

Strata+ Hadoop WORLD

PRES E NTED BY

O'REILLY®

cloudera®



strataconf.com

#StrataHadoop

R QuickStart

Transform and visualize data



Garrett Grolemund

Data Scientist and Master Instructor

March 2016

Email: garrett@rstudio.com

HELLO

my name is

Garrett



@StatGarrett

Warm up

Introduce yourself to your table

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



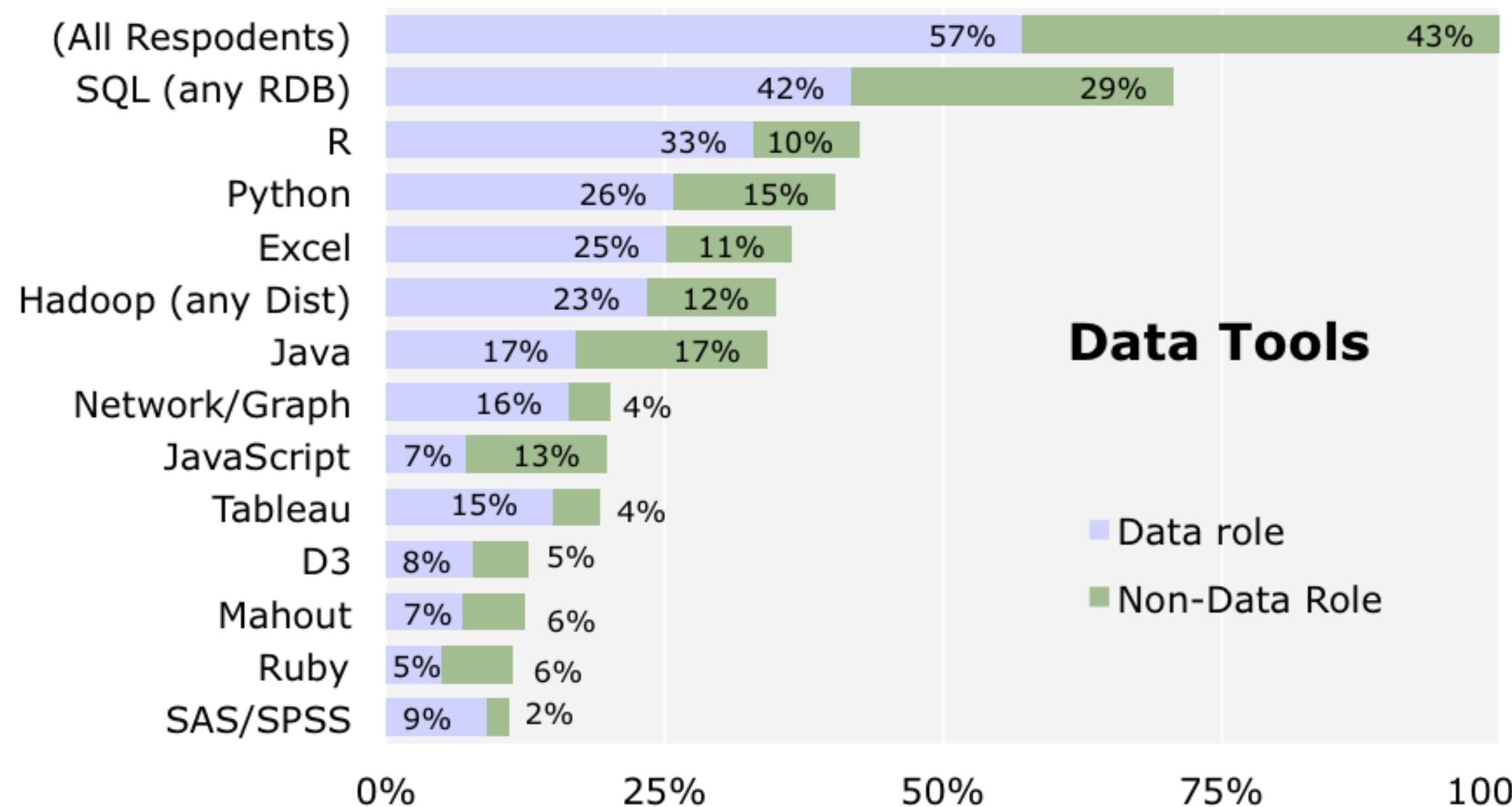
R



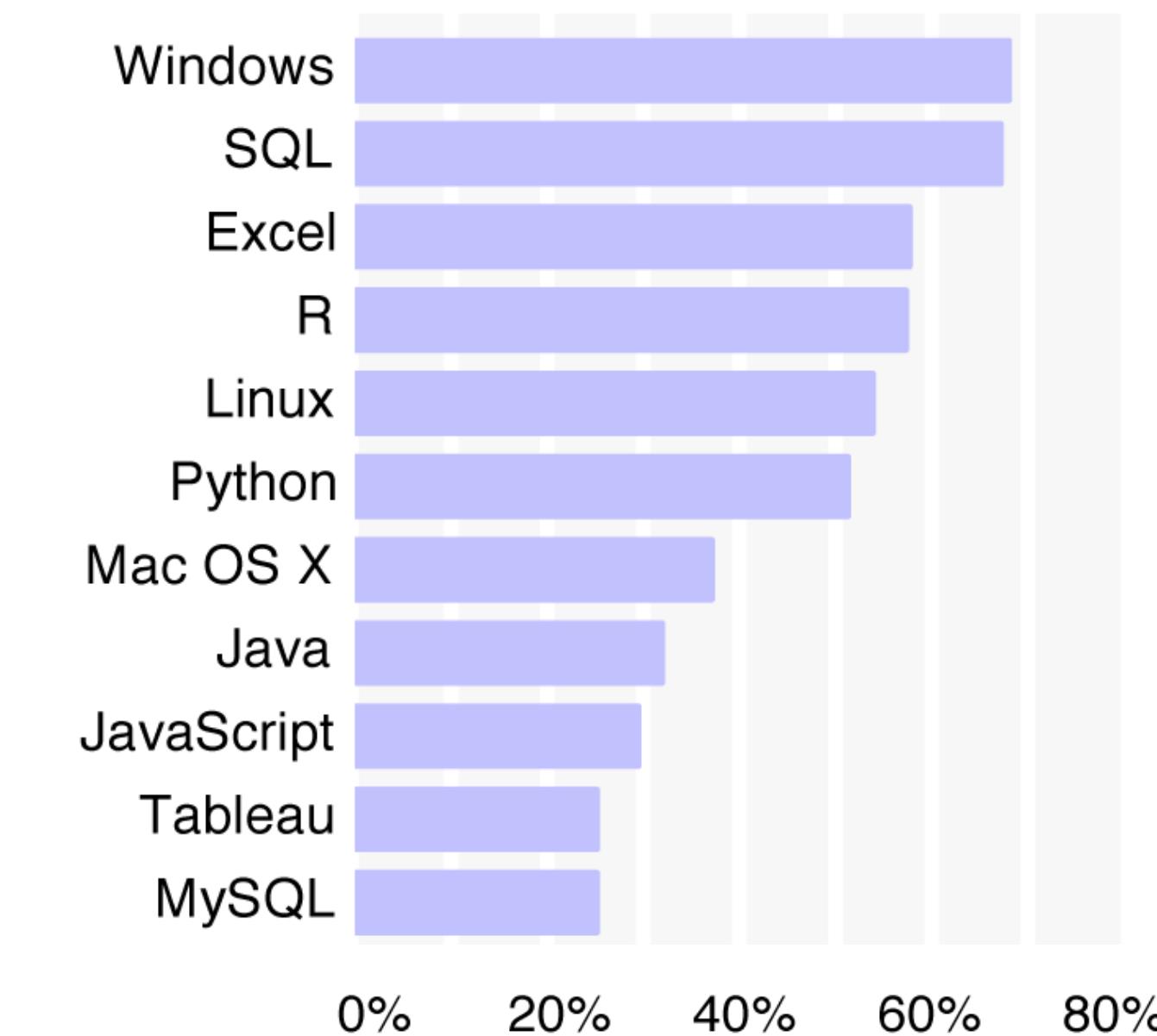
1. A full scripting language
2. Descends from S, Bell Laboratories
3. Evolved in university environment
4. Built-in data science/statistics toolkit

Data Science Salary Survey

2012 & 2013 (combined)



2014



Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo



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

© 2015 RStudio, Inc. All rights reserved.

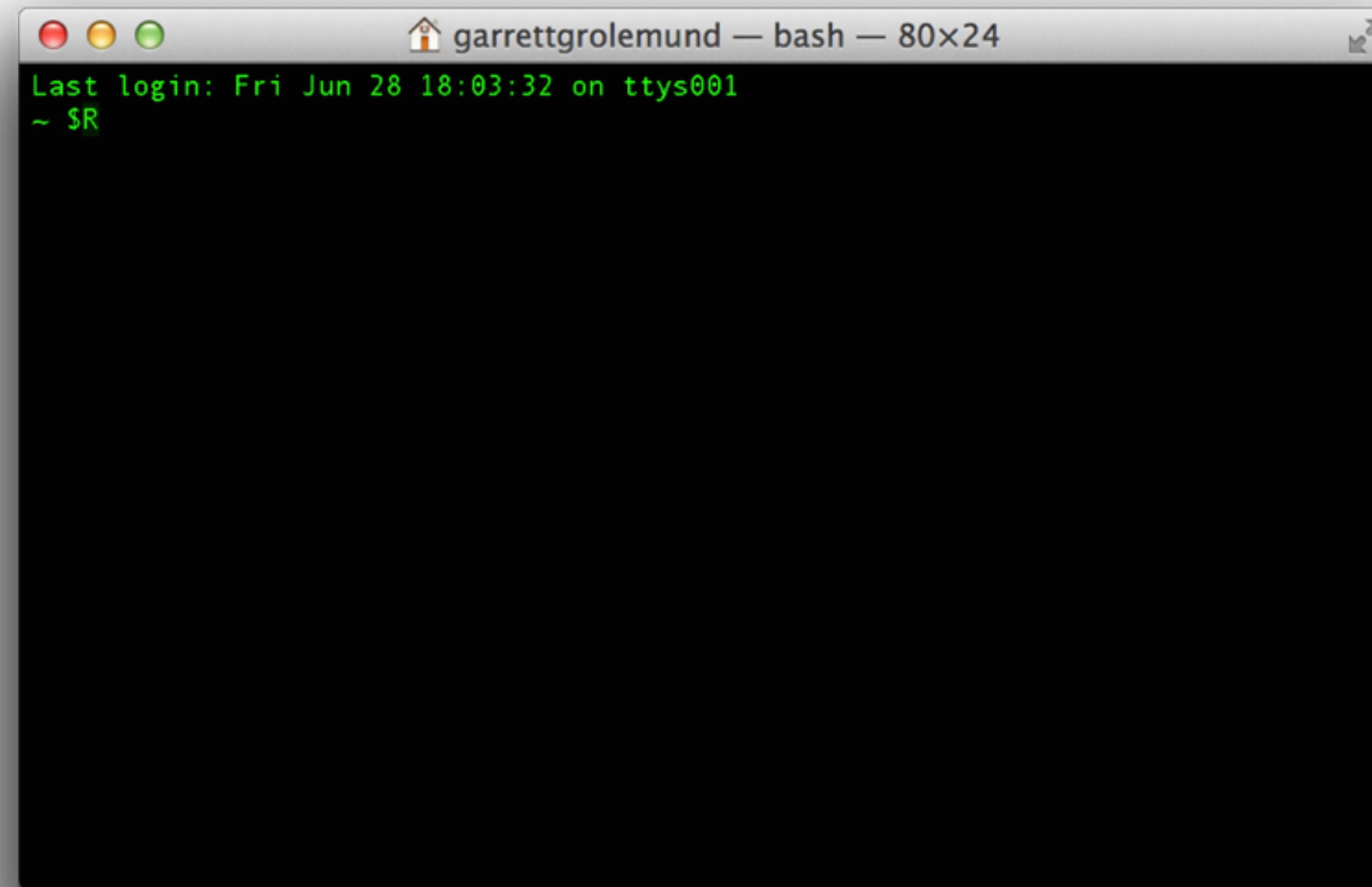


1. A full scripting language
2. Descends from S, Bell Laboratories
3. Evolved in university environment
4. Built-in data science/statistics toolkit

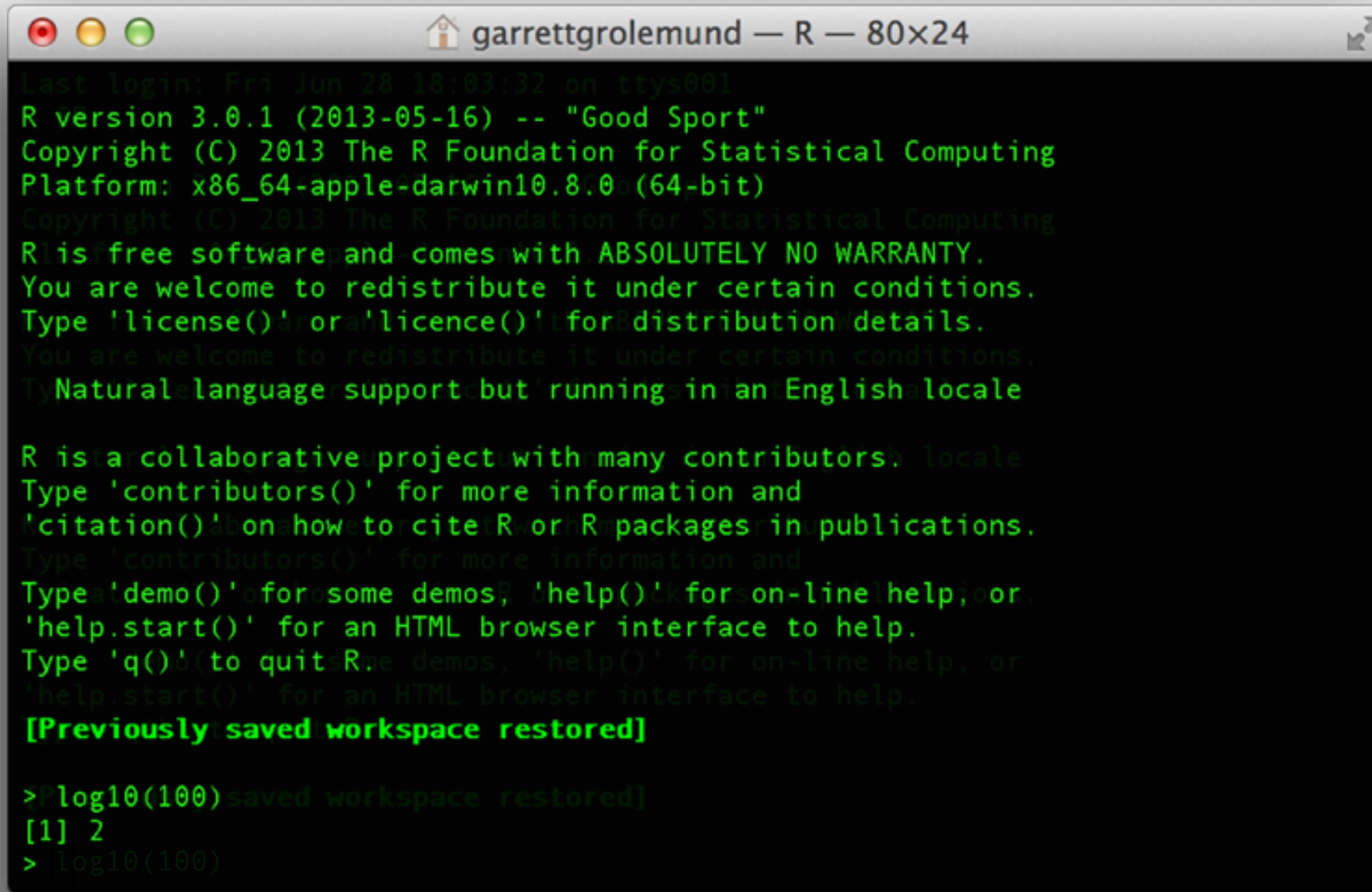


1. An integrated development environment (IDE) for R
2. Software that helps you write, run, debug, and manage code written in R

R: a computer programming language



R: a computer programming language



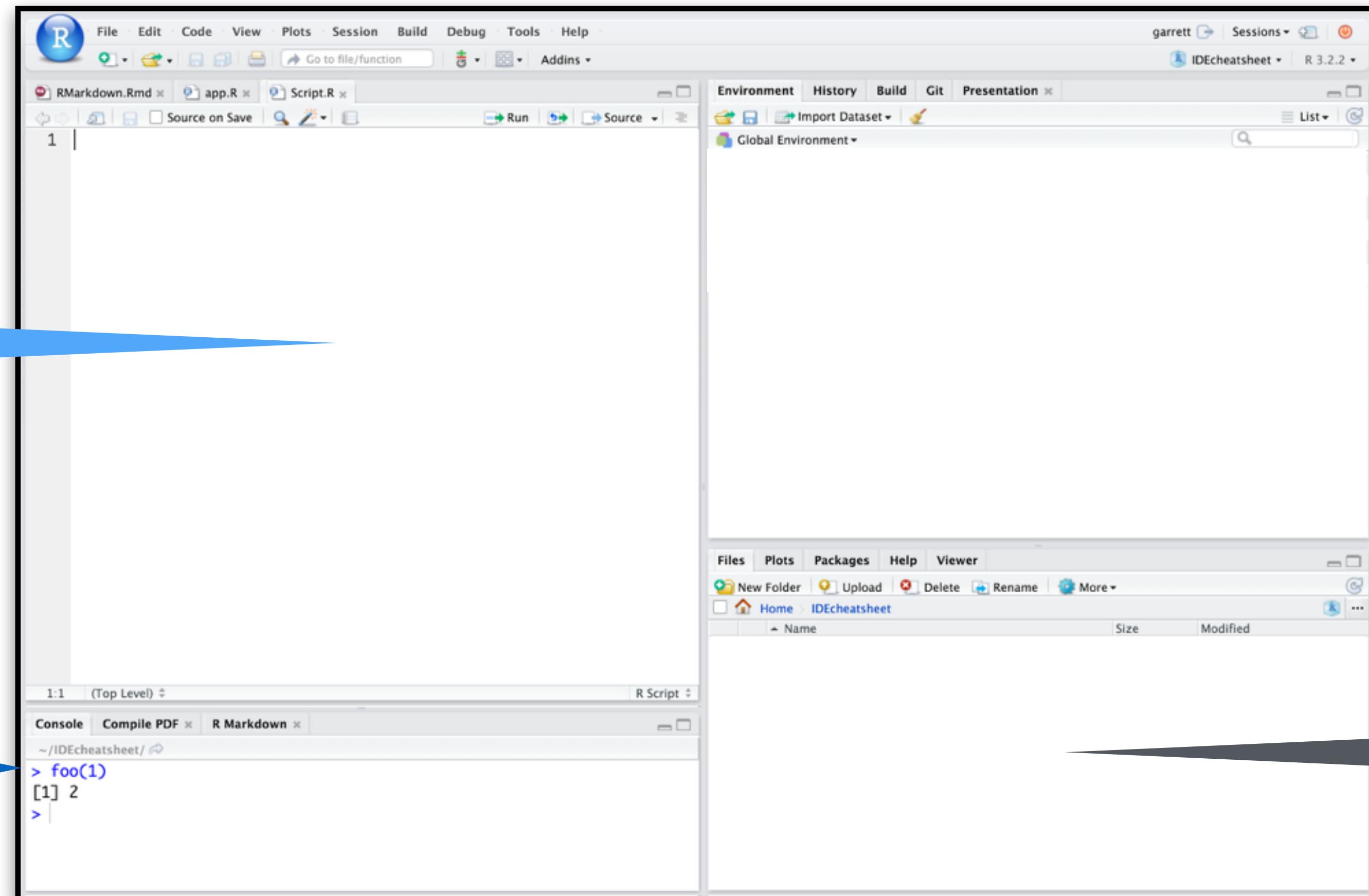
The screenshot shows an R terminal window titled "garrettgrolemund — R — 80x24". It displays the standard R startup message, which includes the last login date, R version, copyright information, and a note about the license. The message ends with "[Previously saved workspace restored]". Below this, a user types the command `> log10(100)`, followed by its output [1] 2, and then re-typing the command.

```
Last login: Fri Jun 28 18:03:32 on ttys001
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)
Copyright (C) 2013 The R Foundation for Statistical Computing
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
You are welcome to redistribute it under certain conditions.
Type 'Natural language support but running in an English locale

R is a collaborative project with many contributors.  Locale
Type 'contributors()' for more information and
'citation()' on how to cite R or packages in publications.
Type 'contributors()' for more information and
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
[Previously saved workspace restored]

> log10(100)
[1] 2
> log10(100)
```

RStudio: Software for R



RStudio IDE Cheat Sheet

learn more at www.rstudio.com



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



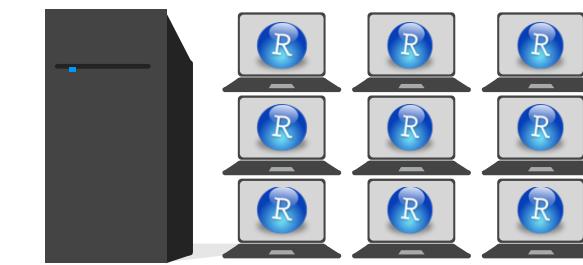
Desktop IDE

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

Press **↑** to see command history

Maximize, minimize panes

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

View in data viewer View function source code

Displays saved objects by type with short description

Path to displayed directory

Maximize, minimize panes

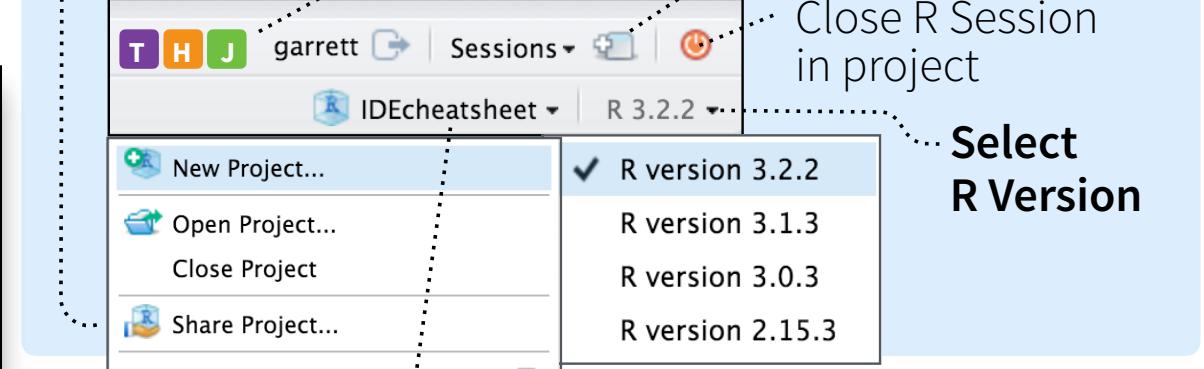
Drag pane boundaries

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

RStudio Pro Features

Share Project

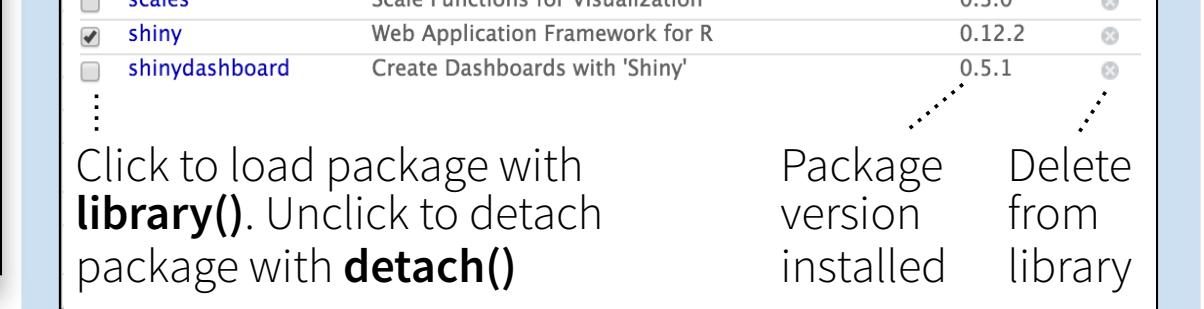
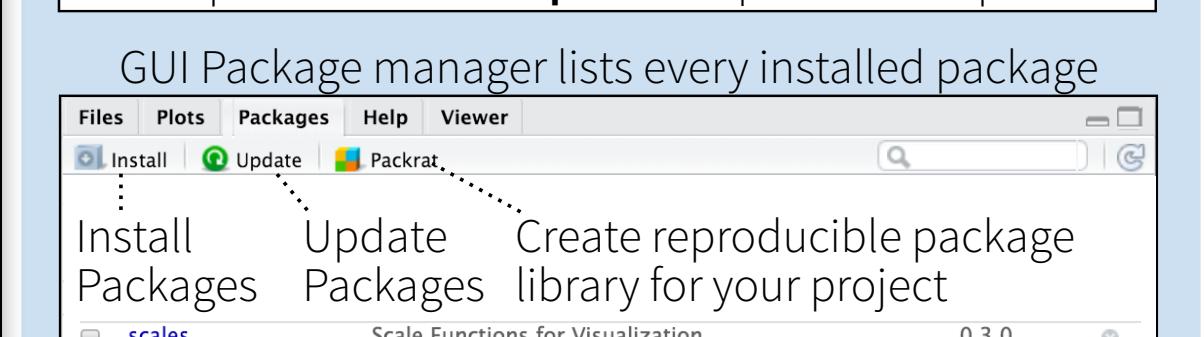
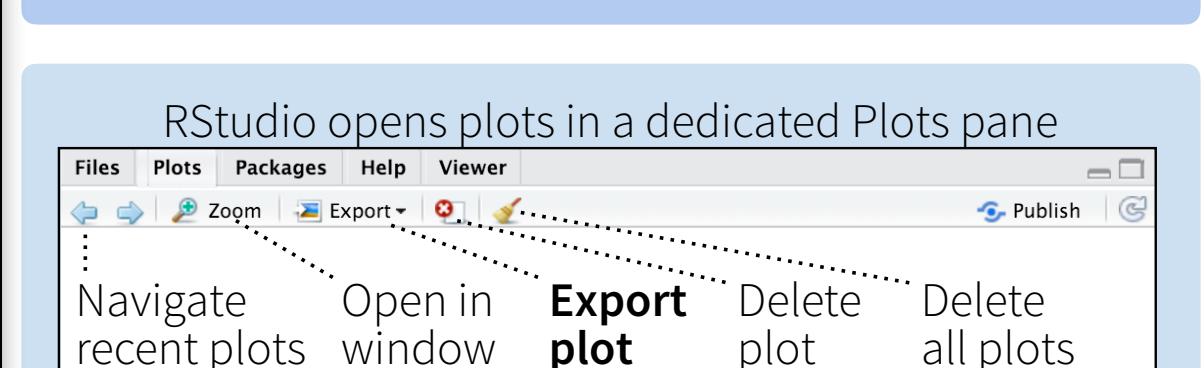
Active shared collaborators Start **new R Session** in current project Close R Session in project



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.



Debug Mode

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

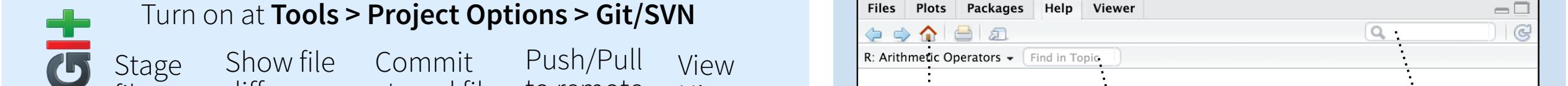
Launch debugger mode from origin of error

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



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

Stage file Show file diff Commit file to local file system Push/Pull file to remote repository



Your Turn

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

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

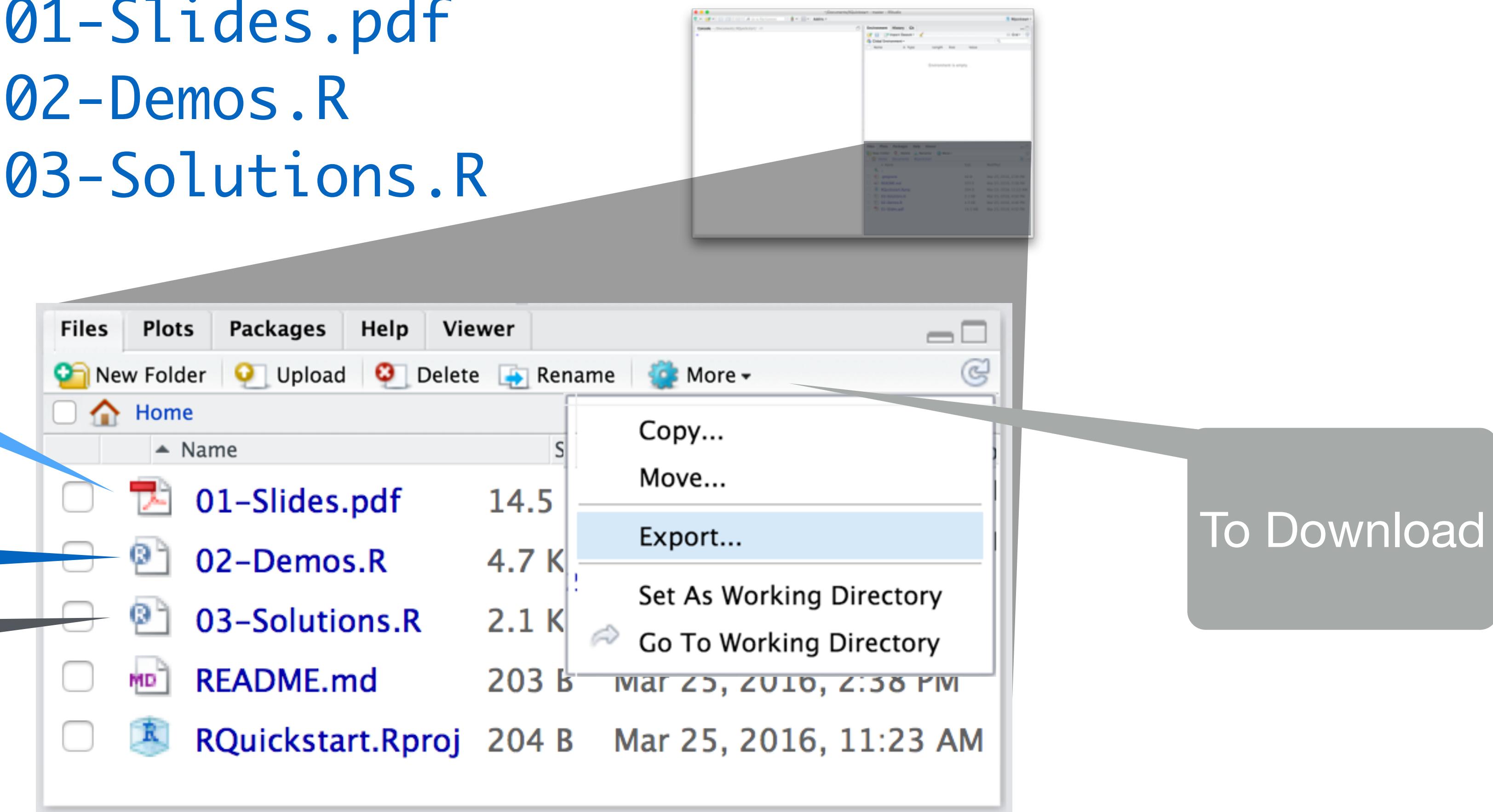
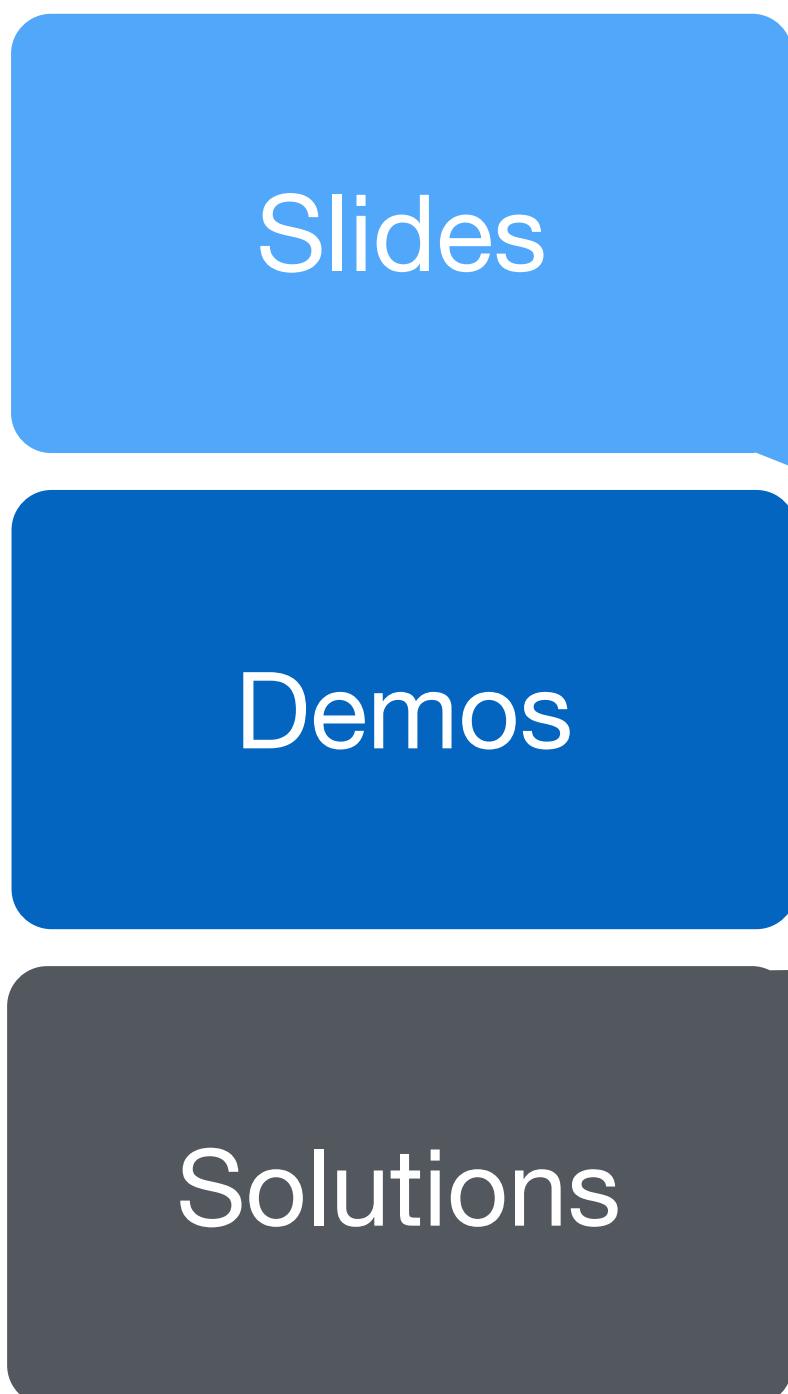
Open the project titled 01-Quickstart.



Class Materials

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

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

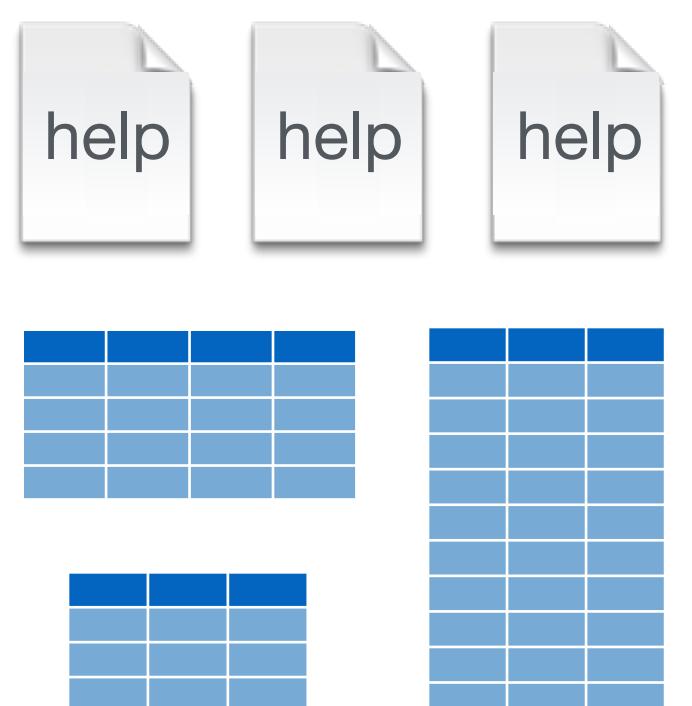


Packages

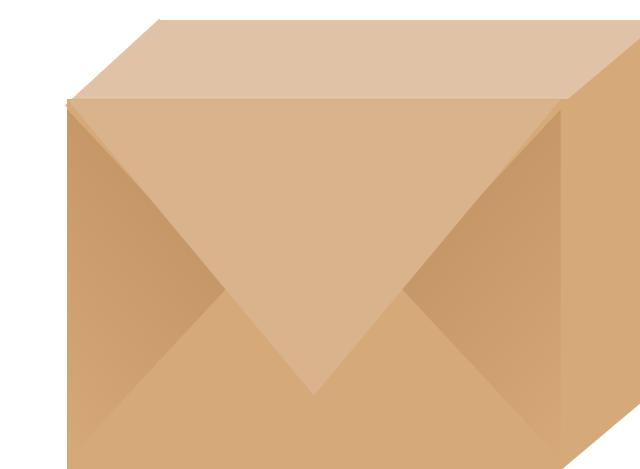
A bundle of functions, data sets, and help pages that you install in addition to the base R installation, e.g.

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

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



```
function1()  
function2()  
function3()  
function4()
```

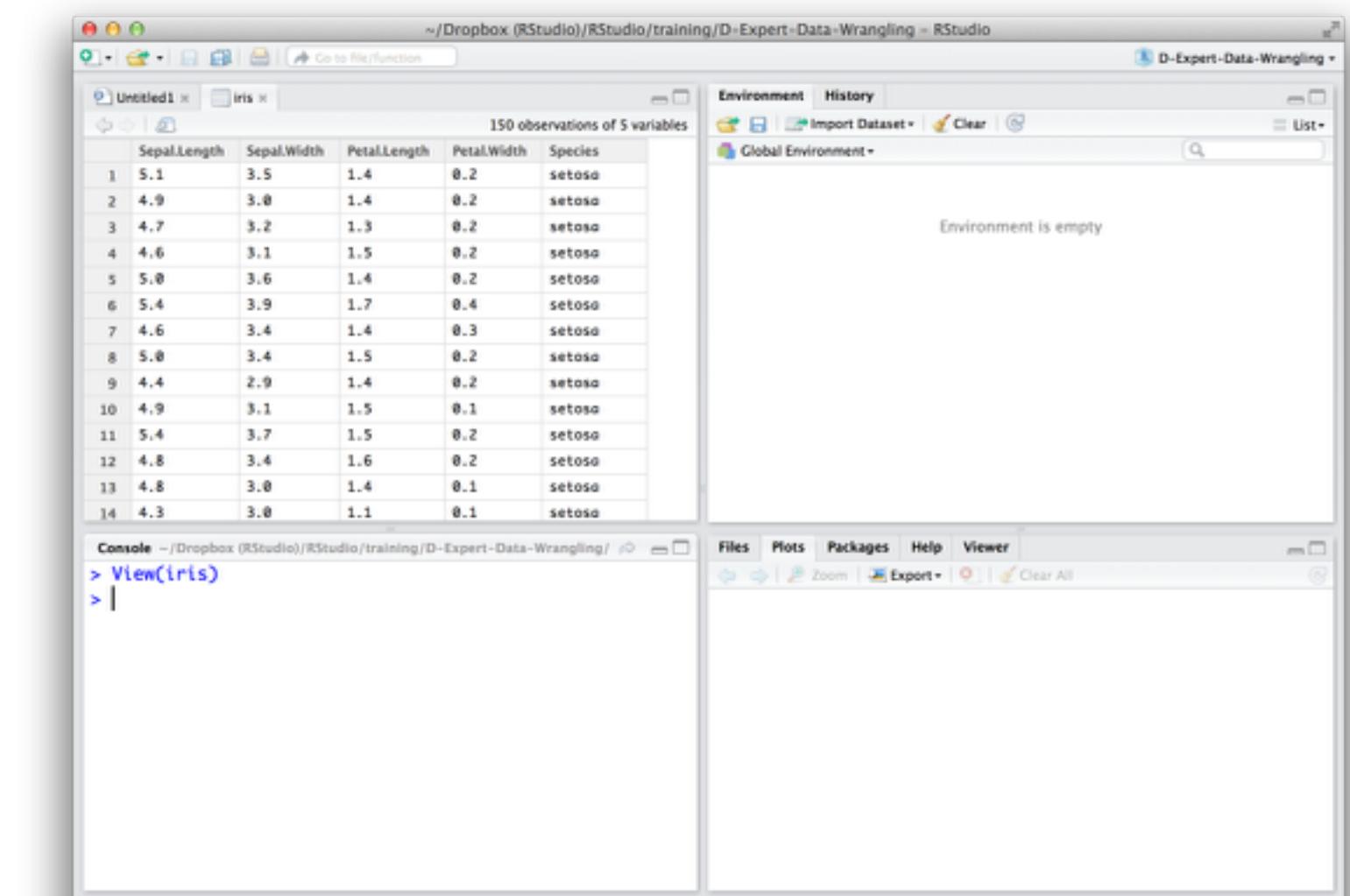


Run `library(ggplot2)`

View()

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

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



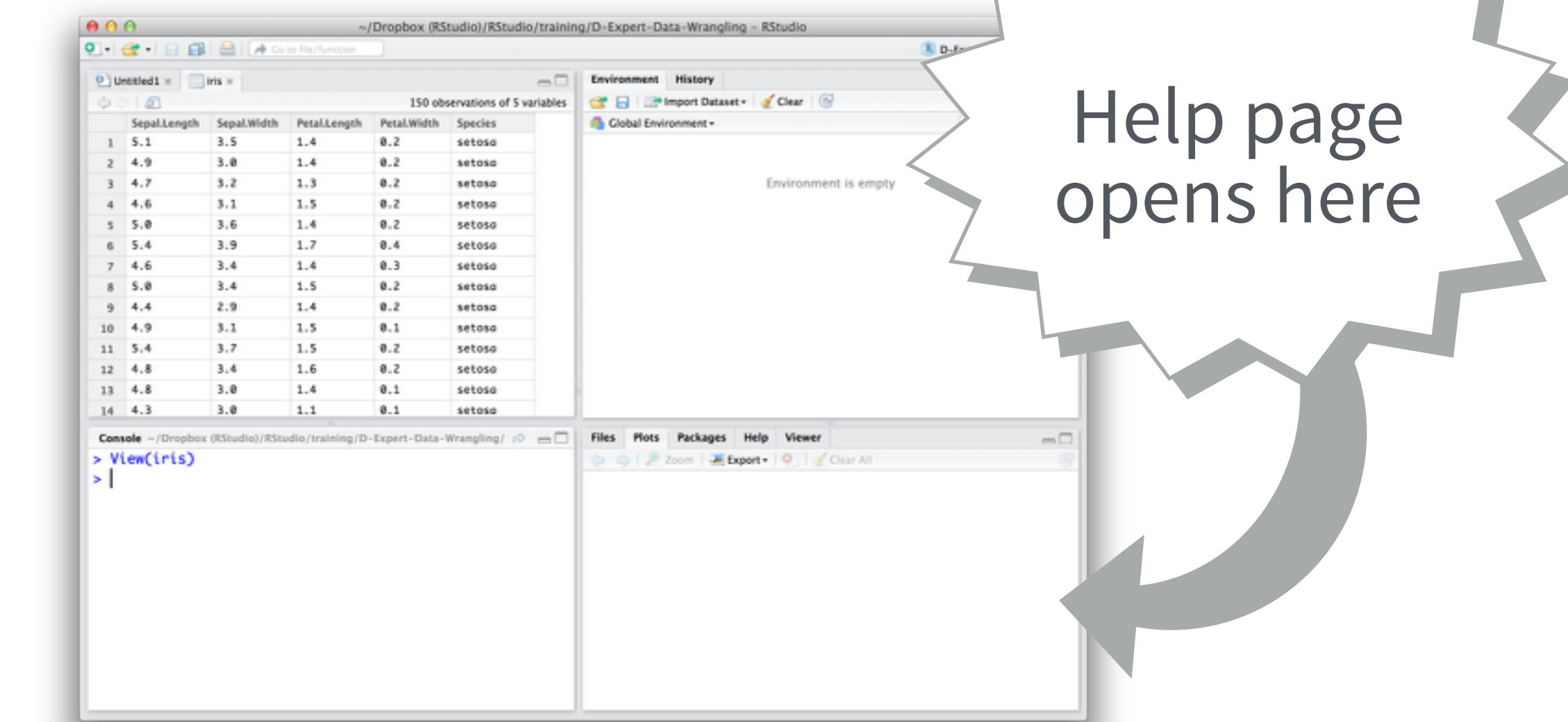
Help

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

?mpg

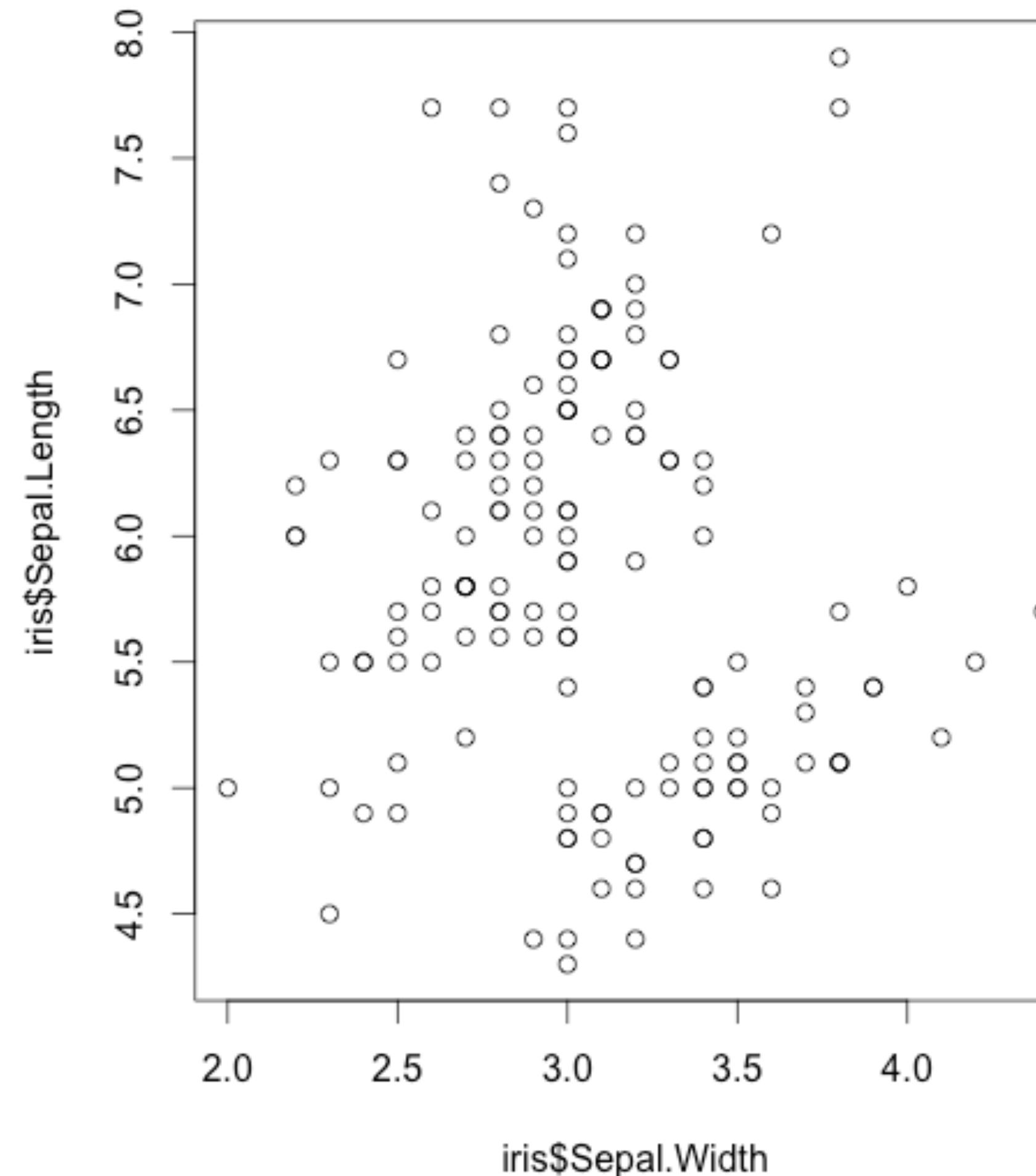
?ggplot

?geom_point



visualizations with ggplot2

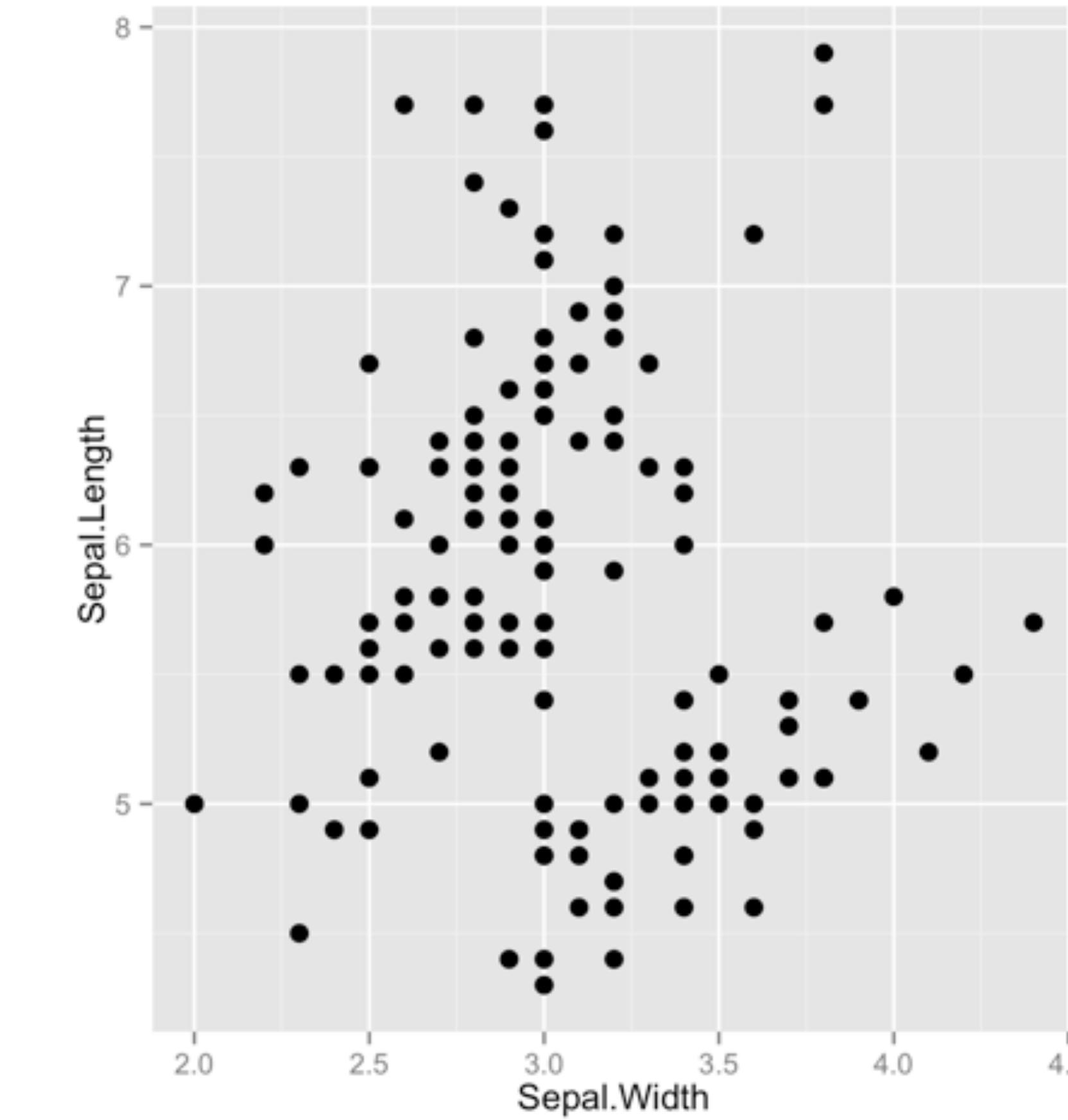
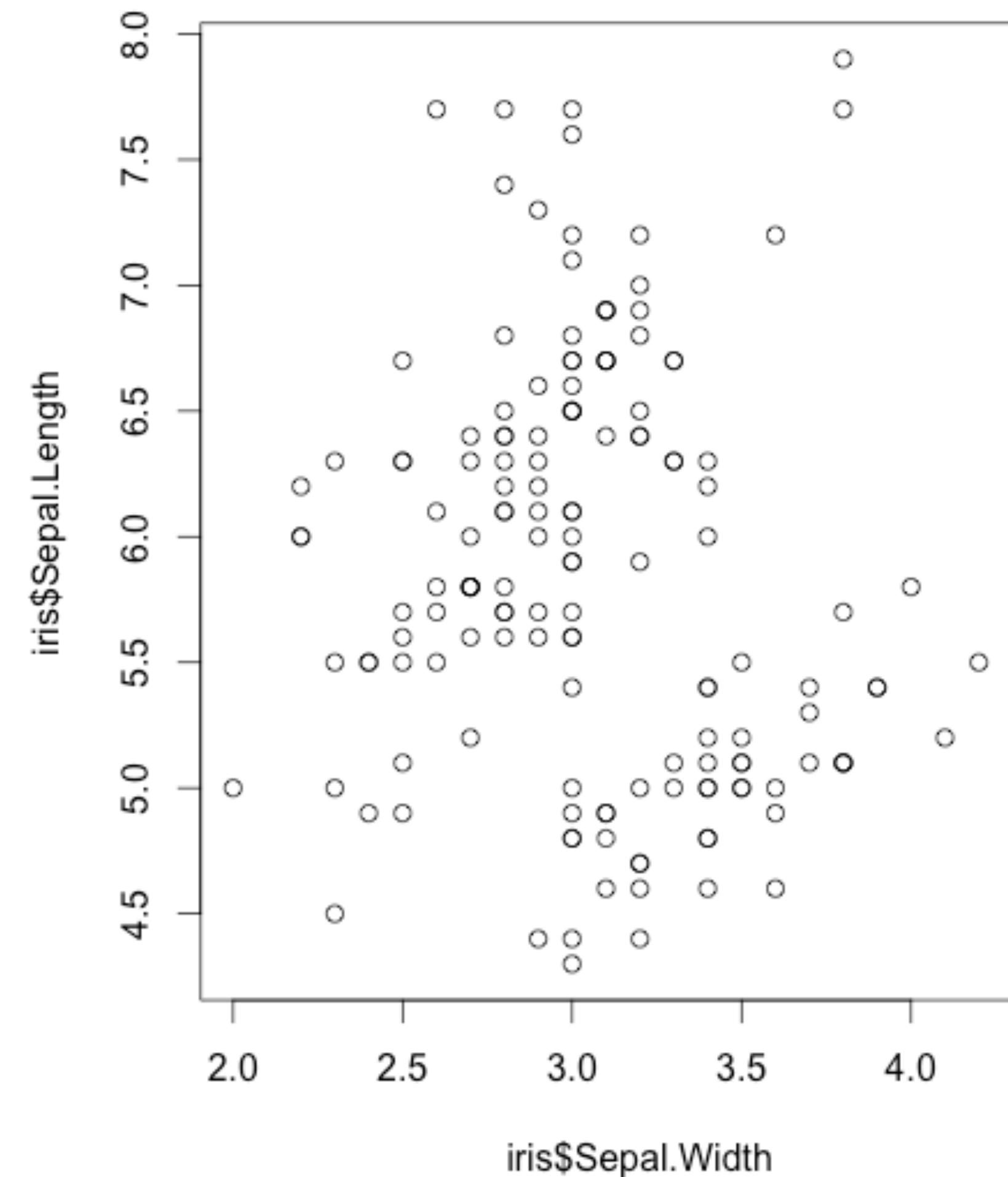
plot



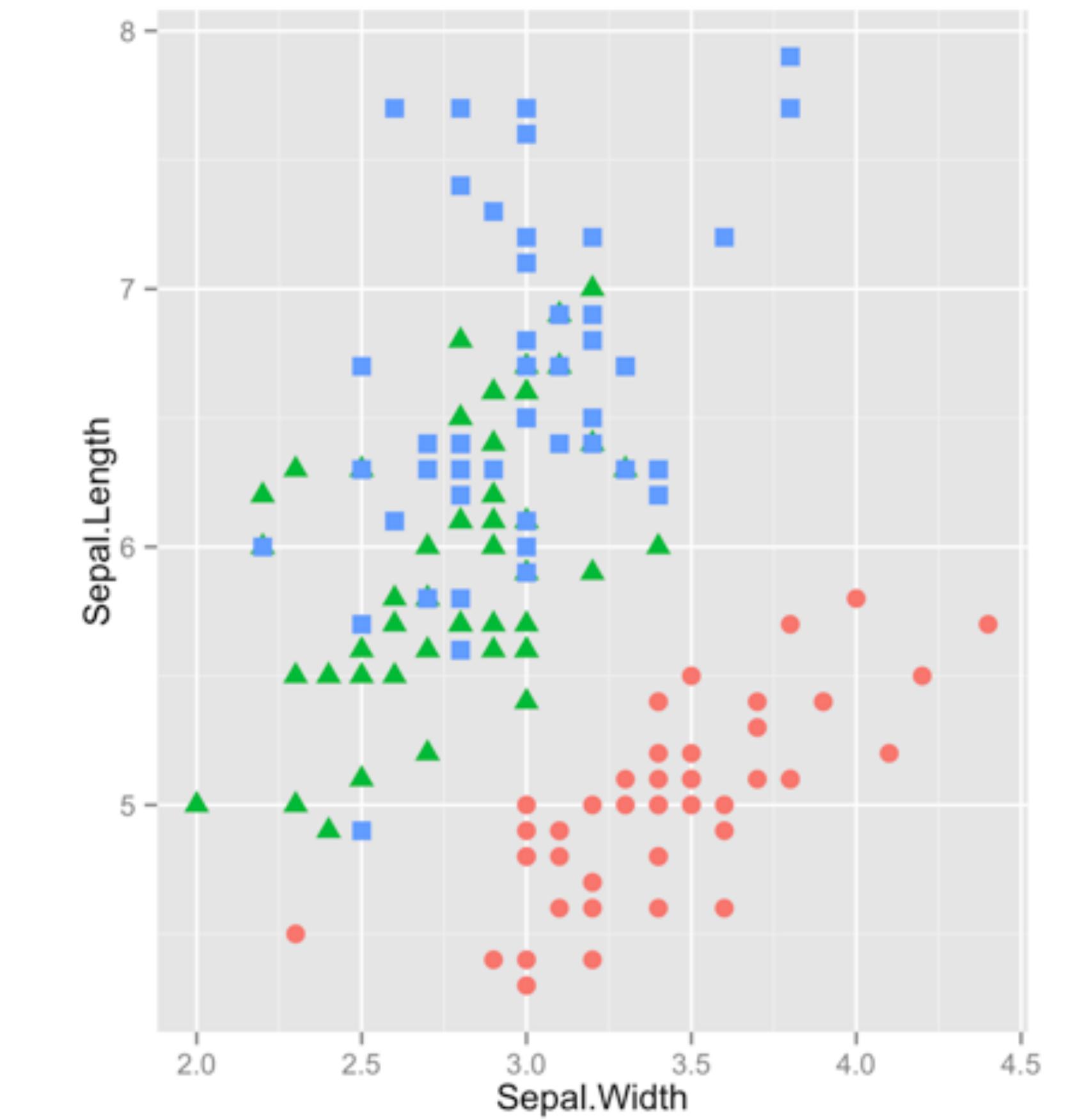
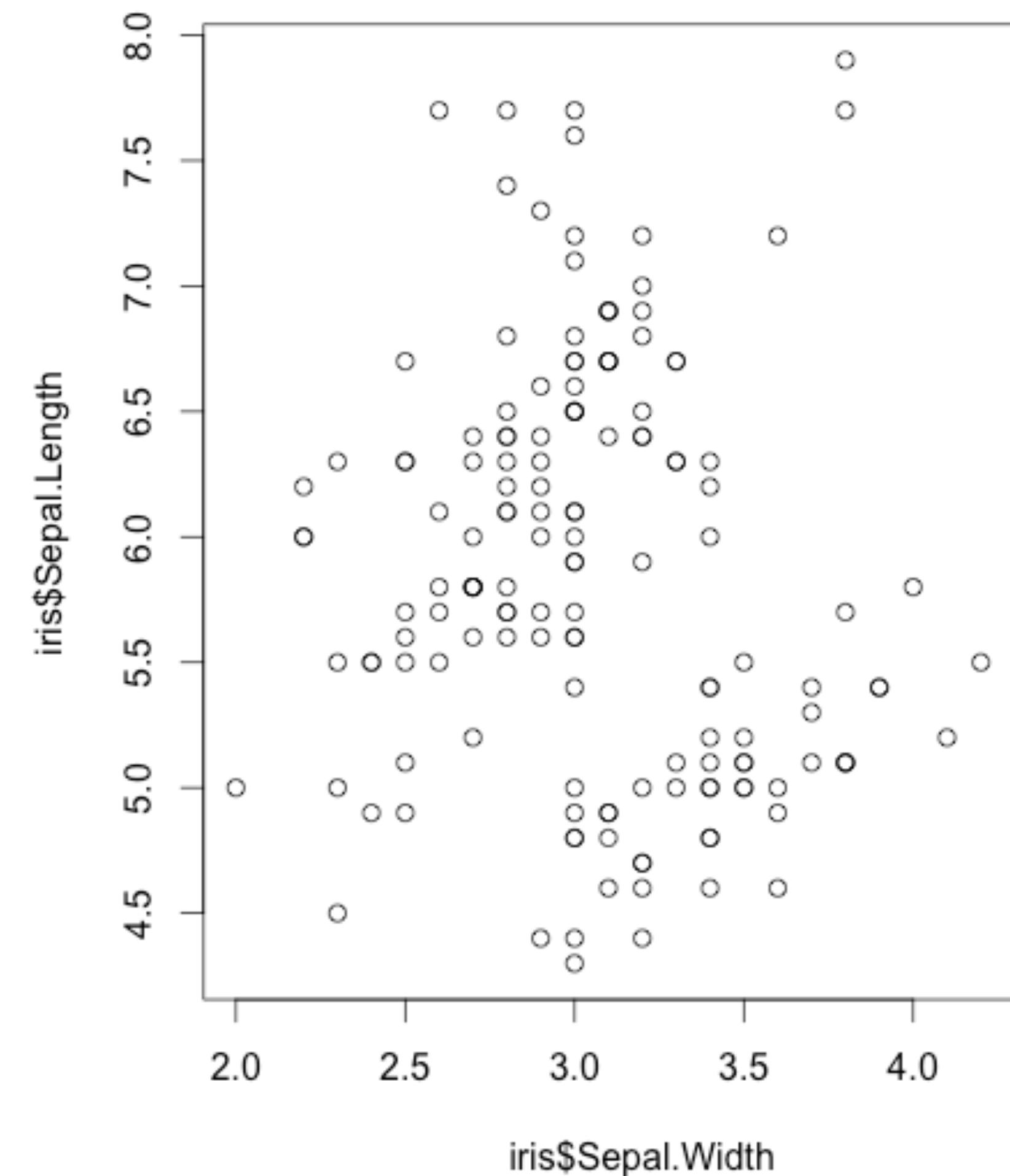
```
plot(iris$Sepal.Width,  
     iris$Sepal.Length)
```

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

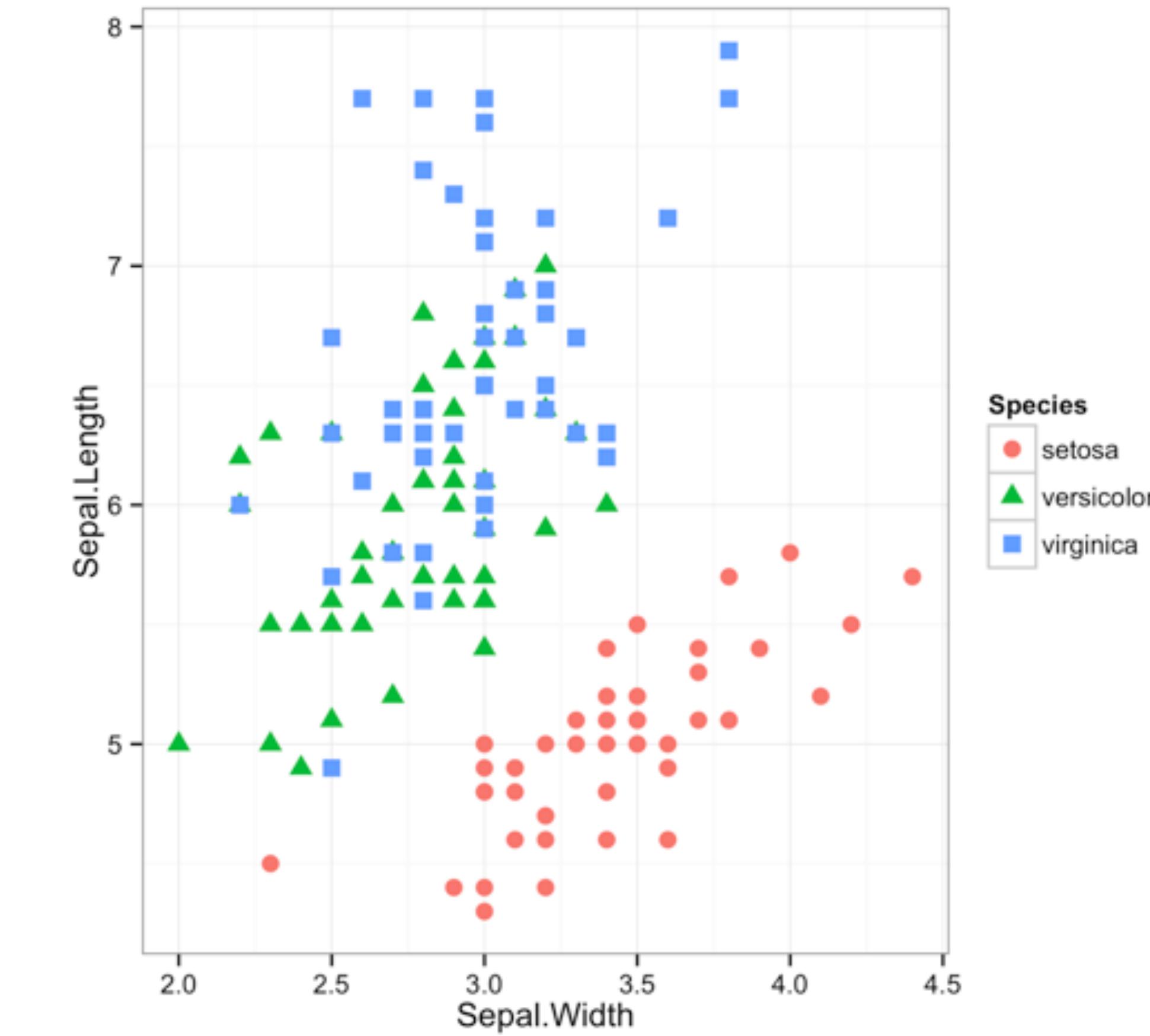
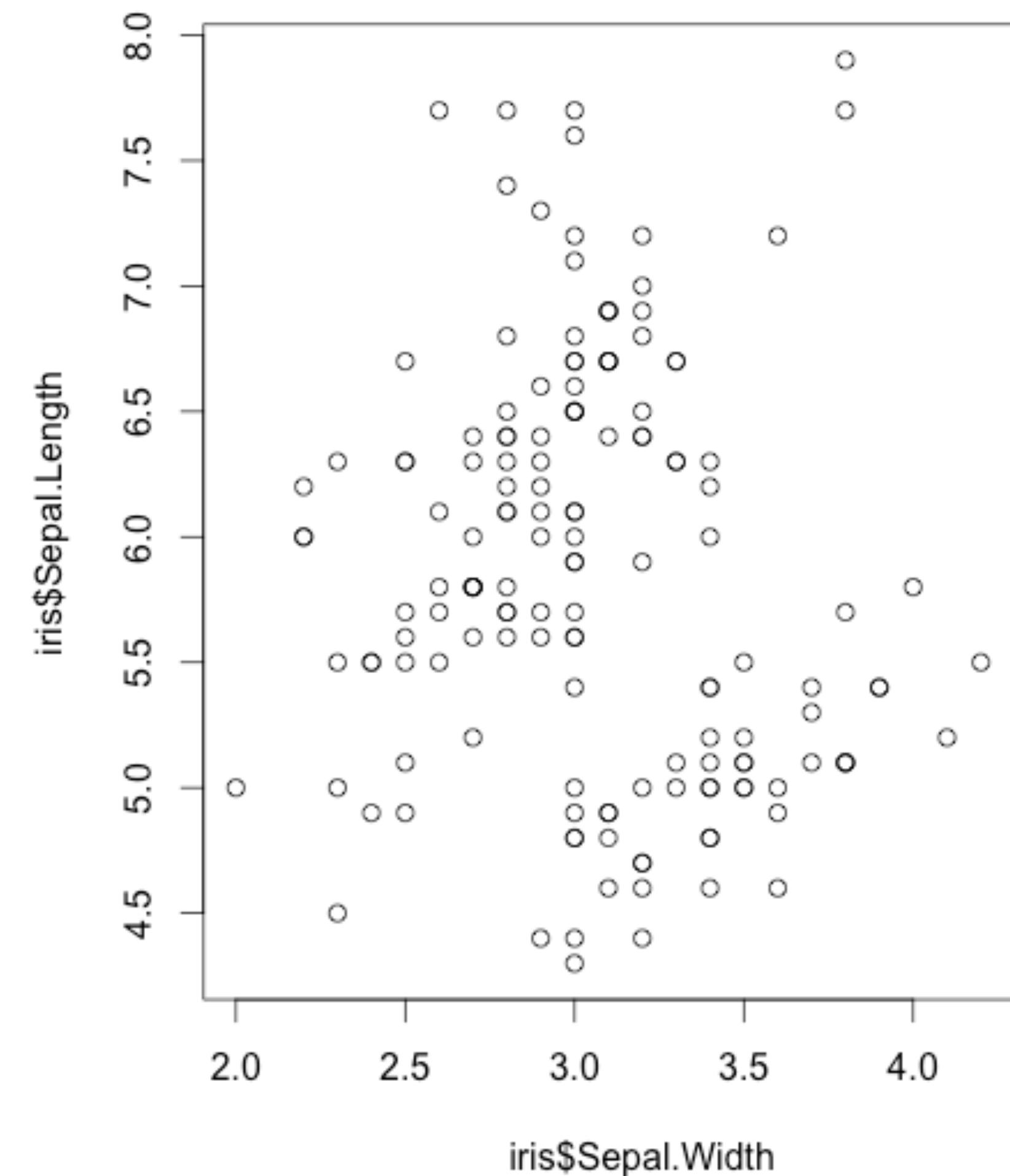
ggplot2



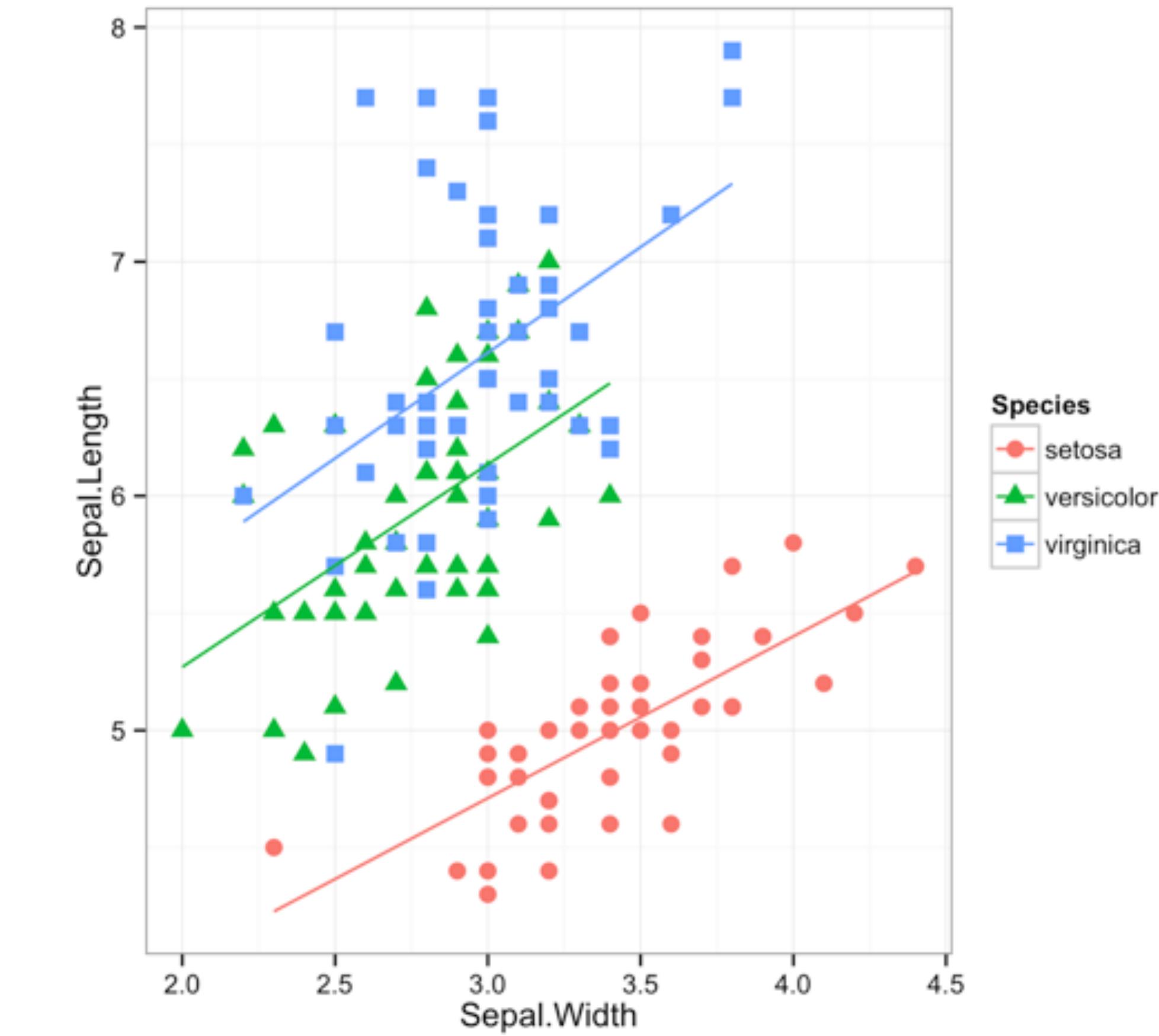
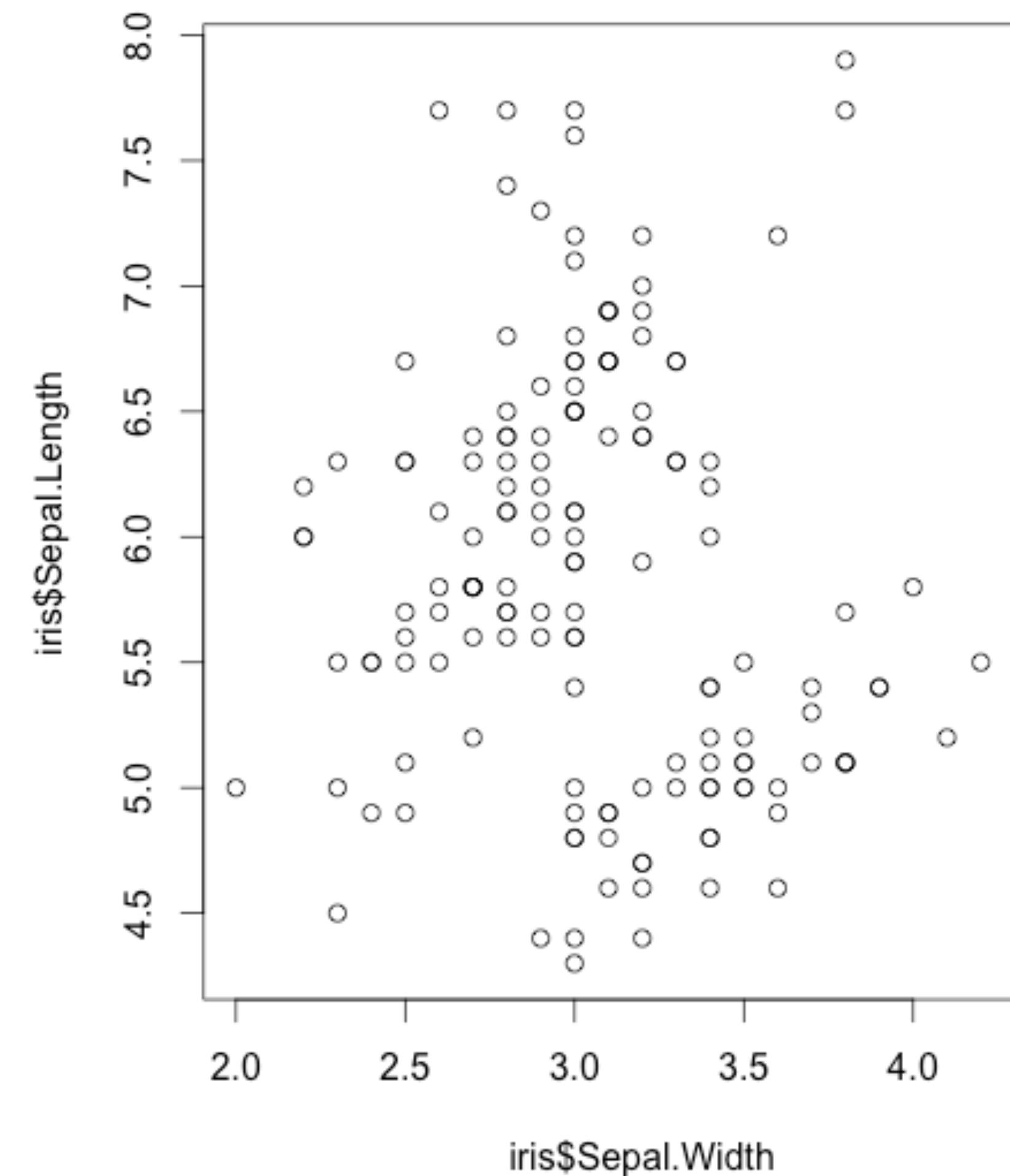
ggplot2



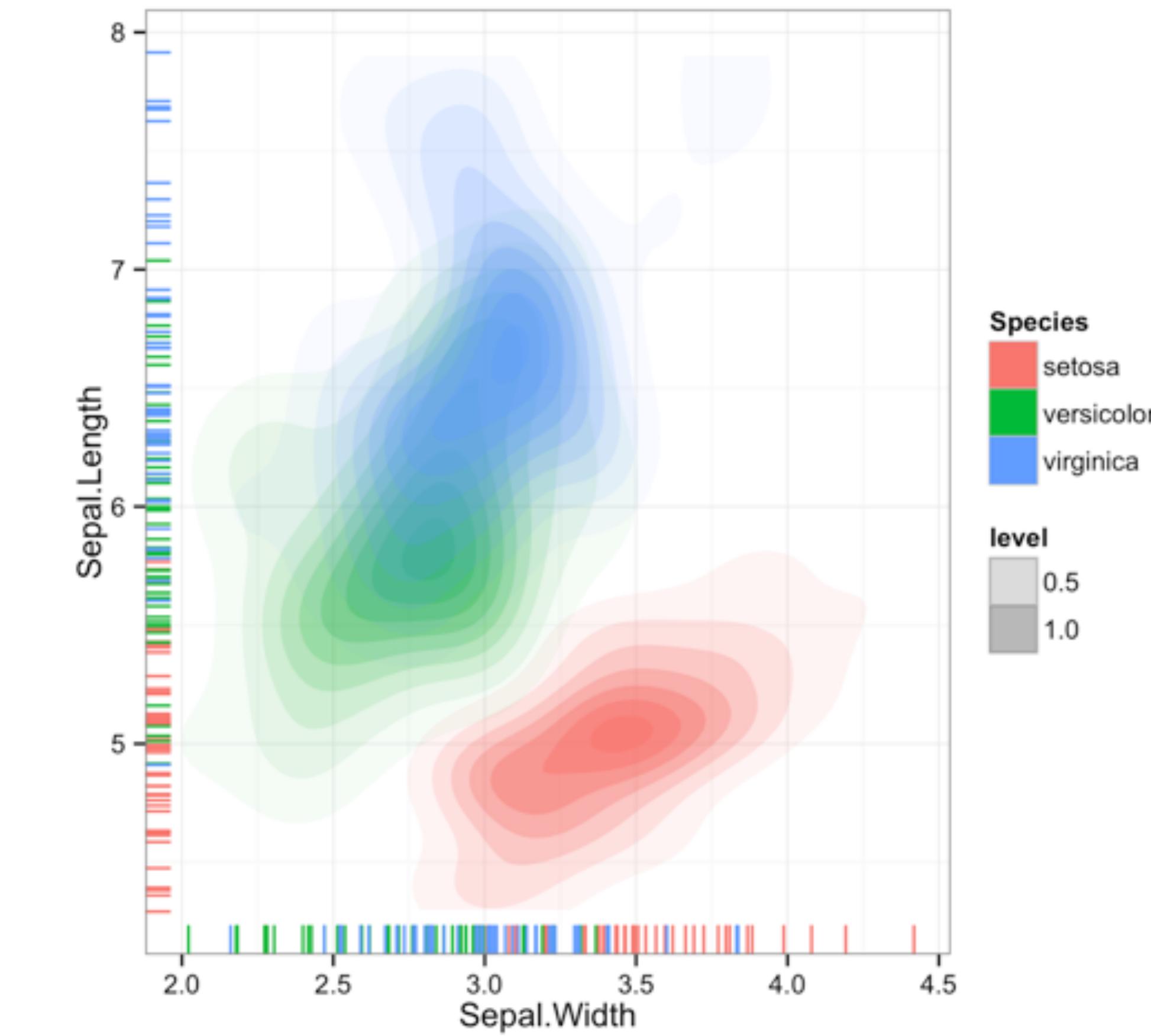
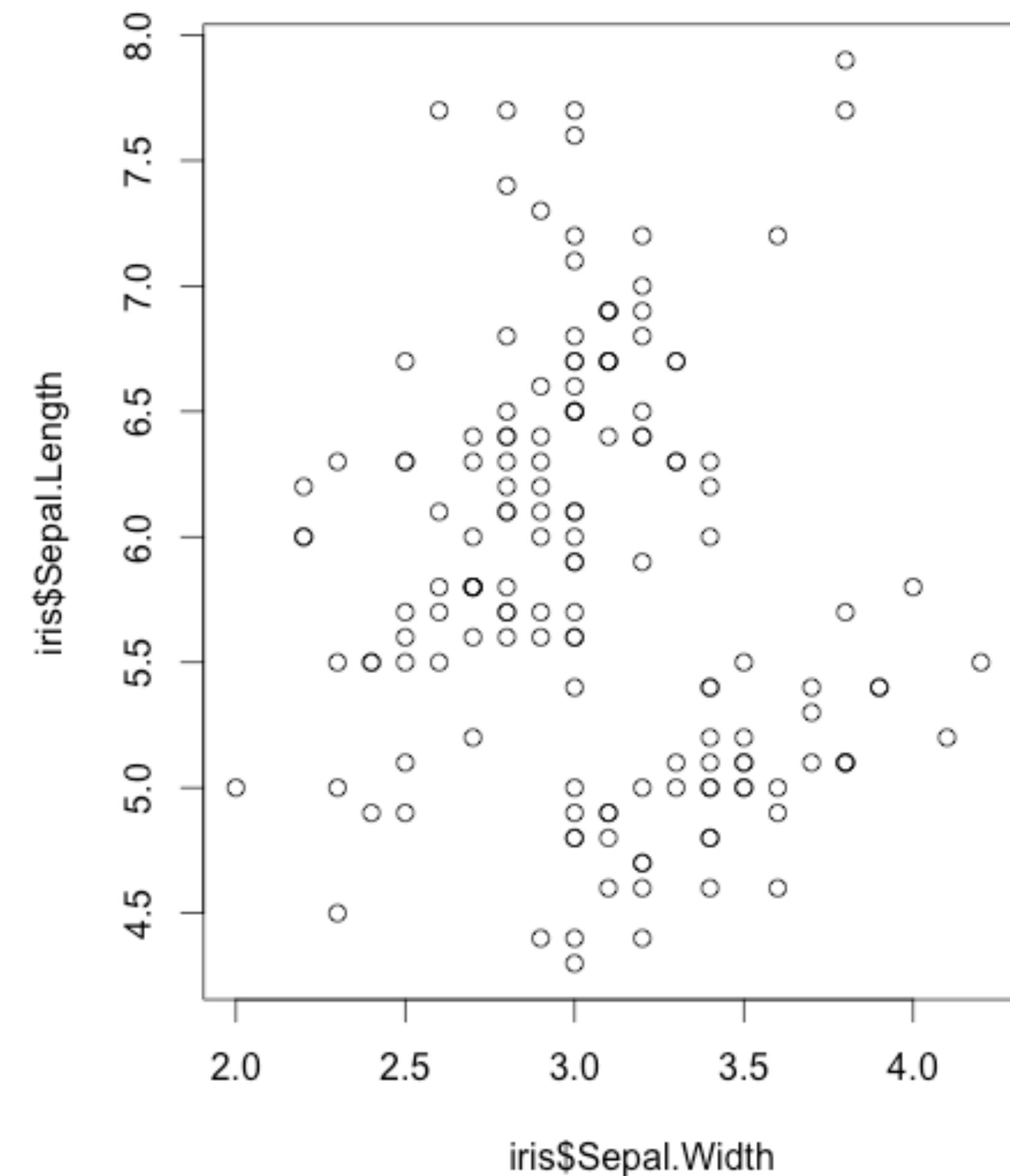
ggplot2



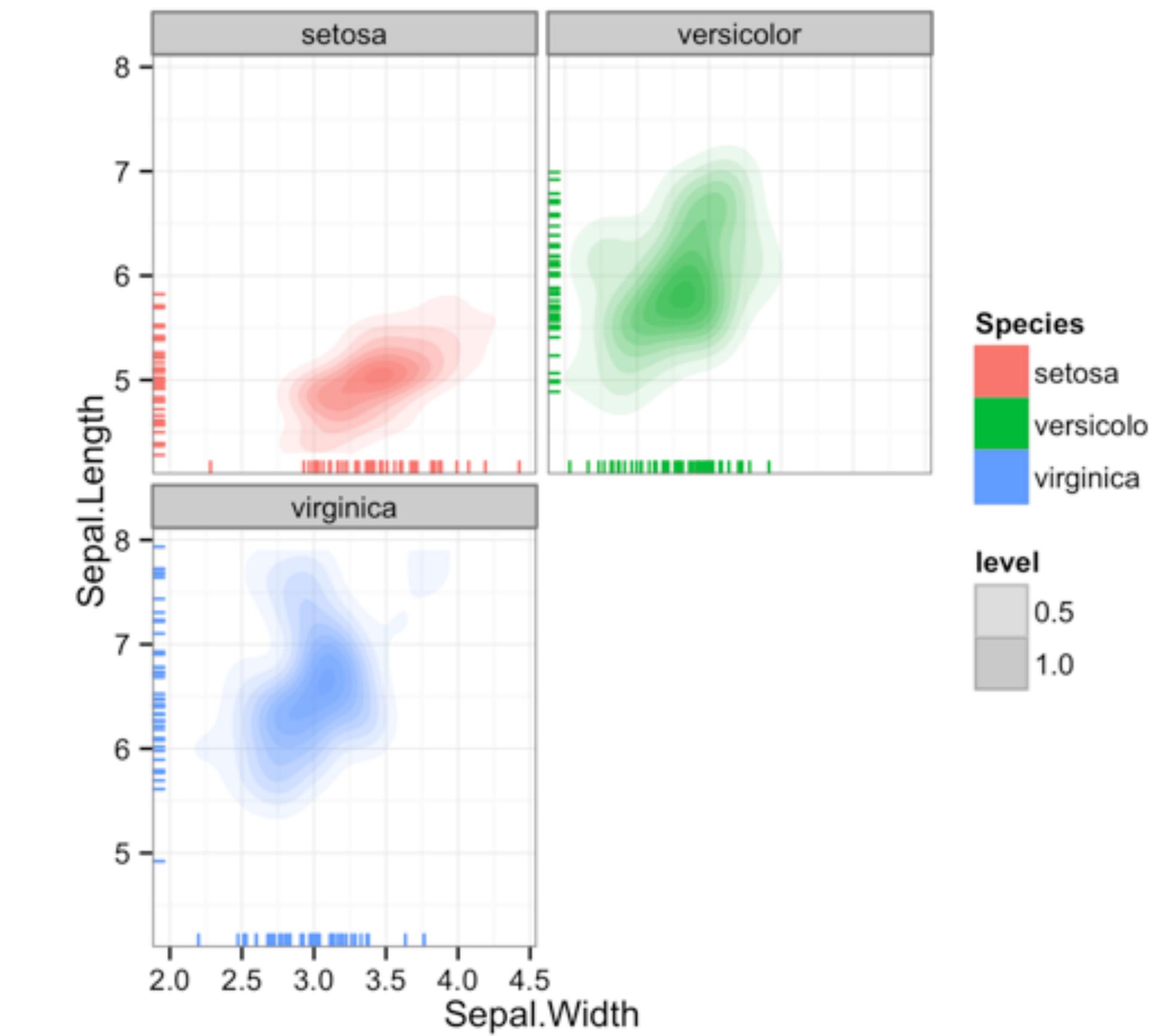
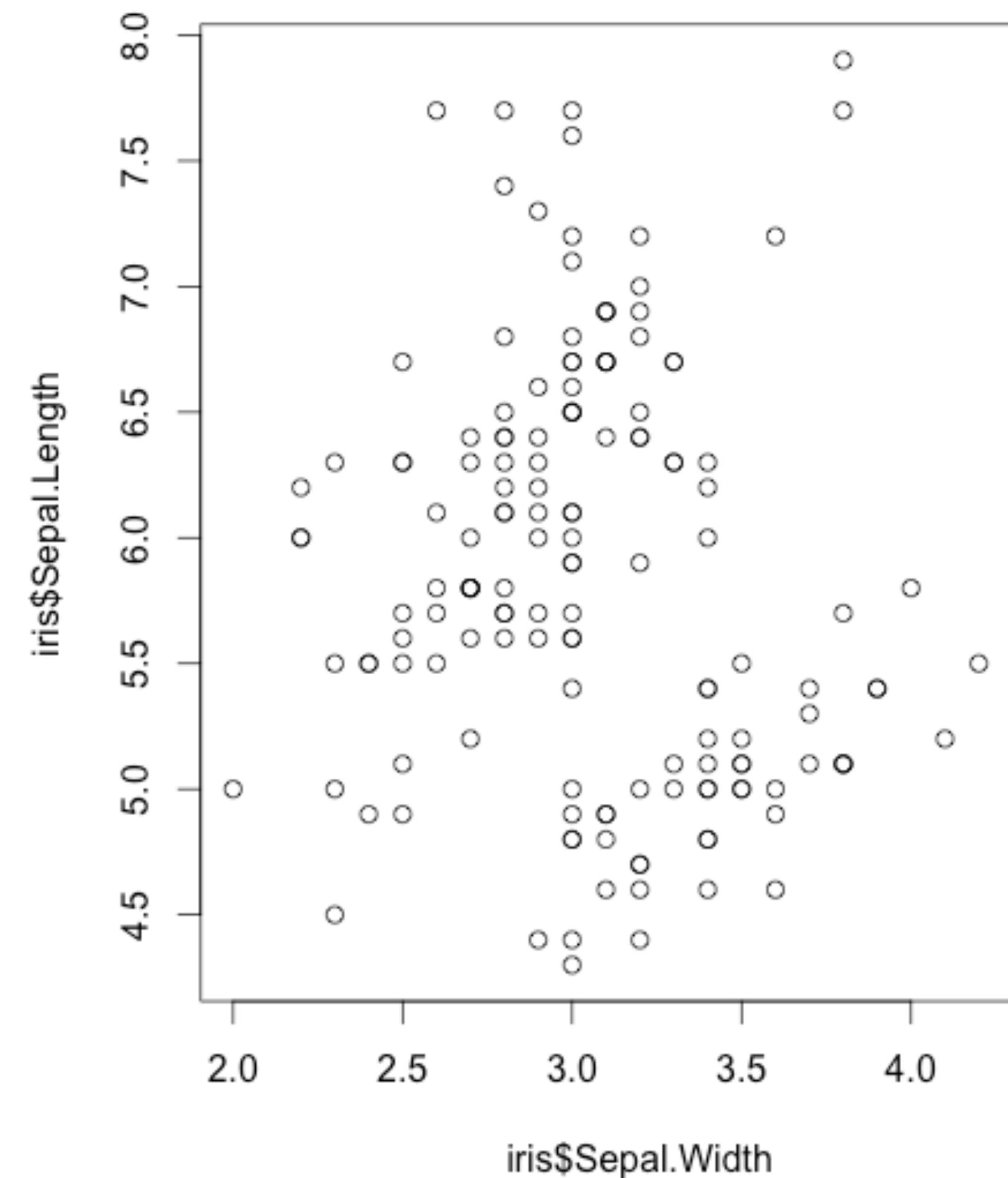
ggplot2

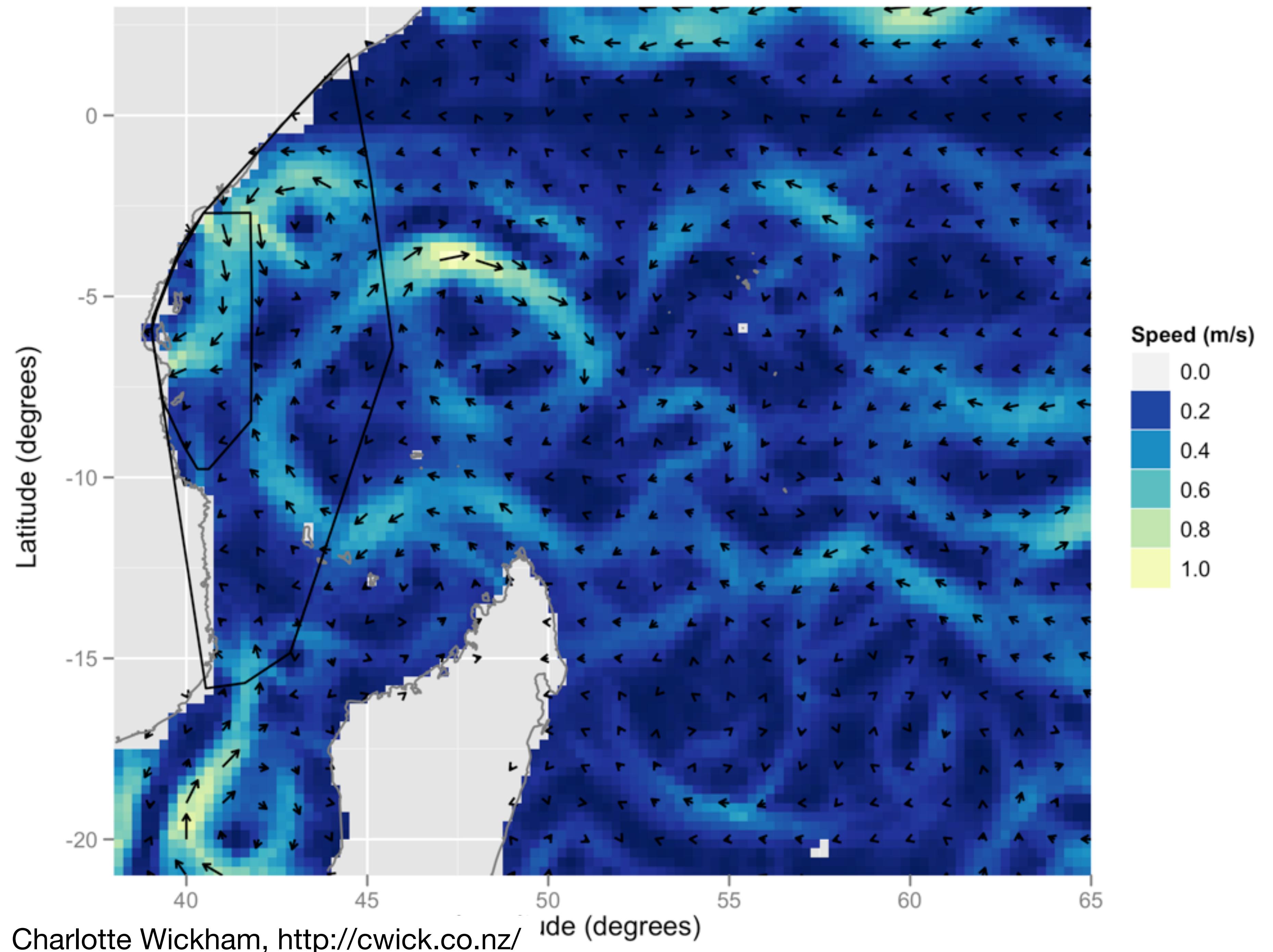


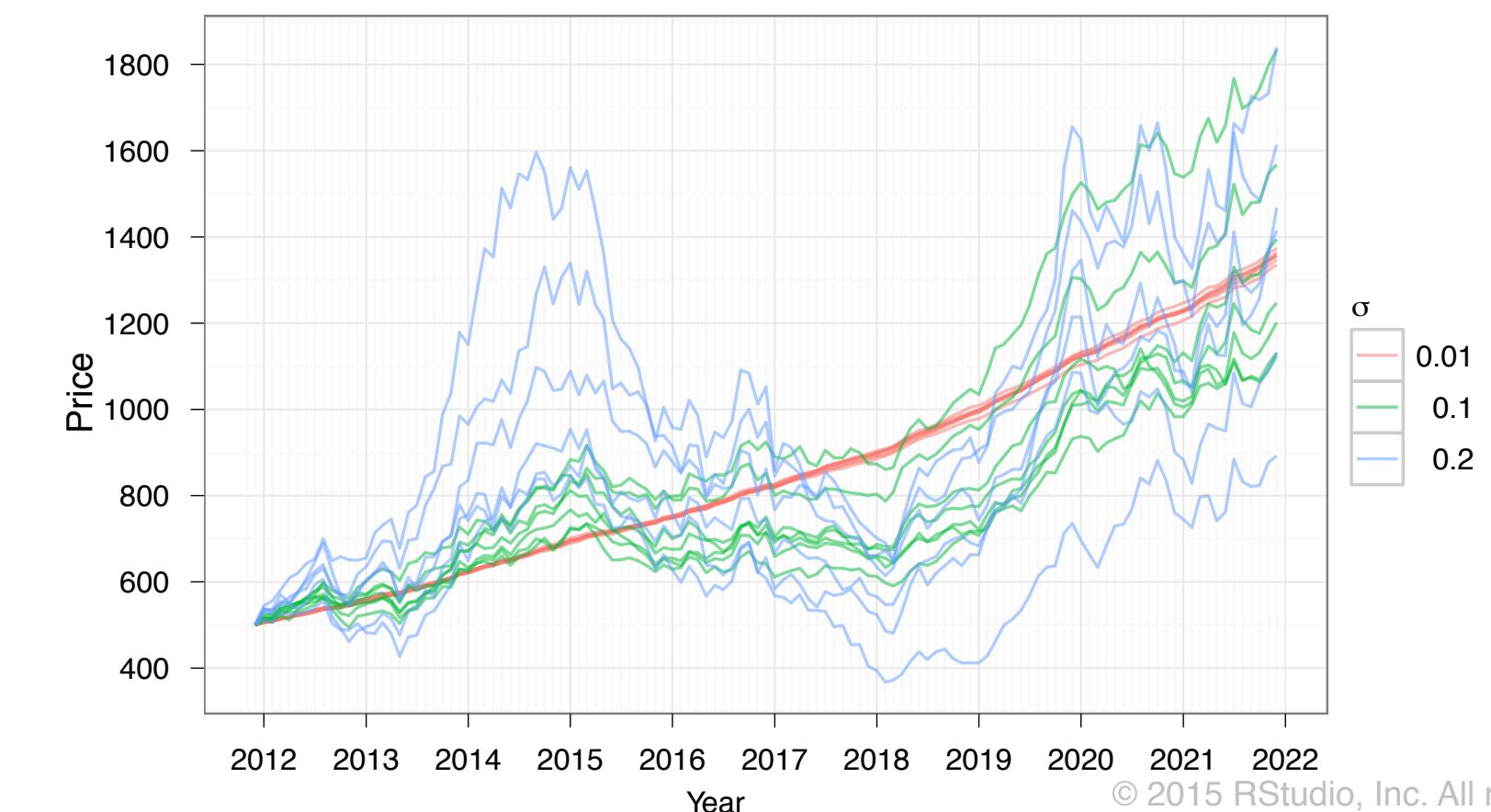
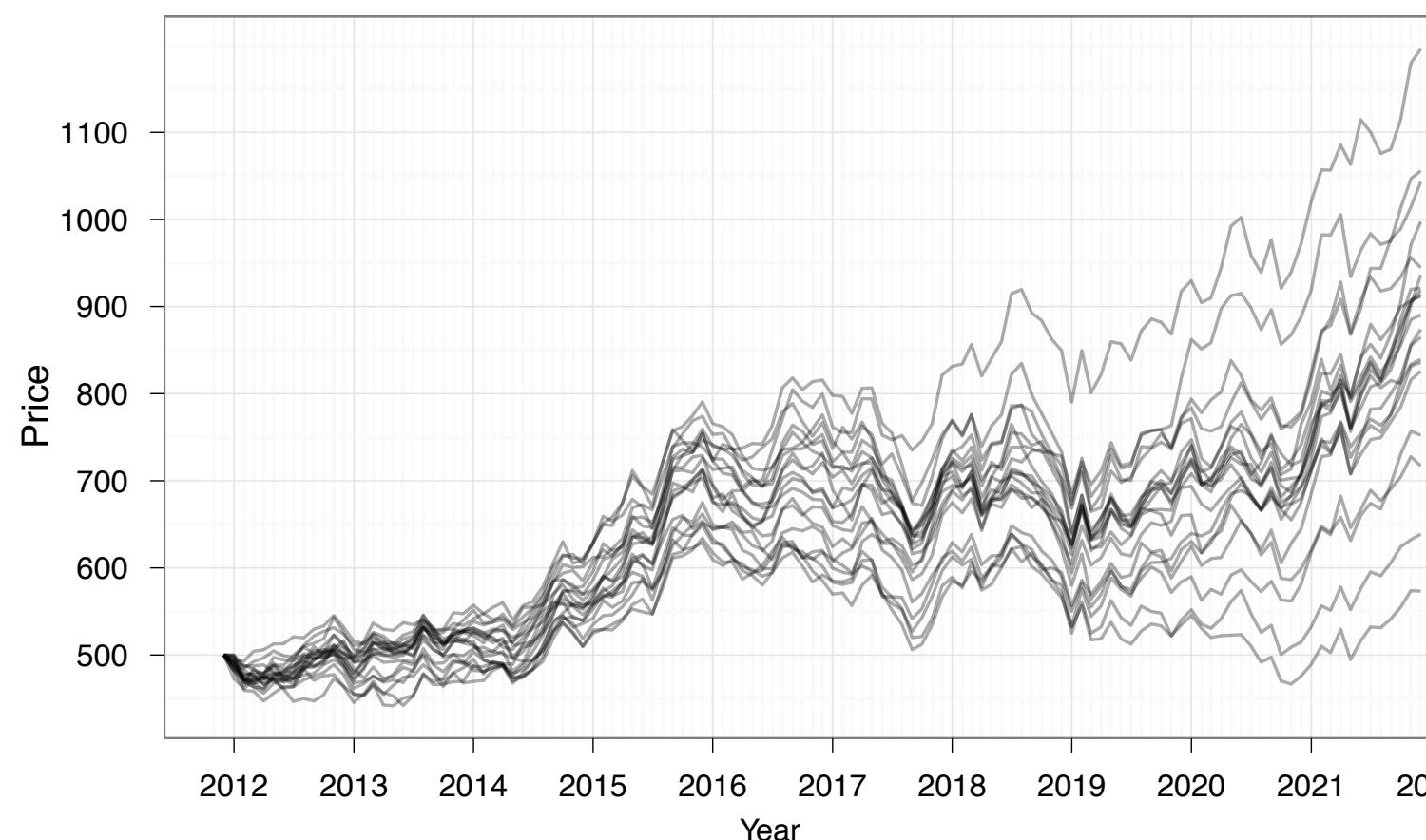
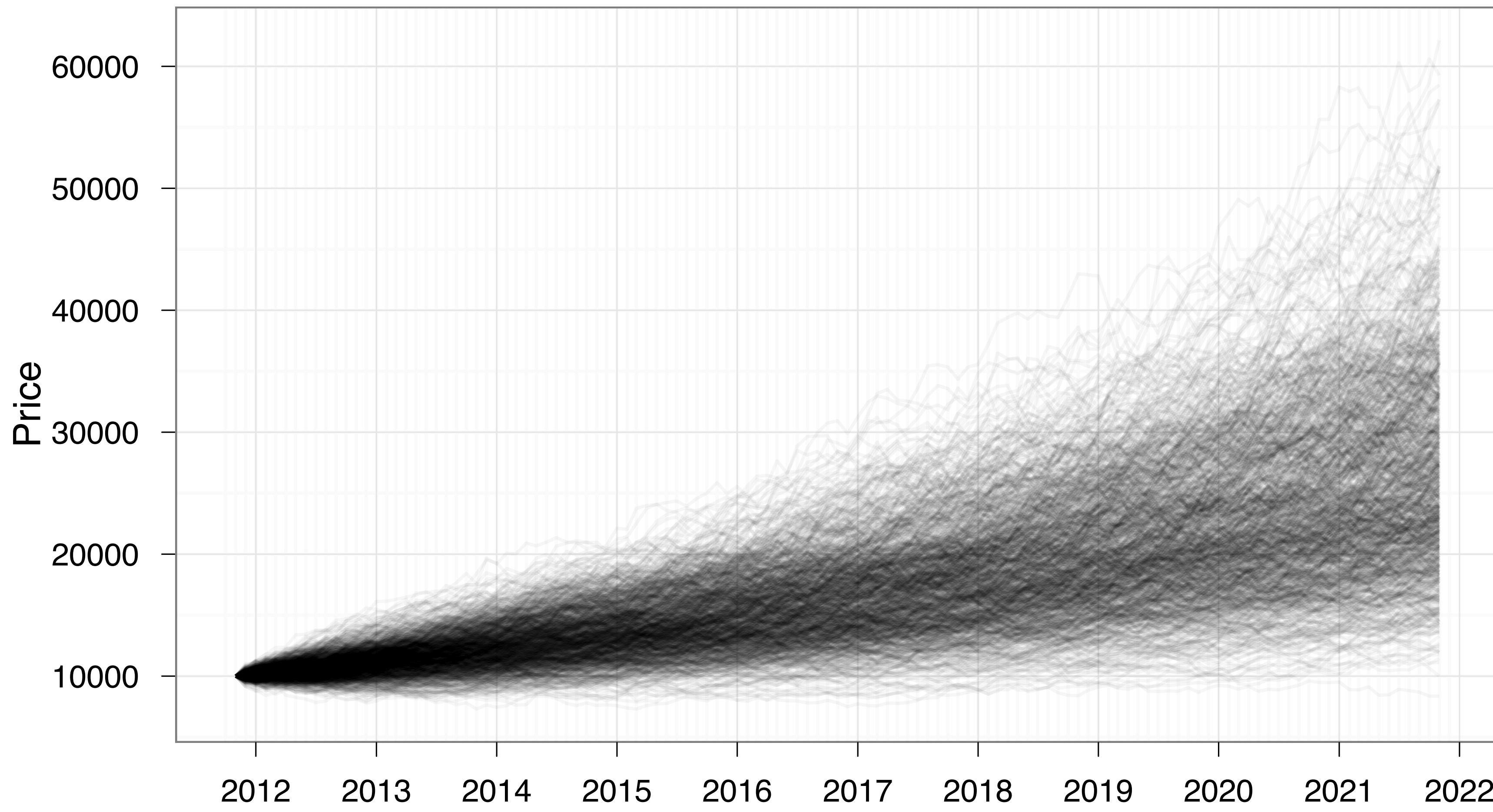
ggplot2

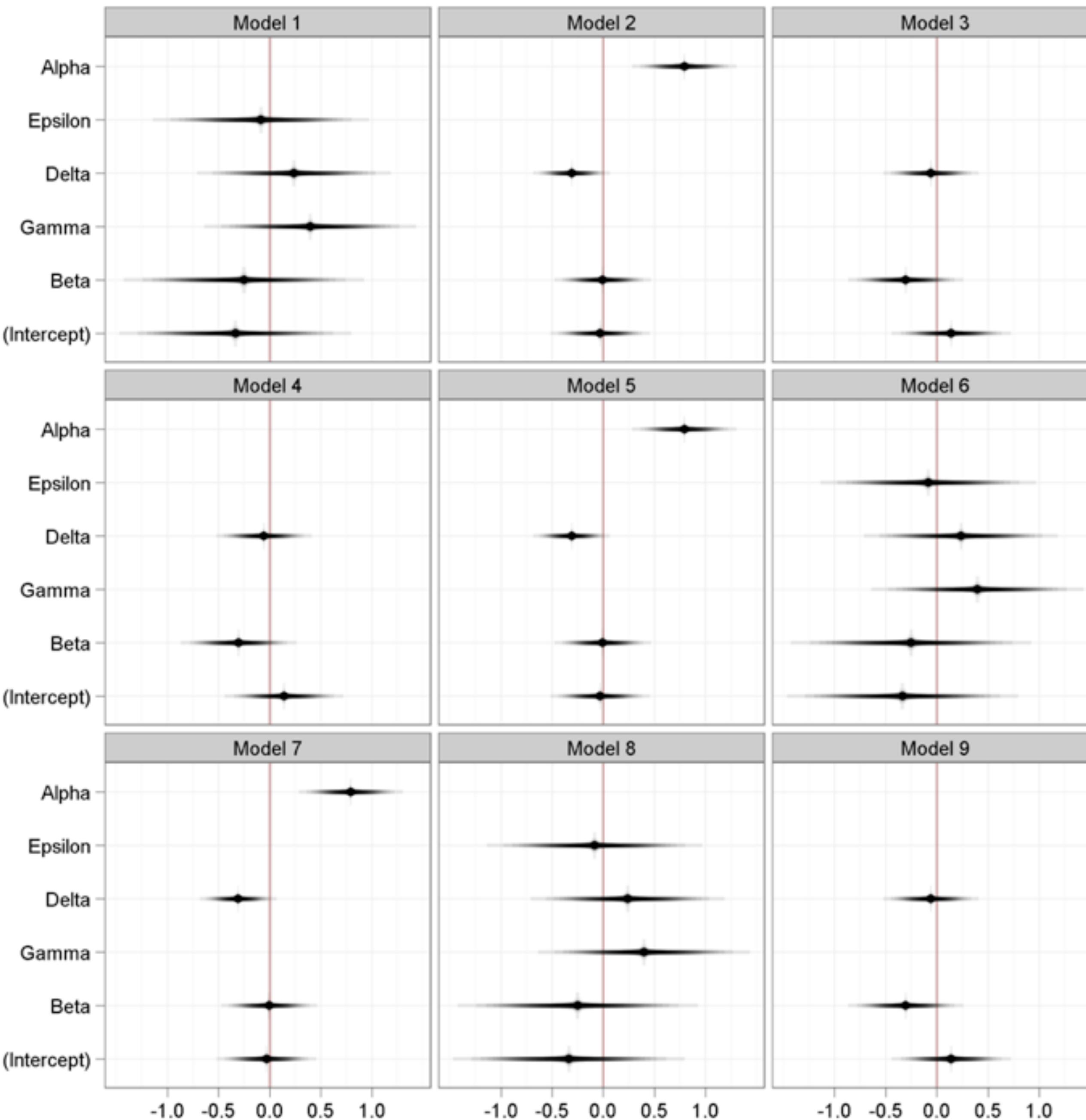


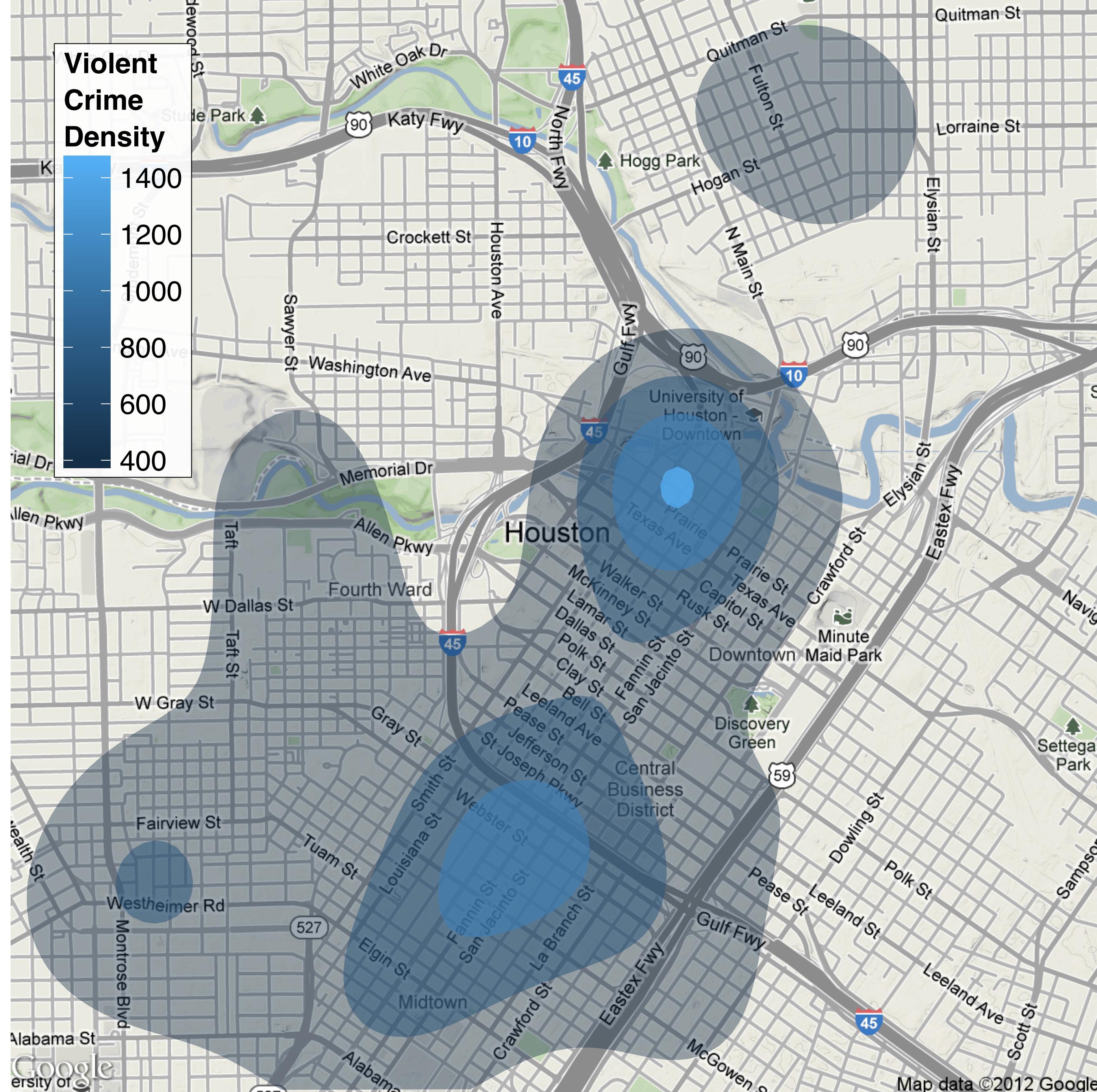
ggplot2











London Cycle Hire Journeys

Thicker, yellower lines mean more journeys



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?



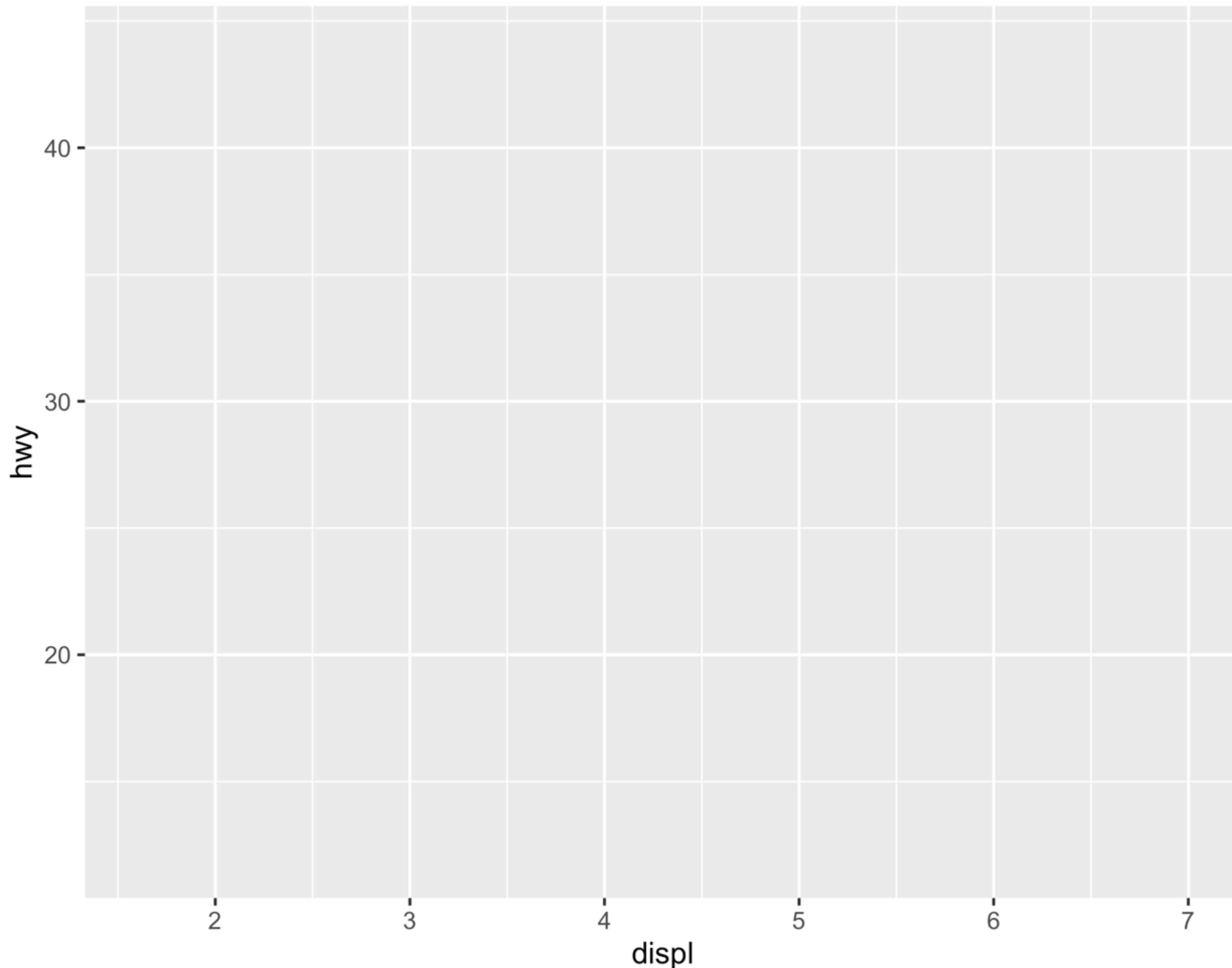
ggplot

1. "Initialize" a plot with ggplot()

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

The diagram illustrates the components of the ggplot() function. It consists of four colored speech bubbles pointing to specific parts of the code:

- A green bubble labeled "data set" points to the first argument "mpg".
- A dark gray bubble labeled "aes()" points to the second argument.
- A blue bubble labeled "x variable" points to the "x" parameter inside the aes() function.
- An orange bubble labeled "y variable" points to the "y" parameter inside the aes() function.

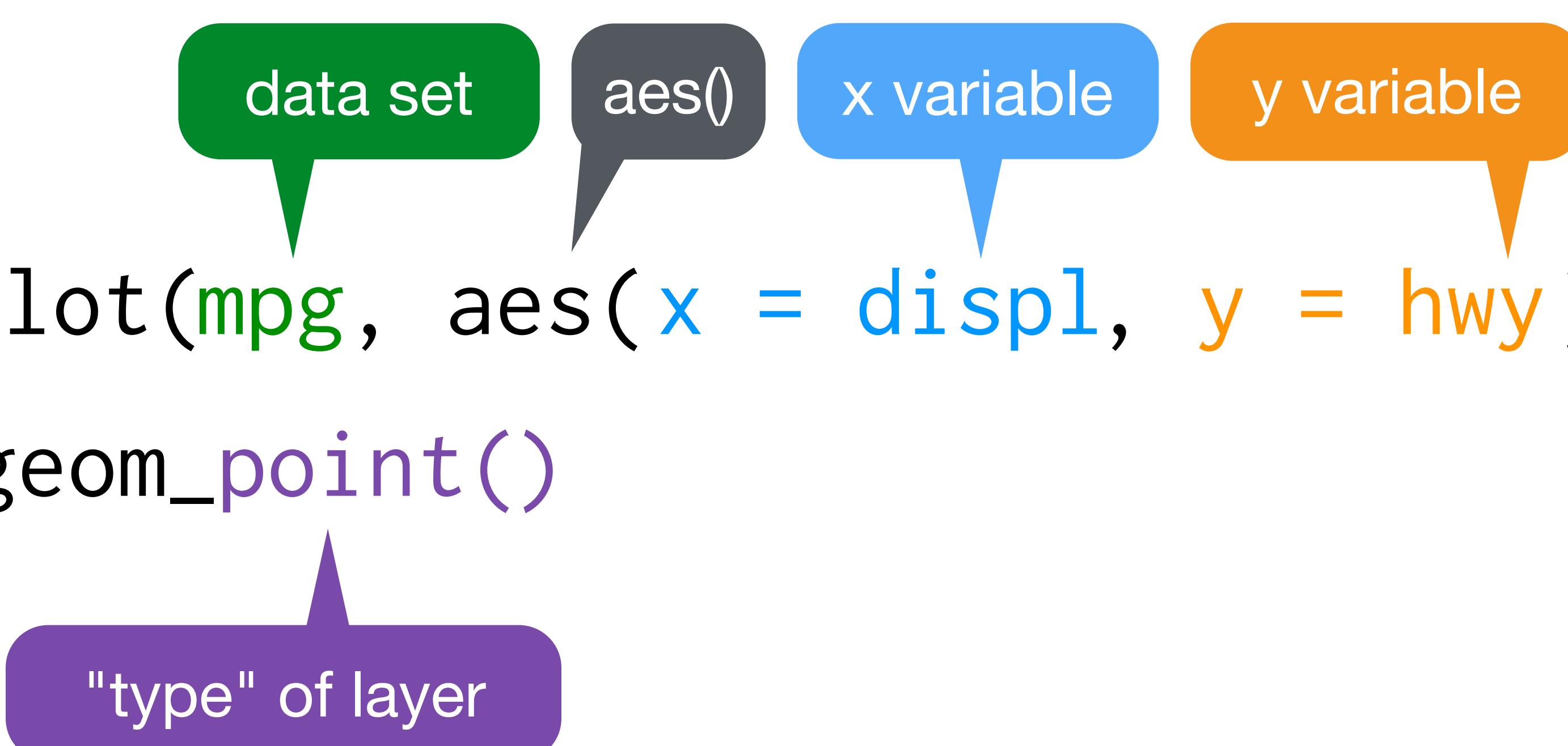


```
ggplot(mpg, aes(displ, hwy))
```

ggplot

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

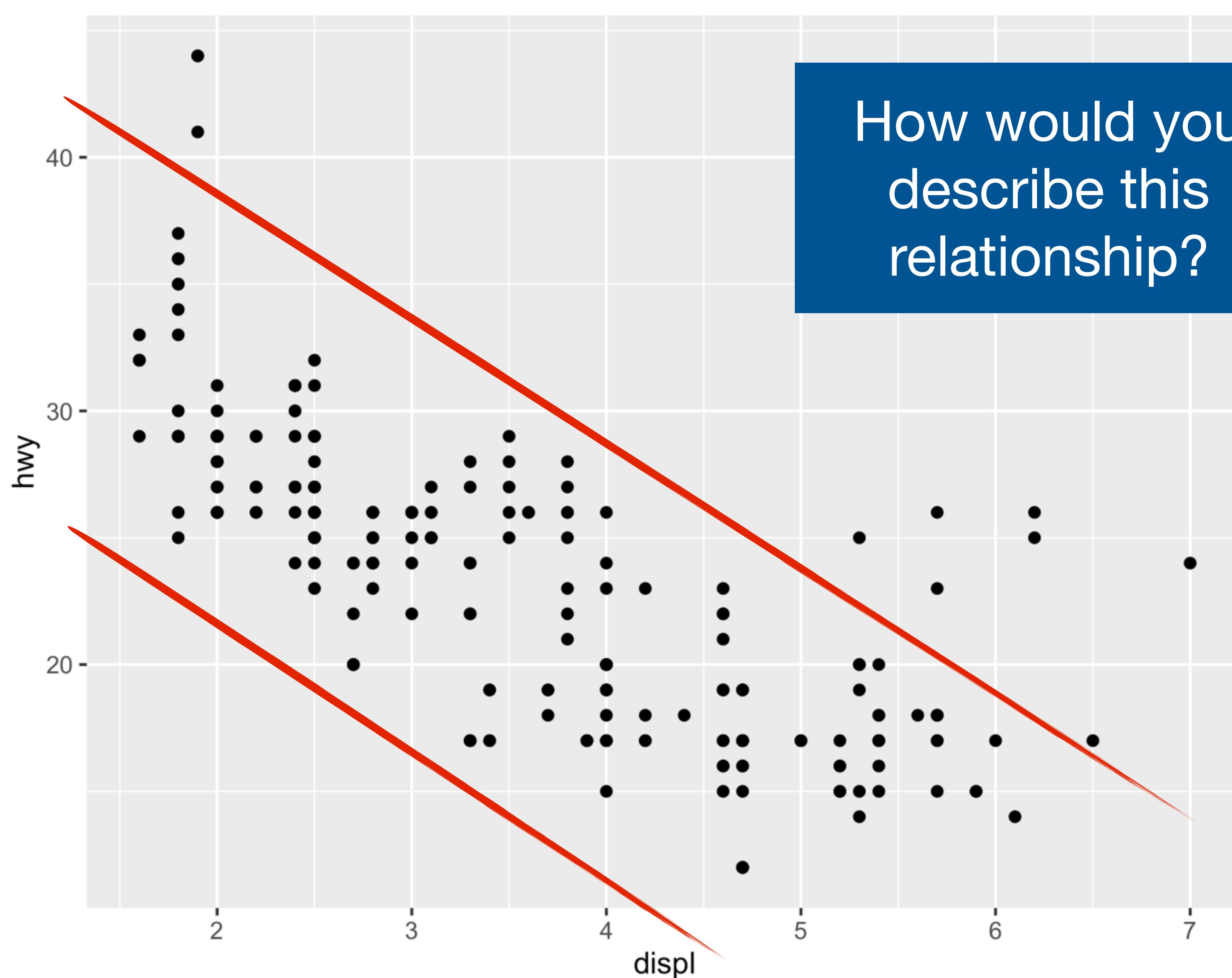
```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point()
```



data set aes() x variable y variable

"type" of layer

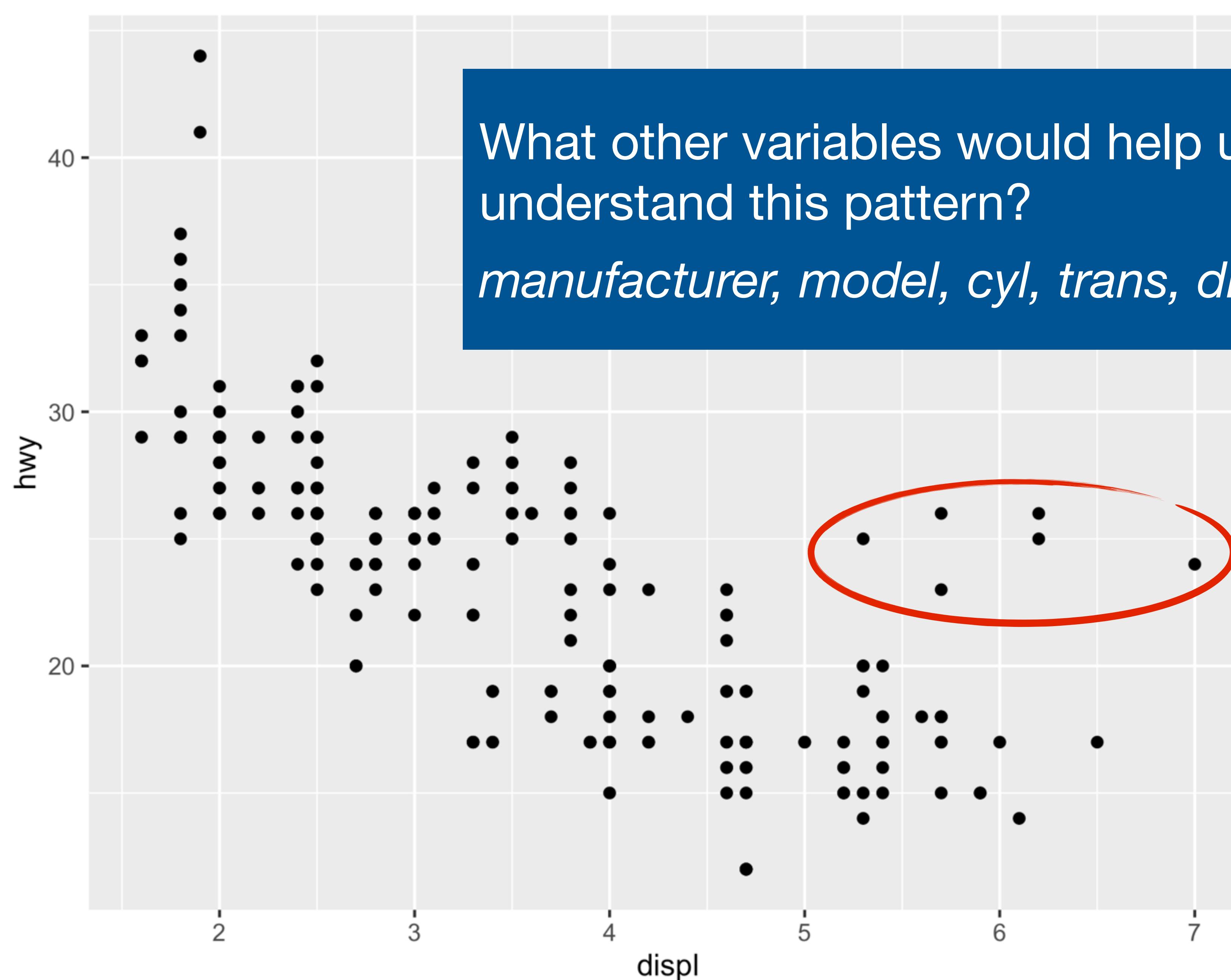
How would you
describe this
relationship?



```
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

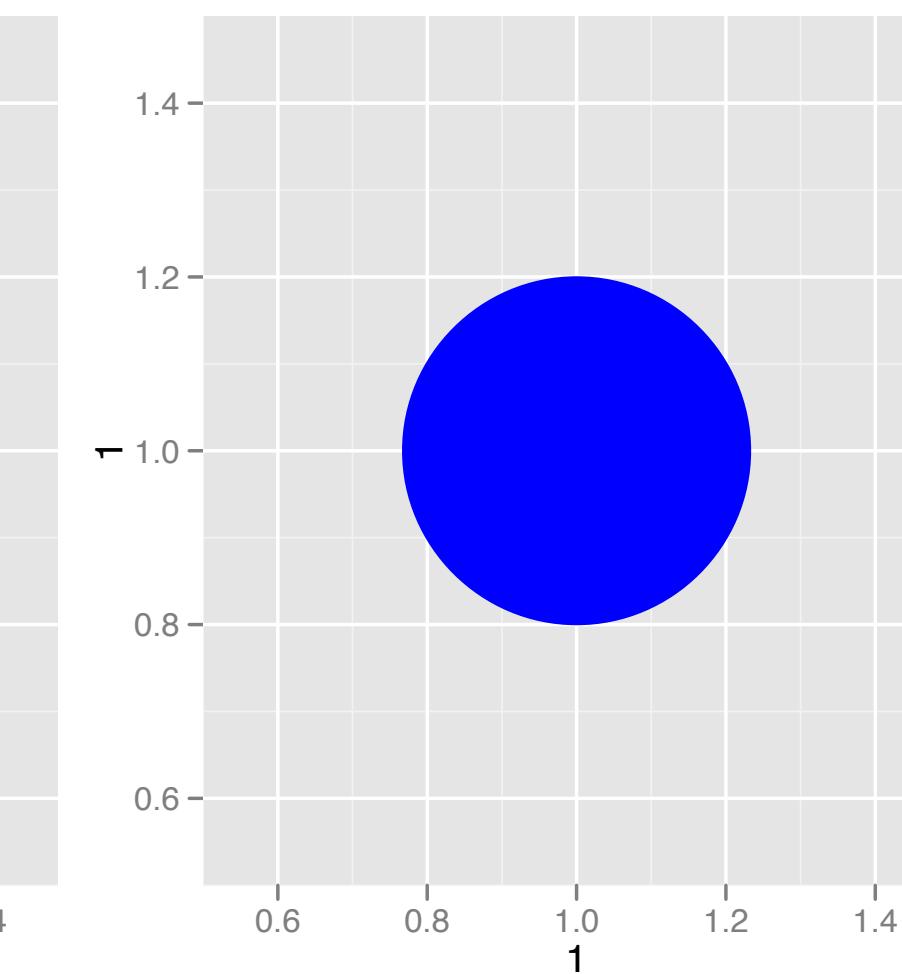
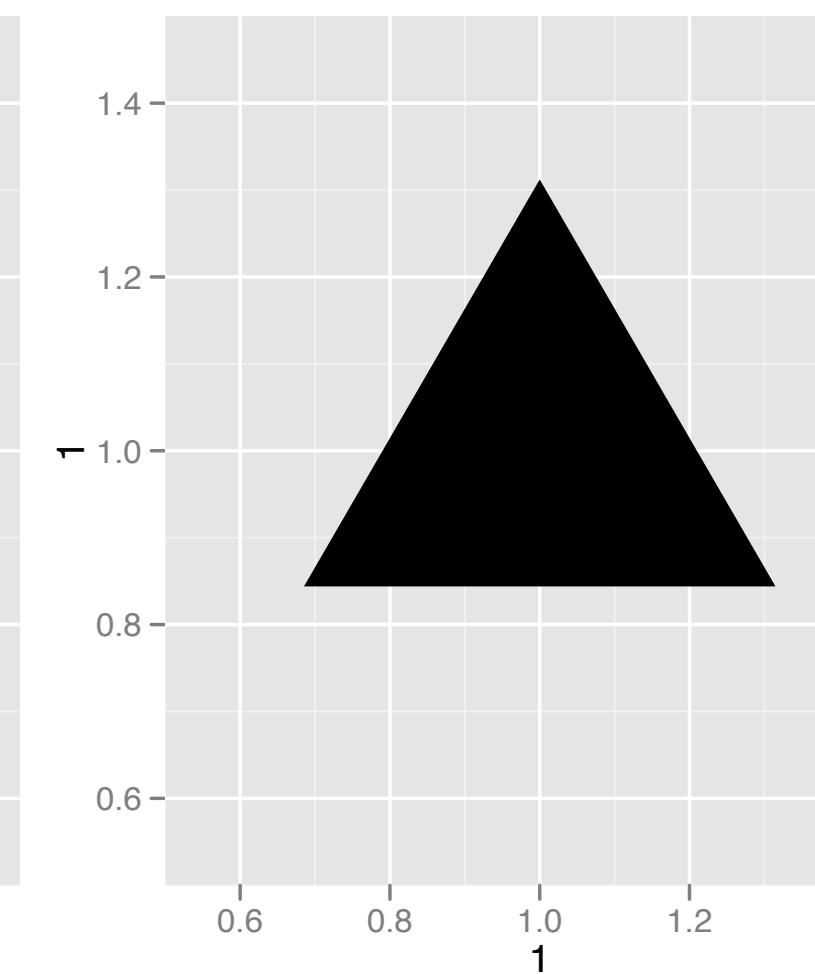
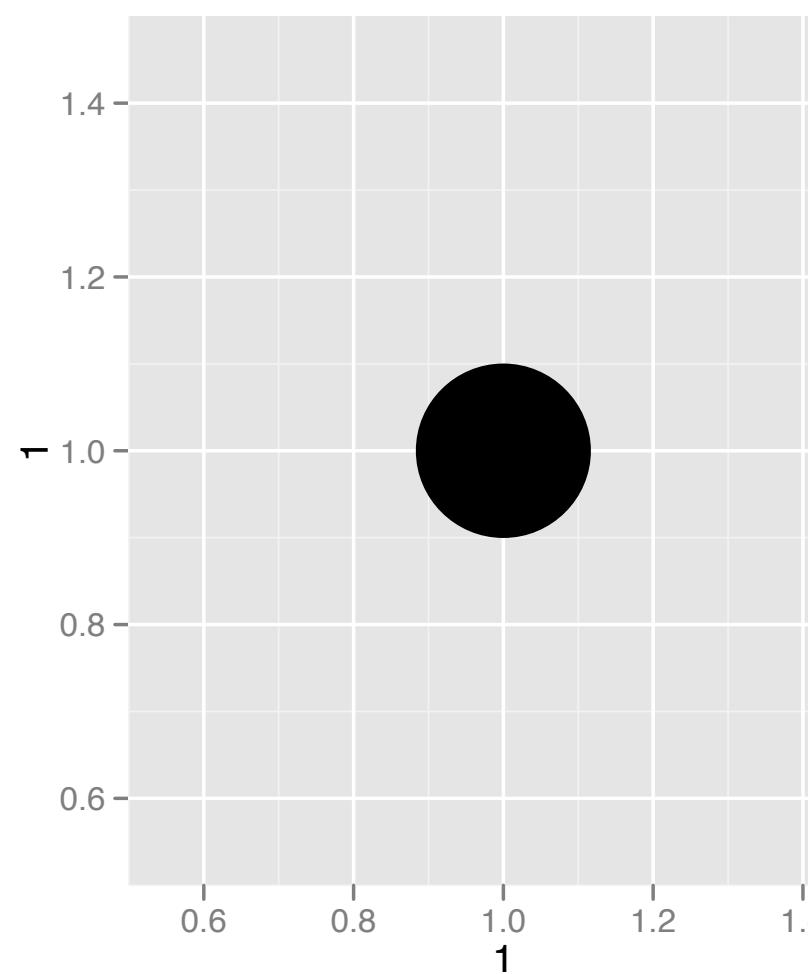
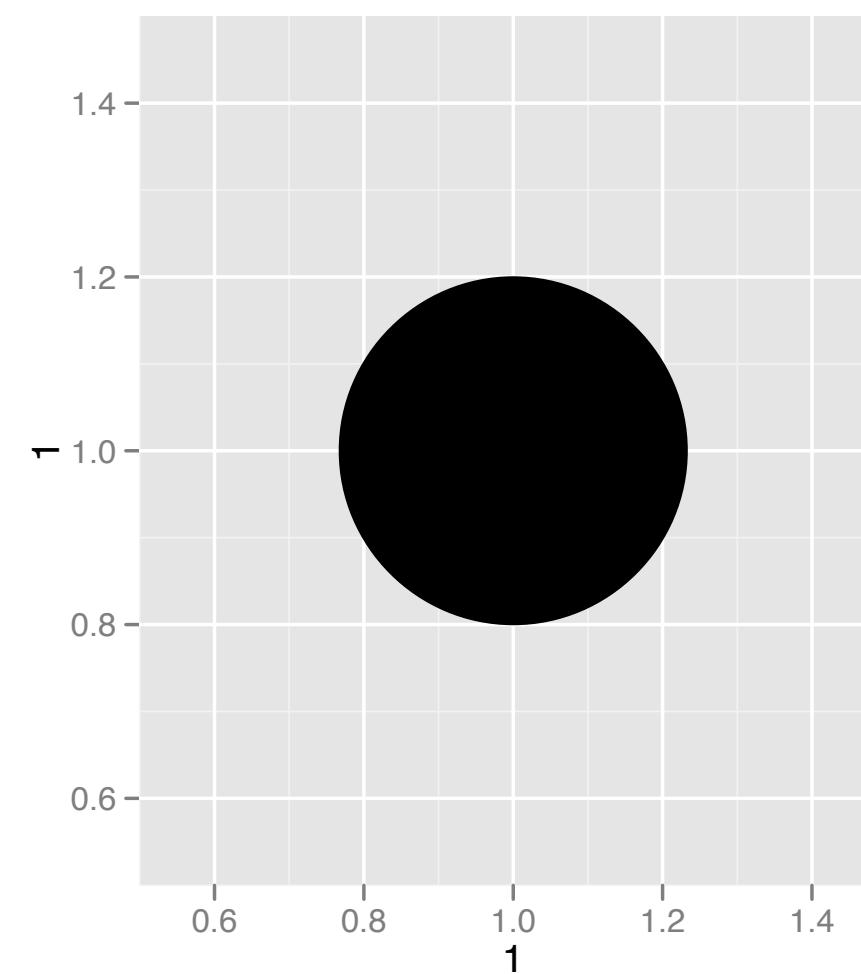


```
ggplot(mpg, aes(displ, hwy)) + geom_point()
```

© 2016 RStudio, Inc. All rights reserved.

Aesthetics

Aesthetics

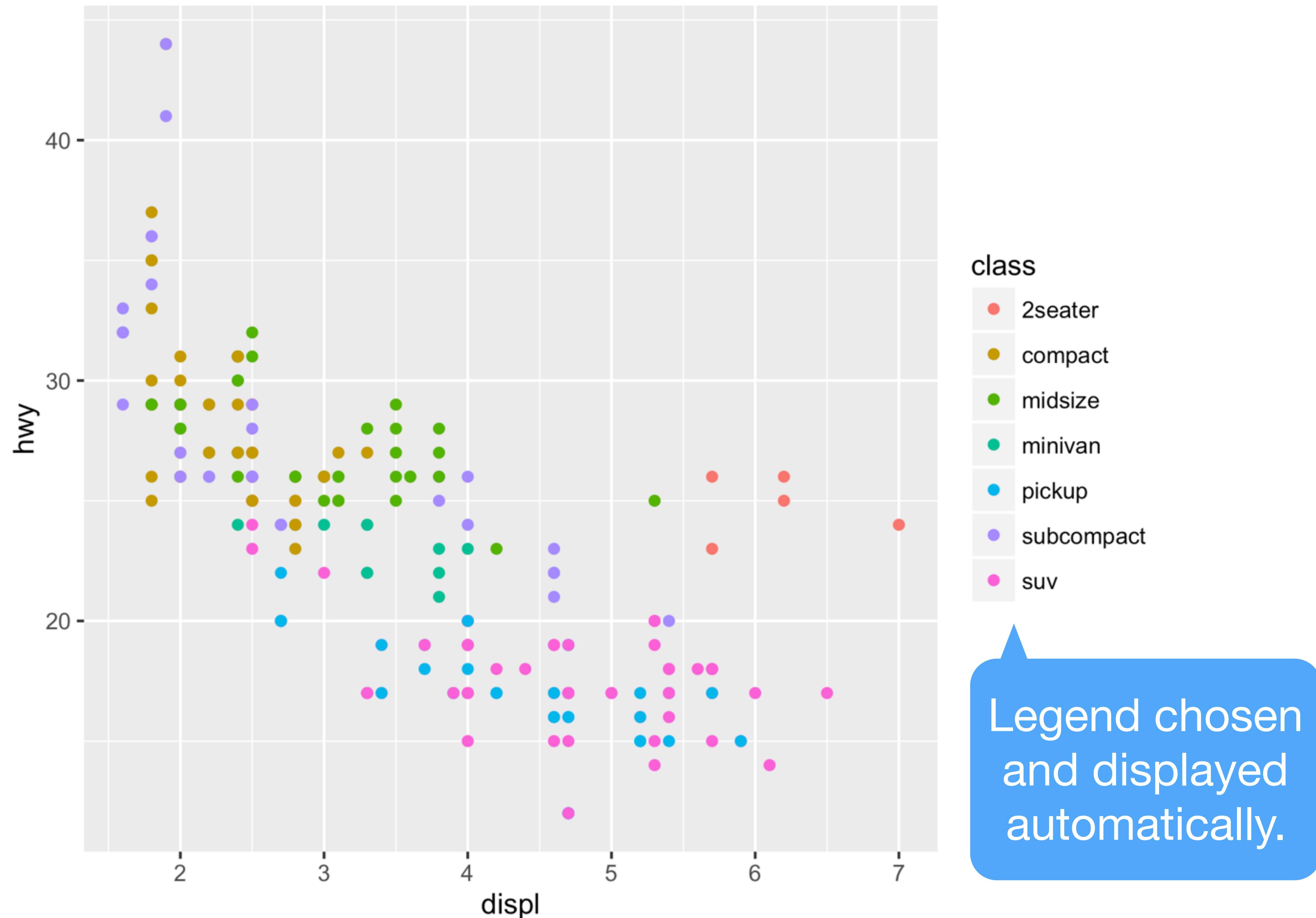


Aesthetics

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))
```



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

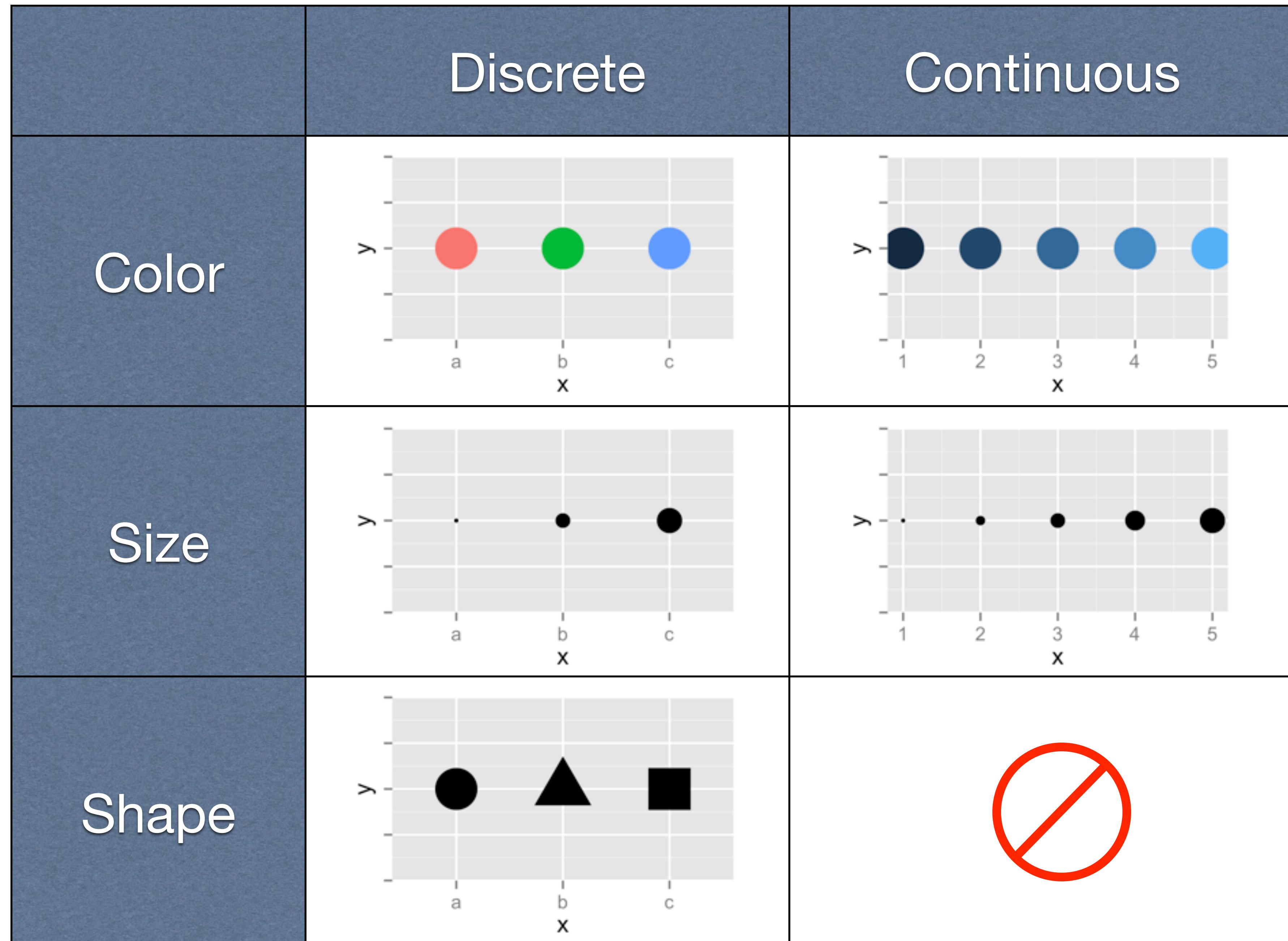
Your Turn

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?





Mapping vs. setting

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

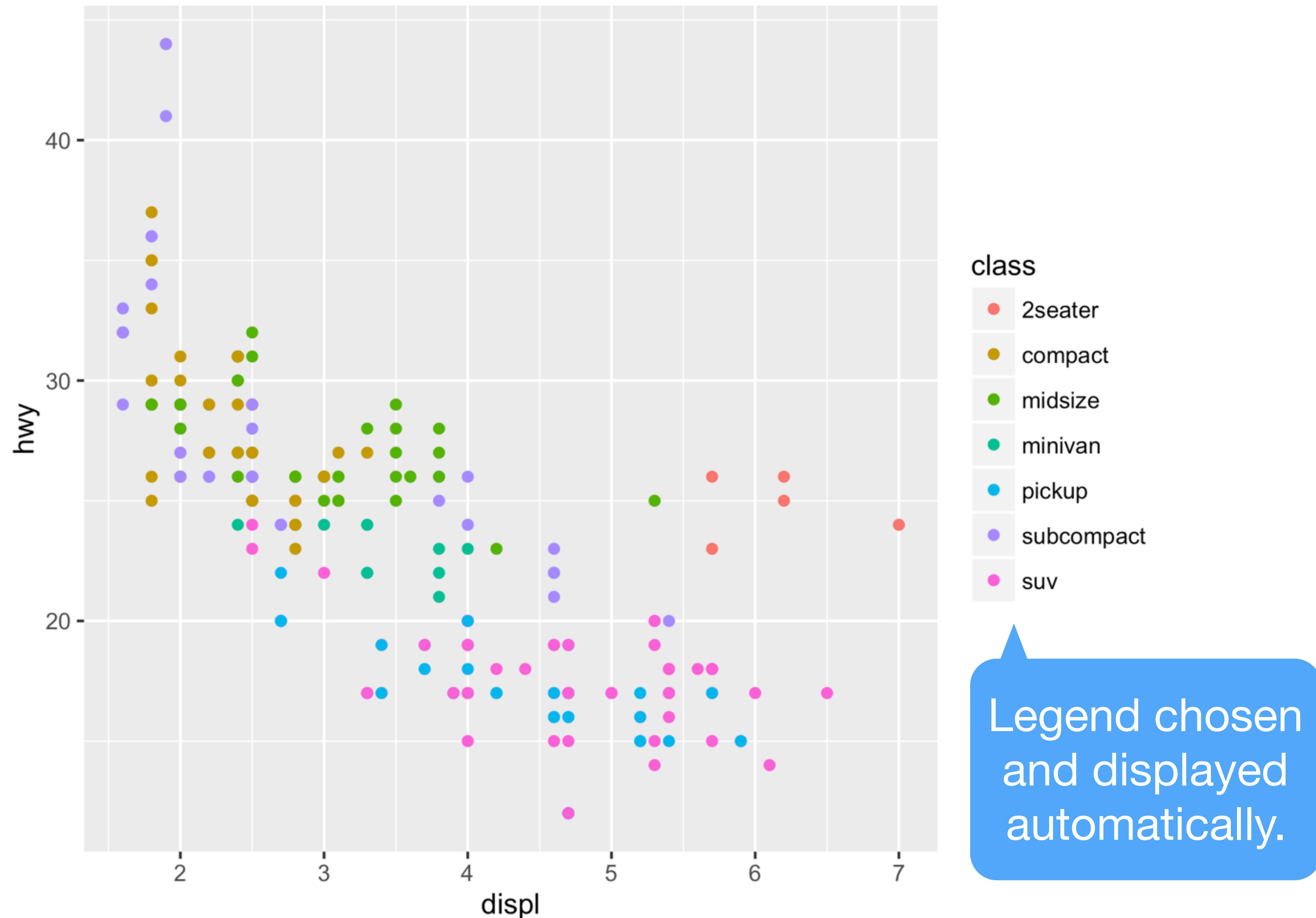
```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(mapping = aes(color = class))
```

Manually set an aesthetic without `aes()`

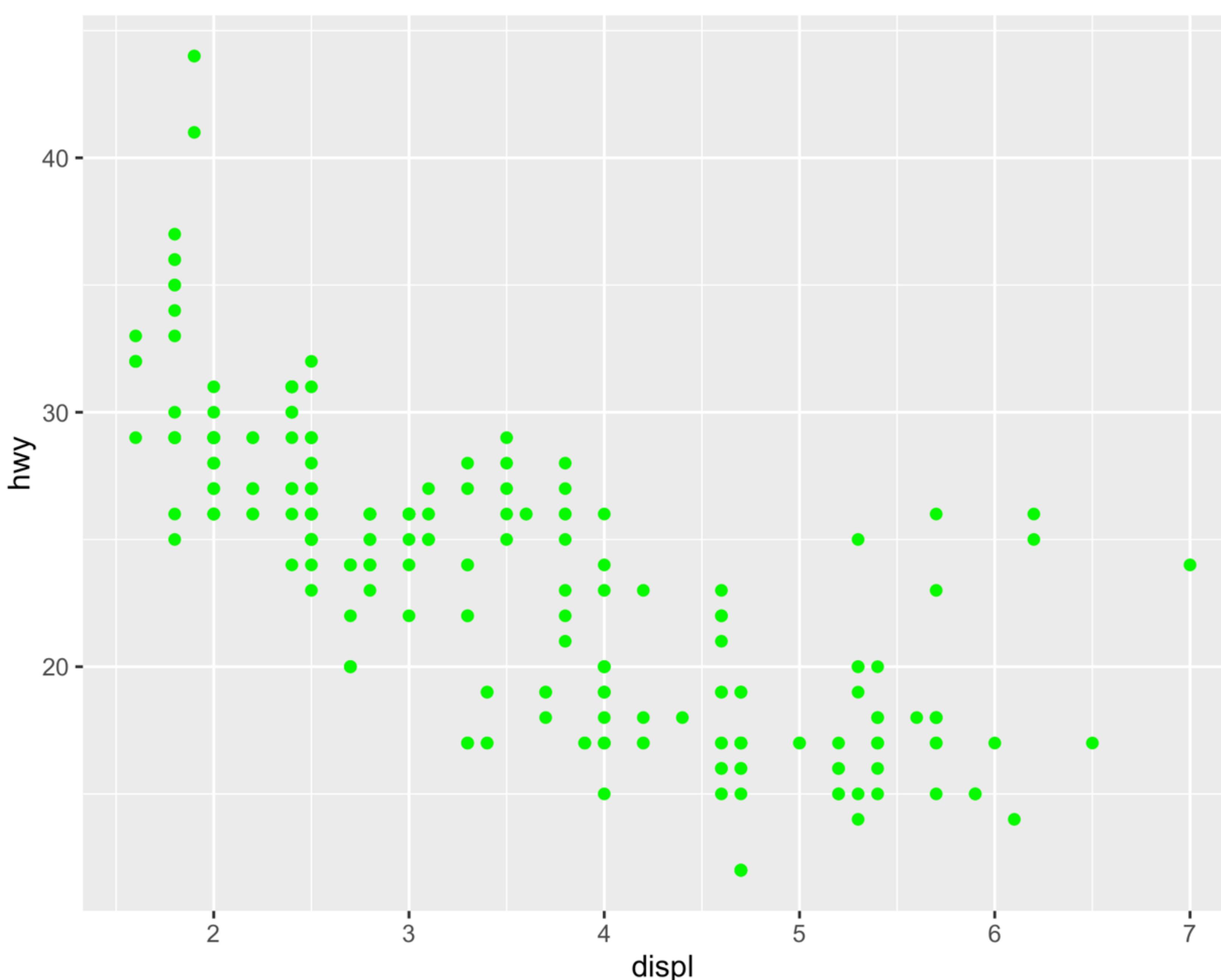
```
ggplot(mpg, aes(displ, hwy)) + geom_point(color = "green")  
ggplot(mpg, aes(displ, hwy)) + geom_point(size = 5)  
ggplot(mpg, aes(displ, hwy)) + geom_point(shape = 3)  
ggplot(mpg, aes(displ, hwy)) + geom_point(alpha = 0.5)
```

no `aes()`

actual values of
color/size/etc.



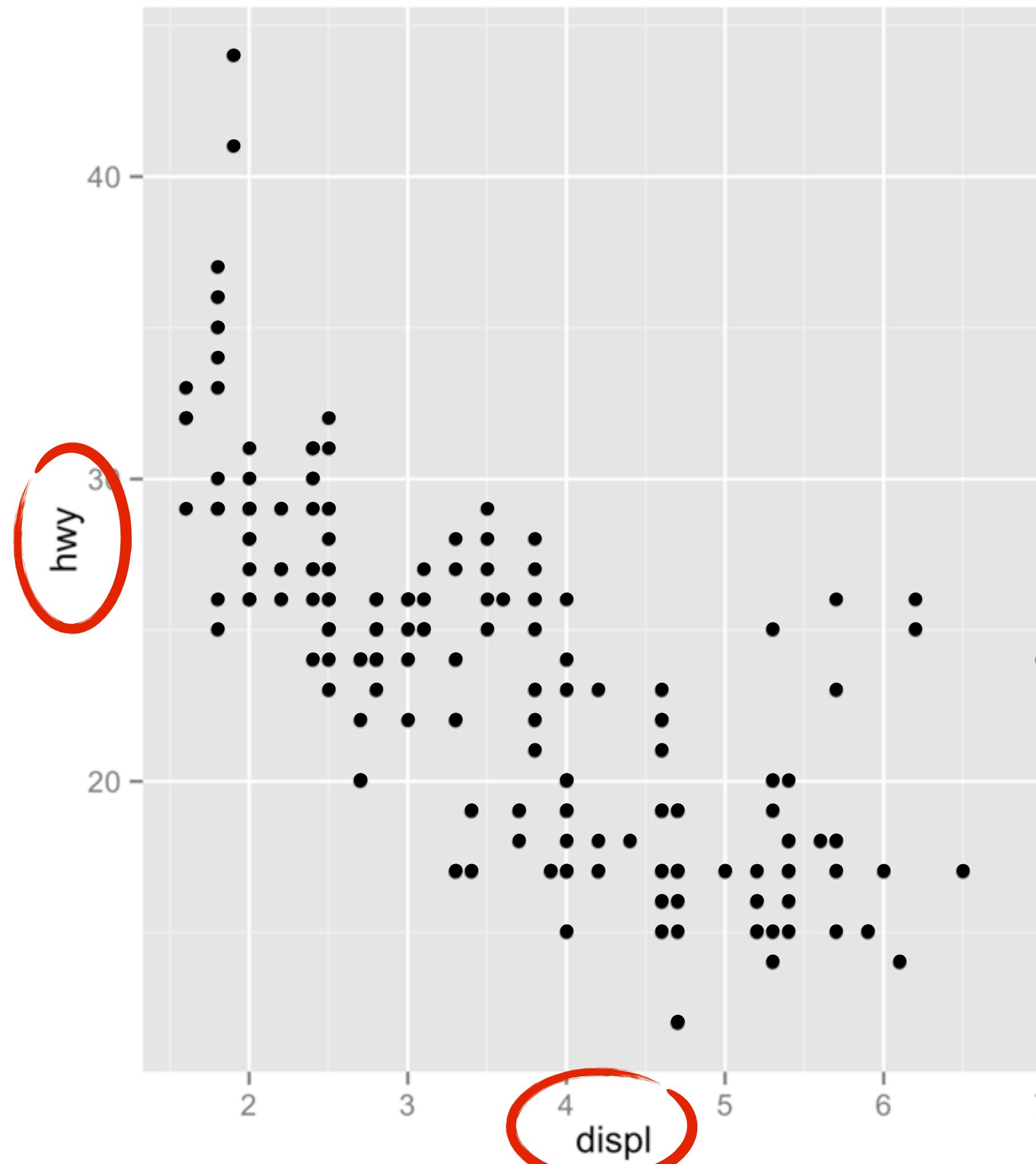
```
ggplot(mpg, aes(displ, hwy)) + geom_point(aes(color = class))
```



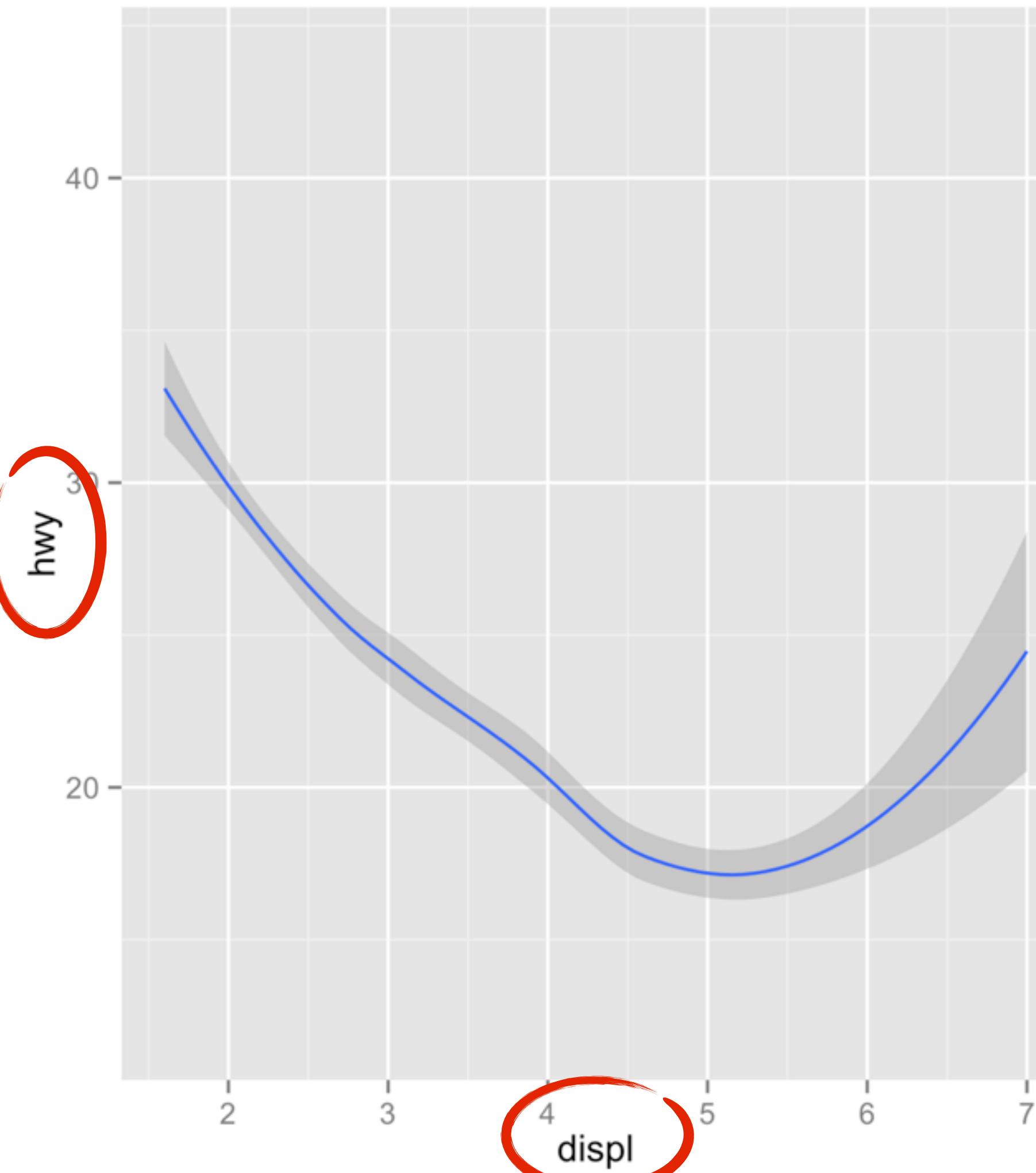
```
ggplot(mpg, aes(displ, hwy)) + geom_point(color = "green")
```

Geoms

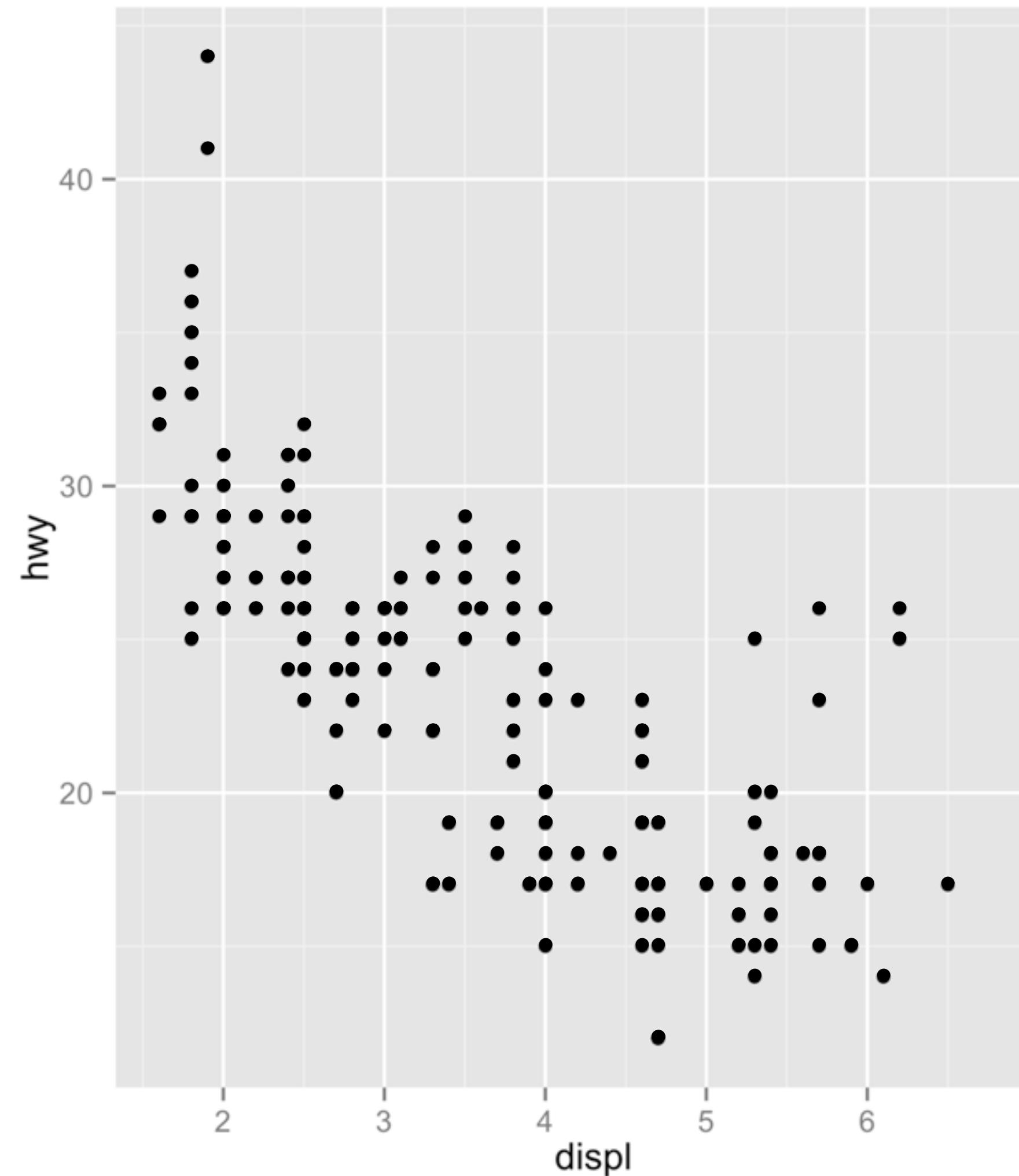
How are these plots similar?



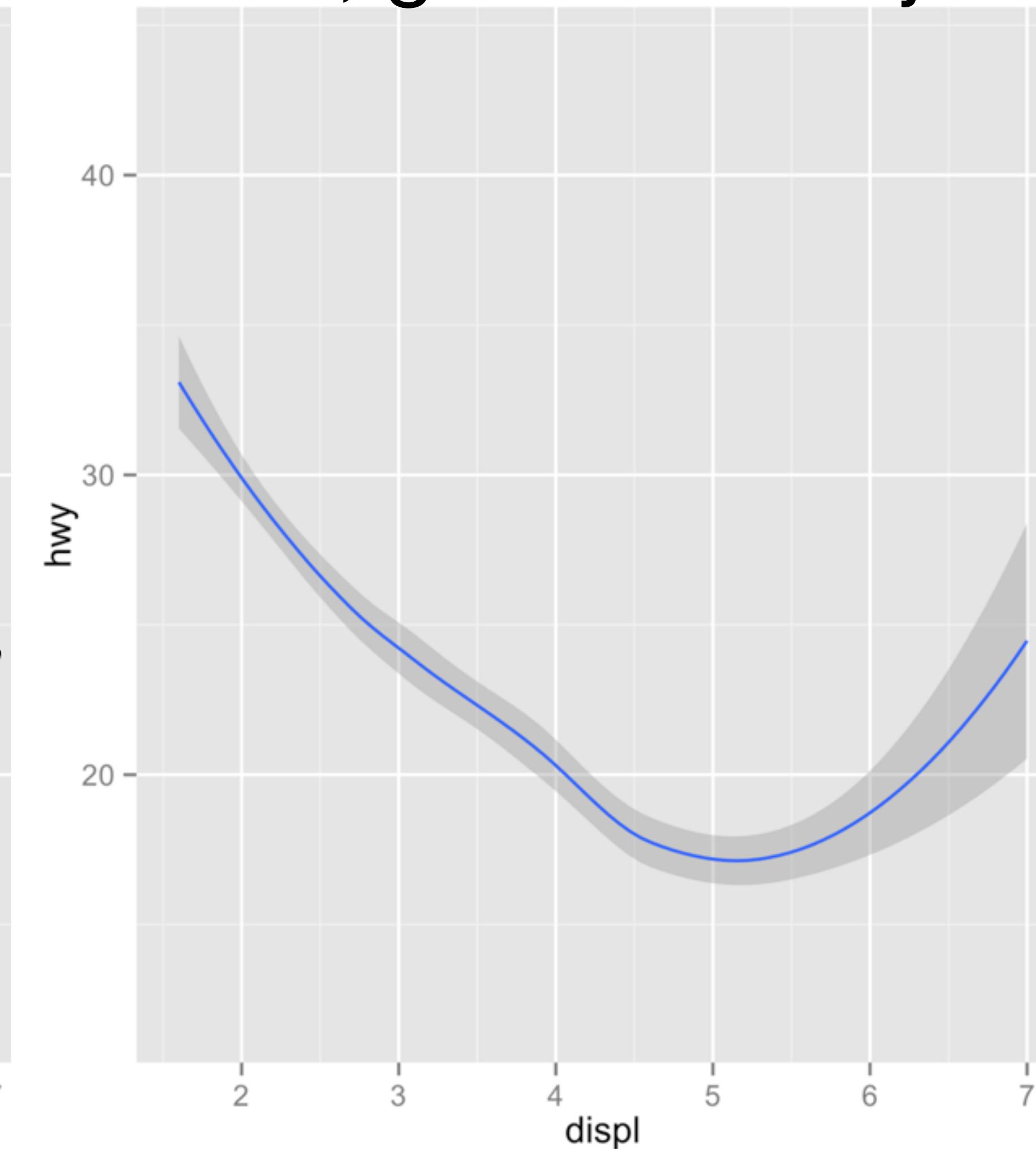
Same: x var, y var, data



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

geom_



geom_

Geometric object

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

`geom_smooth()`

`geom_`

`name of
geom`

Geometric object

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

geom_smooth(aes())

geom_

name of
geom

aes()

Geometric object

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

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

geom_

name of
geom

aes()

aesthetic
mappings

Geometric object

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

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

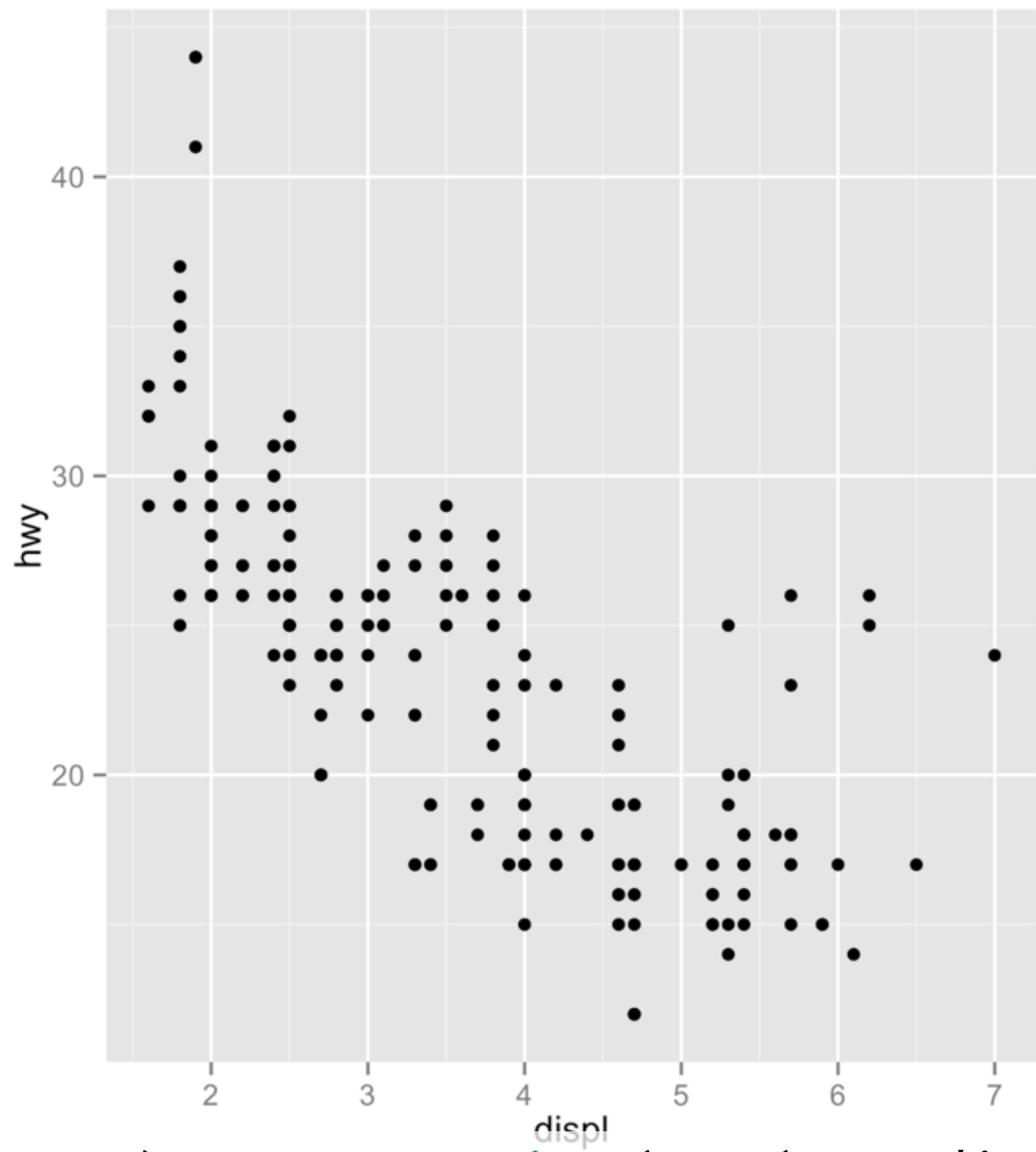
geom_

name of
geom

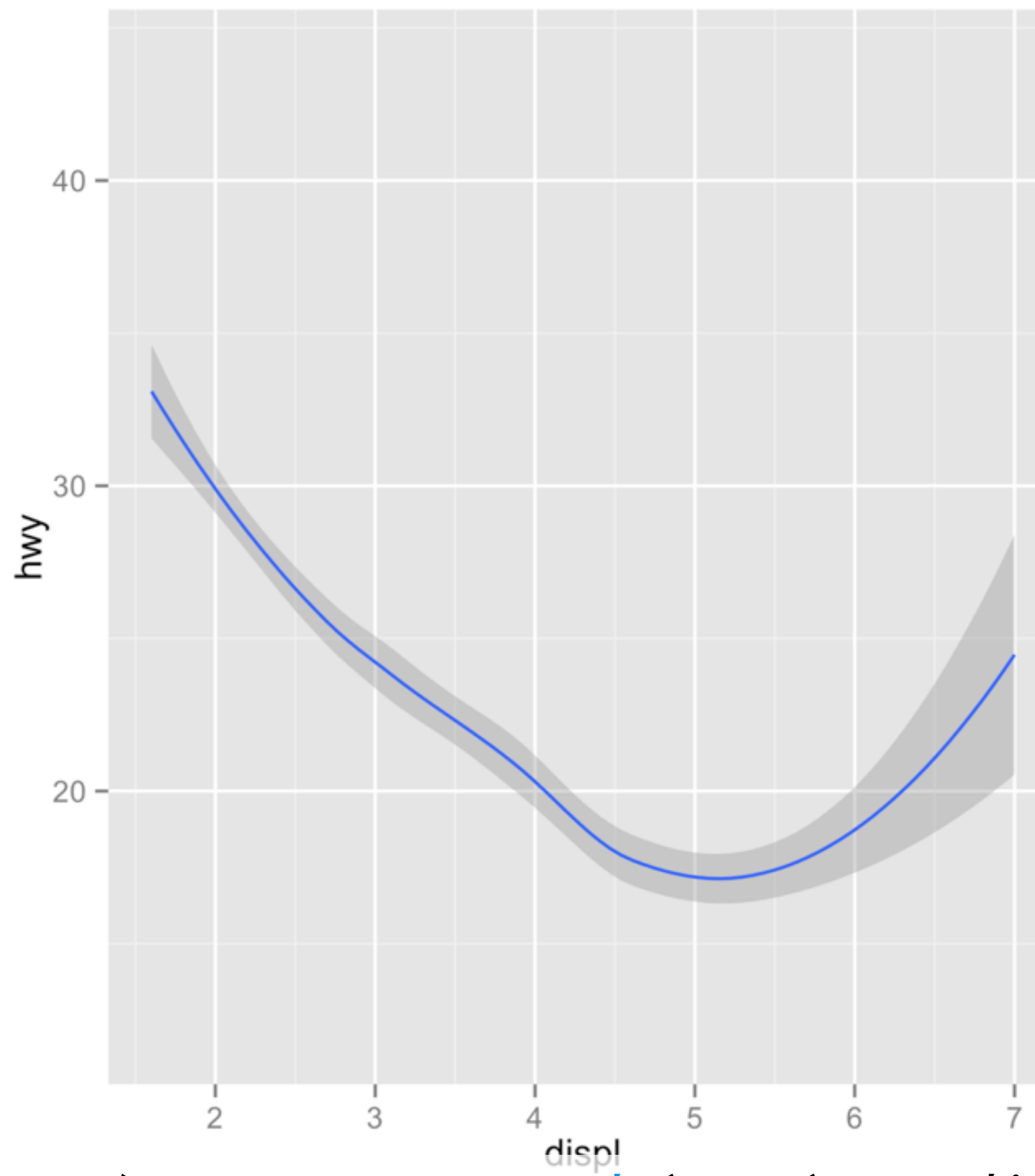
aes()

aesthetic
mappings

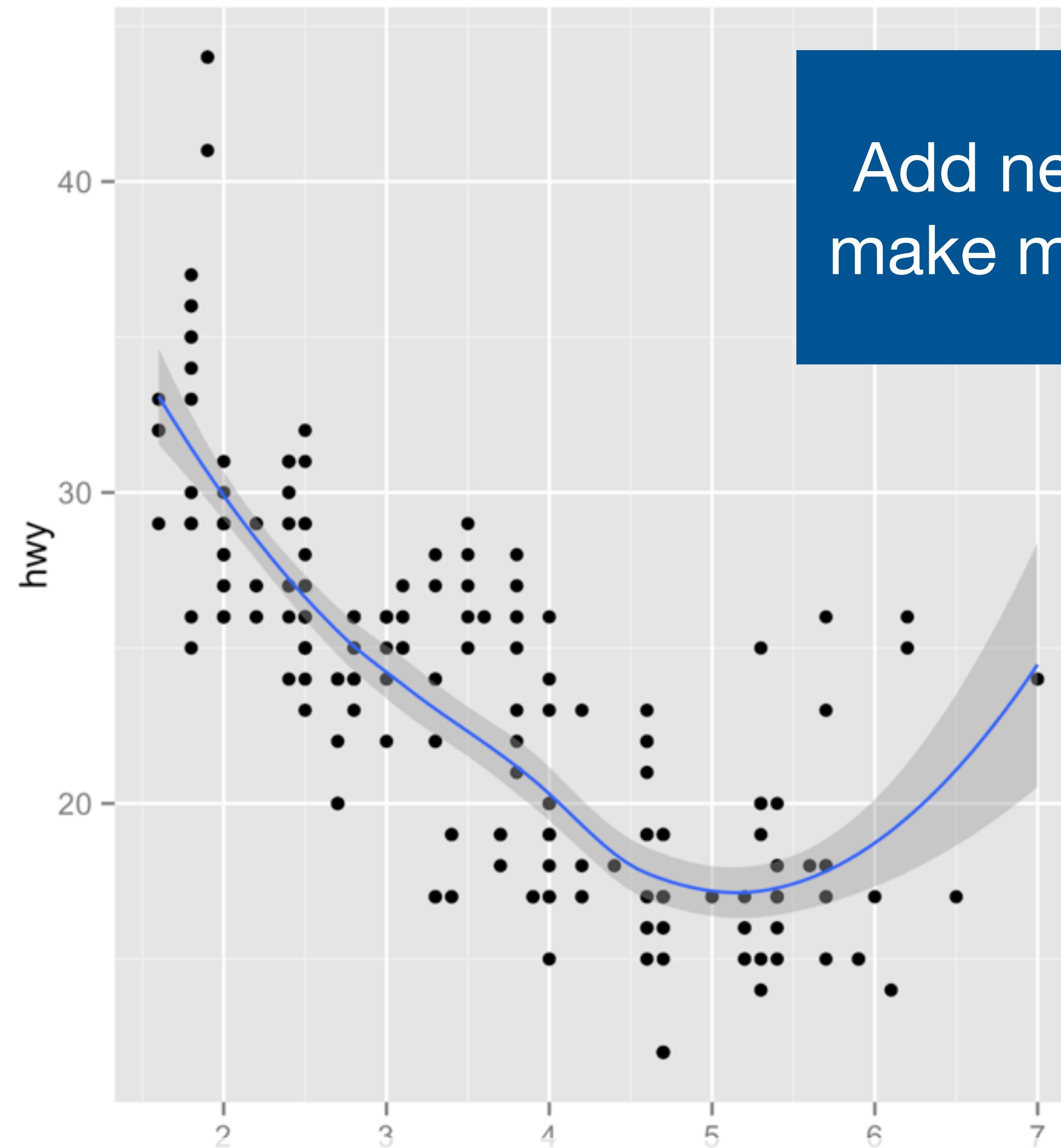
optional
arguments



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



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

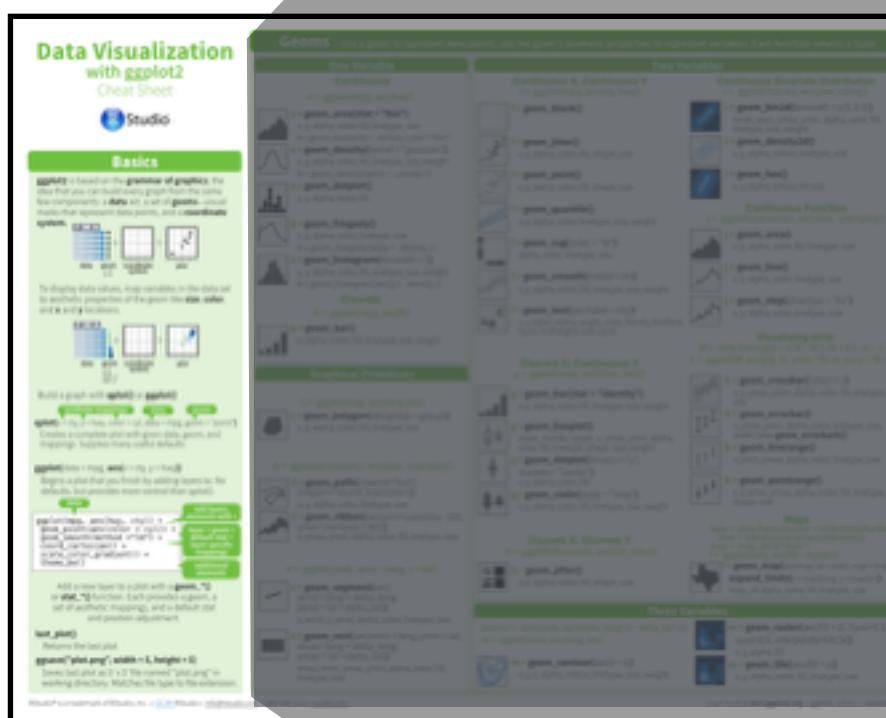


Add new geoms to
make multiple layers

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

Geoms

Help ►
 Cheatsheets ►
 Data Visualization with ggplot2

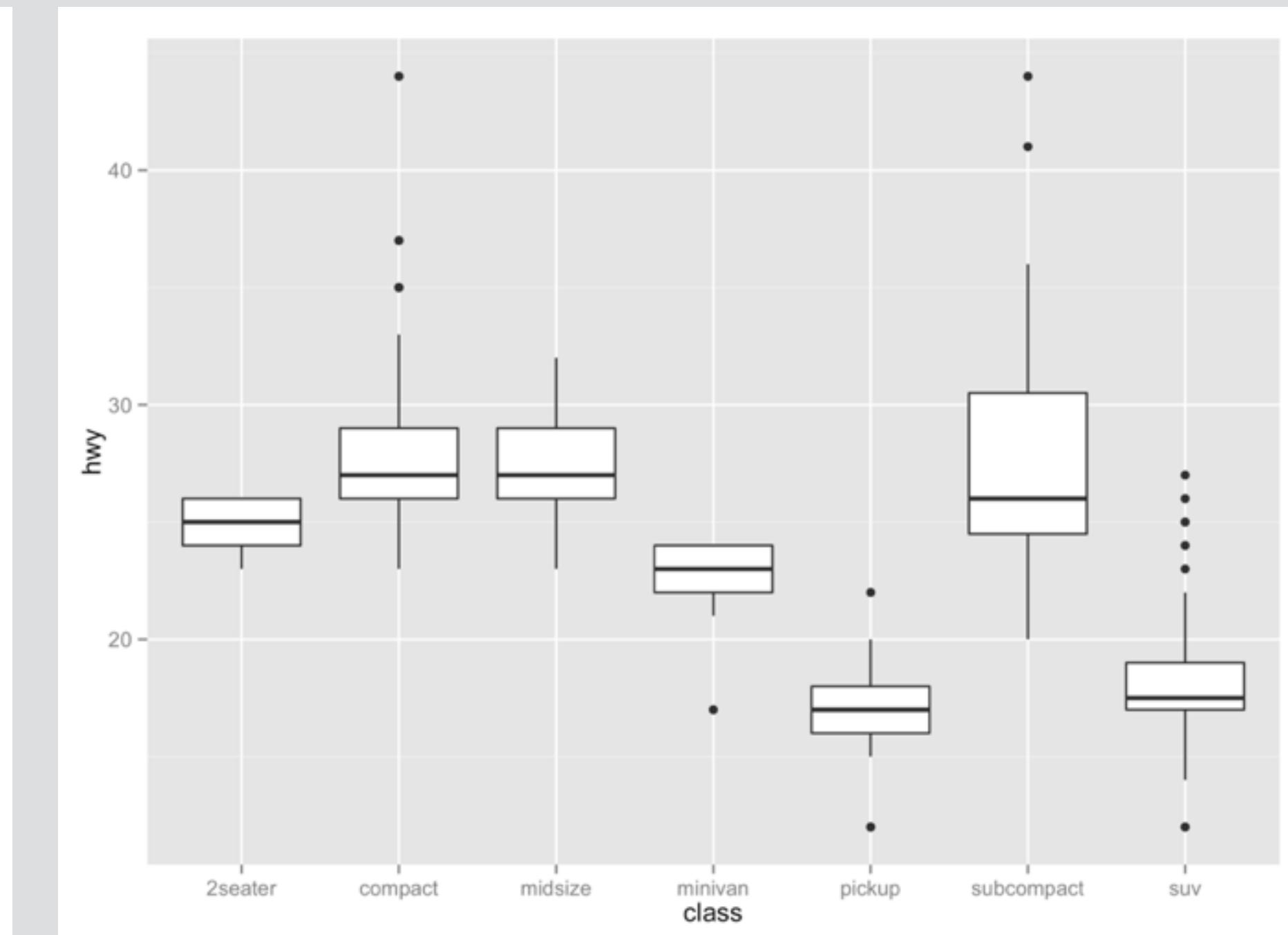
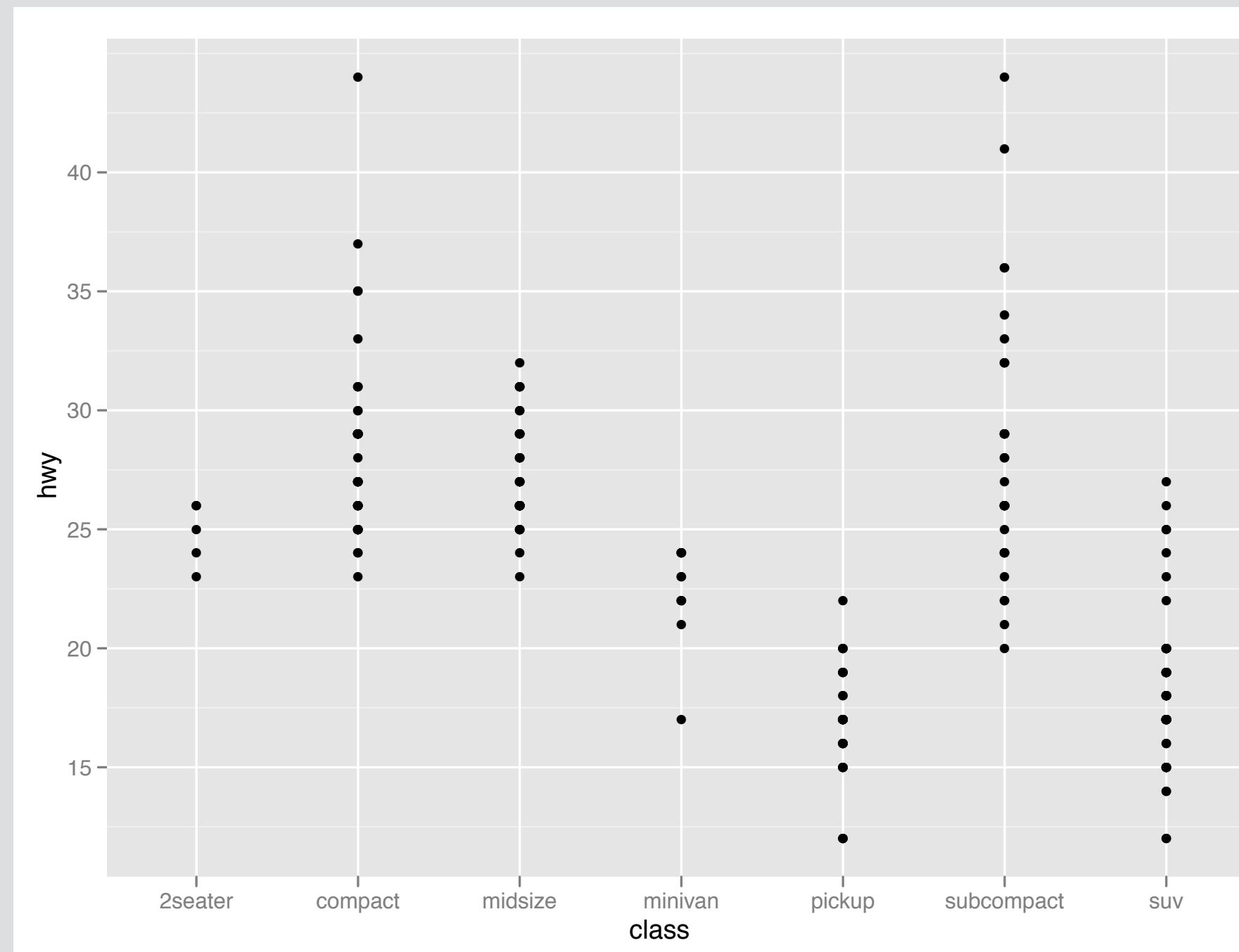


Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable		Two Variables		Three Variables	
Continuous		Continuous X, Continuous Y		Continuous Bivariate Distribution	
a + <code>geom_area(stat = "bin")</code> 	<code>a + geom_density(kernel = "gaussian")</code> 	f + <code>geom_blank()</code> (Useful for expanding limits) 	i + <code>geom_hex()</code> 	j + <code>geom_area()</code> 	m + <code>geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)</code>
b + <code>geom_dotplot()</code> 	c + <code>geom_freqpoly()</code> 	f + <code>geom_jitter()</code> 	f + <code>geom_point()</code> 	j + <code>geom_line()</code> 	m + <code>geom_contour(aes(z = z))</code>
d + <code>geom_histogram(binwidth = 5)</code> 	e + <code>geom_text(aes(label = cty))</code> 	f + <code>geom_quantile()</code> 	g + <code>geom_rug(sides = "bl")</code> 	j + <code>geom_step(direction = "hv")</code> 	m + <code>geom_tile(aes(fill = z))</code>
Discrete	Graphical Primitives	Discrete X, Continuous Y	Maps	Three Variables	
b + <code>geom_bar()</code> 	c + <code>geom_polygon(aes(group = group))</code> 	g + <code>geom_bar(stat = "identity")</code> 	l + <code>geom_map(aes(map_id = state), map = map) + expand_limits(x = map\$long, y = map\$lat)</code> 	seals\$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)) m <- ggplot(seals, aes(long, lat)) m + <code>geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)</code> 	
d + <code>geom_path(lineend="butt", linejoin="round", linemitre=1)</code> 	d + <code>geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900))</code> 	g + <code>geom_boxplot()</code> 	m + <code>geom_tile(aes(fill = z))</code> 		
e + <code>geom_segment(aes(xend = long + delta_long, yend = lat + delta_lat))</code> 	e + <code>geom_rect(aes(xmin = long, ymin = lat, xmax=long + delta_long, ymax=lat + delta_lat))</code> 	g + <code>geom_dotplot(binaxis = "y", stackdir = "center")</code> 	m + <code>geom_contour(aes(z = z))</code> 		
AB	C	Discrete X, Discrete Y	Maps	Three Variables	
AB	C	h + <code>geom_jitter()</code> 	l + <code>geom_map(aes(map_id = state), map = map) + expand_limits(x = map\$long, y = map\$lat)</code> 	m + <code>geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)</code> 	
		h + <code>geom_text(aes(label = cty))</code> 		m + <code>geom_hex()</code> 	

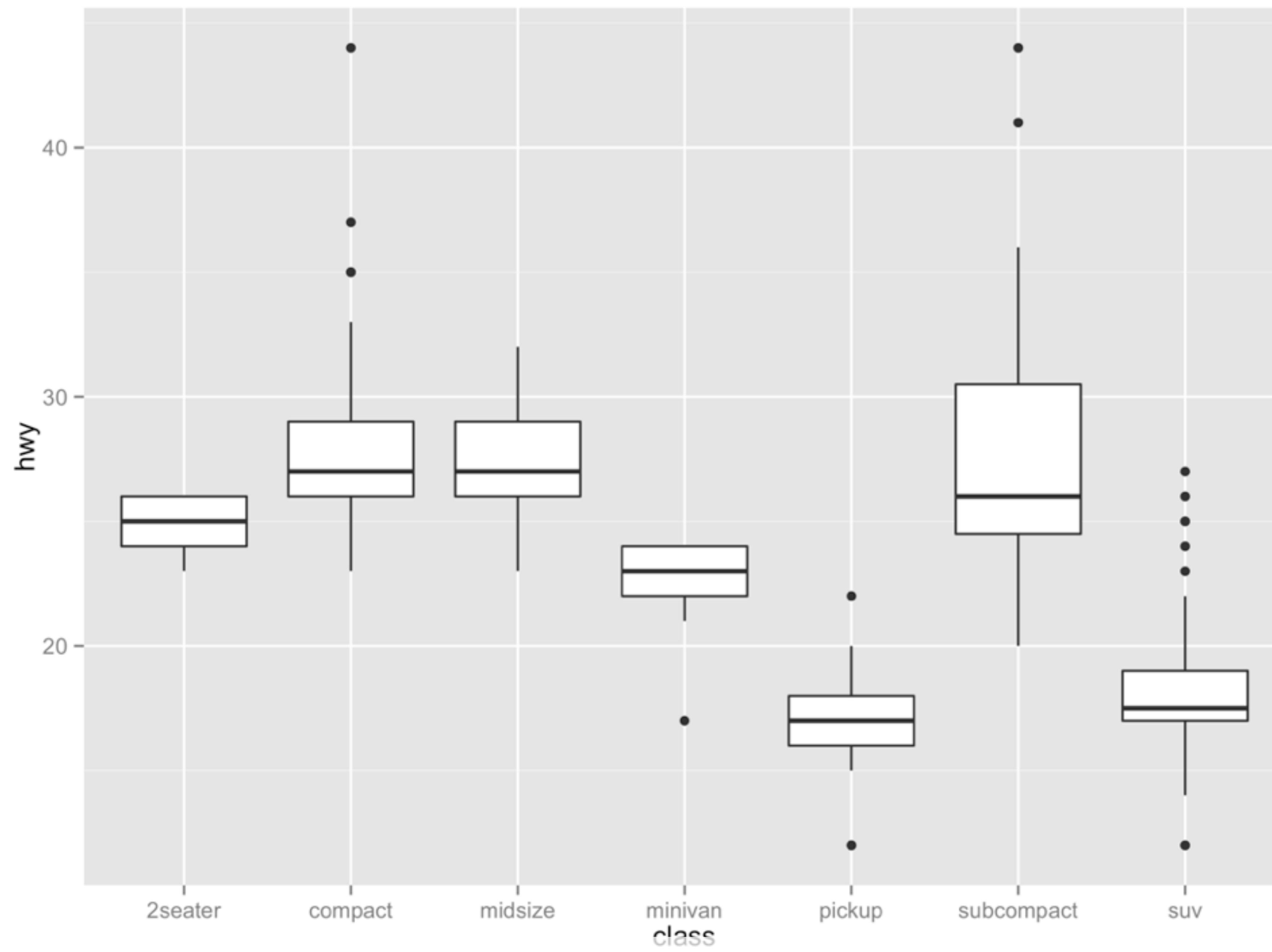
Your Turn

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



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



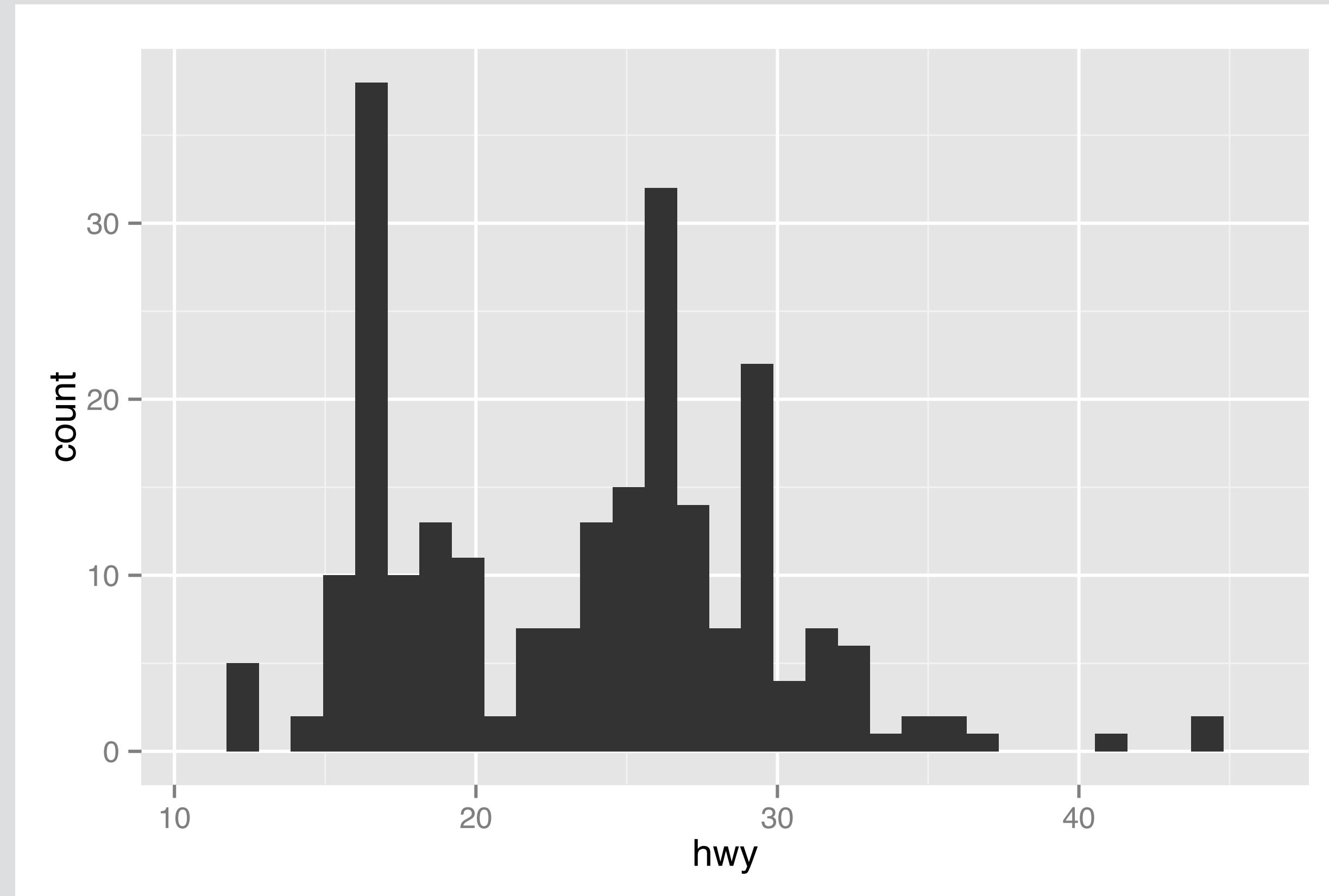


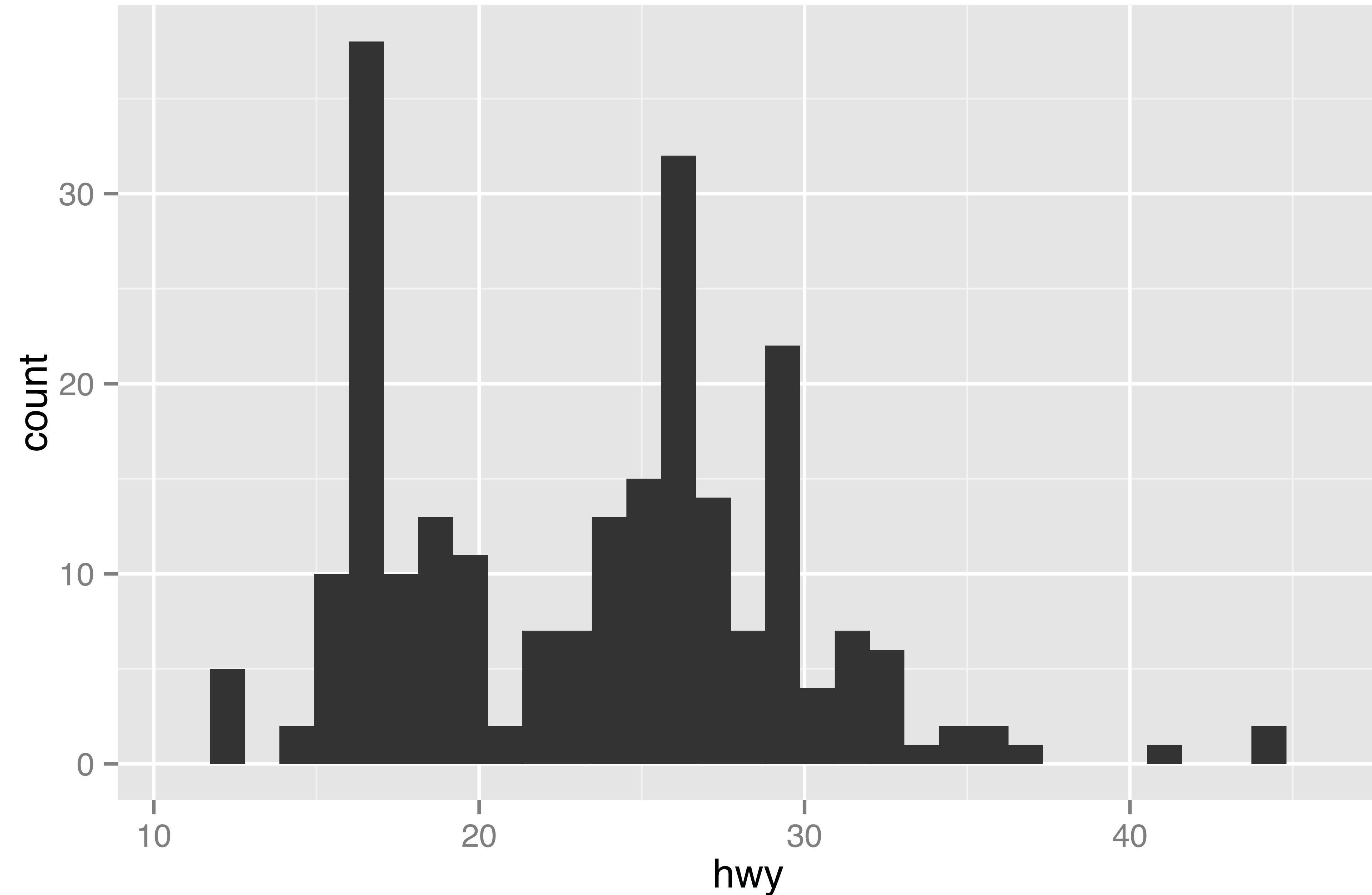
```
ggplot(data = mpg) + geom_boxplot(aes(x = displ, y = hwy))
```

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

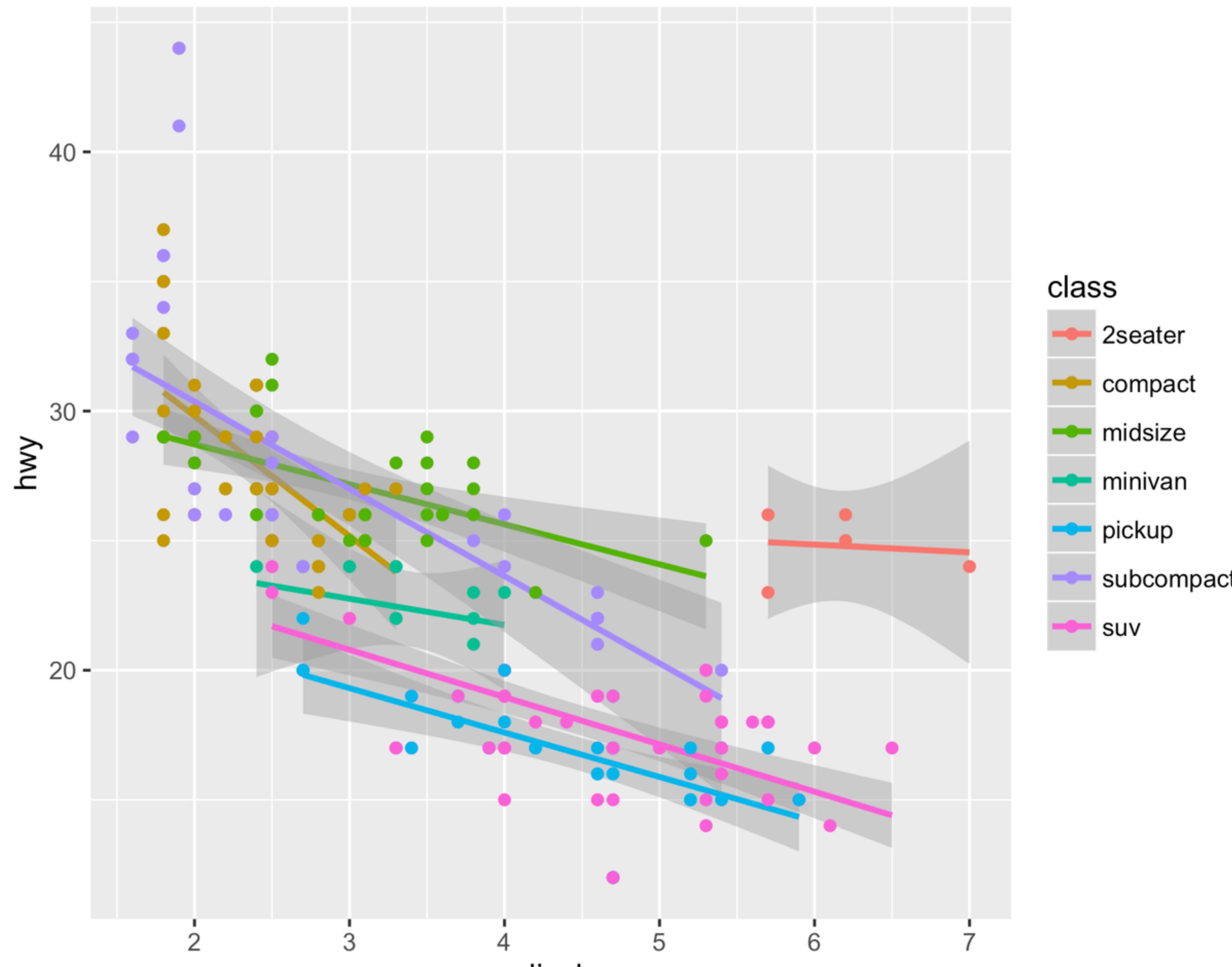
```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_smooth(method = lm) +  
  geom_point(aes(color = cyl), data = mpg[1:10, ])
```

Global vs. Local

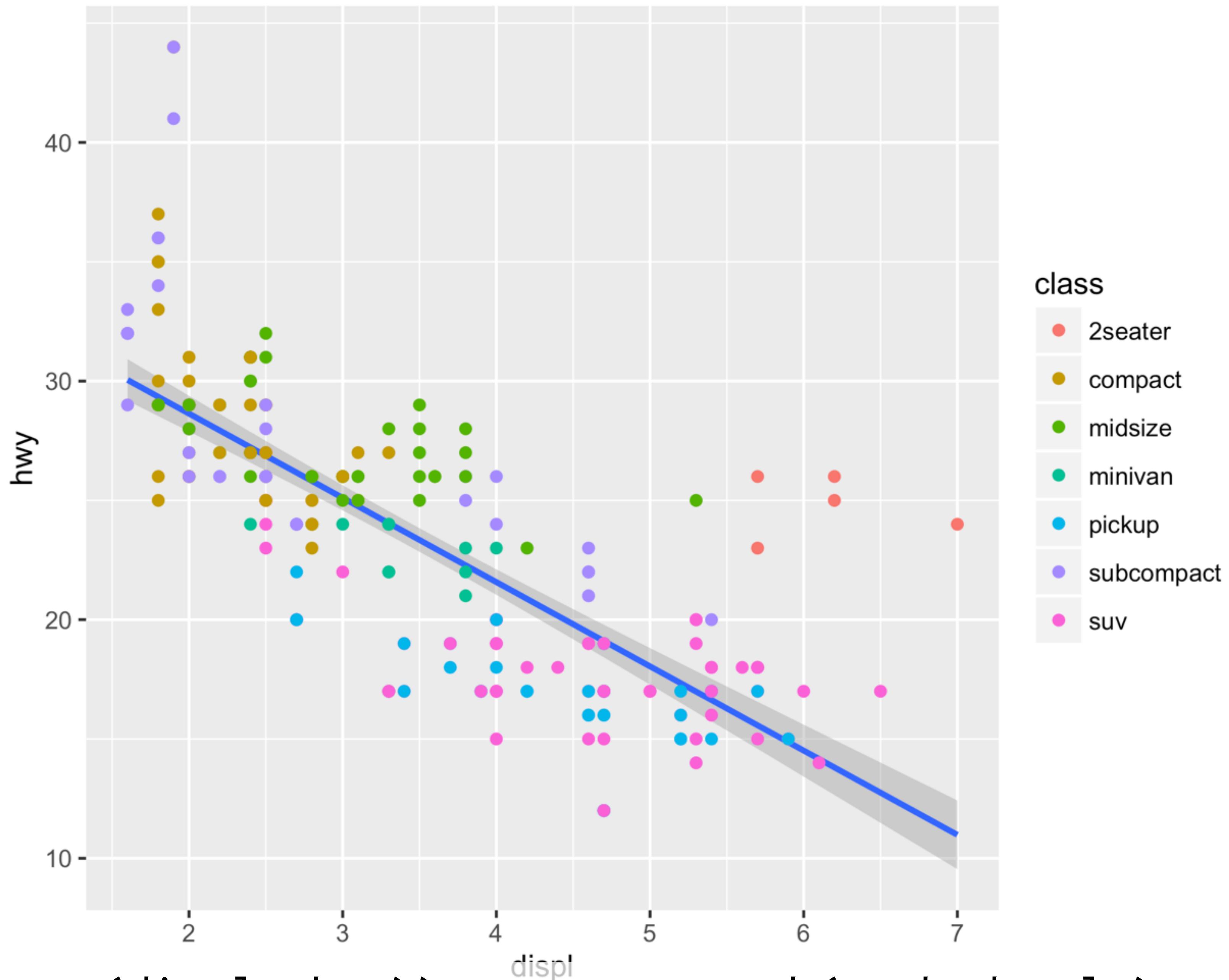
global aesthetics:
default for every layer

```
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()
```



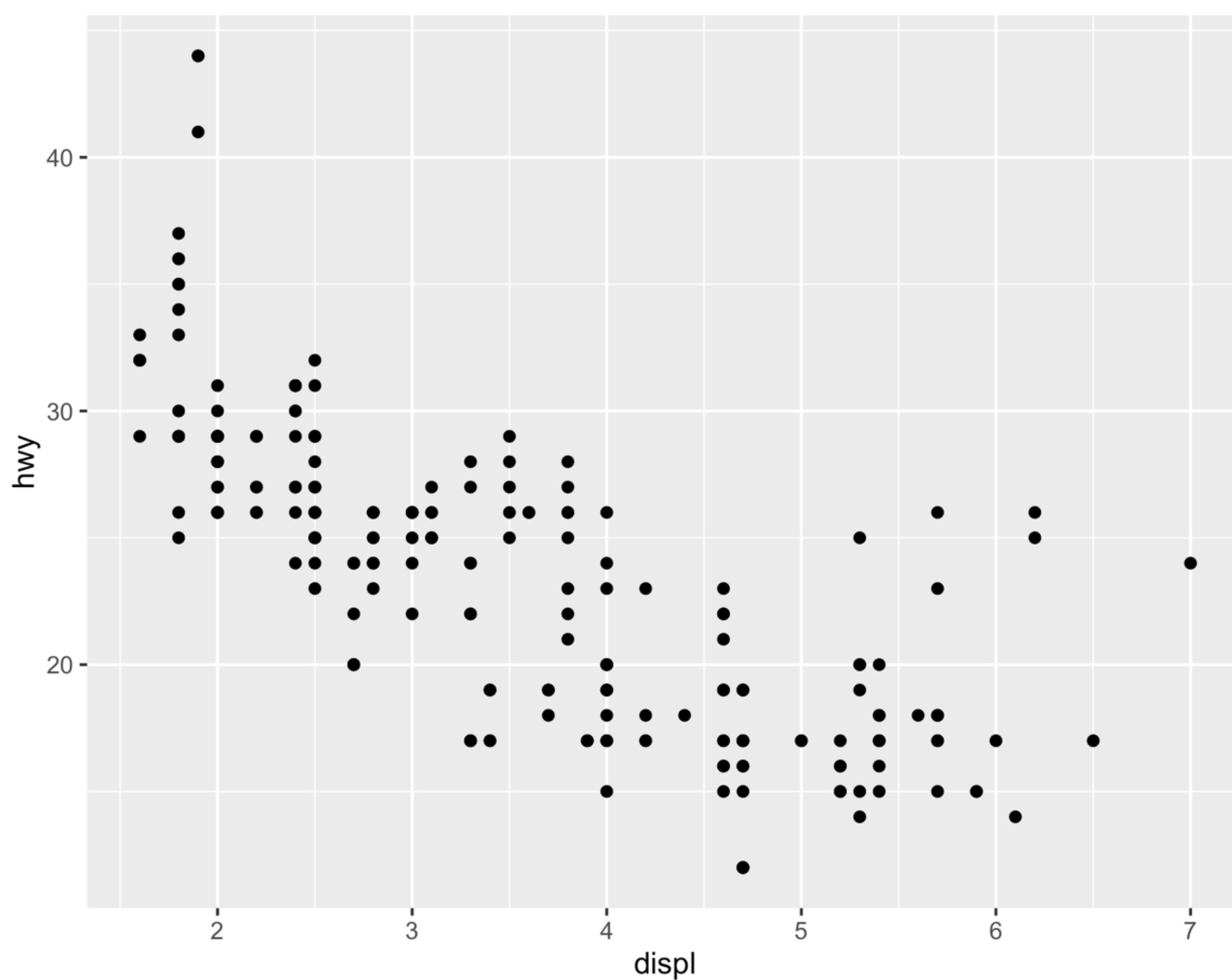
```
ggplot(mpg, aes(displ, hwy)) + geom_smooth(method = lm) +  
  geom_point(aes(color = class))
```

Global vs. Local

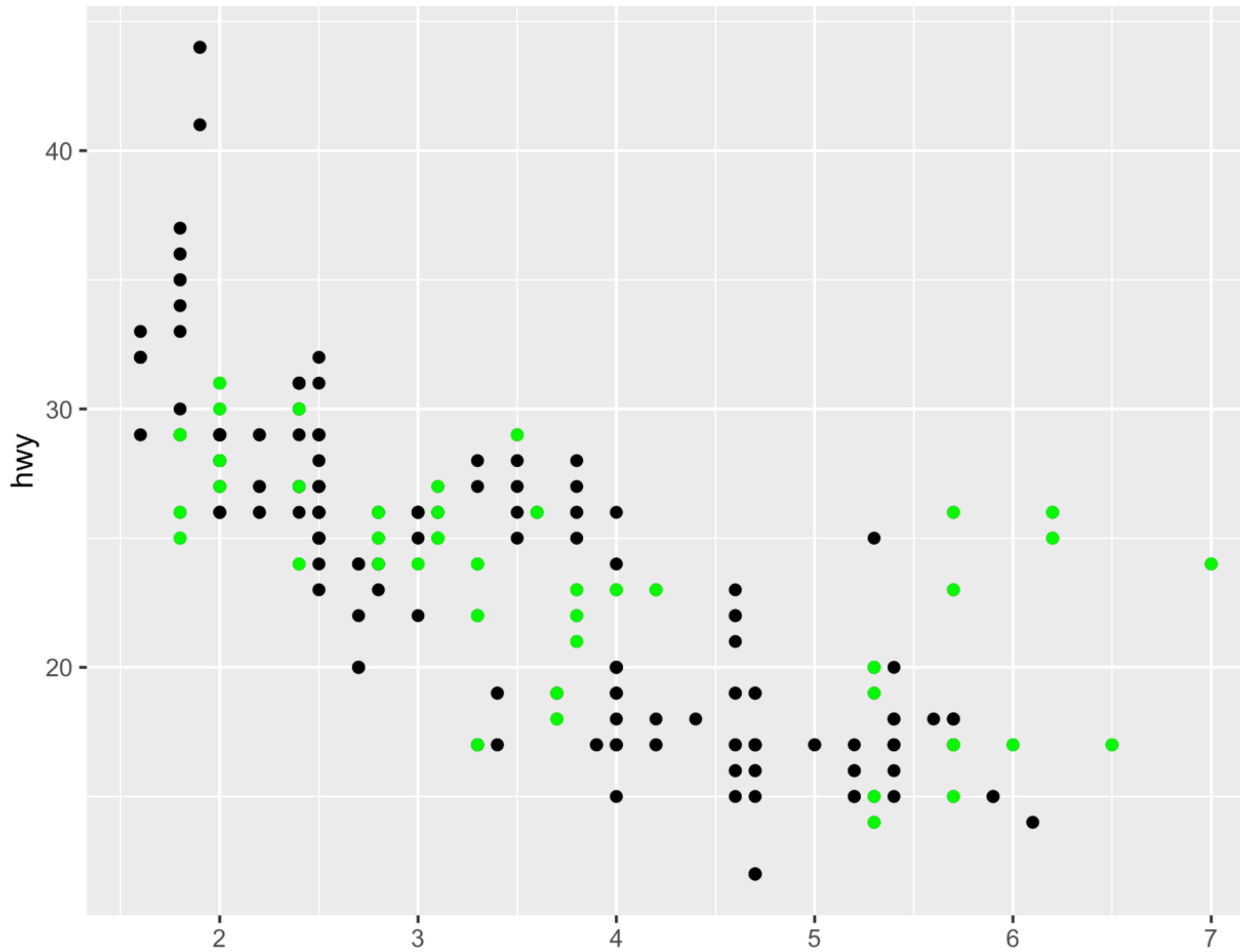
global data:
default for every layer

```
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



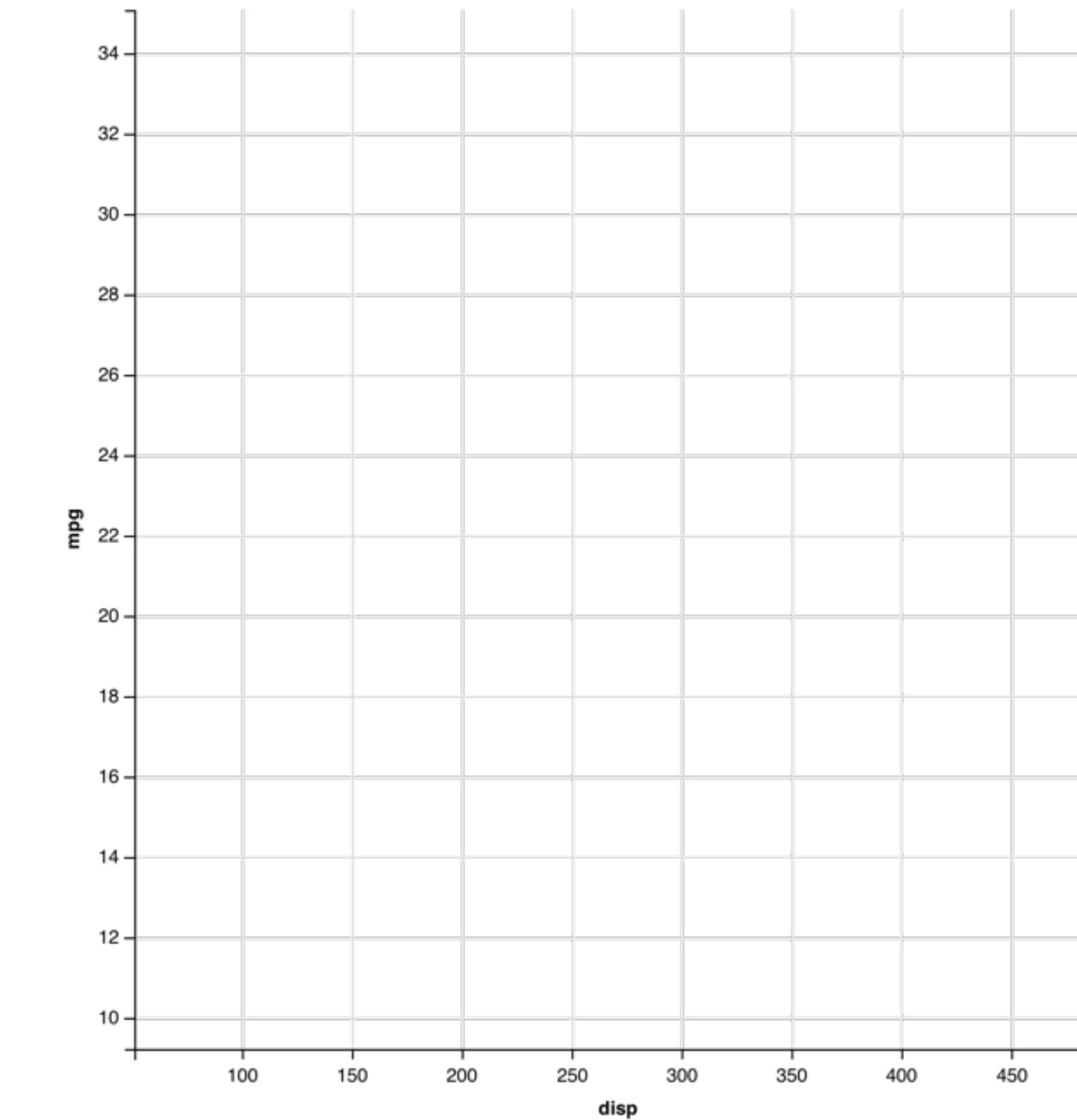
```
ggplot(mpg, aes(displ, hwy)) + geom_point()
```



```
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

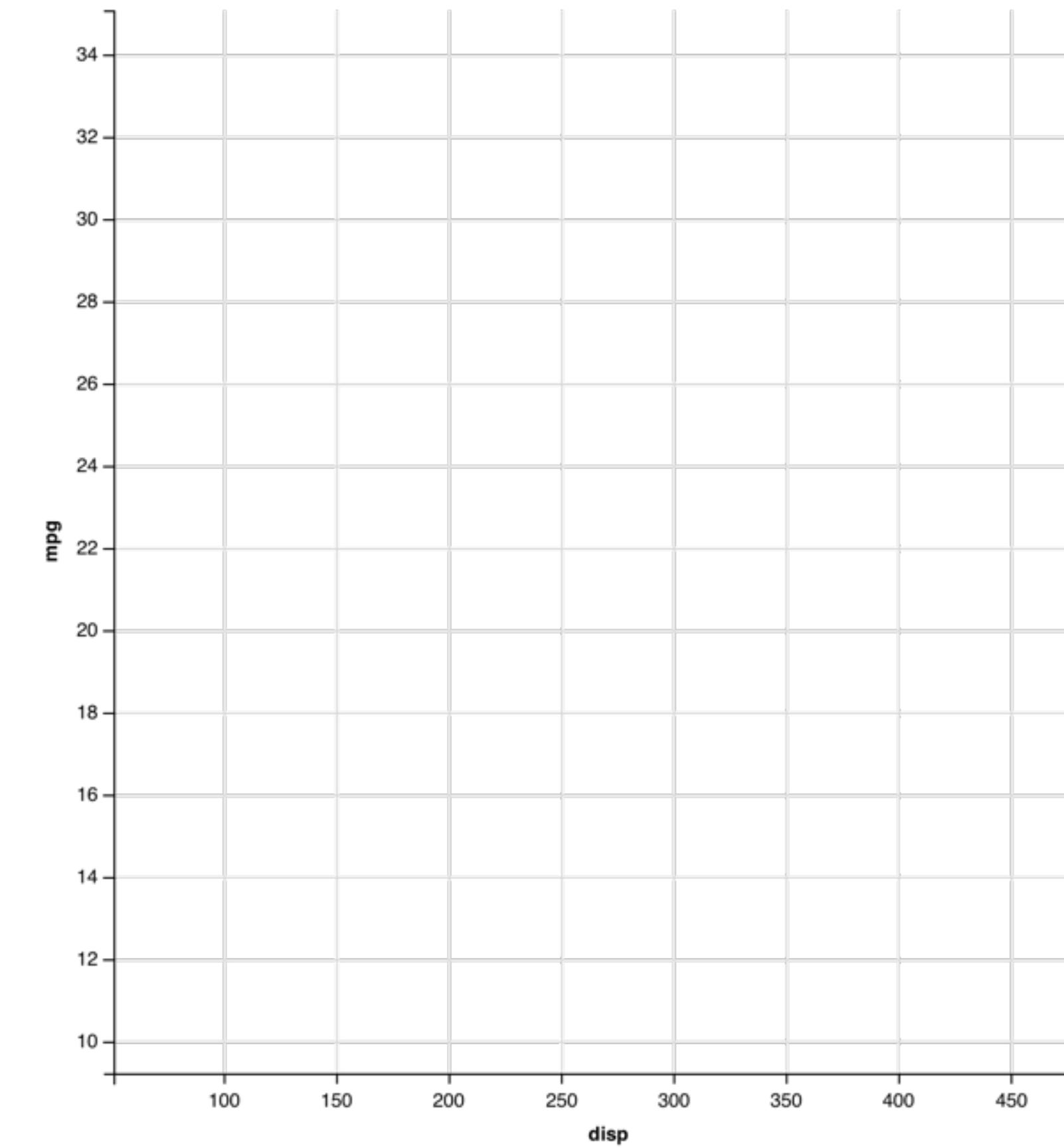
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

fill



data

geom

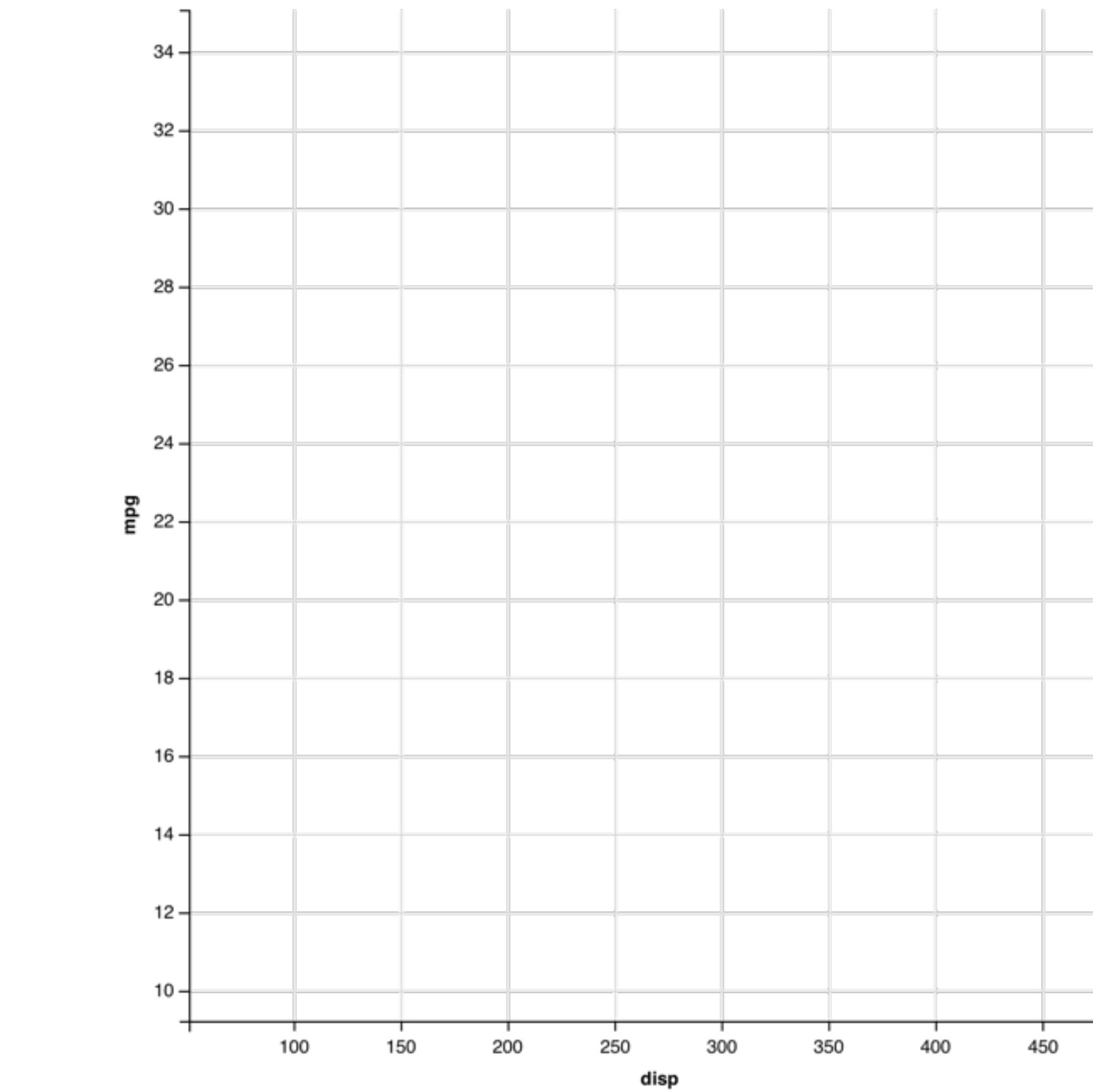
coordinate
system

properties

shape		fill	
	cyl		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



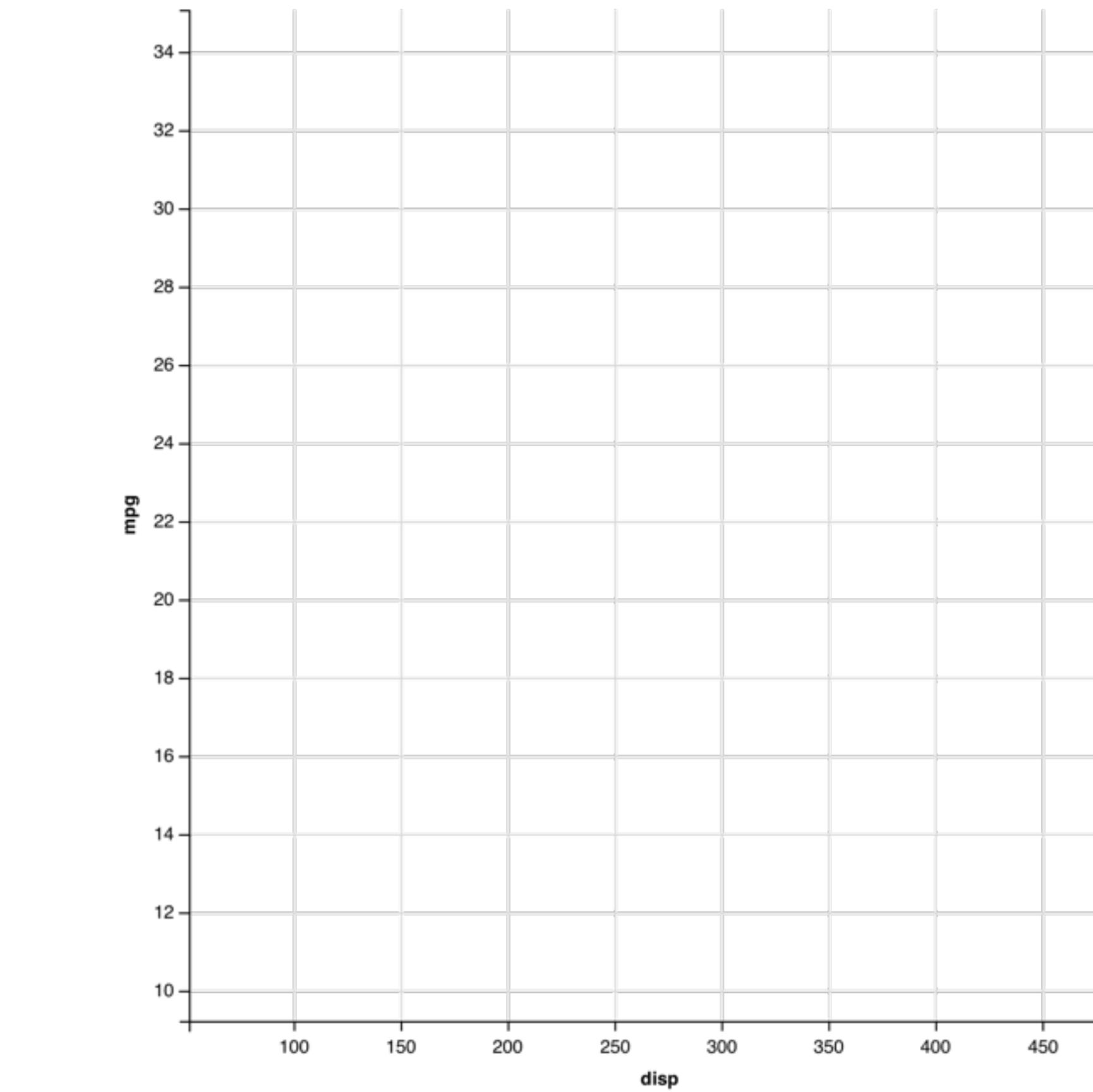
**coordinate
system**

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

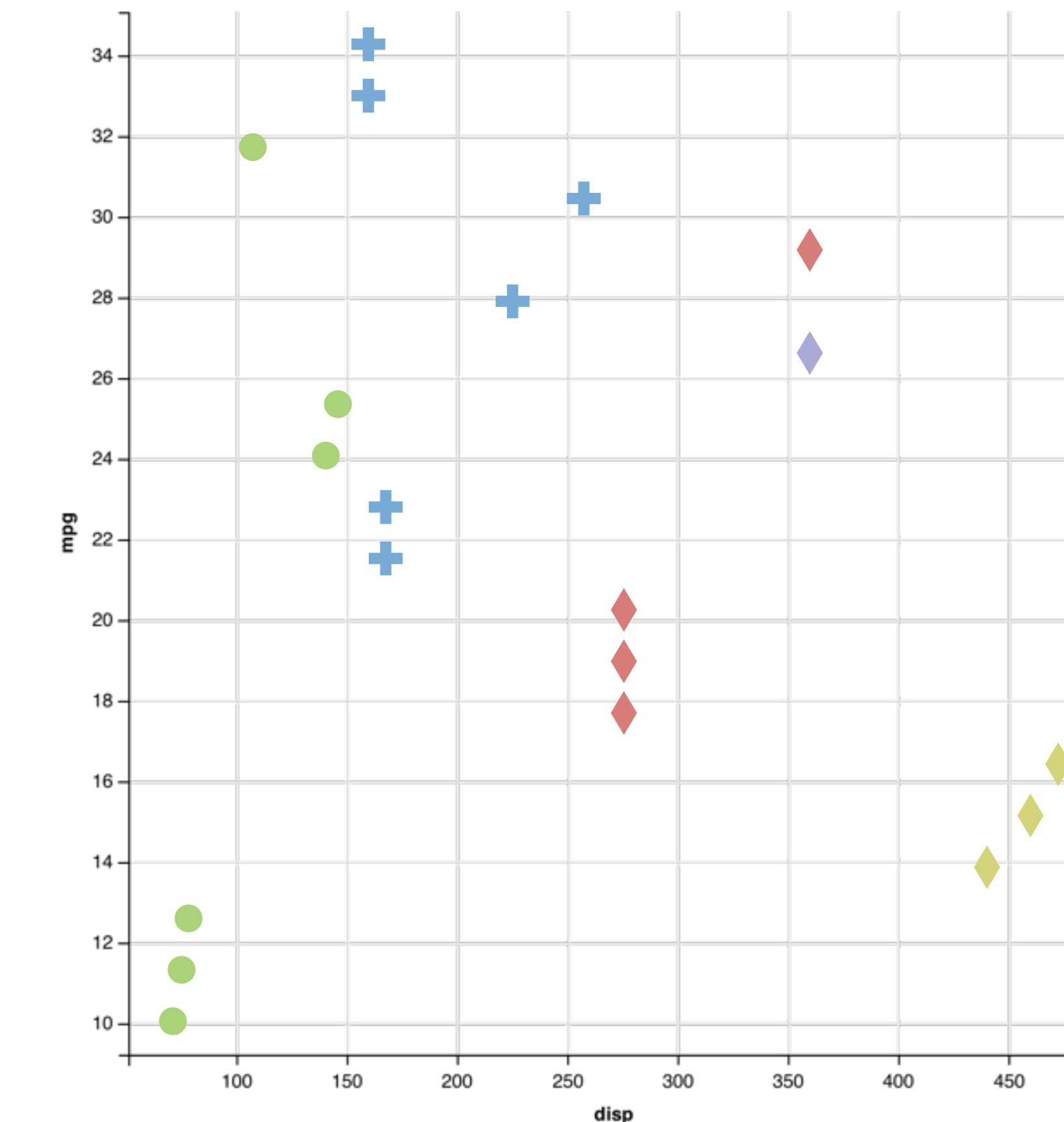
geom



**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**

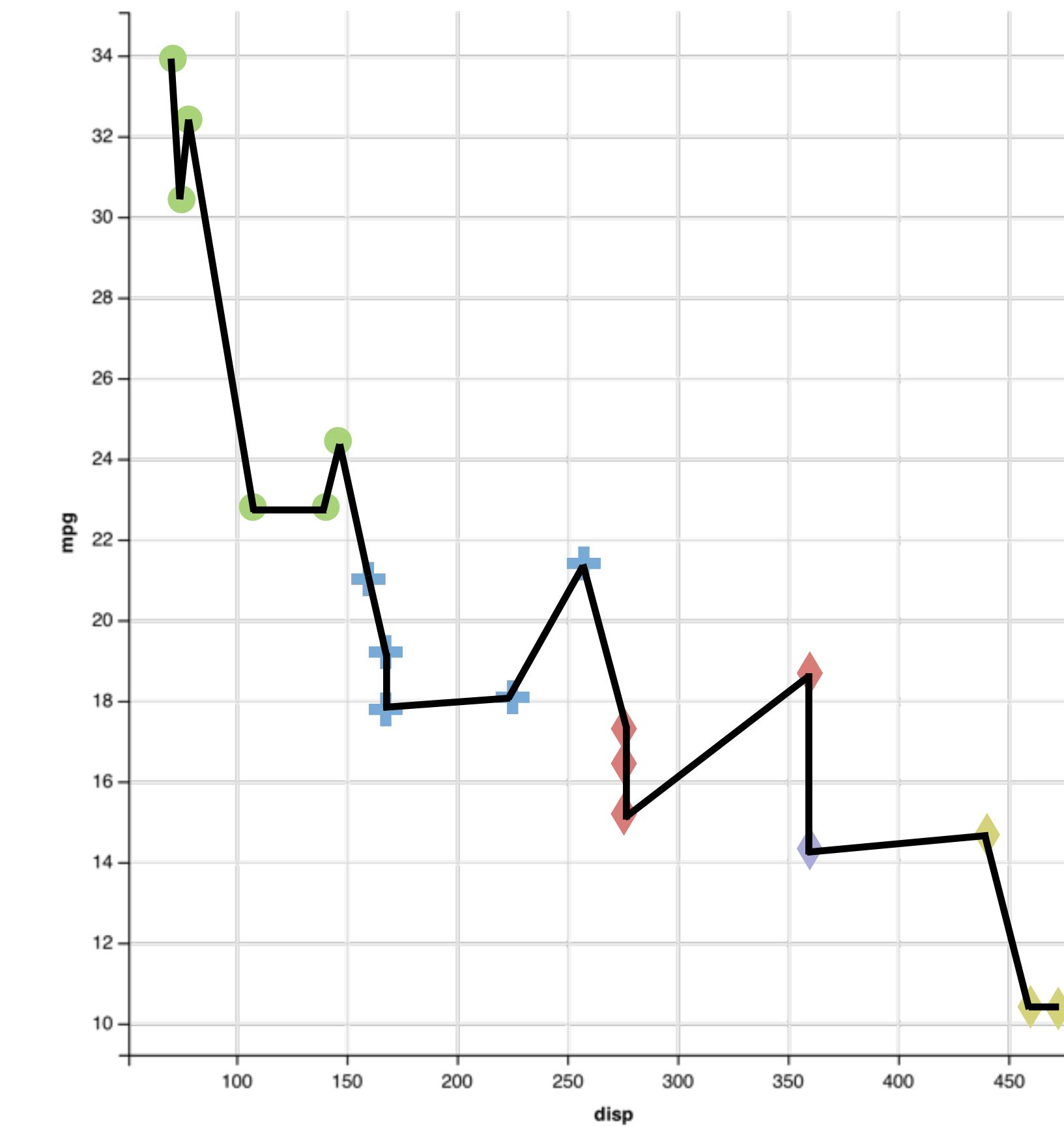
properties

	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

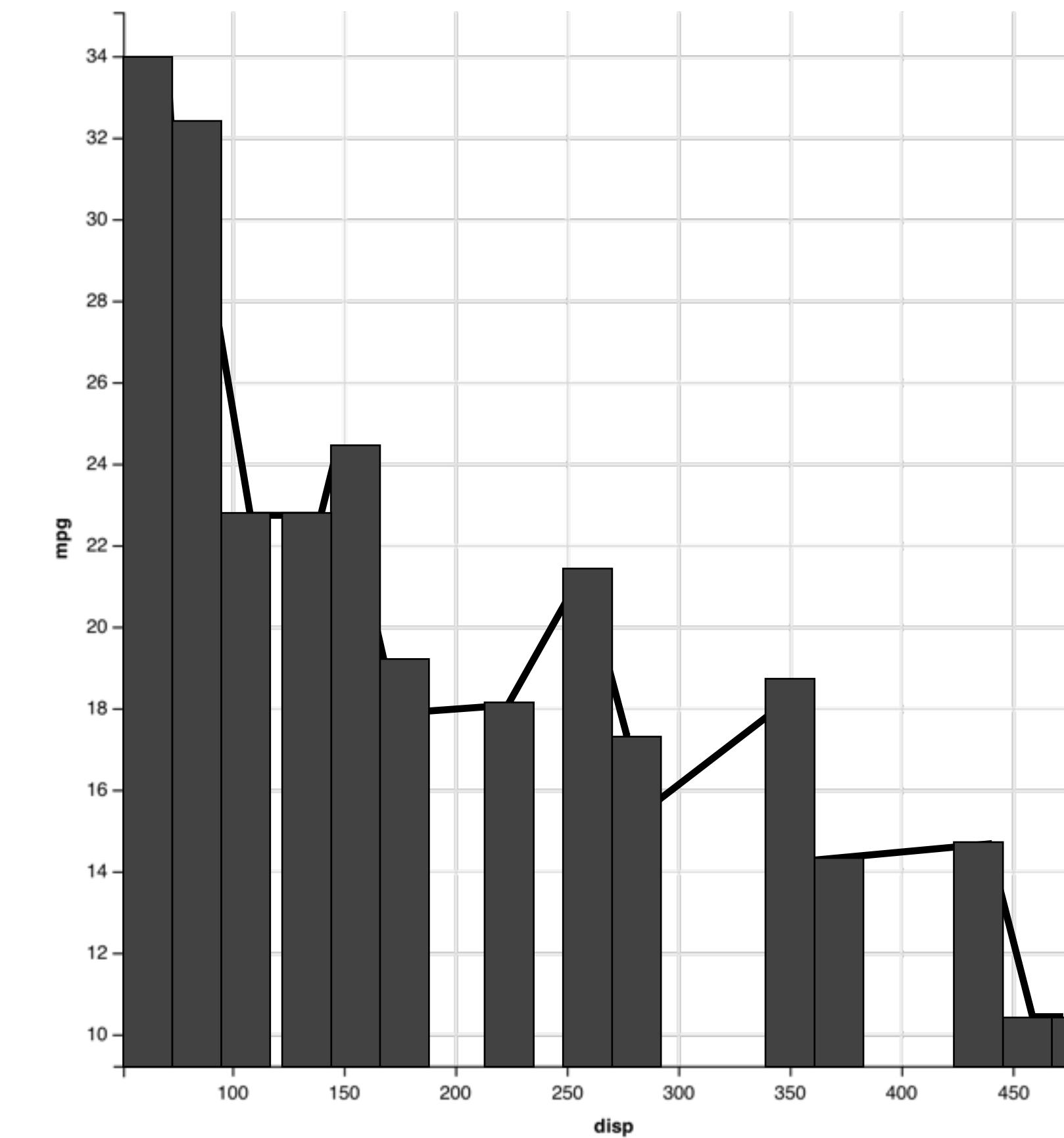
Y
↑ ↓
X
↑ ↓

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

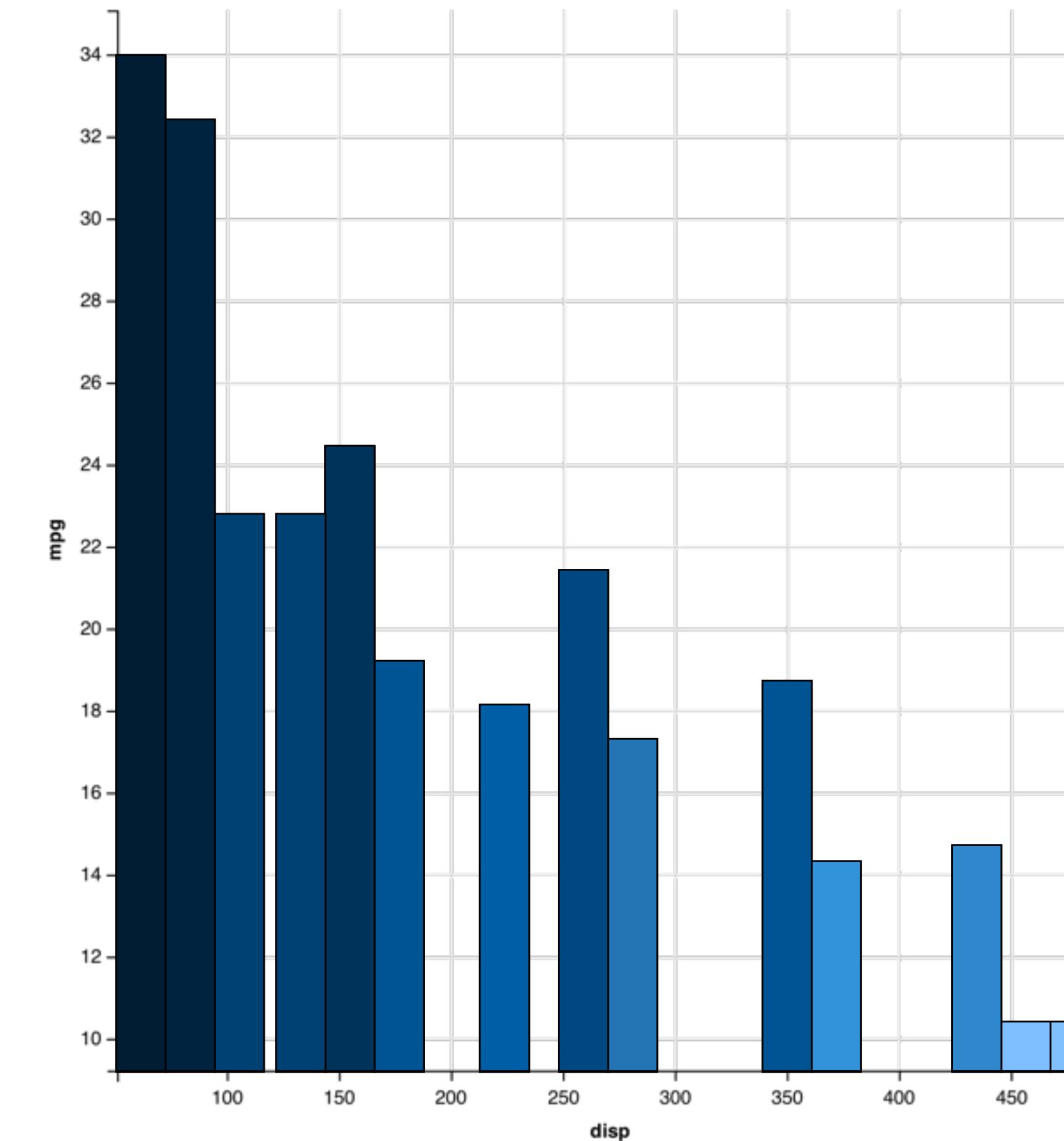


properties

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

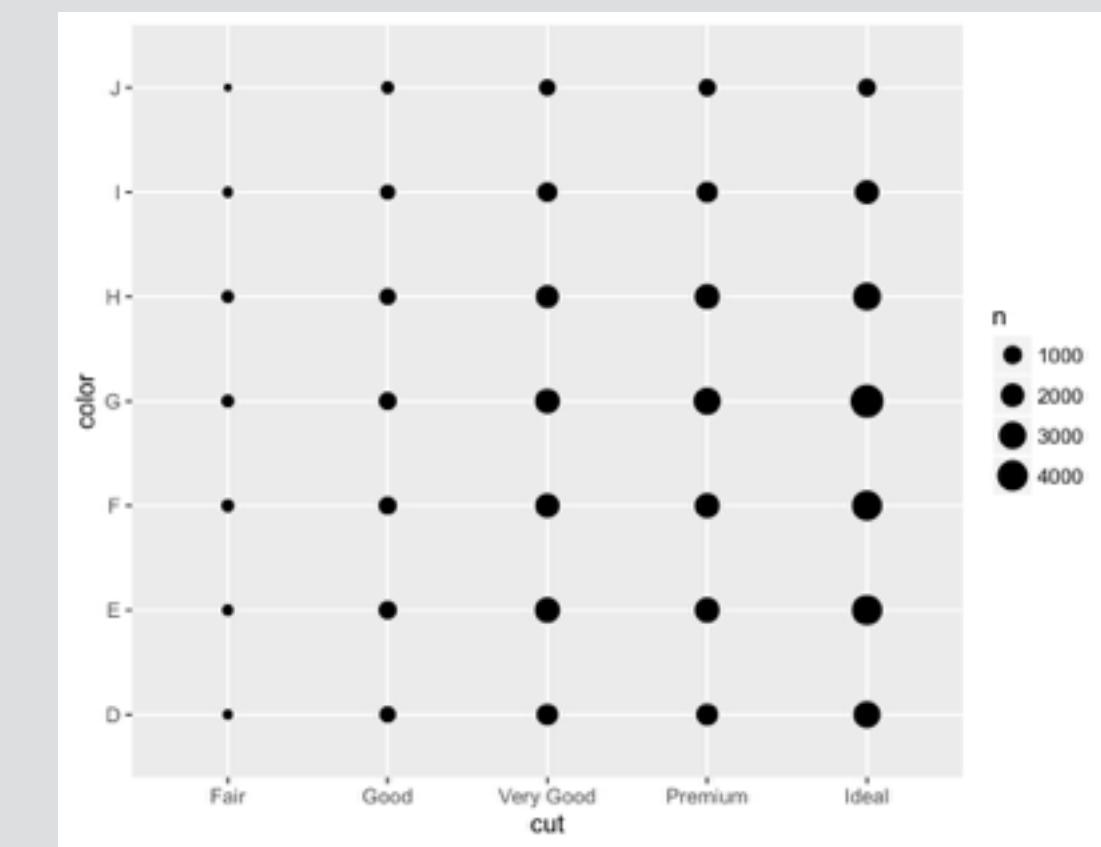
**coordinate
system**

Your turn

Make these plots:

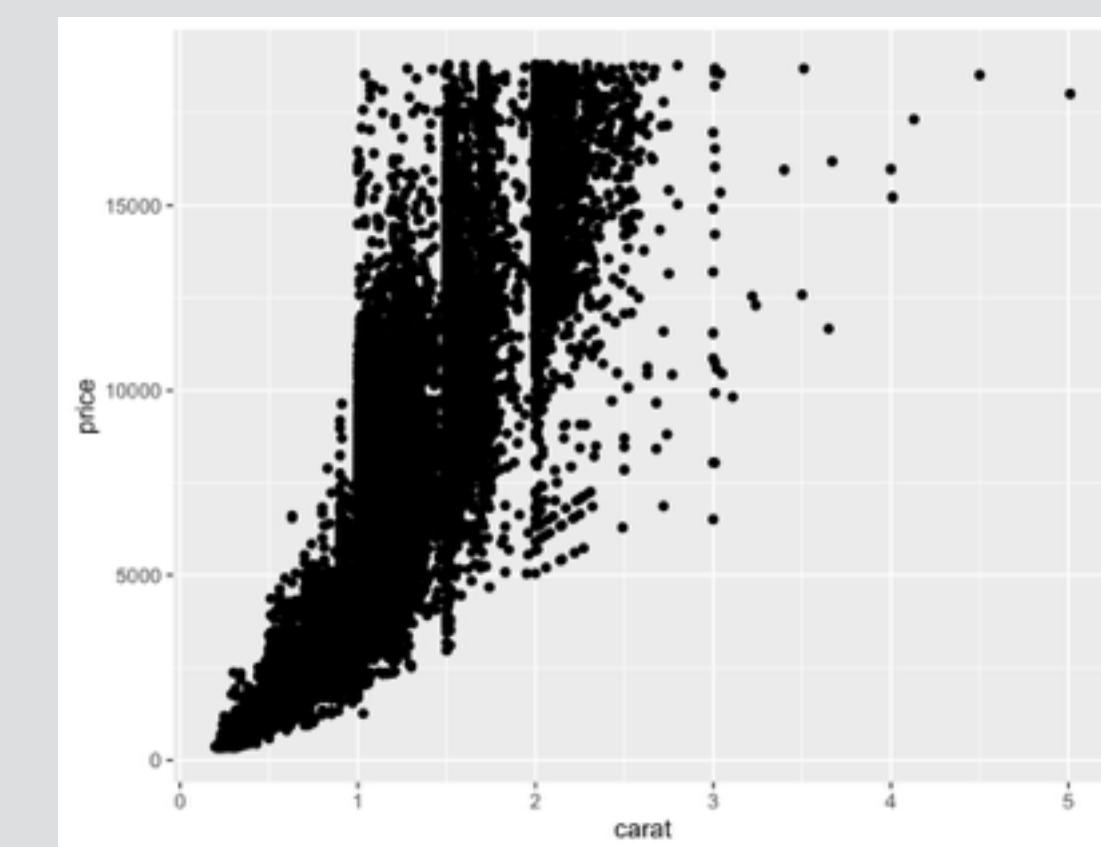
Plot 1

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



Plot 2

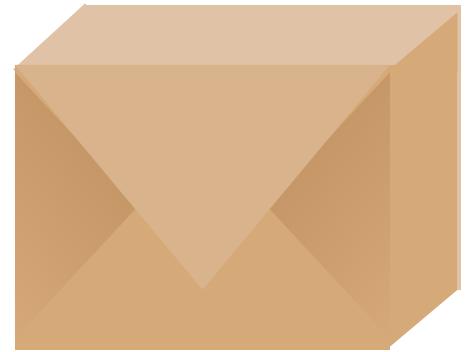
Data = diamonds
geom = point
x = carat
y = price



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

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

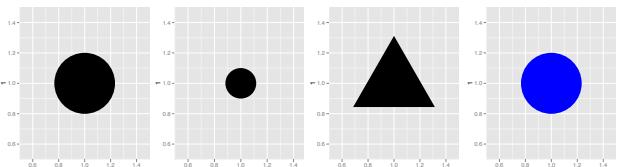
Recap: visualization



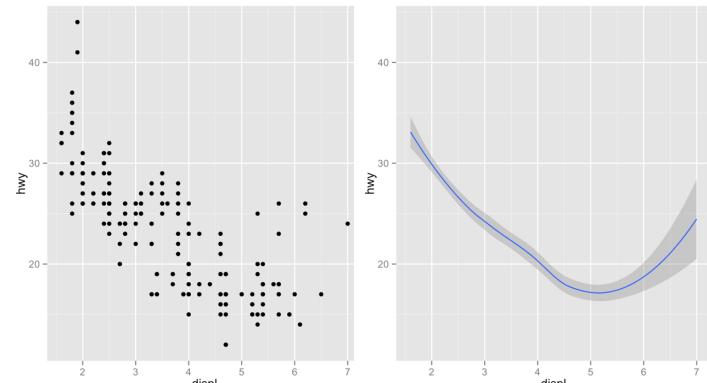
ggplot2: A package that visualizes data.

`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/

The screenshot shows a web browser window displaying the ggplot2 documentation. The title bar reads "Index, ggplot2 0.9.3.1". The address bar shows the URL "docs.ggplot2.org/current/". The main content area has a dark header bar with "ggplot2 0.9.3.1" and "Index". Below this, there are two main sections: "Help topics" and "Dependencies".

Help topics

Geoms

Geoms, short for geometric objects, describe the type of plot you will produce.

- [geom_abline](#)
Line specified by slope and intercept.
- [geom_area](#)
Area plot.
- [geom_bar](#)
Bars, rectangles with bases on x-axis
- [geom_bin2d](#)
Add heatmap of 2d bin counts.
- [geom_blank](#)
Blank, draws nothing.
- [geom_boxplot](#)
Box and whiskers plot.
- [geom_contour](#)
Display contours of a 3d surface in 2d.
- [geom_crossbar](#)
Hollow bar with middle indicated by horizontal line.
- [geom_density](#)
Display a smooth density estimate.



Dependencies

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



R for Data Science

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

The screenshot shows a web browser window for the 'R for Data Science' website. The left sidebar contains a table of contents with chapters numbered 1 through 16. Chapter 3, 'Data visualisation', is expanded, showing sub-sections 3.0.1 through 3.2. The main content area displays a section titled '3.1.2.1 Positions'. Below this, a text block asks if the user can make a chart similar to the one shown. The chart is a grouped bar plot with 'cut' on the x-axis (categories: Fair, Good, Very Good, Premium, Ideal) and 'count' on the y-axis (range: 0 to 5000). Each bar is composed of multiple colored segments representing different 'clarity' levels: I1 (red), SI2 (orange), SI1 (yellow-green), VS2 (green), VS1 (teal), VVS2 (blue), VVS1 (purple), and IF (pink). The total count for each cut increases from approximately 500 for Fair to over 5000 for Ideal.

clarity	I1	SI2	SI1	VS2	VS1	VVS2	VVS1	IF
Fair	~100	~200	~50	~100	~50	~10	~5	~5
Good	~50	~1000	~1500	~100	~50	~10	~5	~5
Very Good	~50	~2100	~3400	~1800	~500	~1200	~800	~300
Premium	~100	~2900	~3600	~3400	~2000	~800	~500	~200
Ideal	~100	~2600	~4300	~3600	~2600	~2500	~2000	~1200

Data Wrangling with dplyr

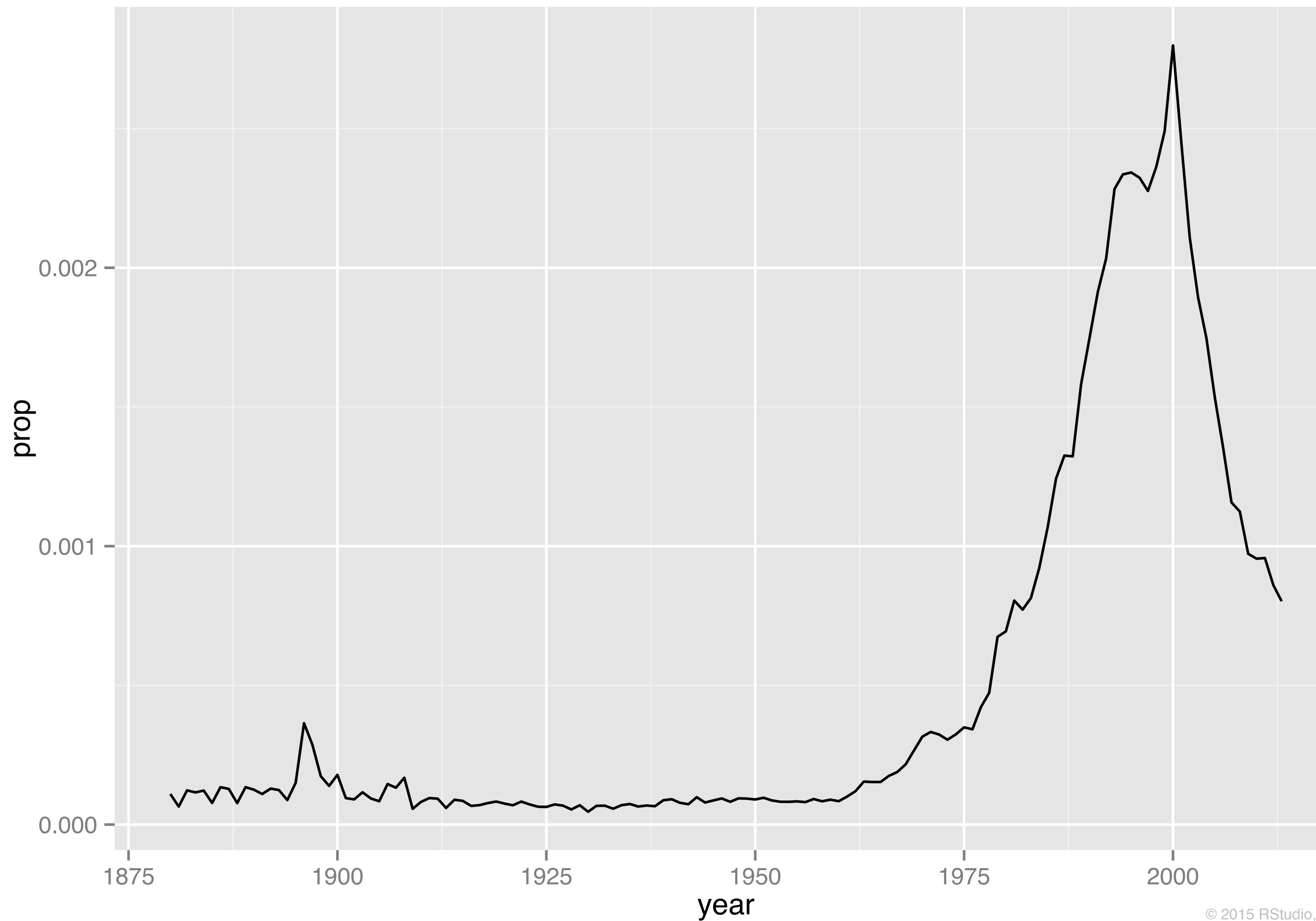
Baby names

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

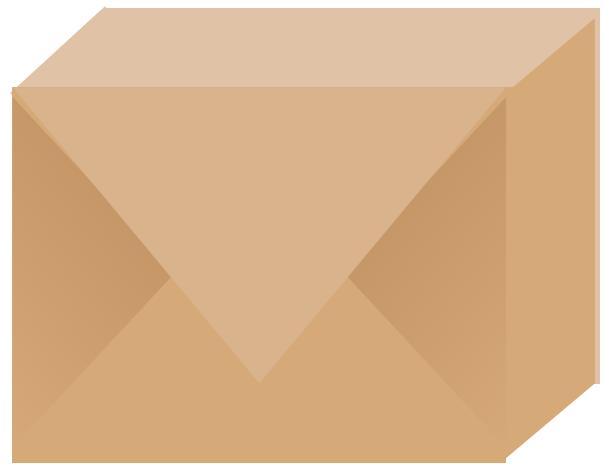
```
library(reportsWS)  
View(bnames)
```

```
# 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 1.515051e-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))
```

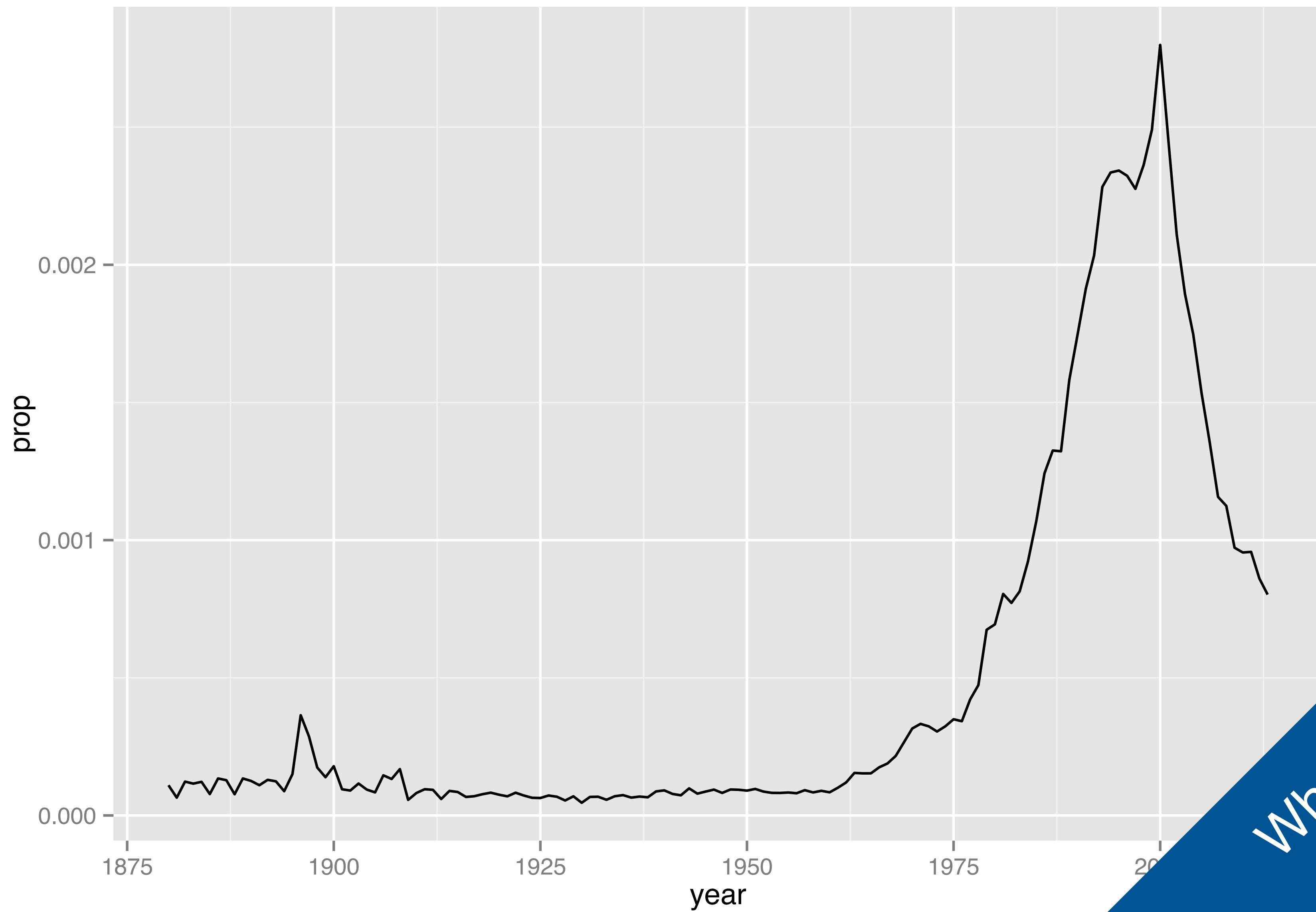


```
library(dplyr)

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

year	sex	births
1880	M	118400
1881	M	108285
1882	M	122032
1883	M	112481
1884	M	122742
1885	M	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, $n = \text{round}(\text{prop} * \text{births})$.

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, $n = \text{round}(\text{prop} * \text{births})$.

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	112285	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)
```

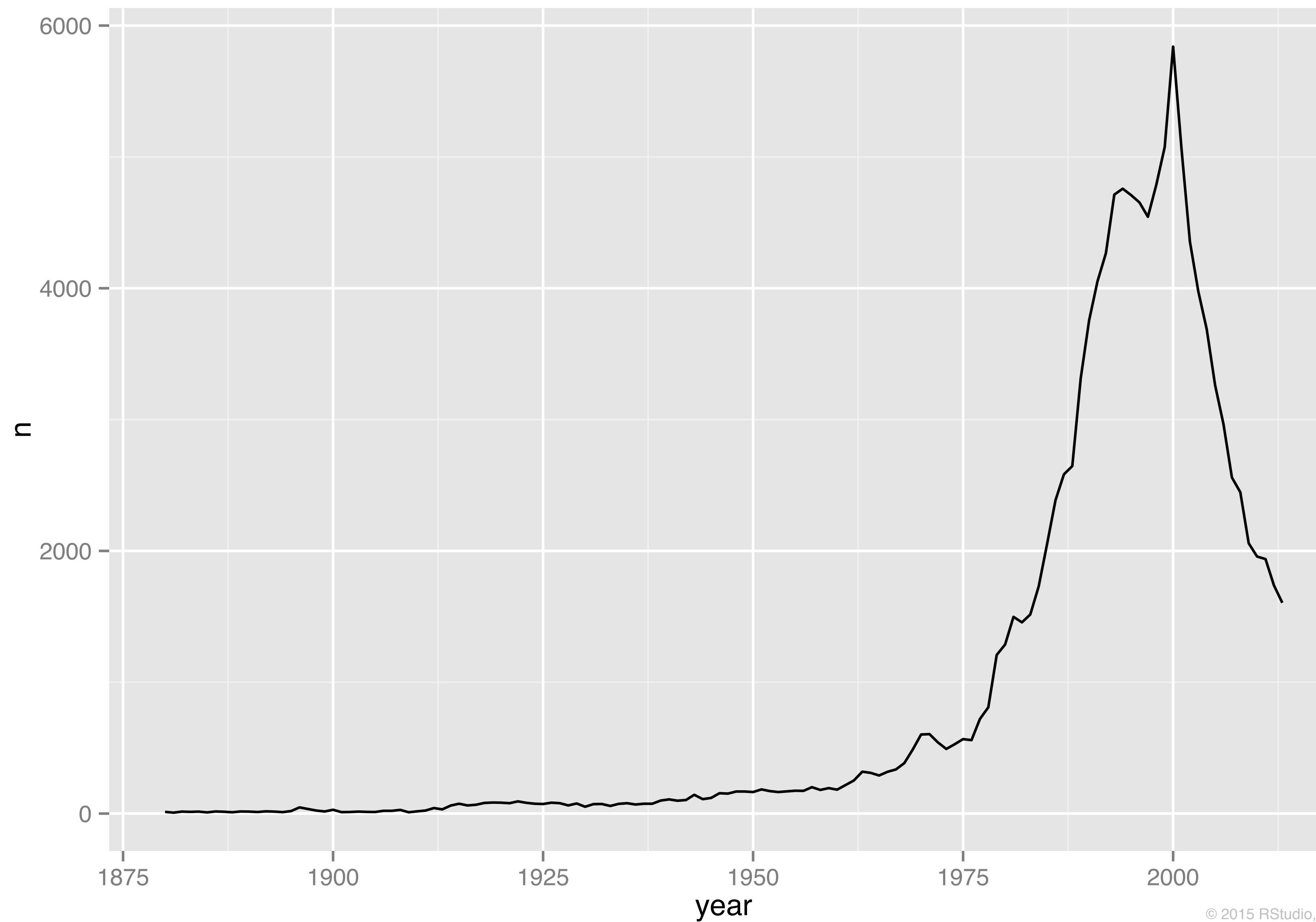
Your Turn

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.

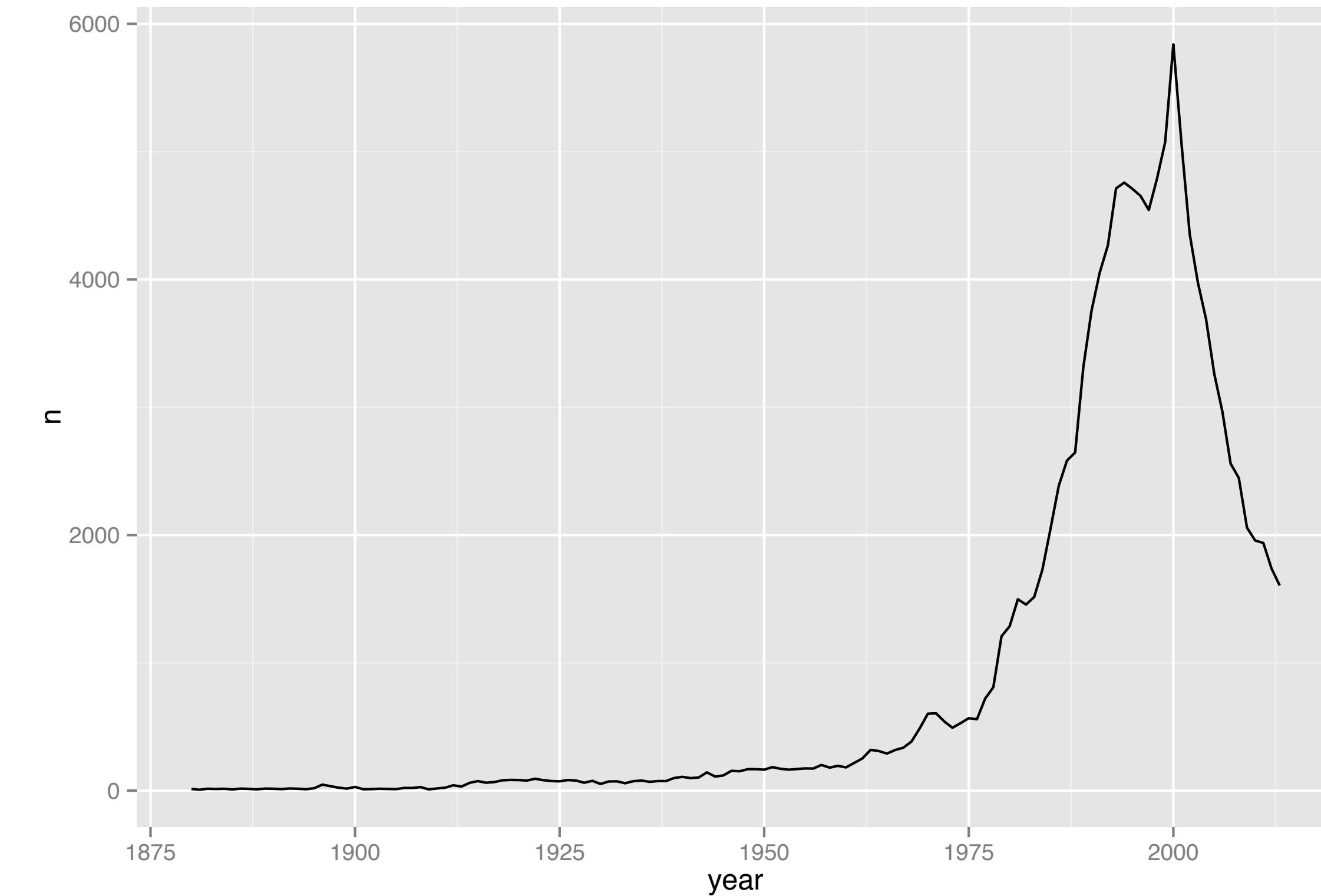
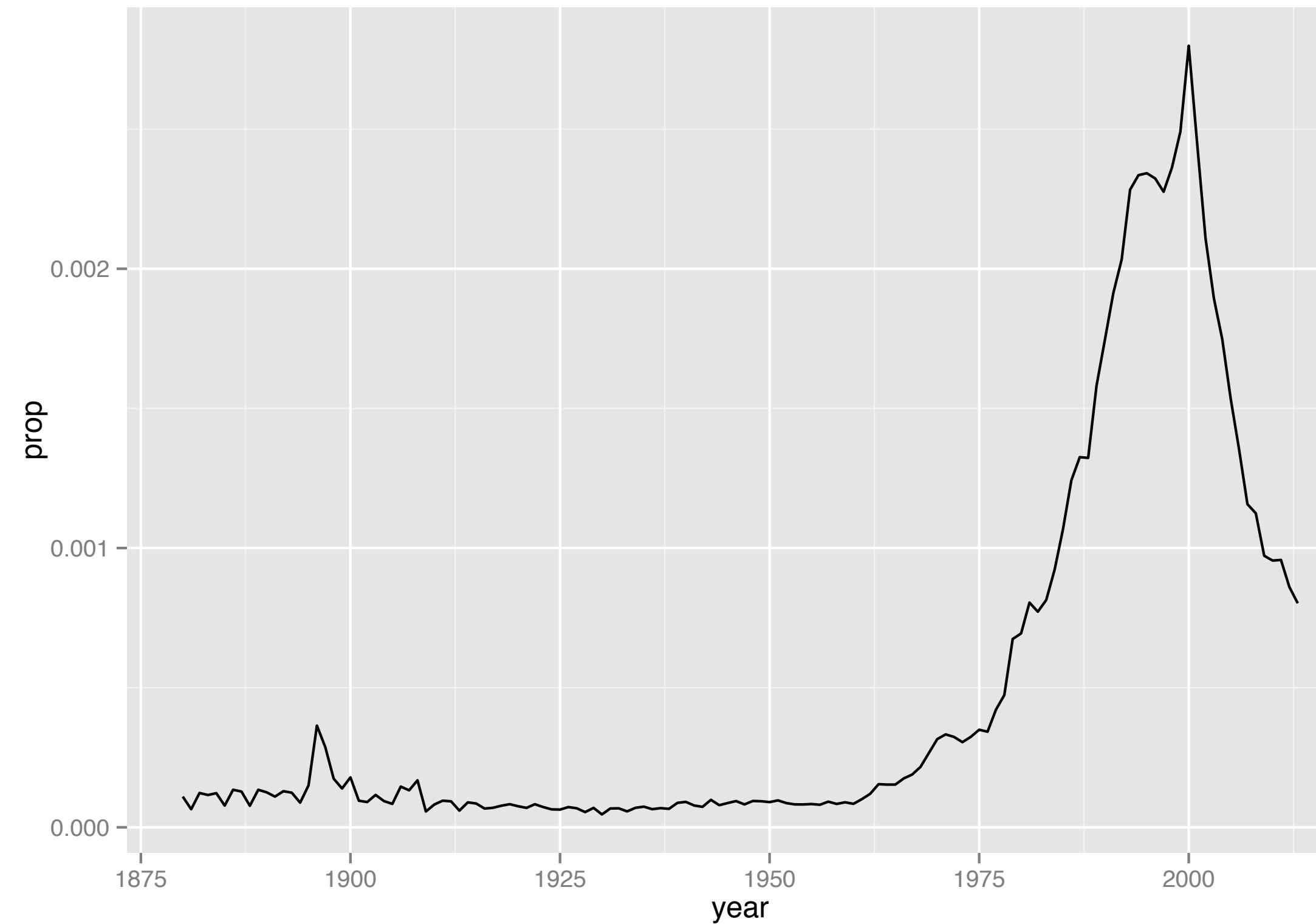


```
boys <- filter(births, sex == "M")  
  
my_name <- left_join(my_name, boys, by = "year")  
  
my_name <- mutate(my_name, n = round(prop * births))  
  
ggplot(my_name) +  
  geom_line(aes(x = year, y = n))
```

Popularity of "Garrett" over time



Popularity of "Garrett" over time



summarise()

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



mean	sum	n
18.5	37	2

London	large	22
London	small	16



19.0	38	2
------	----	---

Beijing	large	121
Beijing	small	56



88.5	177	2
------	-----	---

group_by() + summarise()

group_by()

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

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



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

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



pollution

summarize(_____, median = median(amount))

Shortcut to type %>%

Cmd + Shift + M (Mac)

Ctrl + Shift + M (Windows)

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)) %>%  
  ungroup() %>%  
  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)) %>%
# Ungroup the data (so we do not arrange within groups)
ungroup() %>%
# 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)) %>%  
  ungroup() %>%  
  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 <- ungroup(tmp6)
tmp8 <- arrange(tmp7, desc(total))
```

```
arrange()  
ungroup()  
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)  
,  
)
```

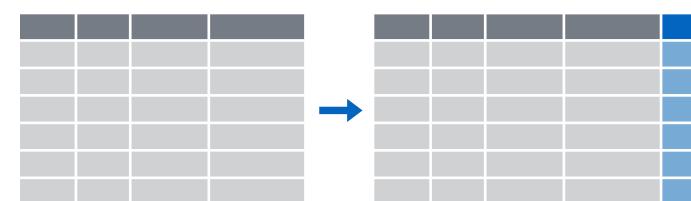
Recap: dplyr



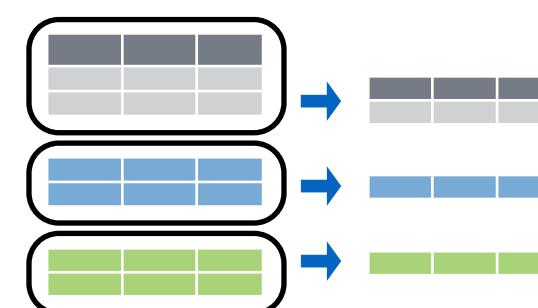
Extract variables and observations with **select()** and **filter()**



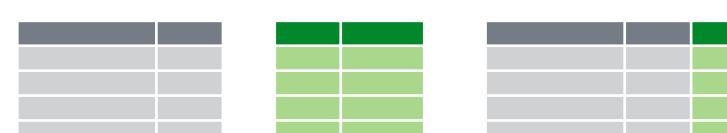
Arrange observations with **arrange()**.



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



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



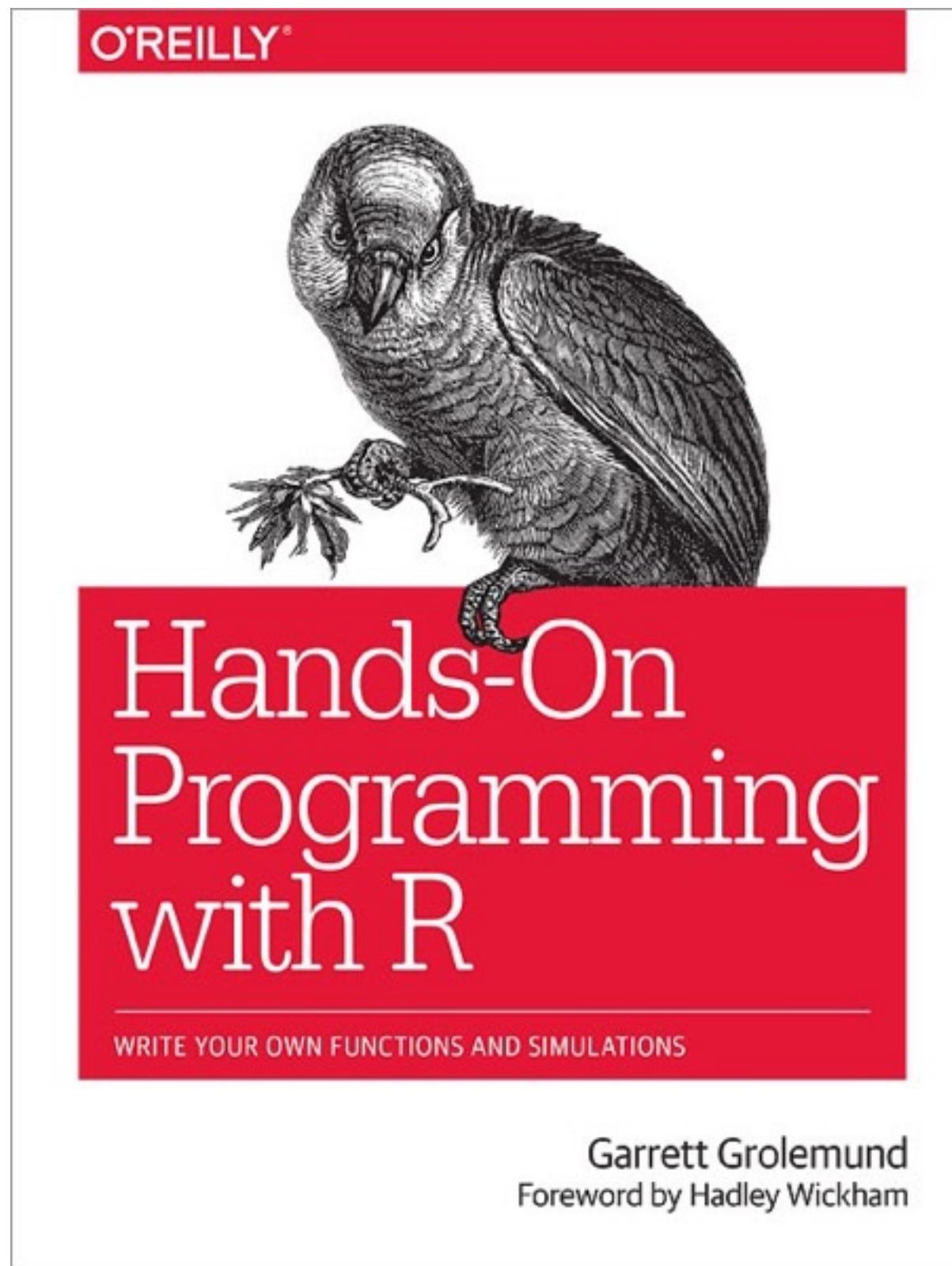
Join data frames with **left_join()**.



Chain operations together with **%>%**

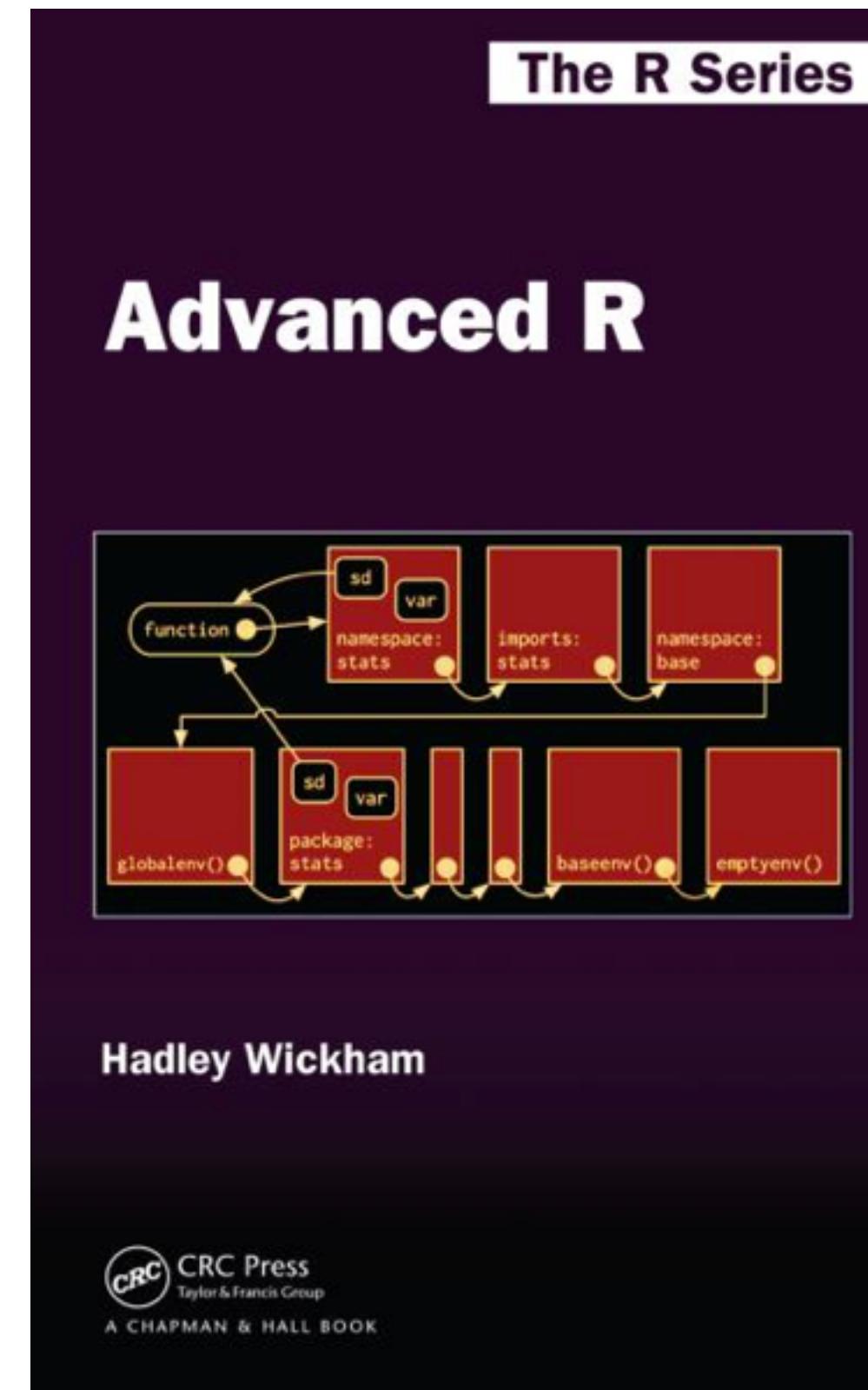
where
next?

Books



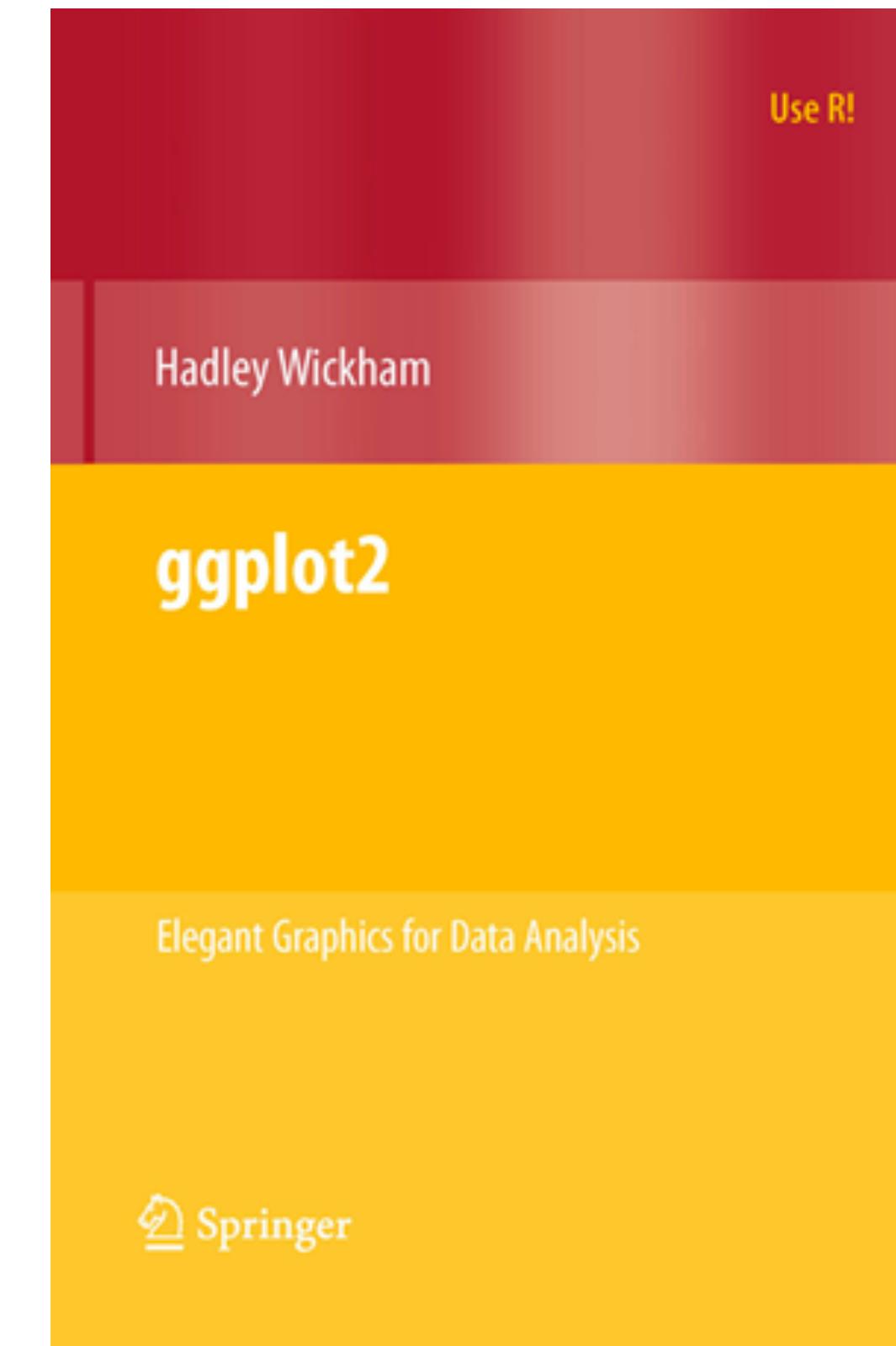
shop.oreilly.com/product/0636920028574.do

Beginner



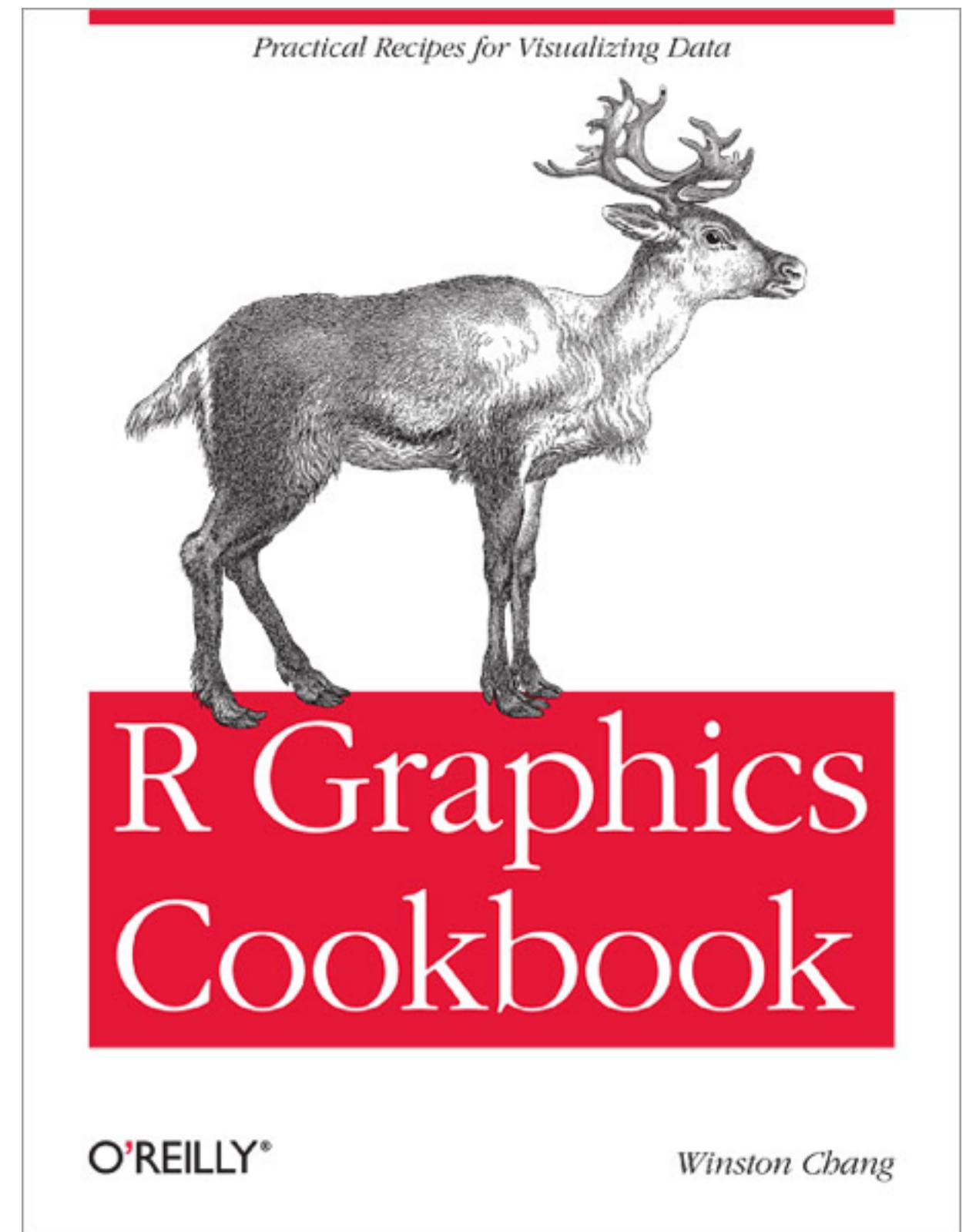
adv-r.had.co.nz/

Advanced



www.amzn.com/0387981403

Intro to
Visualizations



O'REILLY®

Winston Chang

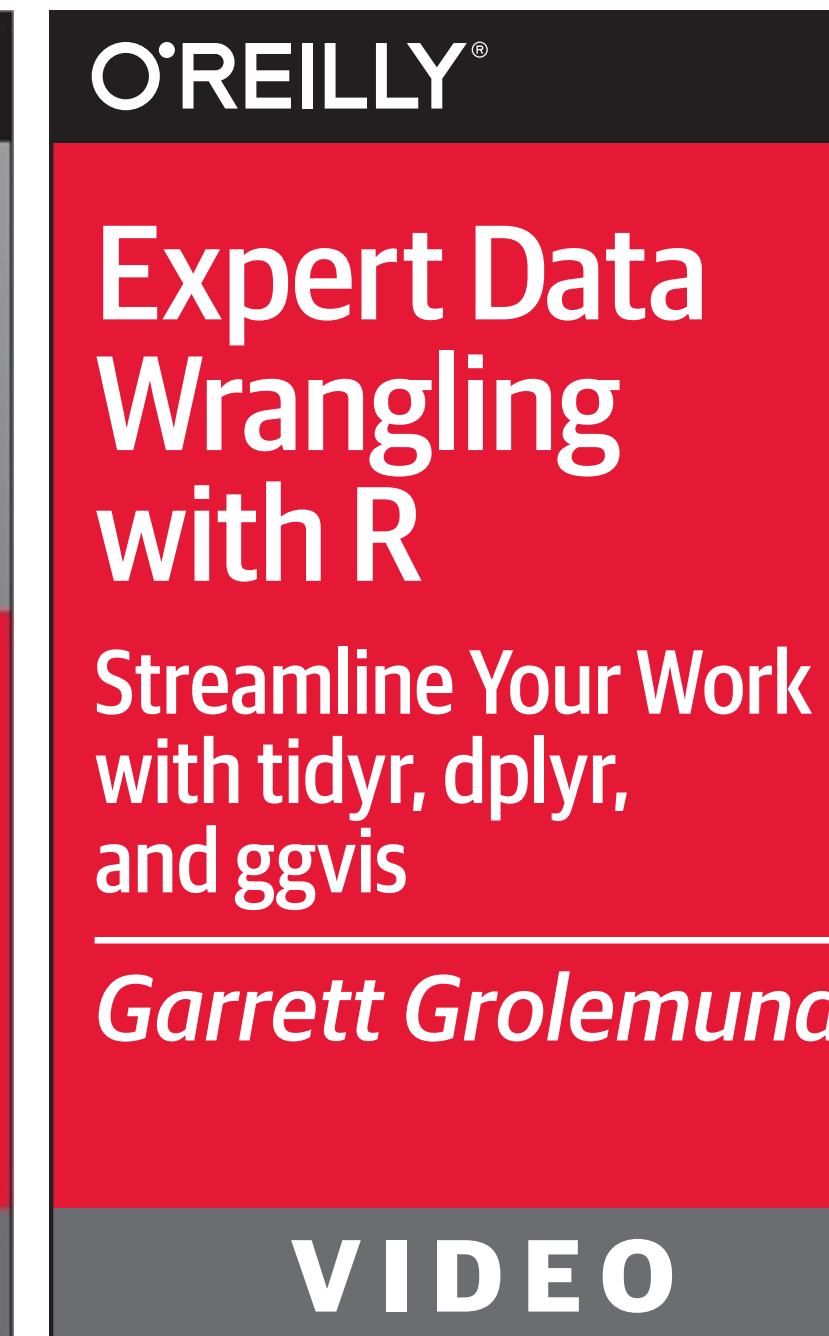
shop.oreilly.com/product/0636920023135.do

Reference for
Visualizations

Videos



[shop.oreilly.com/product/
0636920034834.do](http://shop.oreilly.com/product/0636920034834.do)



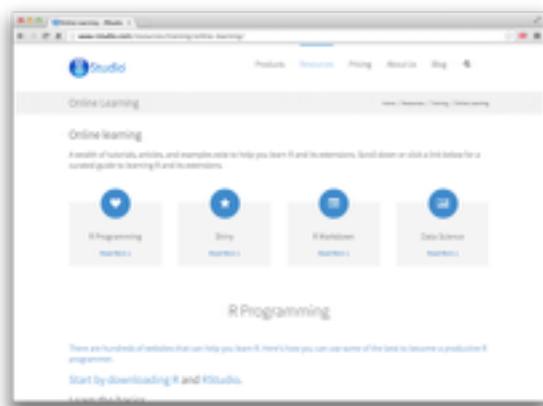
[shop.oreilly.com/product/
0636920035992.do](http://shop.oreilly.com/product/0636920035992.do)

Interactive tutorials



DataCamp

www.datacamp.com



More at

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

R for Data Science

r4ds.had.co.nz/

The screenshot shows a web browser window for the URL r4ds.had.co.nz/. The page title is "R for Data Science". The left sidebar contains a table of contents with 25 numbered sections from "1 Welcome" to "25 Shiny". The main content area features a large block of text describing the book's purpose and content, followed by a note about publication and a small image of the book cover.

This is the book site 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 manage cognitive resources to facilitate discoveries when wrangling, visualizing, and exploring data. (**R for Data Science** was formally called **Data Science with R in Hands-On Programming with R**)

To be published by O'Reilly in July 2016.

The book cover features a red and white design with a black and white photograph of an owl. The title "R for Data Science" is prominently displayed in white on a red background, with the subtitle "VISUALIZE, MODEL, TRANSFORM, Tidy, AND IMPORT DATA" below it. The authors' names, "Garrett Grolemund & Hadley Wickham", are at the bottom.