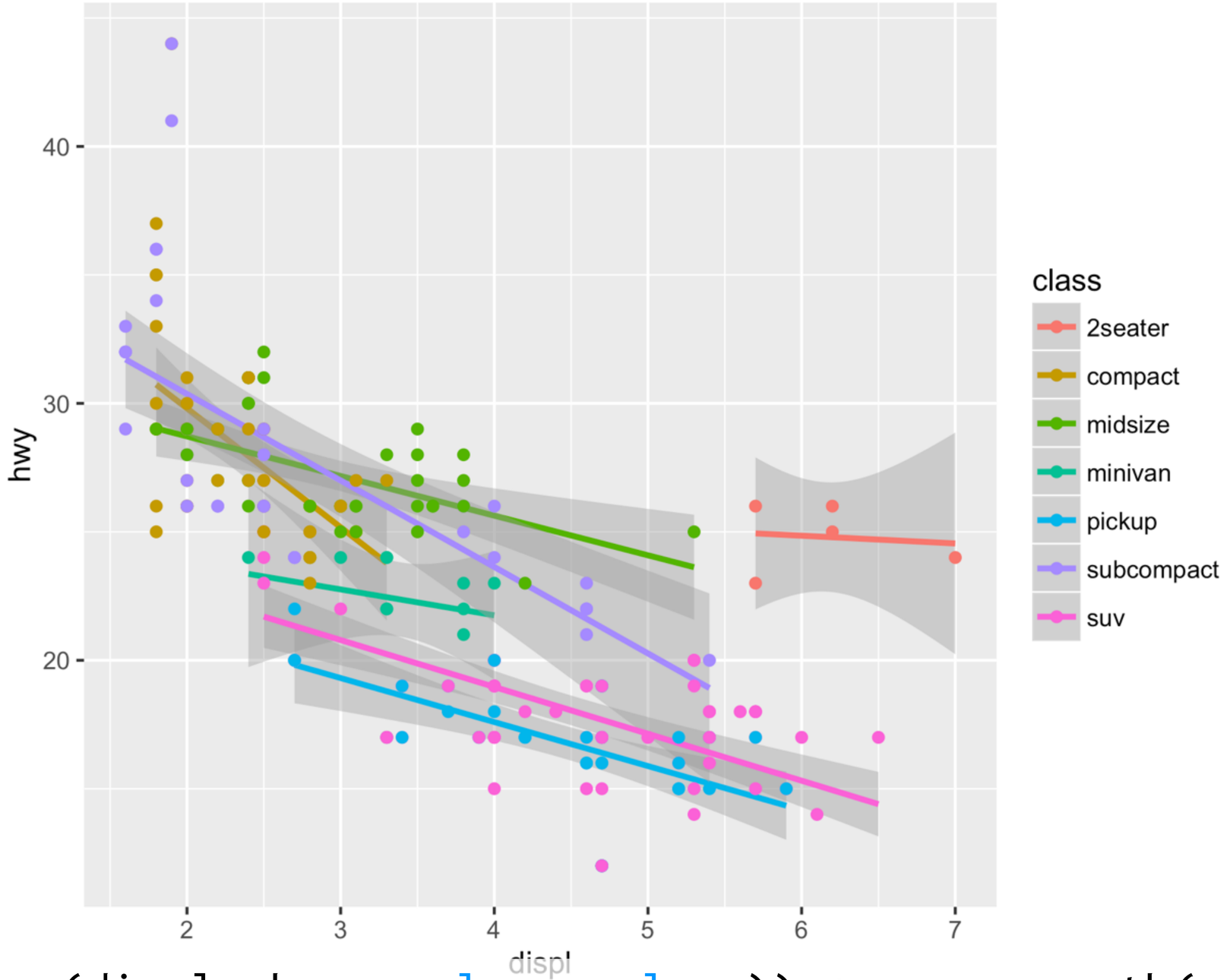
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(method = lm) +
  geom_point(aes(color = cyl), data = mpg[1:10, ])
```

Global vs. Local

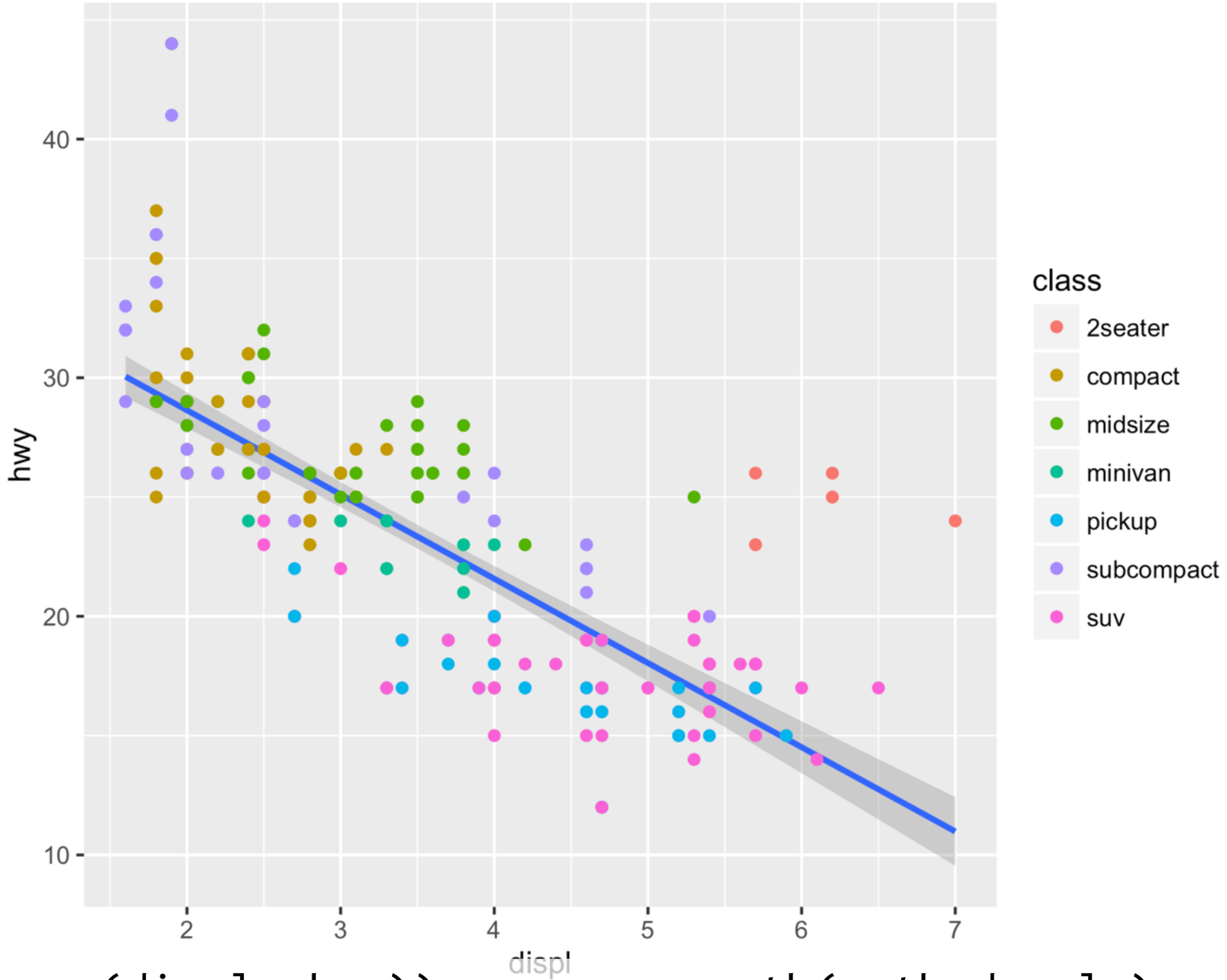
global aesthetics:
default for every layer

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(method = lm) +
  geom_point(aes(color = cyl), data = mpg[1:10, ])
```

local aesthetics:
for this layer only




```
ggplot(mpg, aes(displ, hwy, color = class)) + geom_smooth(method = lm) +
  geom_point()
```



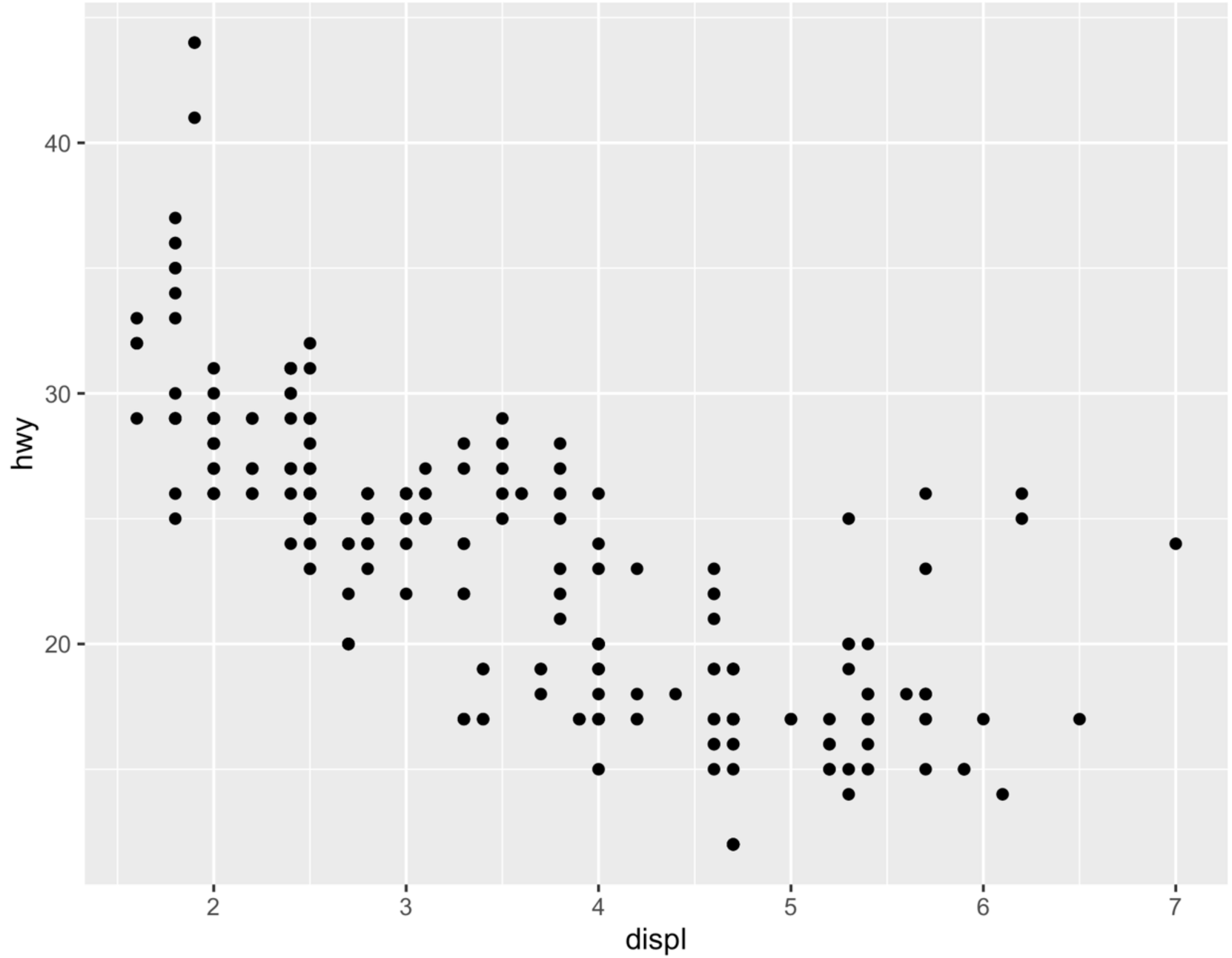

```
ggplot(mpg, aes(displ, hwy)) + geom_smooth(method = lm) +
  geom_point(aes(color = class))
```

Global vs. Local

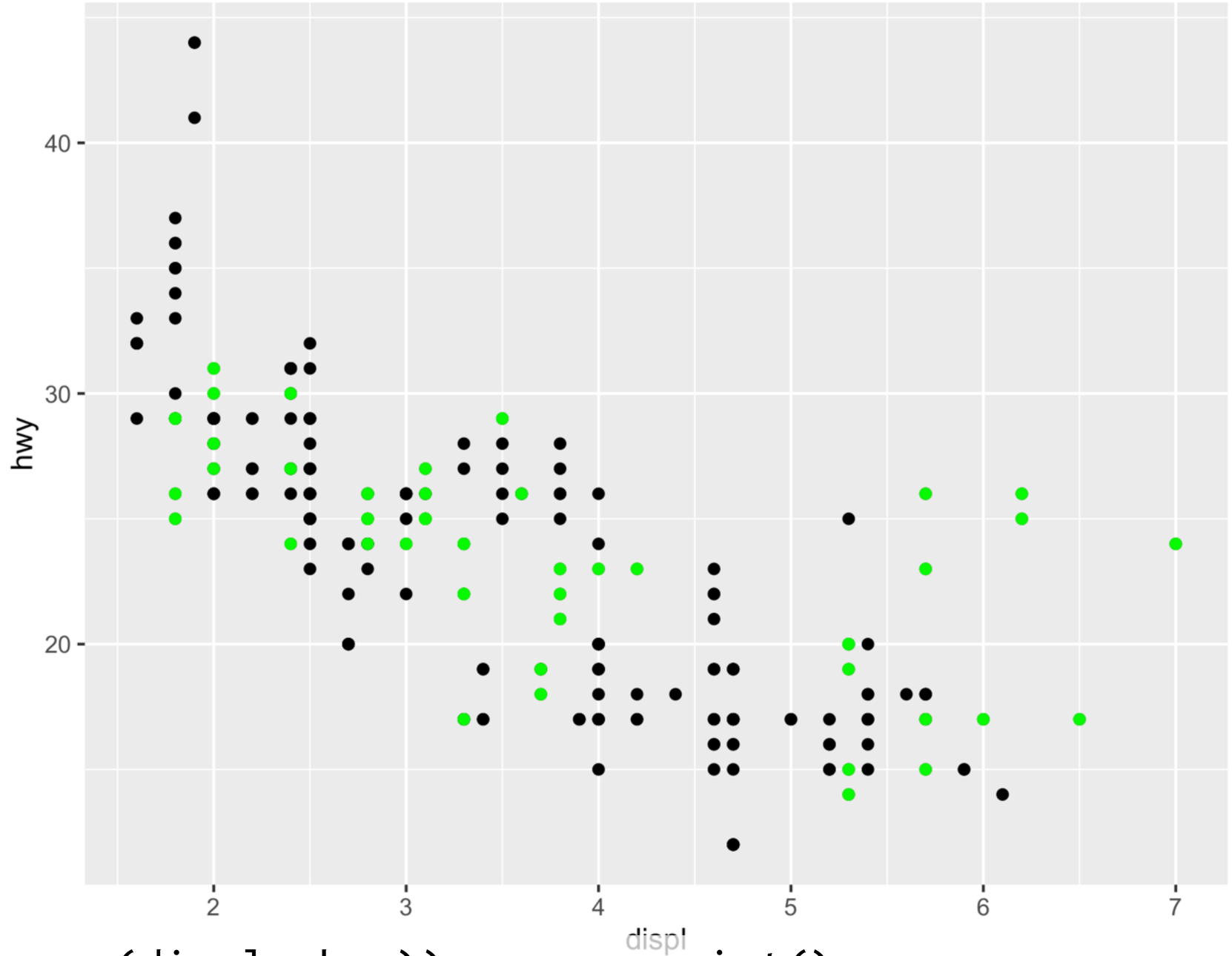
global data:
default for every layer

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(method = lm) +
  geom_point(aes(color = cyl), data = mpg[1:10, ])
```

local data:
data for this layer only



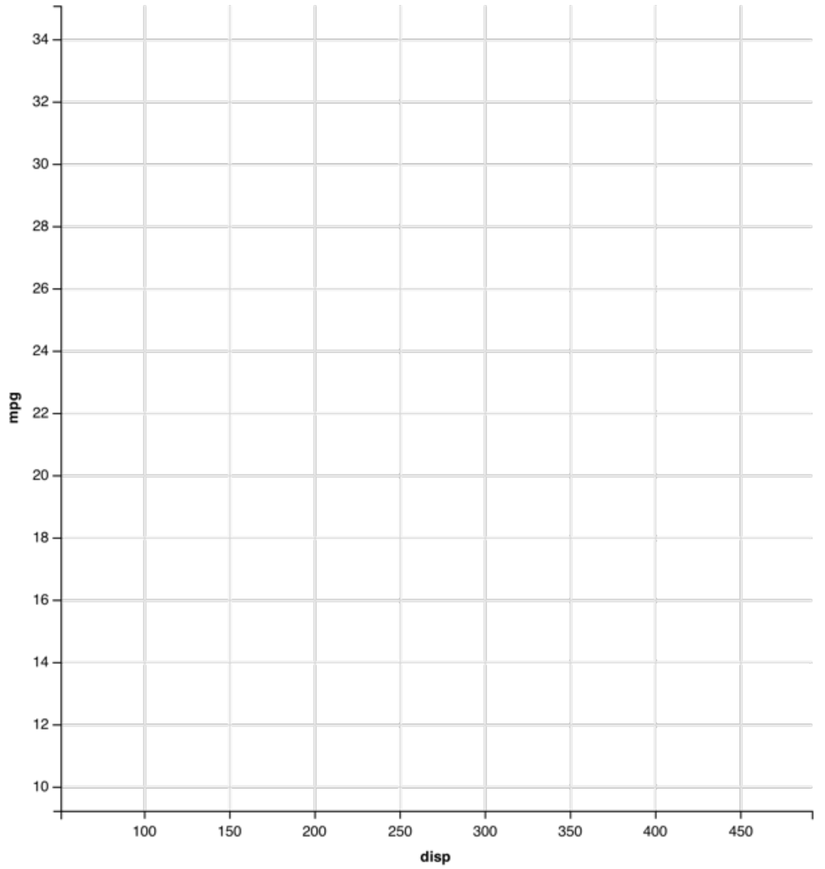

```
ggplot(mpg, aes(displ, hwy)) + geom_point()
```




```
ggplot(mpg, aes(displ, hwy)) + geom_point() +
  geom_point(data = mpg[1:50,], color = "green")
```

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

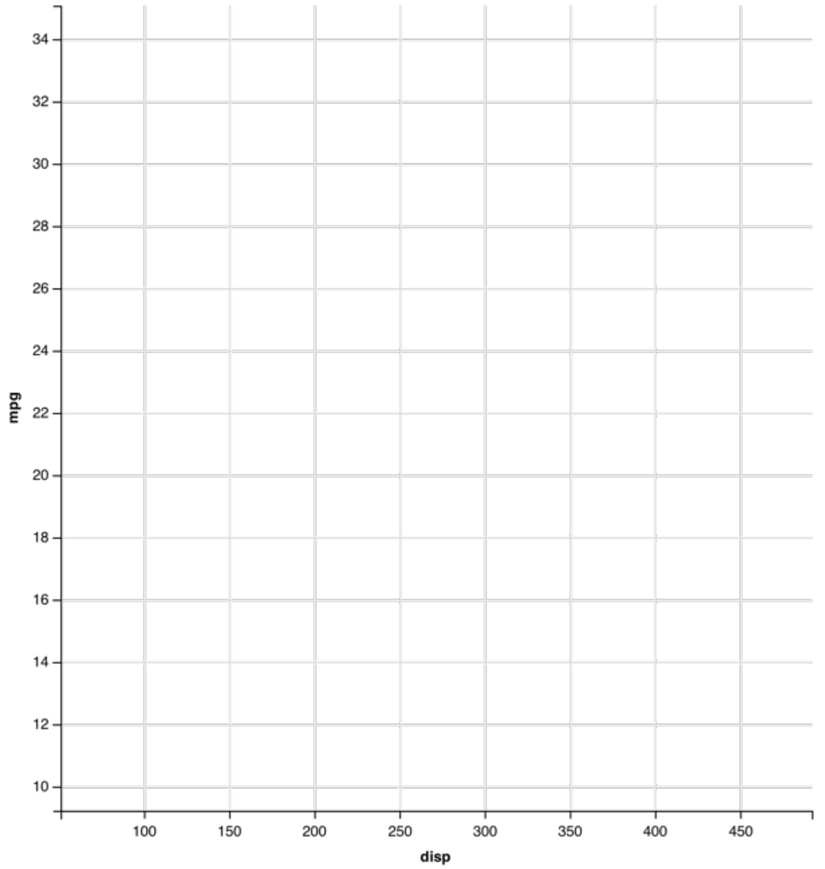


data **geom** **coordinate system**

properties

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

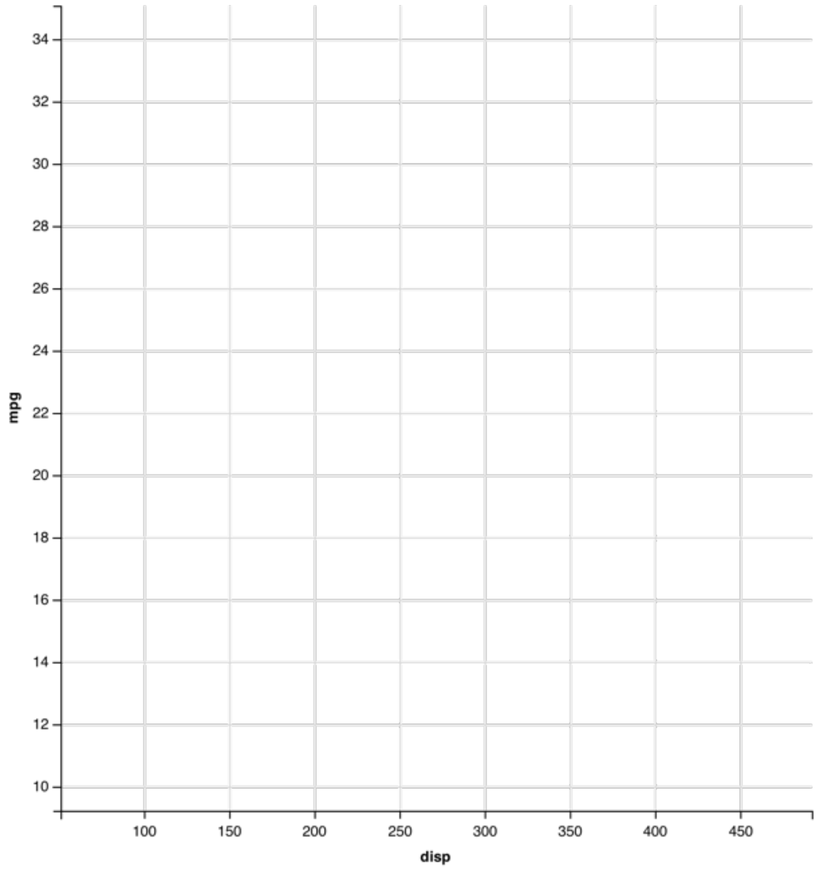


data **geom** **coordinate system**

properties

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



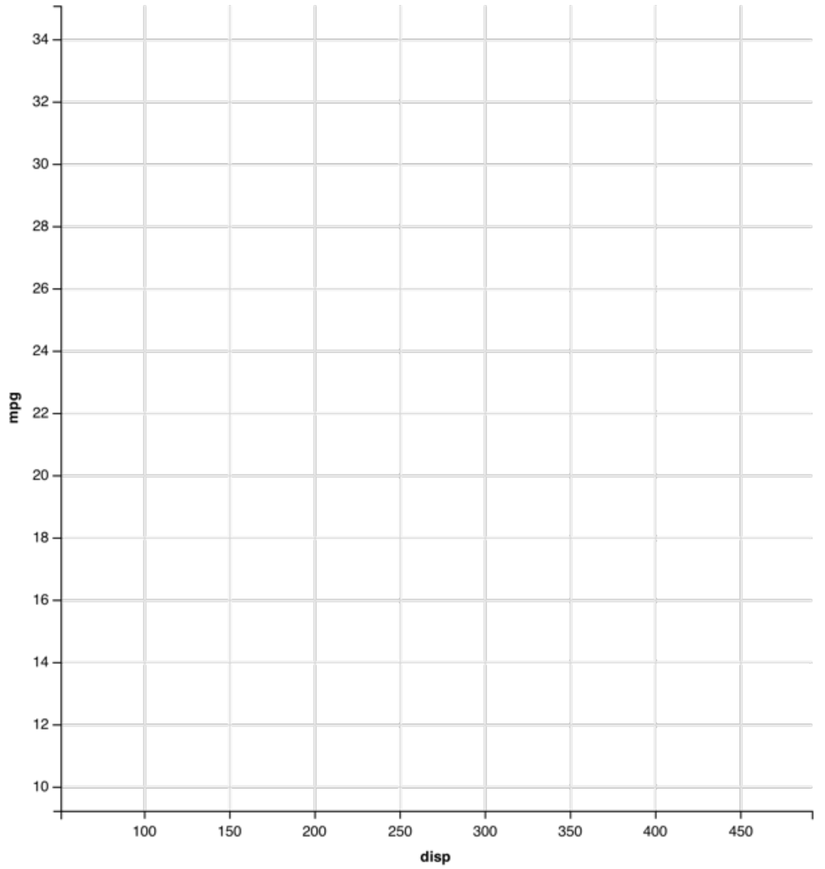
data geom

coordinate system

properties

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



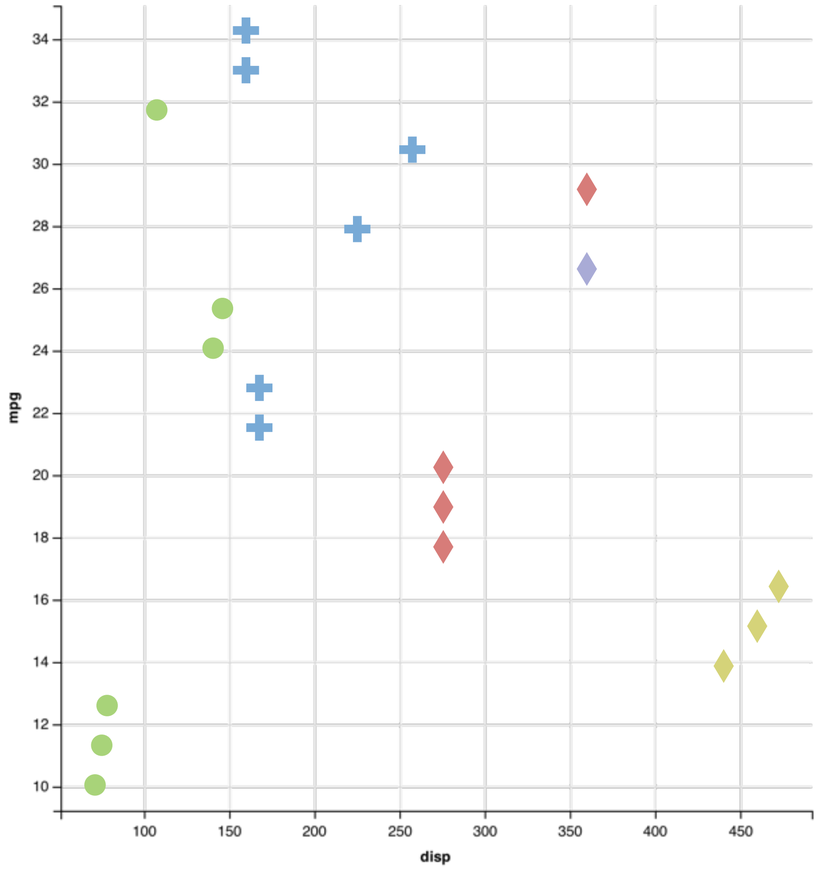
data geom

coordinate system

properties

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

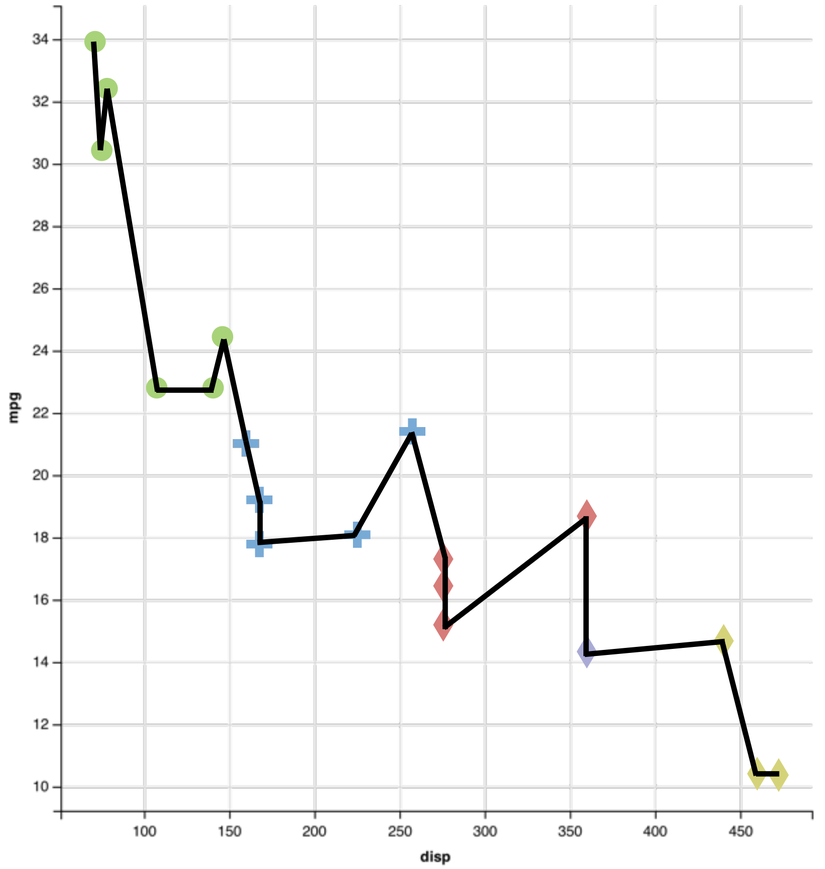


data geom coordinate system

properties

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

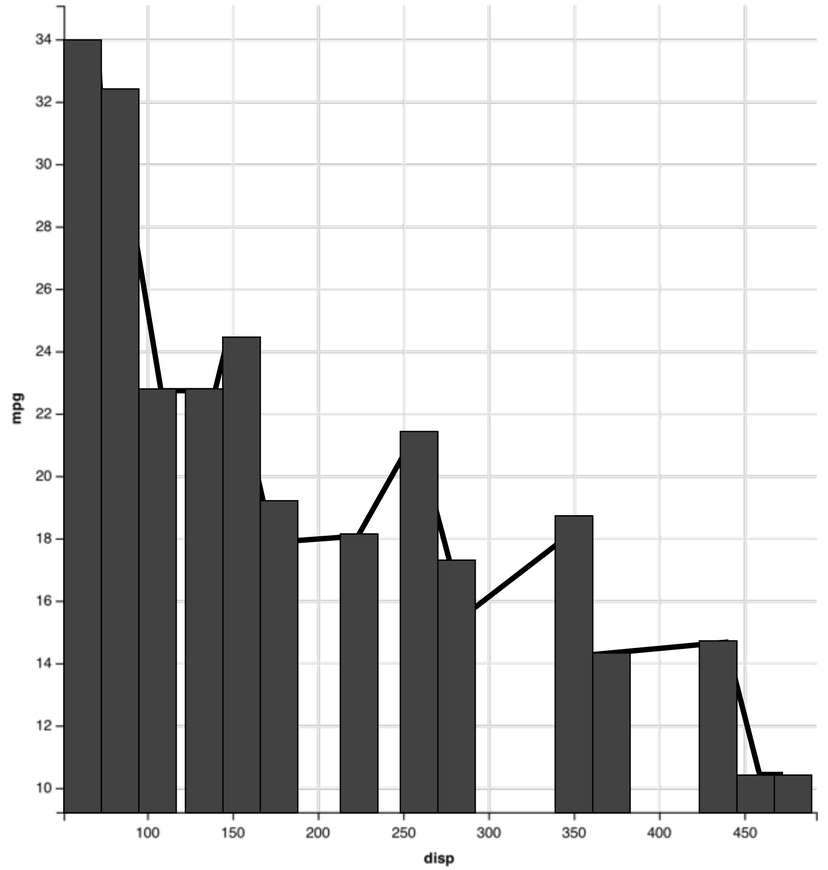
lines

coordinate system

properties

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

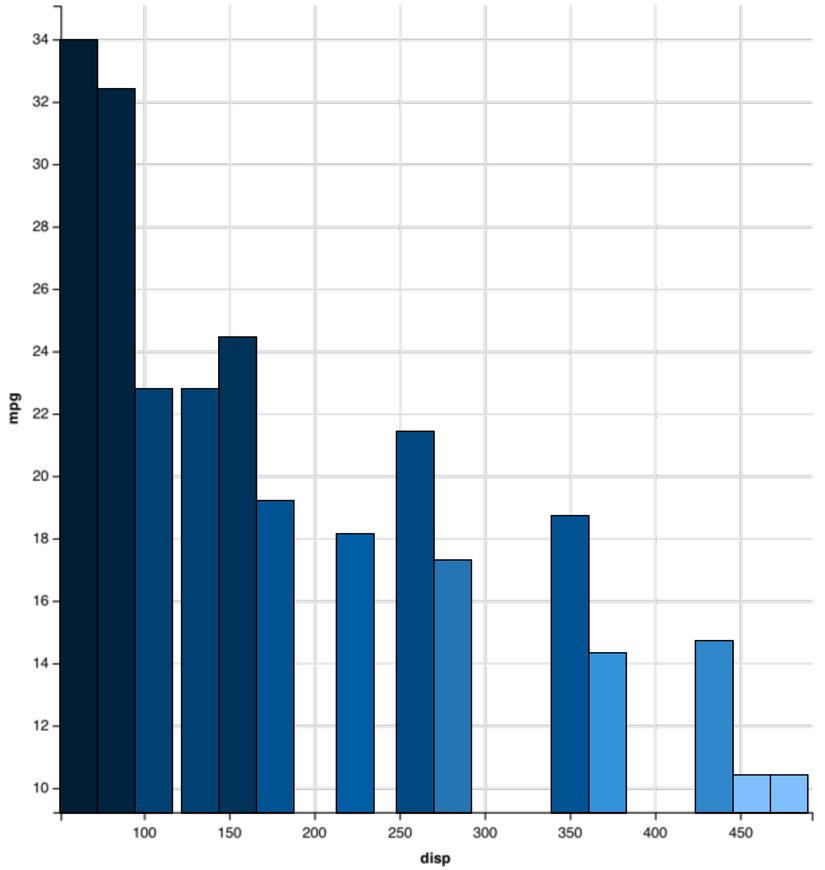
bars

coordinate system

properties

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

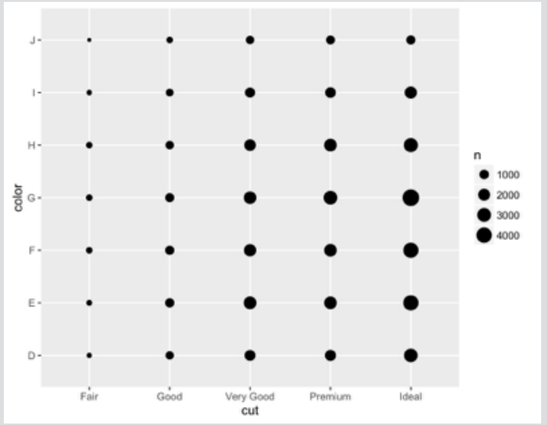
coordinate system

Your turn

Make these plots:

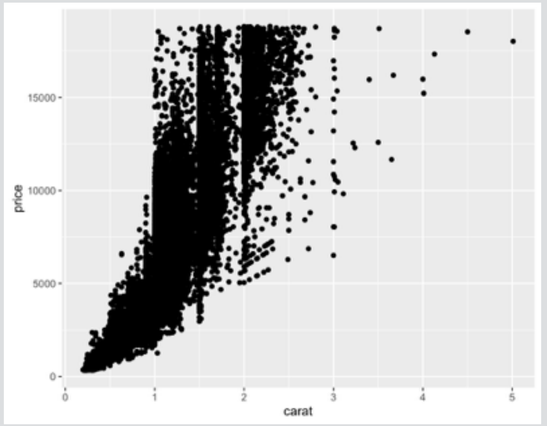
Plot 1

Data = diamonds
geom = count
x = cut
y = color



Plot 2

Data = diamonds
geom = point
x = carat
y = price



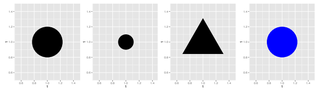
```
ggplot(diamonds, aes(x = cut, y = color)) +
  geom_count()

ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point()
```

Recap: visualization

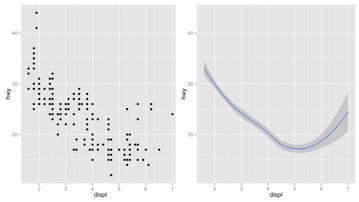
ggplot2: A package that visualizes data.

```
ggplot() +
```

Begin a graph with **ggplot()** then add layers



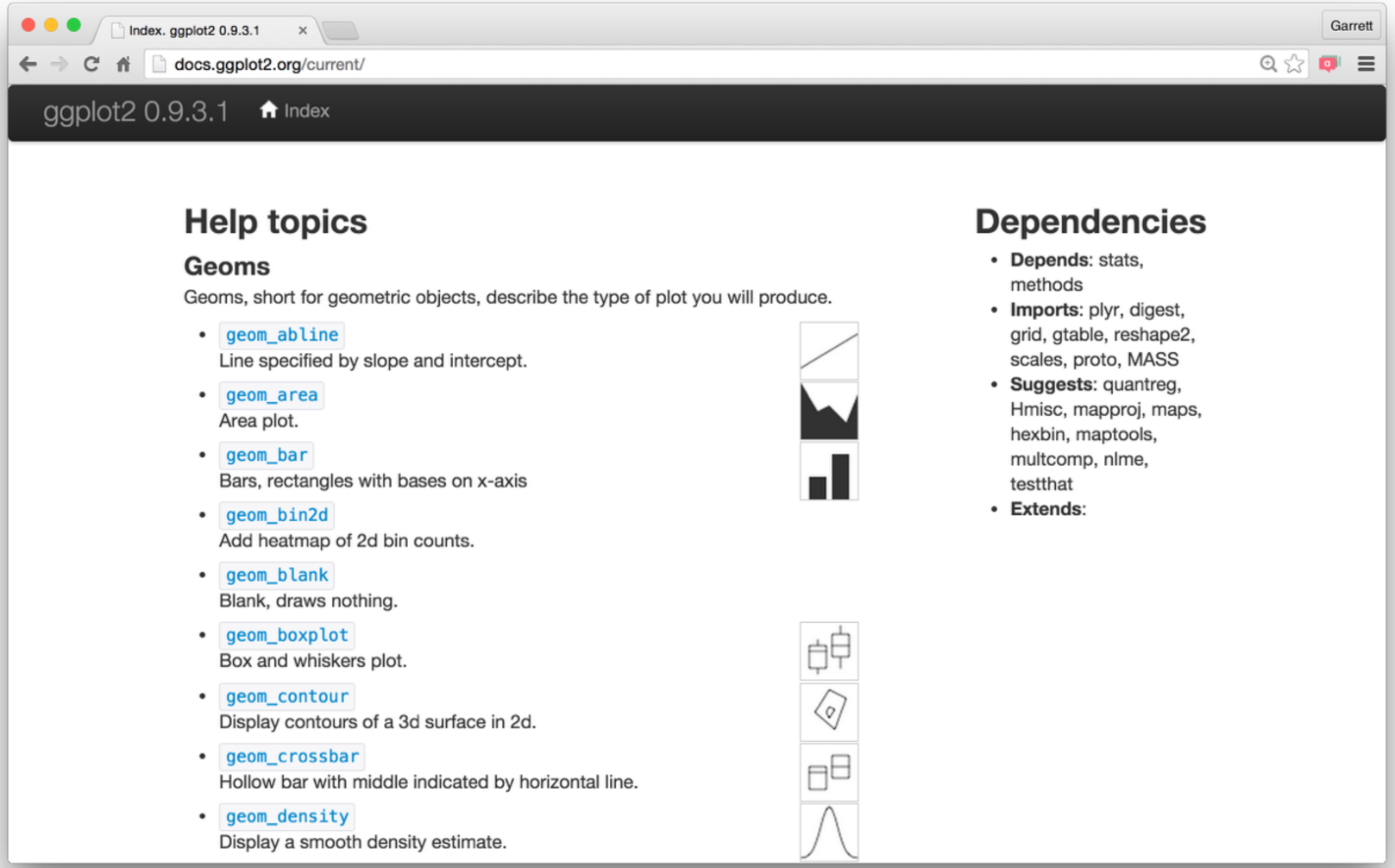
Set aesthetics within **aes()**



Select type of graph with **geom_*()** functions

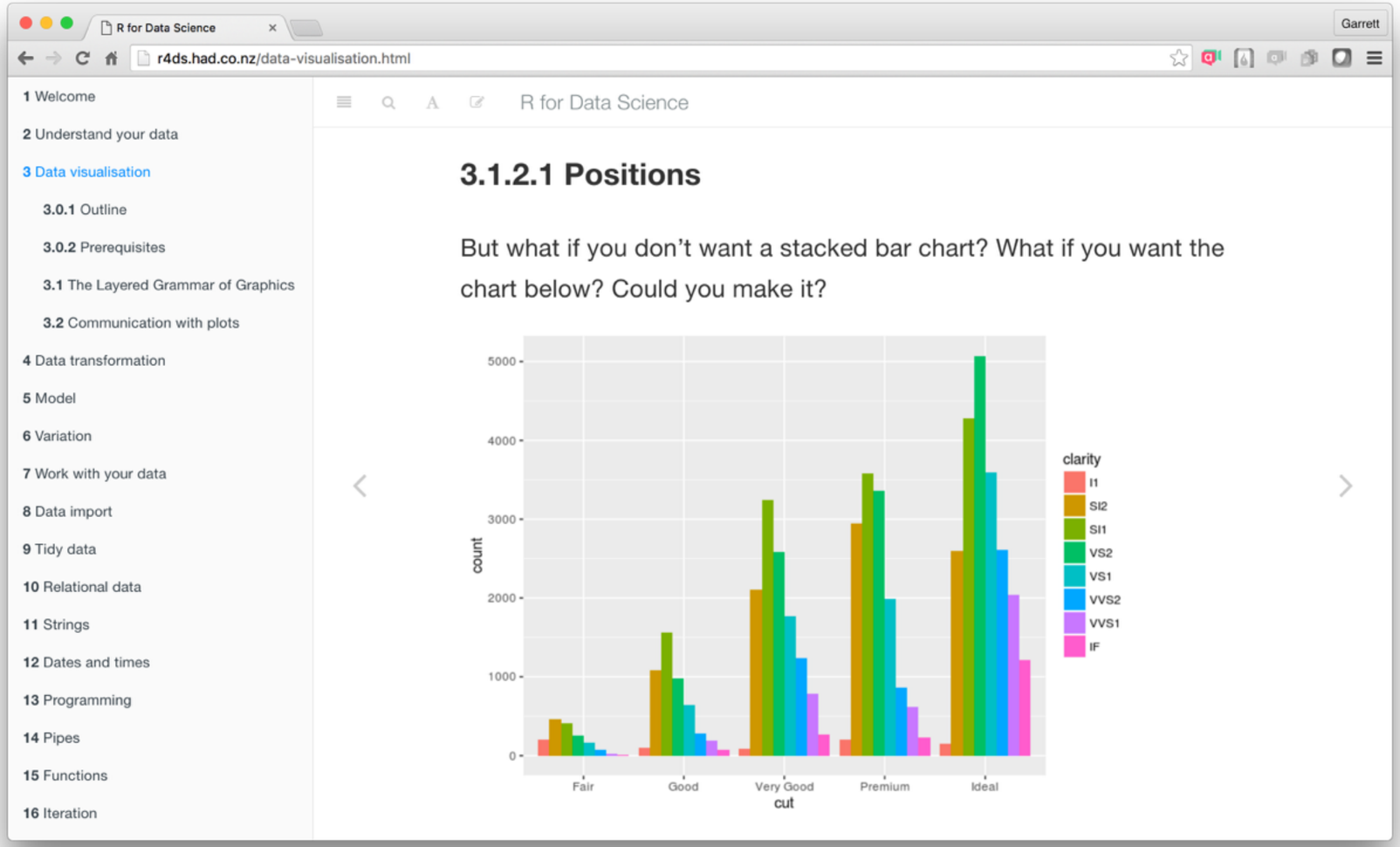
ggplot2 documentation page

docs.ggplot2.org/current/



R for Data Science

r4ds.had.co.nz/data-visualisation.html



Data Wrangling with dplyr

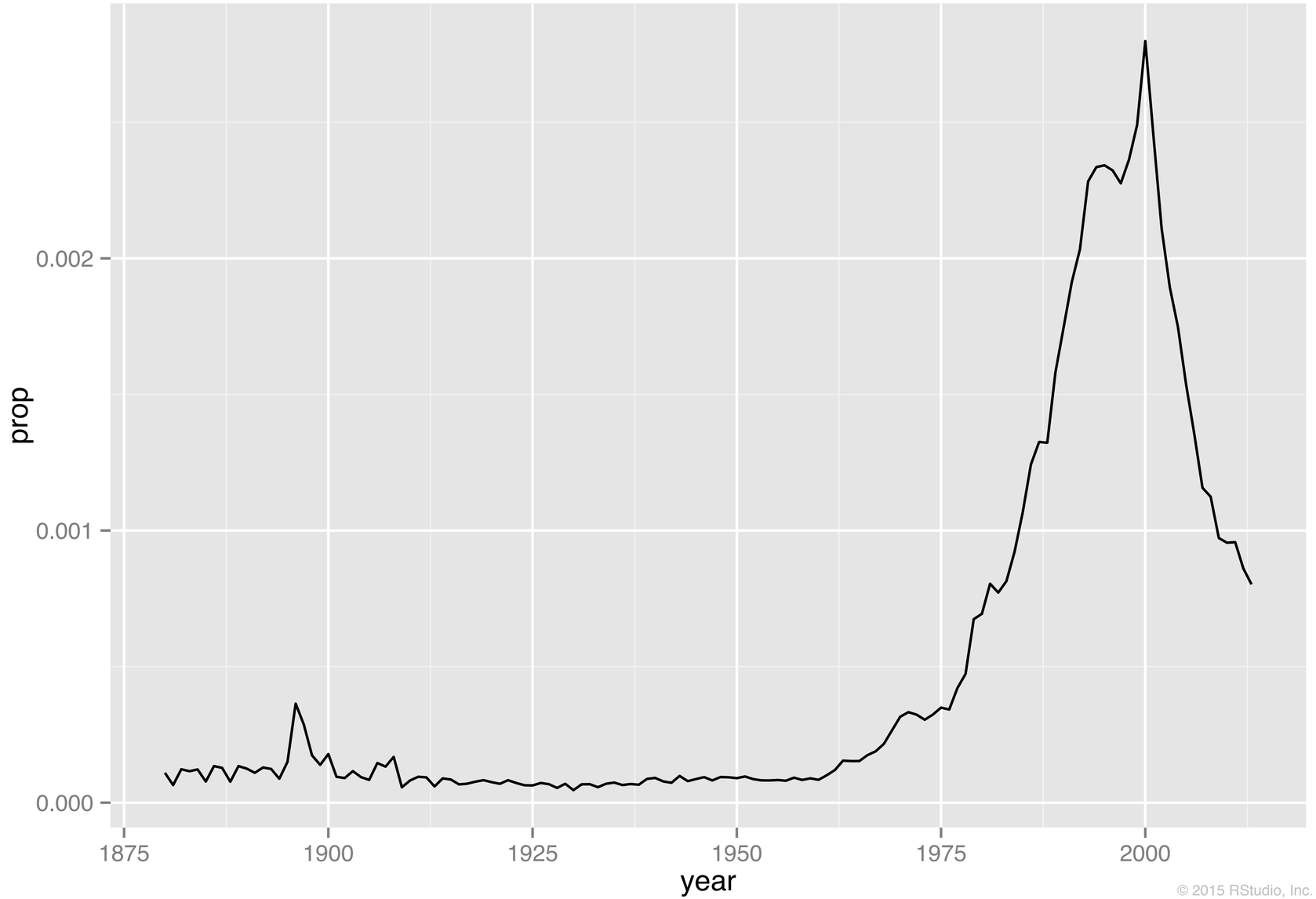
Baby names

- Popularity of baby names in the US from 1880-2013
- Collected by US Social Security Administration

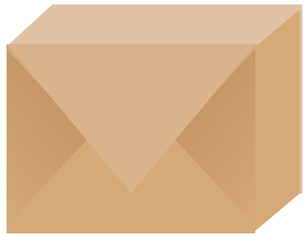
```
library(reportsWS)
View(bnames)
```

```
# devtools::install_github("rstudio/reportsWS")
```

Popularity of "Garrett" over time



dplyr



A package that helps transform tabular data.

```
# install.packages("dplyr")
library(dplyr)
?tbl
?select
?filter
?left_join
?mutate
?summarise
?group_by
?`%>%`
```

Run `library(dplyr)`

```
2477    Kathleen   F 1881 1.315031e-04
2478       Lenna   F 1881 1.315031e-04
2479       Ludie   F 1881 1.315031e-04
2480      Mahala   F 1881 1.315031e-04
2481     Malvina   F 1881 1.315031e-04
2482      Marcia   F 1881 1.315031e-04
2483      Mariah   F 1881 1.315031e-04
```

tbl's

Just like data frames, but easier to work with.

```
2486     Paralee   F 1881 1.315031e-04
2487      Serena   F 1881 1.315031e-04
2488        Sina   F 1881 1.315031e-04
2489     Tressie   F 1881 1.315031e-04
2490      Vernie   F 1881 1.315031e-04
2491     Camille   F 1881 1.213875e-04
2492      Connie   F 1881 1.213875e-04
2493        Dell   F 1881 1.213875e-04
2494        Faye   F 1881 1.213875e-04
2495    Magnolia   F 1881 1.213875e-04
2496       Minta   F 1881 1.213875e-04
2497     Natalie   F 1881 1.213875e-04
2498       Patsy   F 1881 1.213875e-04
2499    Permelia   F 1881 1.213875e-04
2500     Rosella   F 1881 1.213875e-04
 [ reached getOption("max.print") --
omitted 1789591 rows ]
```

bnames

```
Source: local data frame [1,792,091 x 4]

        name   sex  year       prop
       (chr) (chr) (dbl)      (dbl)
1       Mary     F  1880 0.07238359
2       Anna     F  1880 0.02667896
3       Emma     F  1880 0.02052149
4  Elizabeth     F  1880 0.01986579
5     Minnie     F  1880 0.01788843
6   Margaret     F  1880 0.01616720
7        Ida     F  1880 0.01508119
8      Alice     F  1880 0.01448696
9     Bertha     F  1880 0.01352390
10     Sarah     F  1880 0.01319605
..       ...   ...   ...        ...
```

tbl_df(bnames)

```
library(dplyr)

bnames <- tbl_df(bnames)
# Can undo with
# bnames <- as.data.frame(bnames)
```

arrange()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Ana	40	1013	1997-07-01
Alex	45	1009	1998-07-30
Arthur	45	1010	1996-06-21
Arlene	50	1010	1999-06-13
Allison	65	1005	1995-06-04
Alberto	110	1007	2000-08-12

```
arrange(storms, wind)
```

arrange()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Ana	40	1013	1997-07-01
Alex	45	1009	1998-07-30
Arthur	45	1010	1996-06-21
Arlene	50	1010	1999-06-13
Allison	65	1005	1995-06-04
Alberto	110	1007	2000-08-12

```
arrange(storms, wind)
```

arrange()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

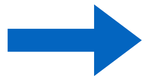
storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21
Alex	45	1009	1998-07-30
Ana	40	1013	1997-07-01

```
arrange(storms, desc(wind))
```

select()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21



storm	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

```
select(storms, storm, pressure)
```

filter()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Arlene	50	1010	1999-06-13

```
filter(storms, wind == 50)
```

logical tests in R

?Comparison

<	Less than
>	Greater than
==	Equal to
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to
%in%	Group membership
is.na	Is NA
!is.na	Is not NA

?base::Logic

&	boolean and
|	boolean or
xor	exactly or
!	not
any	any true
all	all true

filter()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13

```
filter(storms, wind >= 50)
```

filter()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01

```
filter(storms, wind > 60, wind == 40)
```

Your Turn

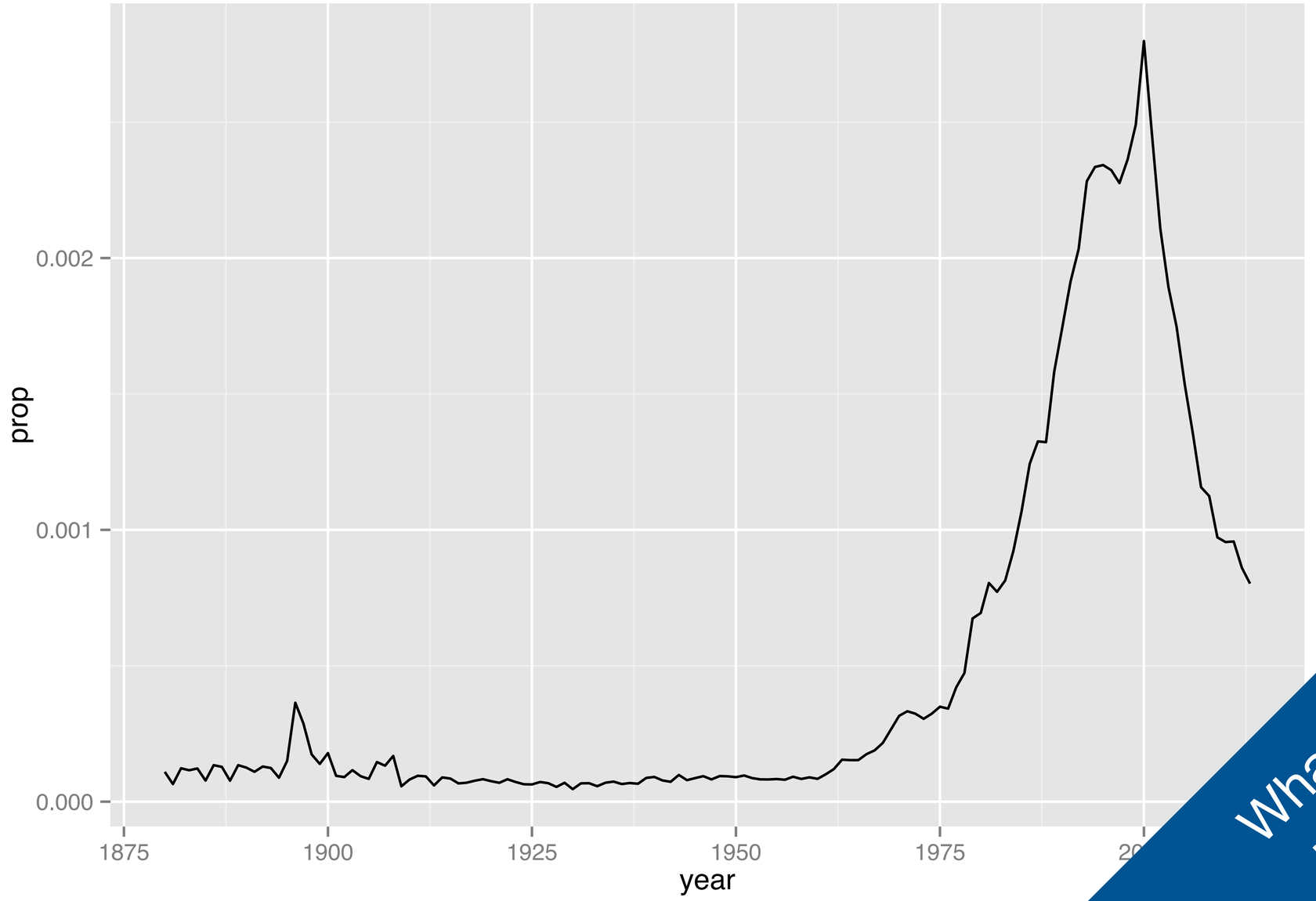
Create a data set that contains only rows with **your name** and **sex**, and only the columns **name**, **year**, and **prop**.

Then plot the data with

```
ggplot(<data name here>) +
  geom_line(aes(x = year, y = prop))
```

```
library(dplyr)

my_name <- filter(bnames, name == "Garrett", sex == "M")
my_name <- select(my_name, name, year, prop)

ggplot(my_name) +
  geom_line(aes(x = year, y = prop))
```

Popularity of "Garrett" over time



What is another way to measure popularity?

Births

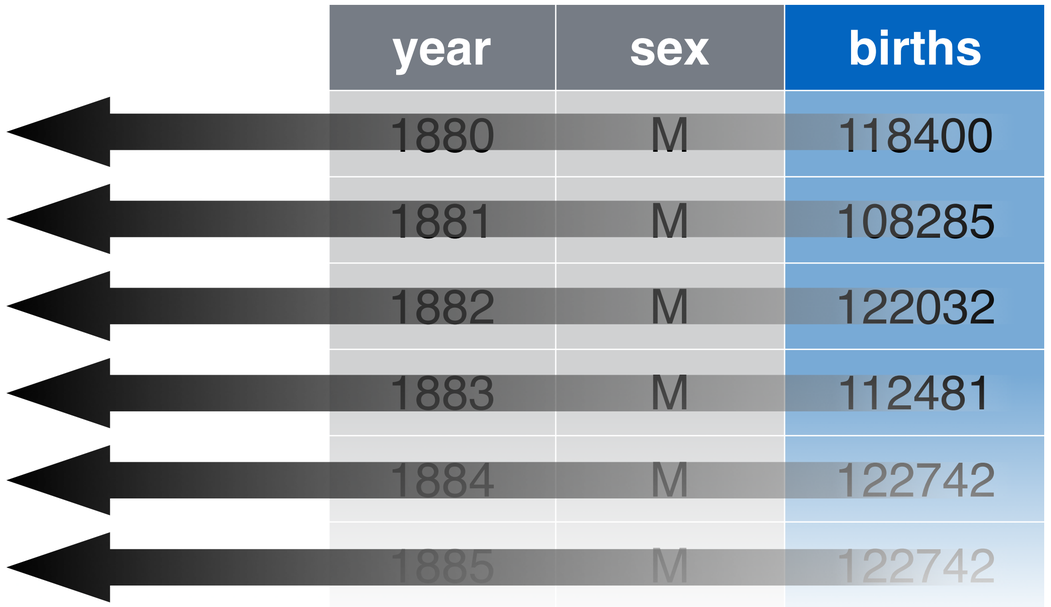
Number of children born in the US from 1880-2013

```
View(births)
```

Strategy

1. Join the births data to the my_name data set.

name	year	prop	births
Garrett	1880	1.097973E-04	118400
Garrett	1881	6.464423E-05	108285
Garrett	1882	1.229186E-04	122032
Garrett	1883	1.155751E-04	112481
Garrett	1884	1.222076E-04	122742
Garrett	1885	7.762100E-05	122742



Strategy

1. Join the births data to the my_name data set.
2. Use the births and prop variables to calculate the number of children, `n = round(prop * births)`.

name	year	prop	births	n
Garrett	1880	1.097973E-04	118400	13
Garrett	1881	6.464423E-05	108285	7
Garrett	1882	1.229186E-04	122032	15
Garrett	1883	1.155751E-04	112481	13
Garrett	1884	1.222076E-04	122742	15
Garrett	1885	7.762100E-05	122742	10

Strategy

1. Join the births data to the my_name data set.
2. Use the births and prop variables to calculate the number of children, `n = round(prop * births)`.

name	year	prop	births	n
Garrett	1880	1.097973E-04	118400	13
Garrett	1881	6.464423E-05	108285	7
Garrett	1882	1.229186E-04	122032	15
Garrett	1883	1.155751E-04	112481	13
Garrett	1884	1.222076E-04	122742	15
Garrett	1885	7.762100E-05	122212	10

left_join()

songs

song	name
Across the Universe	John
Come Together	John
Hello, Goodbye	Paul
Peggy Sue	Buddy

+

artists

name	plays
George	sitar
John	guitar
Paul	bass
Ringo	drums

=

song	name	plays
Across the Universe	John	guitar
Come Together	John	guitar
Hello, Goodbye	Paul	bass
Peggy Sue	Buddy	**<NA>**

```
left_join(songs, artists, by = "name")
```

mutate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date	ratio
Alberto	110	1007	2000-08-12	9.15
Alex	45	1009	1998-07-30	22.42
Allison	65	1005	1995-06-04	15.46
Ana	40	1013	1997-07-01	25.32
Arlene	50	1010	1999-06-13	20.20
Arthur	45	1010	1996-06-21	22.44

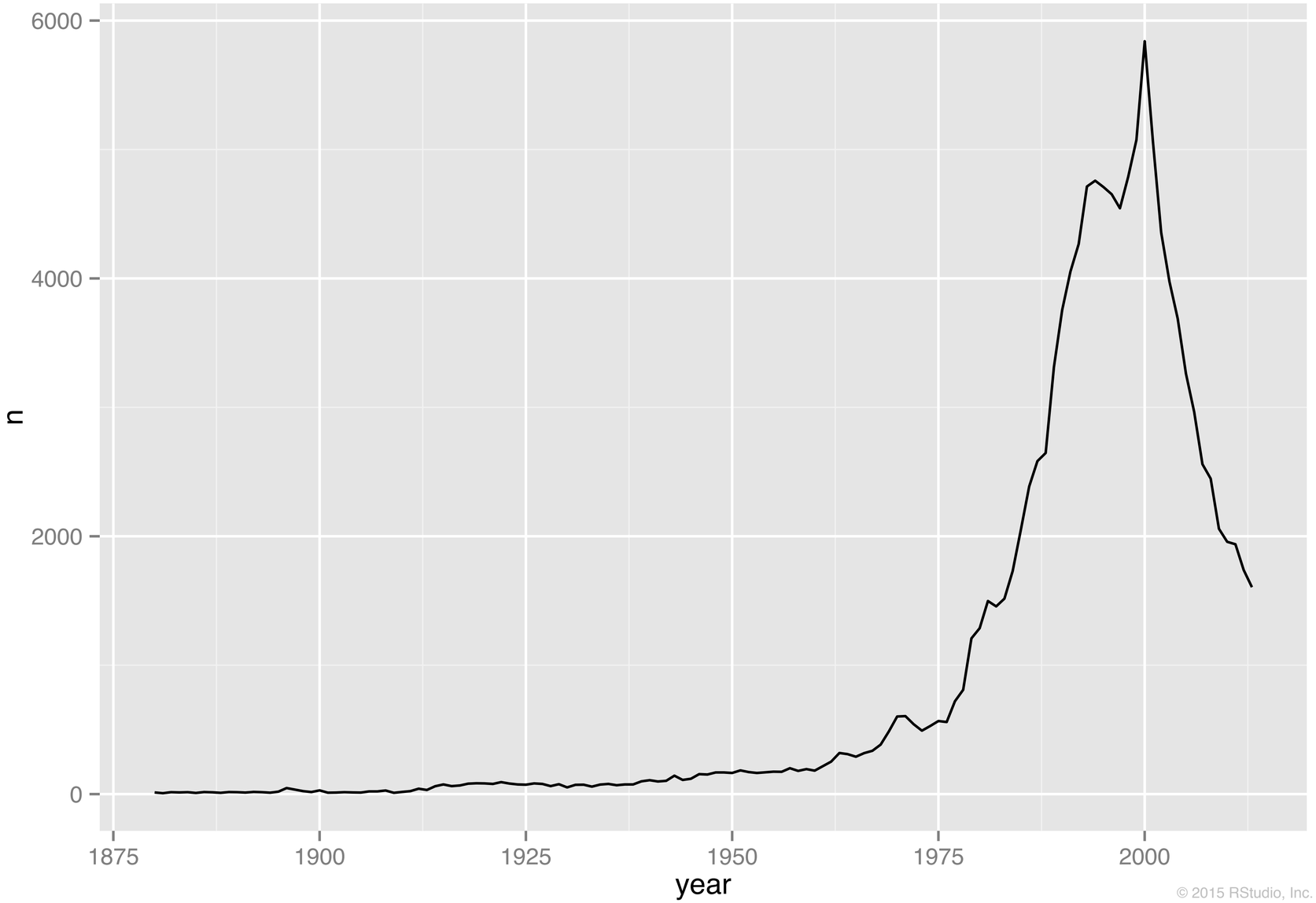
```
mutate(storms, ratio = pressure / wind)
```

Your Turn

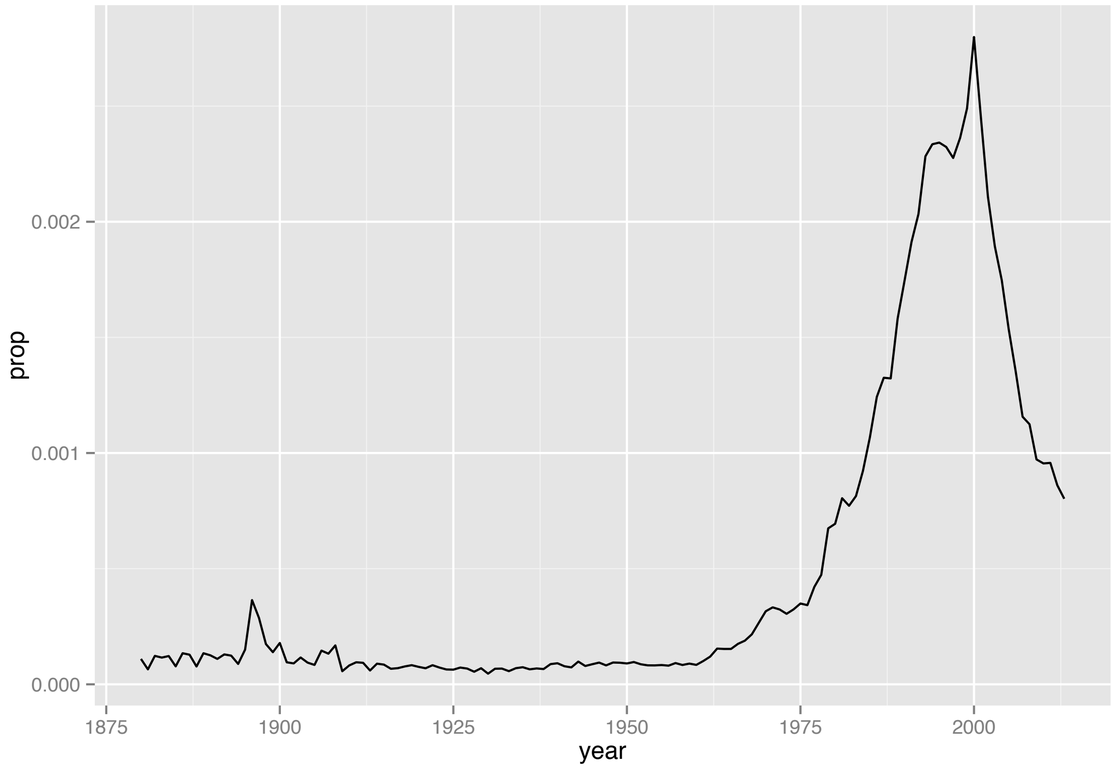
1. `filter()` **births** to just rows with **your sex**.
2. Join the result to `my_name` by **year**.
3. Add a new variable to the data: `n = round(prop * births)`
4. Save the new data. Then plot **n** over time.

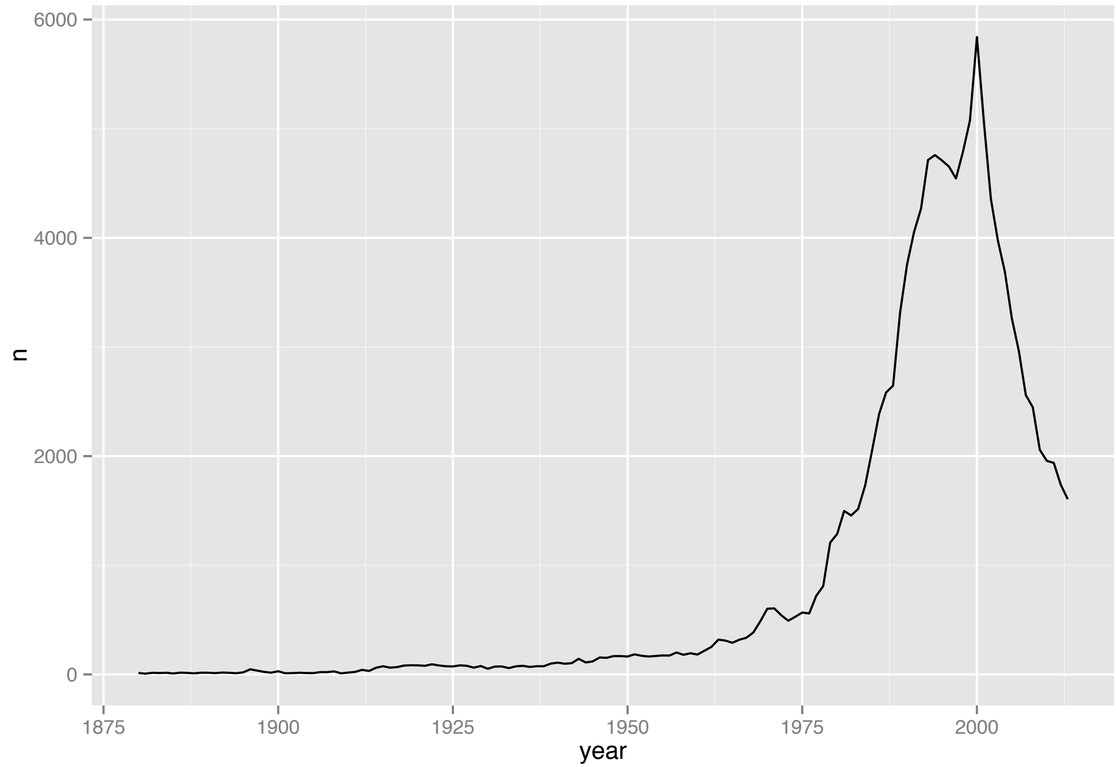
```
boys <- filter(births, sex == "M")

my_name <- left_join(my_name, boys, by = "year")

my_name <- mutate(my_name, n = round(prop * births))

ggplot(my_name) +
  geom_line(aes(x = year, y = n))
```

Popularity of "Garrett" over time



Popularity of "Garrett" over time





summarise()

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

median
22.5

```
summarise(pollution, median = median(amount))
```

summarise()

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

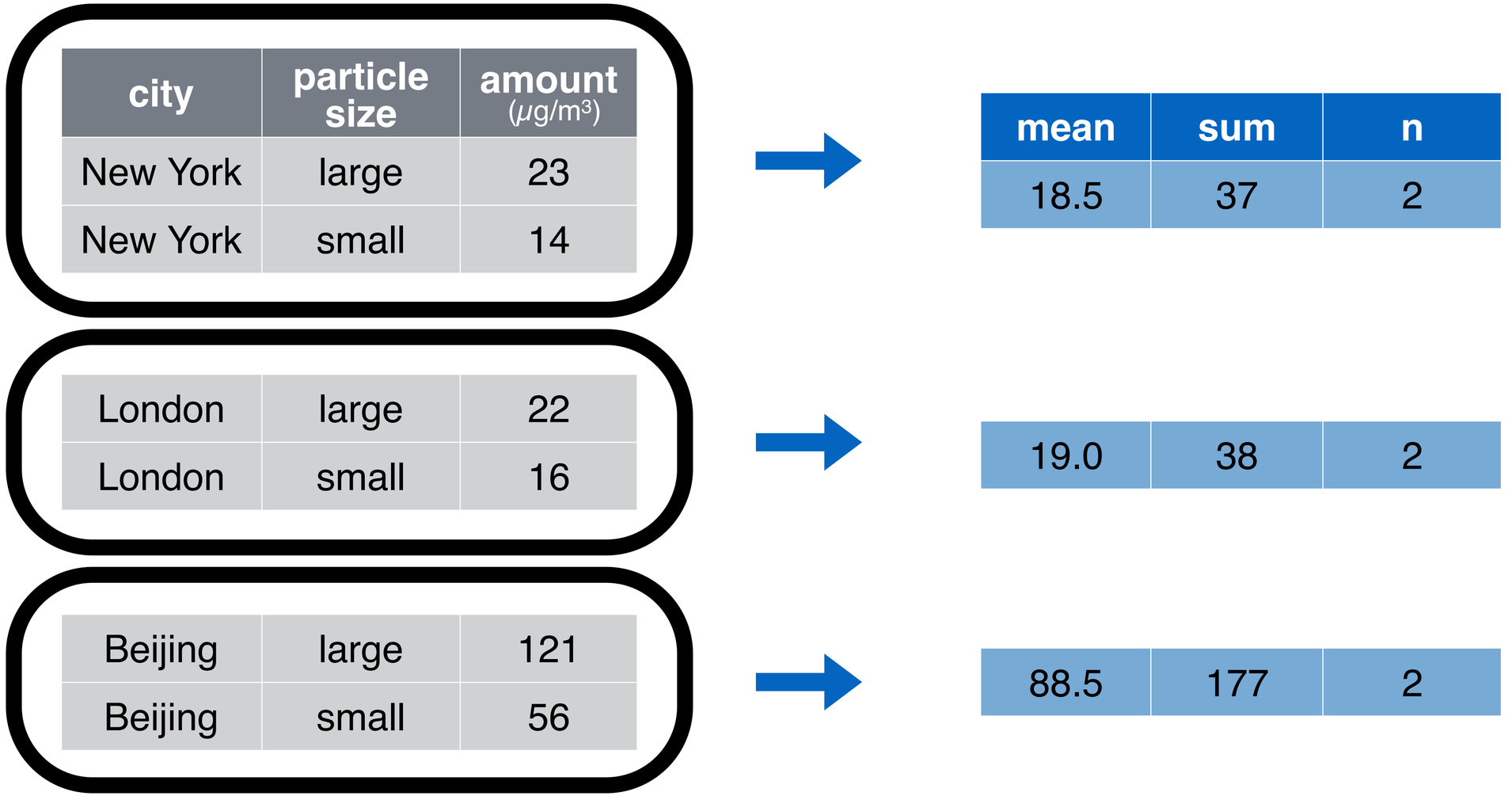
```
summarise(pollution, mean = mean(amount), sum = sum(amount), n = n())
```

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6



group_by() + summarise()

group_by()

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14

city	particle size	amount ($\mu g/m^3$)
London	large	22
London	small	16

city	particle size	amount ($\mu g/m^3$)
Beijing	large	121
Beijing	small	56

mean	sum	n
18.5	37	2
19.0	38	2
88.5	177	2

```
p <- group_by(pollution, city)
summarise(p, mean = mean(amount), sum = sum(amount), n = n())
```

%>%

Ceci n'est pas une pipe.

```
p <- group_by(pollution, city)
summarise(p, mean = mean(amount), sum=sum(amount), n=n())
```

```
my_name <- filter(bnames, name == "Garrett", sex == "M")
my_name <- select(my_name, name, year, prop)
```

```
my_name <- left_join(my_name, boys, by = "year")
my_name <- mutate(my_name, n = round(prop * births))
```

The pipe operator %>%

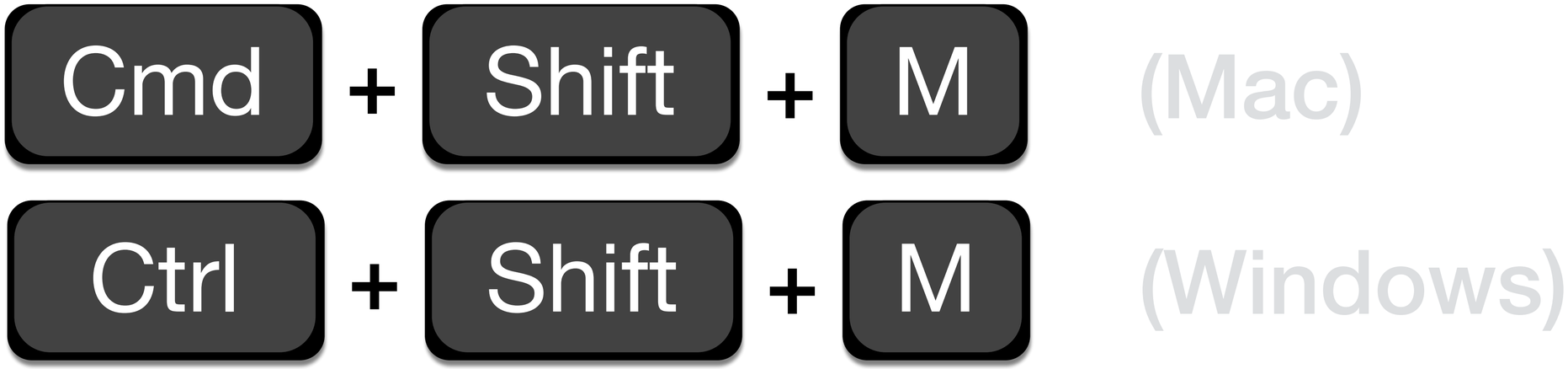
These do the same thing

Try it!

```
summarize(pollution, median = median(amount))
pollution %>% summarize(median = median(amount))
```


```
pollution      summarize( ______, median = median(amount))
```

Shortcut to type %>%



Your Turn

Work with a neighbor to determine what each line of the code below does.

```
bnames %>%
  left_join(births, by = c("year", "sex")) %>%
  mutate(n = round(prop * births)) %>%
  select(name, sex, year, n) %>%
  filter(!is.na(n)) %>%
  group_by(name, sex) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))
```

```
# Take bnames and
bnames %>%
  # join to it births by year and sex.
  left_join(births, by = c("year", "sex")) %>%
  # Then use the result to calculate a new variable, n
  mutate(n = round(prop * births)) %>%
  # Select from that four columns: name, sex, year, and n
  select(name, sex, year, n) %>%
  # Filter out rows where n = NA
  filter(!is.na(n)) %>%
  # Then group by the combination of name and gender
  group_by(name, sex) %>%
  # Calculate the total number of children for each group
  summarise(total = sum(n)) %>%
  # Then order the groups from the largest total to the smallest
  arrange(desc(total))
```

```
bnames %>%
  left_join(births, by = c("year", "sex")) %>%
  mutate(n = round(prop * births)) %>%
  select(name, sex, year, n) %>%
  filter(!is.na(n)) %>%
  group_by(name, sex) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))
```

```
tmp1 <- left_join(bnames, births, by = c("year", "sex"))
tmp2 <- mutate(tmp1, n = round(prop * births))
tmp3 <- select(tmp2, name, sex, year, n)
tmp4 <- filter(tmp3, !is.na(n))
tmp5 <- group_by(tmp4, name, sex)
tmp6 <- summarise(tmp5, total = sum(n))
tmp7 <- arrange(tmp6, desc(total))
```

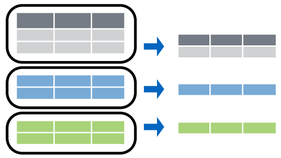
```
arrange(
  summarise(
    group_by(
      filter(
        select(
          mutate(
            left_join(bnames, births, by = c("year", "sex")),
            n = round(prop * births)
          ), name, sex, year, n
        ), !is.na(n)
      ), name, sex
    ), total = sum(n)
  ), desc(total)
)
```

Recap: dplyr

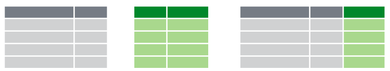
Extract variables and observations with **select()** and **filter()**

Arrange observations with **arrange()**.

Make new variables with **mutate()**.



Make groupwise summaries with **group_by()** and **summarise()**.

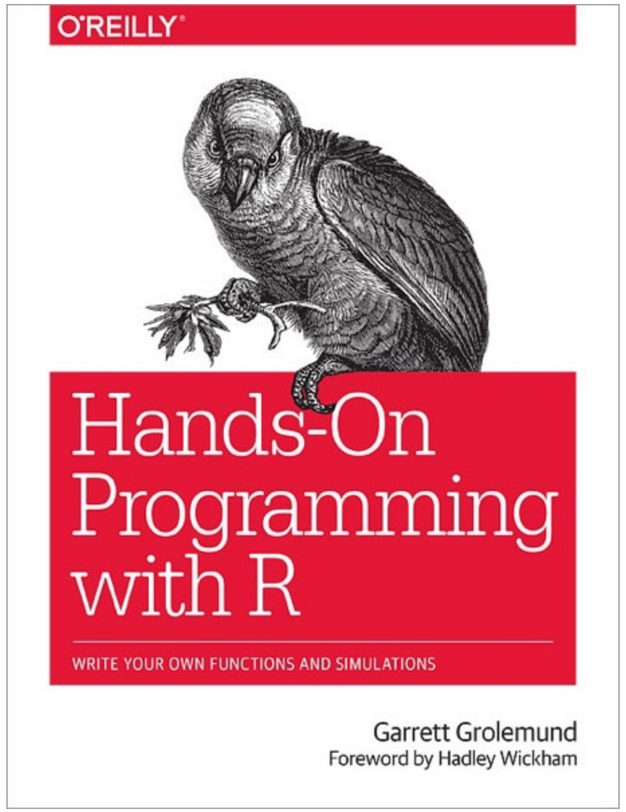


Join data frames with **left_join()**.

Chain operations together with **%>%**

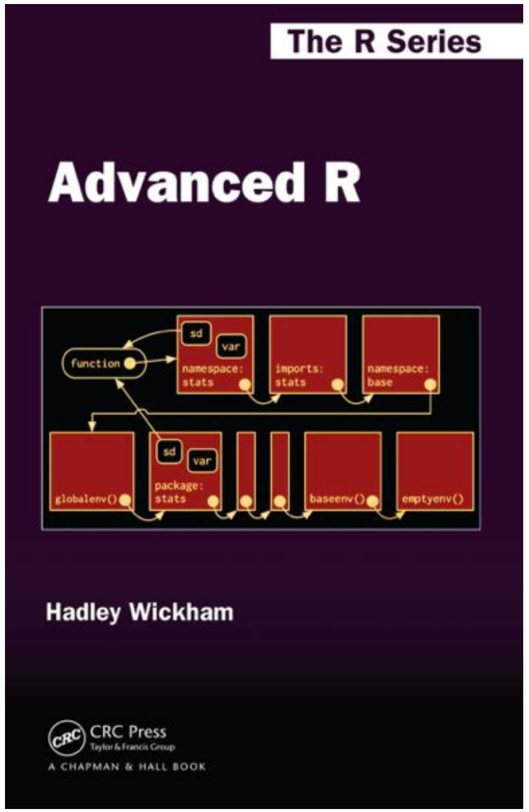
Where next?

Books



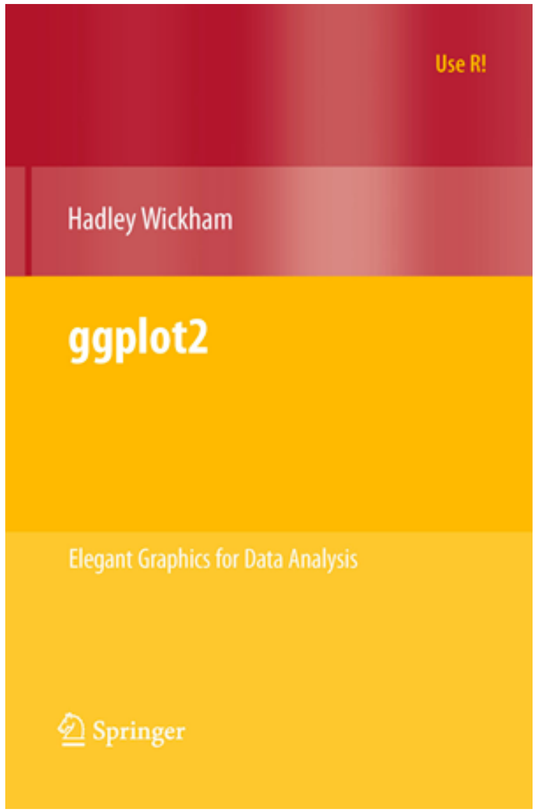
shop.oreilly.com/product/0636920028574.do

Beginner



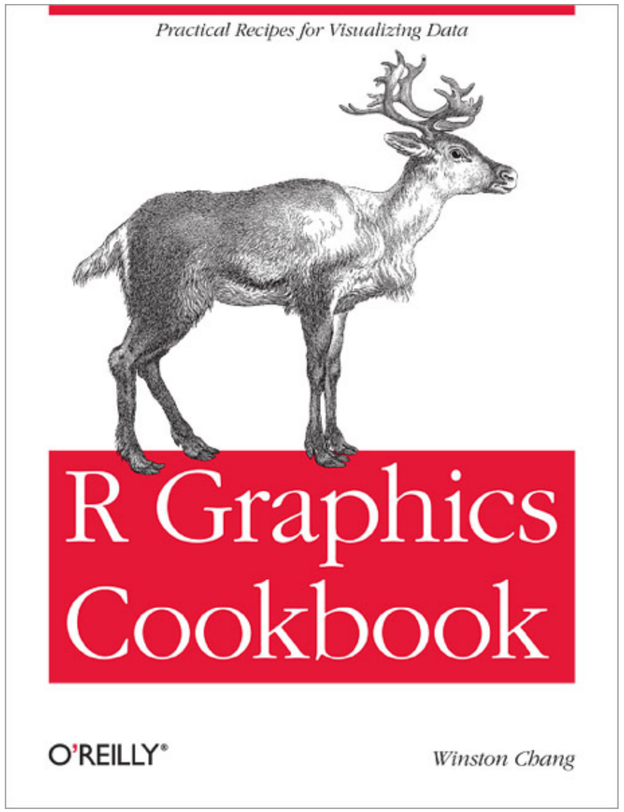
adv-r.had.co.nz/

Advanced



www.amzn.com/0387981403

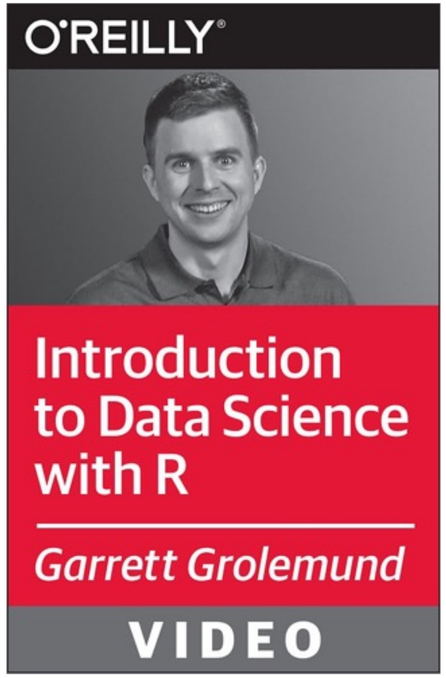
Intro to Visualizations



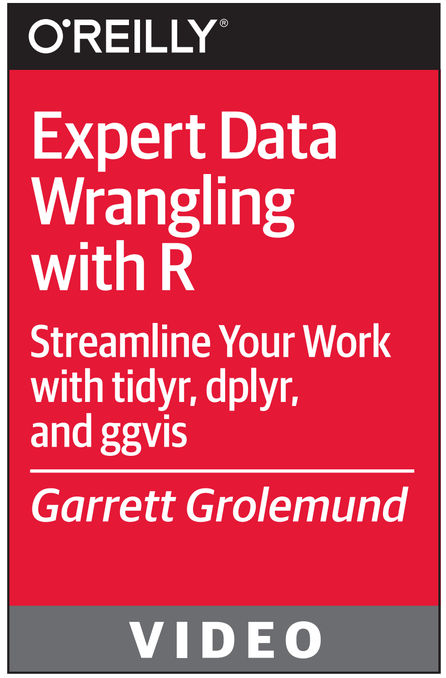
shop.oreilly.com/product/0636920023135.do

Reference for Visualizations

Videos



shop.oreilly.com/product/0636920034834.do



shop.oreilly.com/product/0636920035992.do

Interactive tutorials

DataCamp

www.datacamp.com

More at
www.rstudio.com/resources/training/online-learning/

R for Data Science

[r4ds.had.co.nz/](r4ds.had.co.nz/)