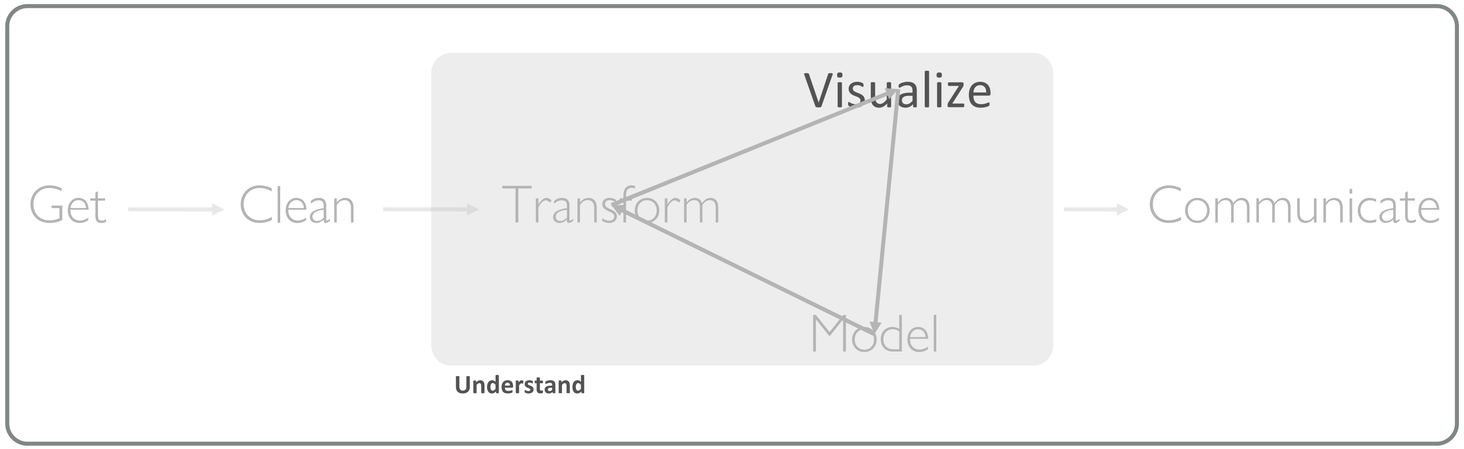
TODAY'S CLASS

6:00PM – 6:50PM: Data Visualization with **ggplot2**

7:00PM – 7:50PM: Movie Challenge

8:00PM – 9:50PM: **ggplot** Extensions

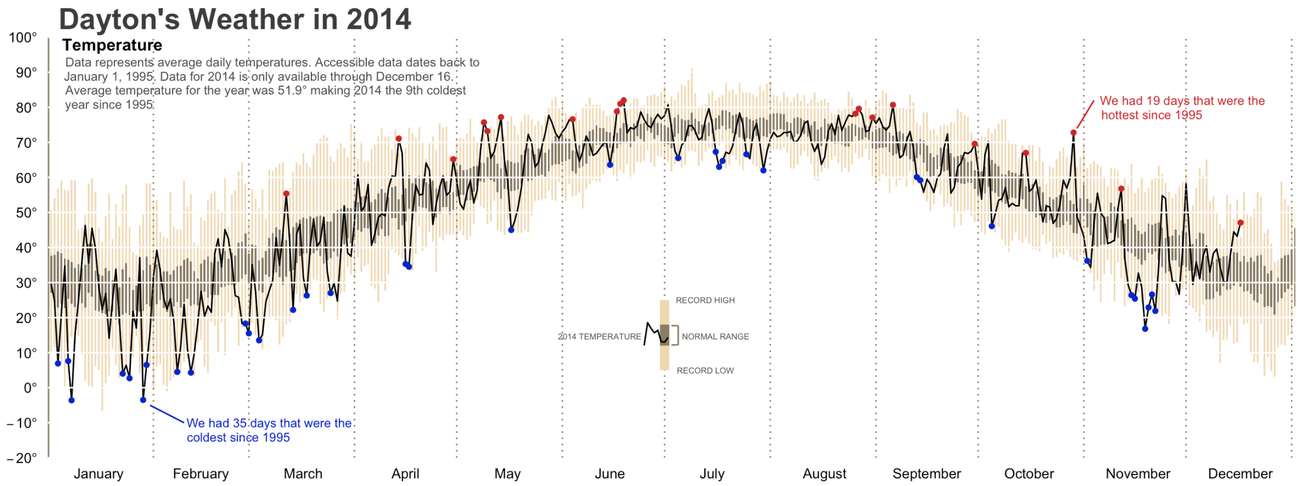
VISUALIZATION

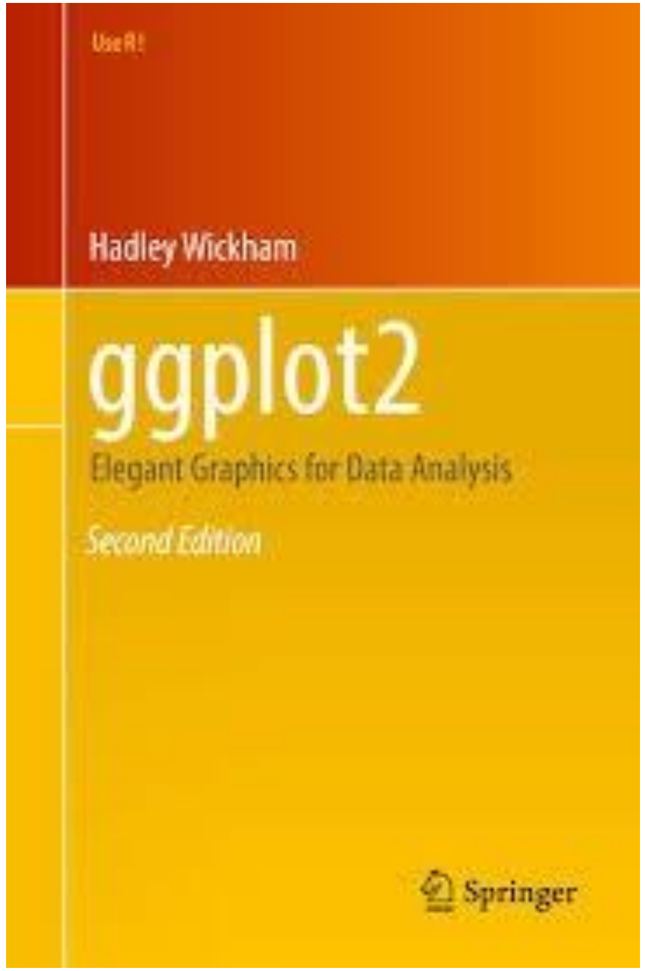


†A modified version of Hadley Wickham's analytic process

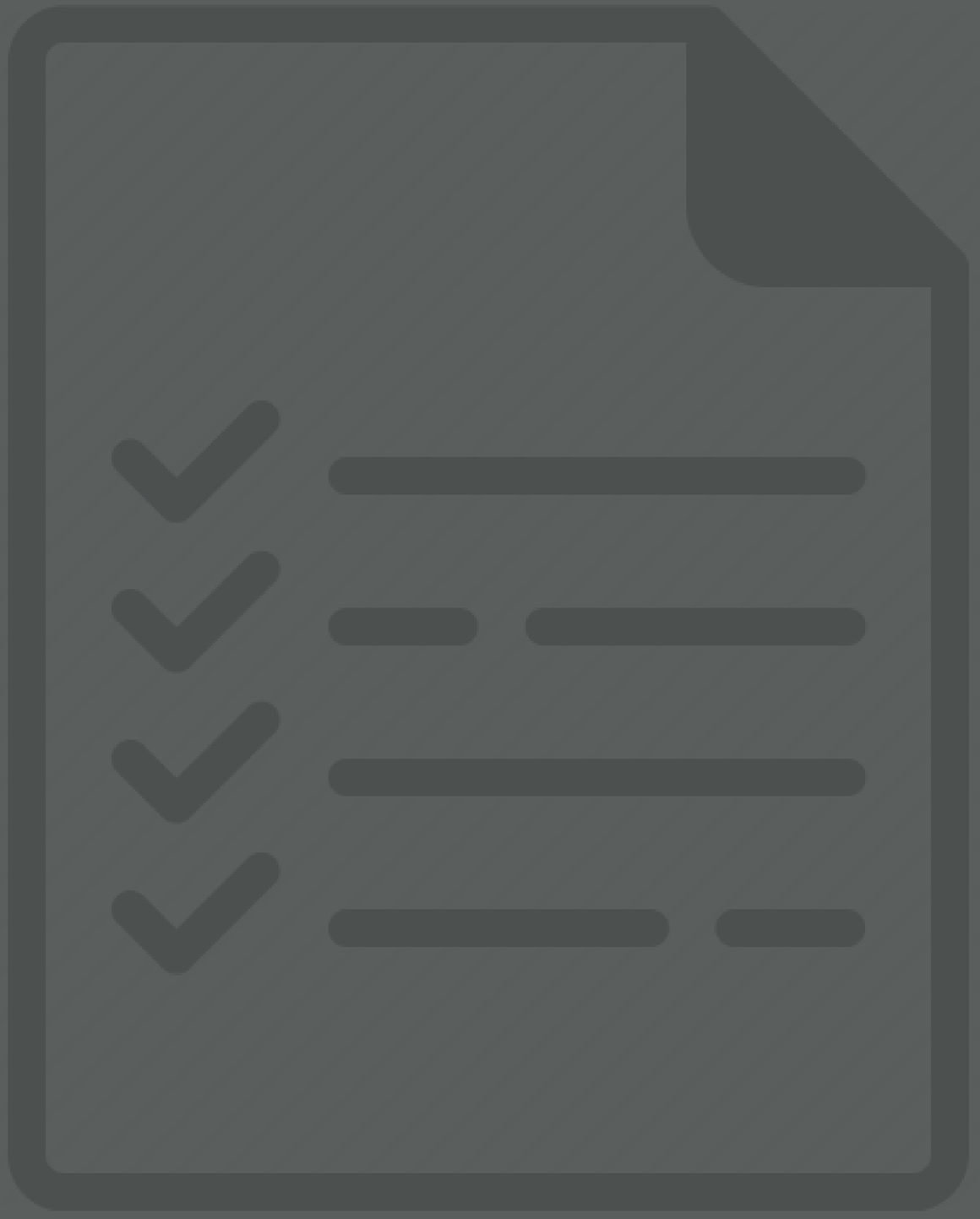
ggplot2

- R has several systems for making graphs
- **ggplot2** is the most elegant and versatile
- Implements the grammar of graphics theory behind data visualization





PREREQUISITES



PACKAGE PREREQUISITE

```
library(tidyverse)
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages ----------------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

DATA PREREQUISITE

```
mpg
#> # A tibble: 234 × 11
#>   manufacturer model displ  year   cyl      trans   drv   cty   hwy    fl
#>          <chr> <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>
#> 1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p
#> 2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p
#> 3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p
#> 4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p
#> 5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p
#> 6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p
#> # ... with 228 more rows, and 1 more variables: class <chr>
```

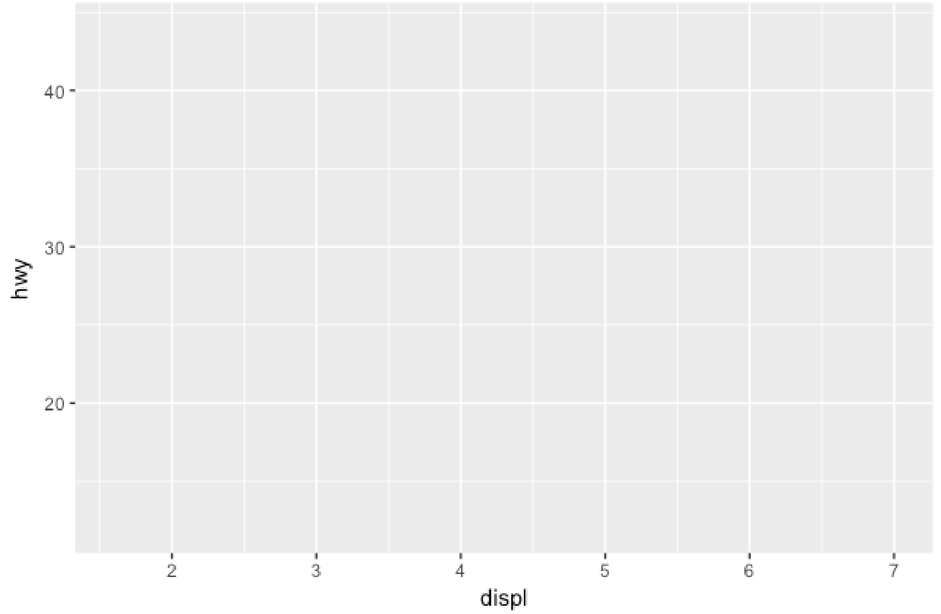
Type View(mpg) in the console for a spreadsheet view of the data

CANVAS

LET'S CREATE OUR "CANVAS"

```
# left
ggplot(data = mpg)

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

GEOMS

geom_abline
geom_bar
geom_bin2d
geom_blank
geom_boxplot
geom_contour
geom_count
geom_hex
geom_crossbar
geom_density
geom_density_2d
geom_dotplot
geom_errorharh
geom_freqpoly
geom_histogram
geom_jitter
geom_label
geom_map
geom_path
geom_point
geom_polygon
geom_quantile
geom_raster
geom_ribbon
geom_rug
geom_segment
geom_smooth
geom_violin

LETS ADD “GEOMS”

- We display data with geometric shapes
- ~ 30 built-in geoms (with many more offered by other pkgs)

Type **geom_** *+ tab in the console*

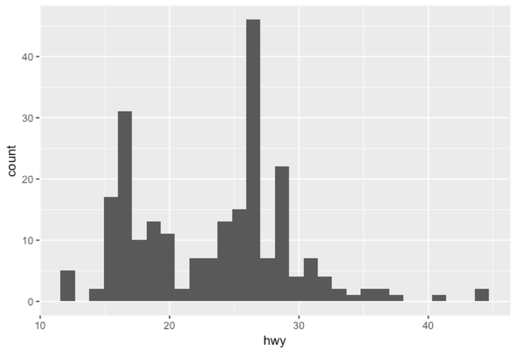
geom_abline
geom_bar
geom_bin2d
geom_blank
geom_boxplot
geom_contour
geom_count
geom_hex
geom_crossbar
geom_density
geom_density_2d
geom_dotplot
geom_errorharh
geom_freqpoly
geom_histogram
geom_jitter
geom_label
geom_map
geom_path
geom_point
geom_polygon
geom_quantile
geom_raster
geom_ribbon
geom_rug
geom_segment
geom_smooth
geom_violin

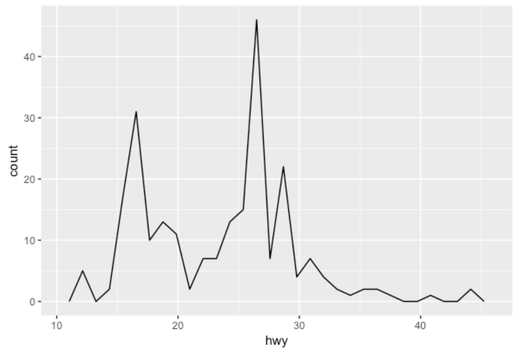
UNIVARIATE GEOMS

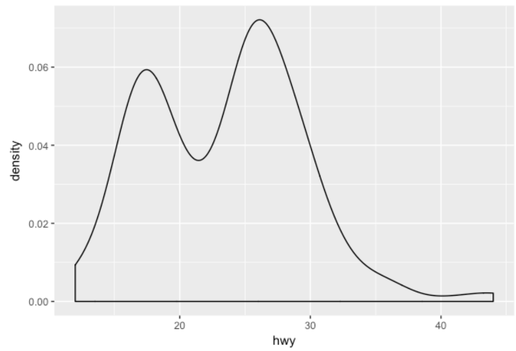
```
ggplot(data = mpg, aes(x = hwy)) +
  geom_histogram()

ggplot(data = mpg, aes(x = hwy)) +
  geom_freqpoly()

ggplot(data = mpg, aes(x = hwy)) +
  geom_density()
```



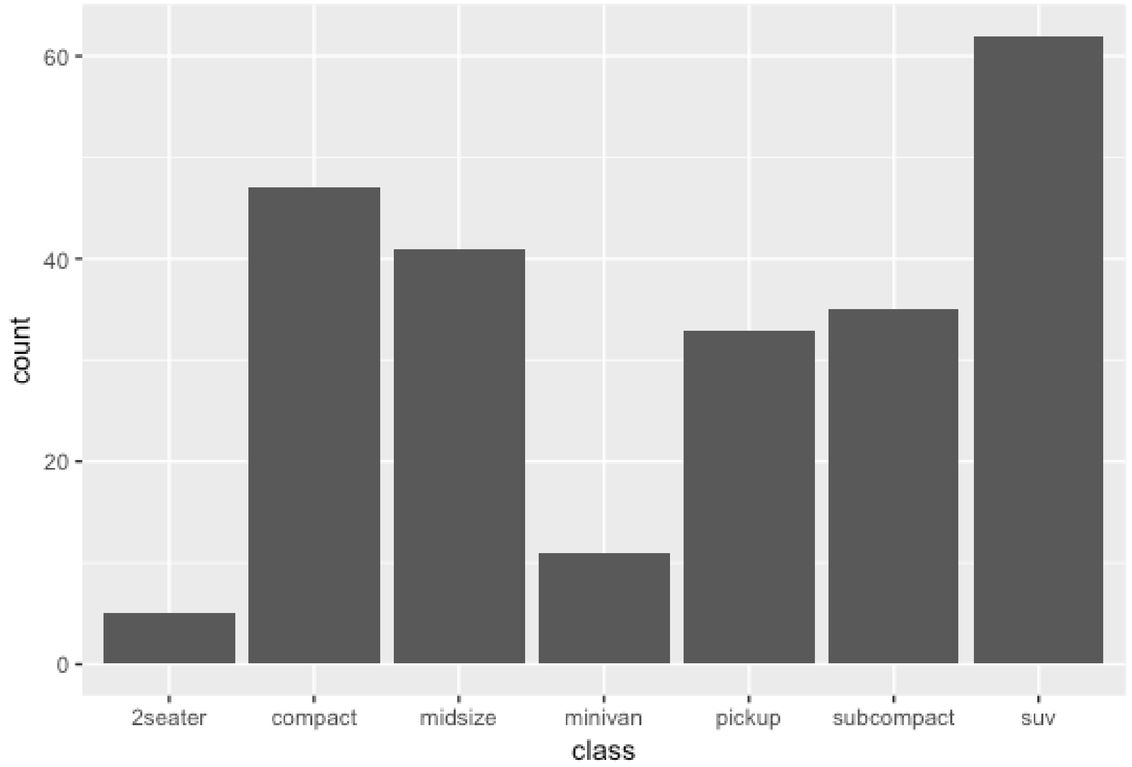




geom_abline
geom_bar
geom_bin2d
geom_blank
geom_boxplot
geom_contour
geom_count
geom_hex
geom_crossbar
geom_density
geom_density_2d
geom_dotplot
geom_errorharh
geom_freqpoly
geom_histogram
geom_jitter
geom_label
geom_map
geom_path
geom_point
geom_polygon
geom_quantile
geom_raster
geom_ribbon
geom_rug
geom_segment
geom_smooth
geom_violin

UNIVARIATE GEOMS

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



geom_abline
geom_bar
geom_bin2d
geom_blank
geom_boxplot
geom_contour
geom_count
geom_hex
geom_crossbar
geom_density
geom_density_2d
geom_dotplot
geom_errorharh
geom_freqpoly
geom_histogram
geom_jitter
geom_label
geom_map
geom_path
geom_point
geom_polygon
geom_quantile
geom_raster
geom_ribbon
geom_rug
geom_segment
geom_smooth
geom_violin

UNIVARIATE GEOMS

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

- This is called an **aes**thetic mapping argument
- Every geom requires a mapping argument
  - Some geoms require just one (x variable)
  - While other geoms require two (x & y variable)

geom_abline
geom_bar
geom_bin2d
geom_blank
geom_boxplot
geom_contour
geom_count
geom_hex
geom_crossbar
geom_density
geom_density_2d
geom_dotplot
geom_errorharh
geom_freqpoly

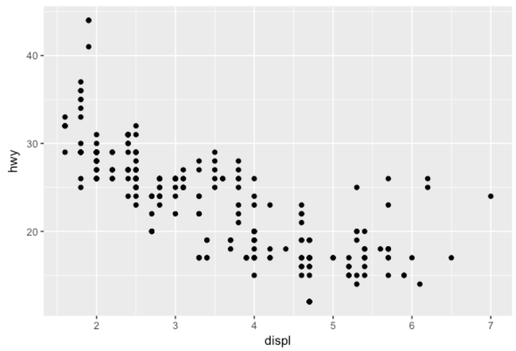
geom_histogram
geom_jitter
geom_label
geom_map
geom_path
geom_point
geom_polygon
geom_quantile
geom_raster
geom_ribbon
geom_rug
geom_segment
geom_smooth
geom_violin

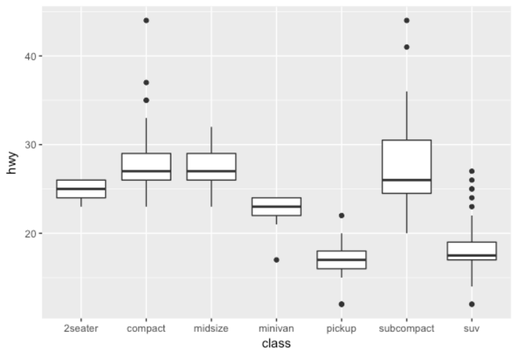
BIVARIATE GEOMS

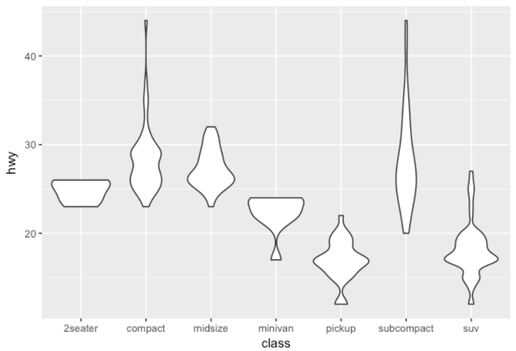
```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point()

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

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







geom_abline
geom_bar
geom_bin2d
geom_blank
geom_boxplot
geom_contour
geom_count
geom_hex
geom_crossbar
geom_density
geom_density_2d
geom_dotplot
geom_errorharh
geom_freqpoly
geom_histogram
geom_jitter
geom_label
geom_map
geom_path
geom_point
geom_polygon
geom_quantile
geom_raster
geom_ribbon
geom_rug
geom_segment
geom_smooth
geom_violin

YOUR TURN!

1. Create a chart that illustrates the distribution of the **cty** variable

2. Create a chart that shows the number of observations for each **manufacturer**

3. Create a scatter plot of **cty** vs **displ**

SOLUTION

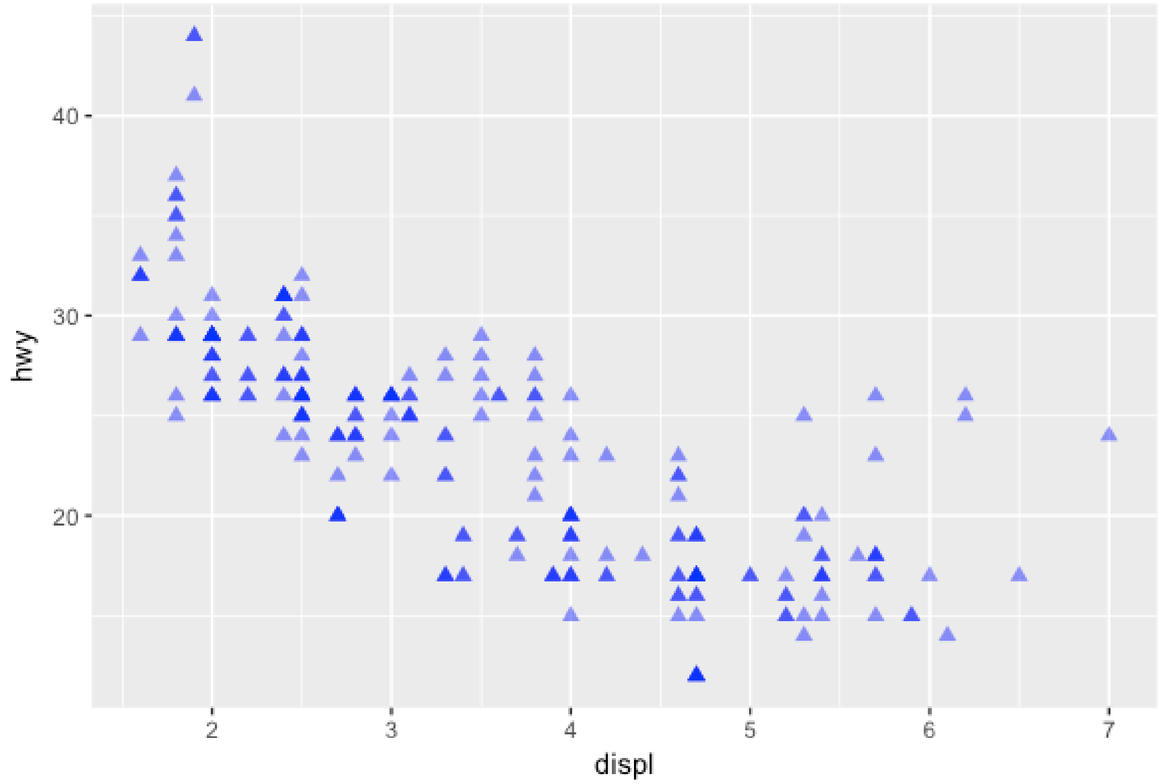
```
# distribution of cty variable
ggplot(data = mpg, aes(x = cty)) +
  geom_histogram()


# number of observations for each manufacturer
ggplot(data = mpg, aes(x = manufacturer)) +
  geom_bar()


# scatter plot for cty vs displ
ggplot(data = mpg, aes(x = displ, y = cty)) +
  geom_point()
```

NON-MAPPING AESTHETICS

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point(color = "blue", size = 2, shape = 17, alpha = .5)
```



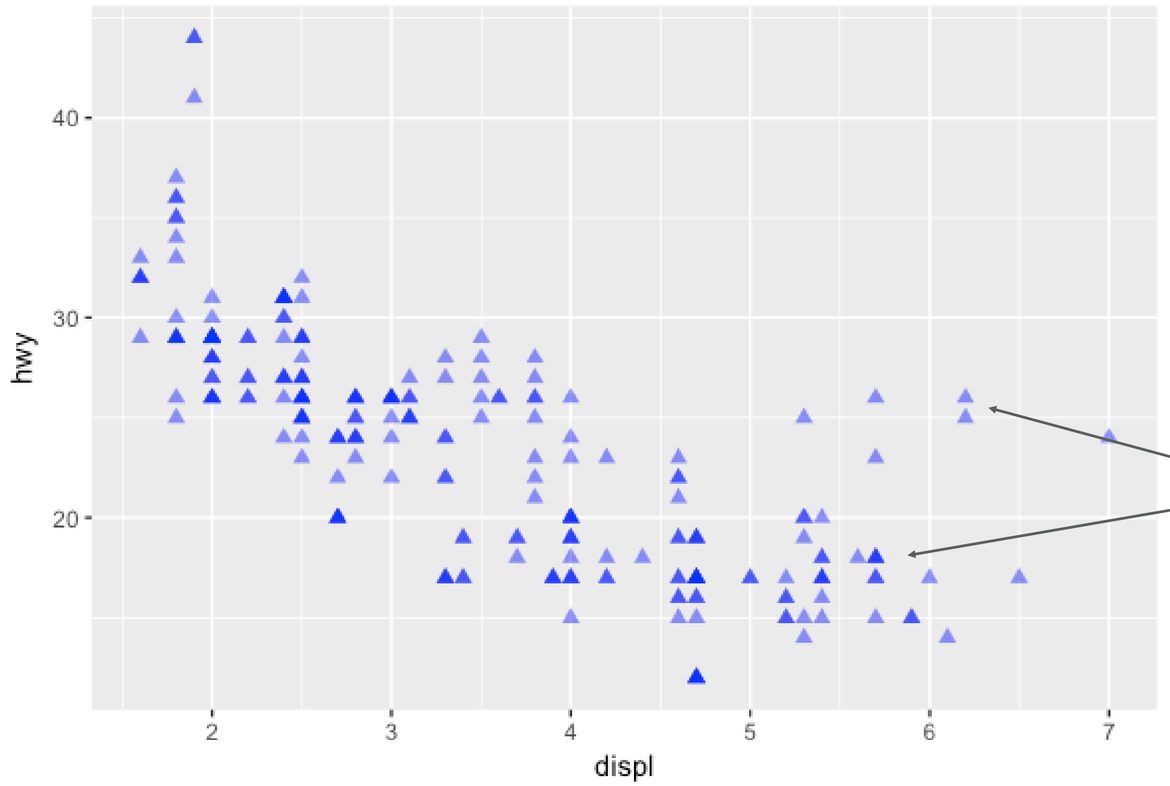
We can also change other visual aesthetics in our graphics

- color
- size
- shape (0-25 ?**pch**)
- opacity

Lots of color and shape options; just google and you'll find plenty of references

NON-MAPPING AESTHETICS

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point(color = "blue", size = 2, shape = 17, alpha = .5)
```



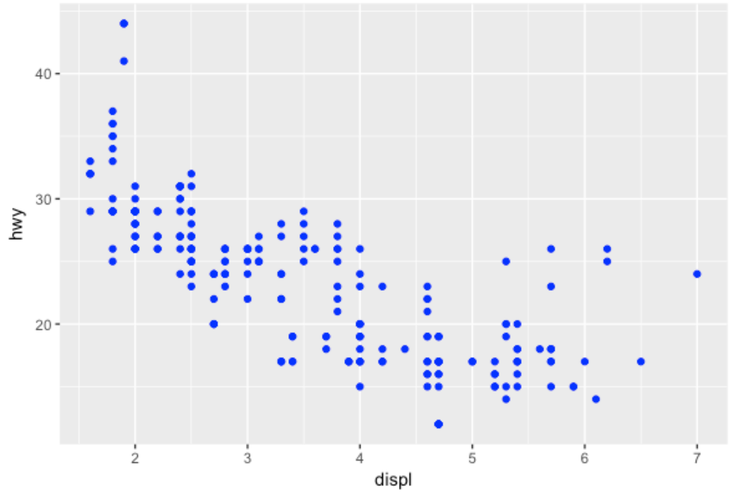
We can also change other visual aesthetics in our graphics

- color
- size
- sh△pe
- opacity

- Why are some points darker than others?
- Try **`geom_jitter`** in place of **`geom_point`**
- What do you think **`geom_jitter`** does?

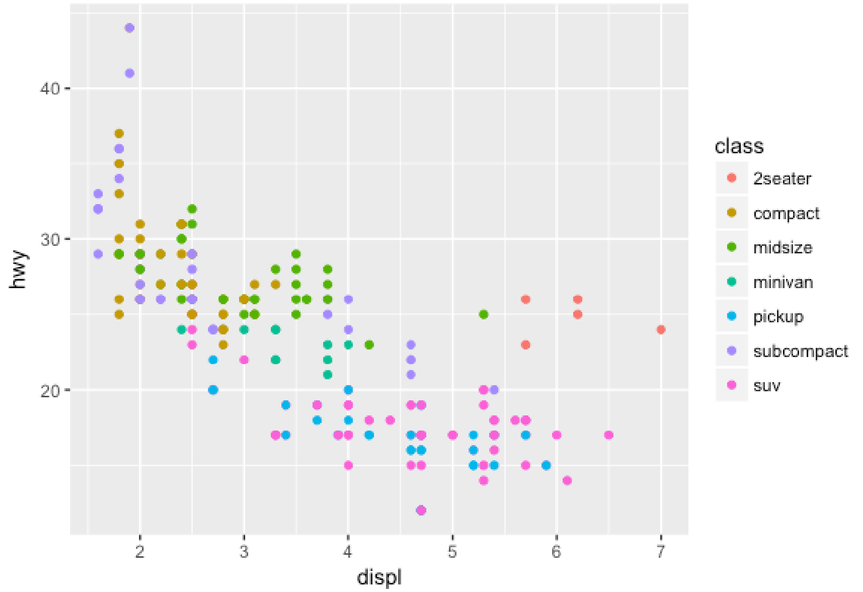
ADDING A 3RD DIMENSION

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



By moving the **`color`** argument to within **aes()**, we can map a 3rd variable to our plot

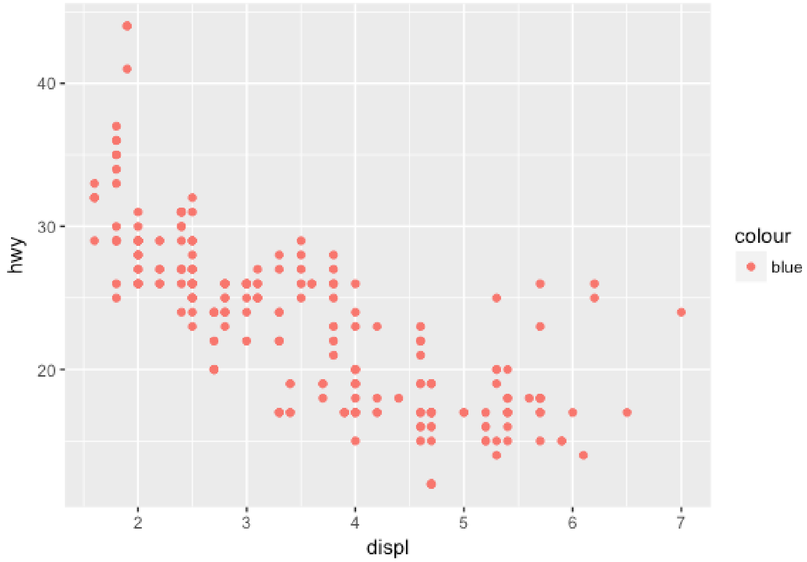
```
ggplot(data = mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point()
```



ADDING A 3RD DIMENSION

A common error…what happened???

```
ggplot(data = mpg, aes(x = displ, y = hwy, color = “blue”)) +
  geom_point()
```



YOUR TURN!

1. Which variables in mpg are categorical? Which variables are continuous? (Hint: type **?mpg** to read the documentation for the dataset). How can you see this information when you run mpg?

2. Map a continuous variable to **`color`**, **`size`**, and **`shape`**. How do these aesthetics behave differently for categorical vs. continuous variables?

3. What happens if you map the same variable to multiple aesthetics?

4. What does the **`stroke`** aesthetic do? What shapes does it work with? (Hint: use **?geom_point**)

5. What happens if you map an aesthetic to something other than a variable name, like **aes(colour = displ < 5)**?

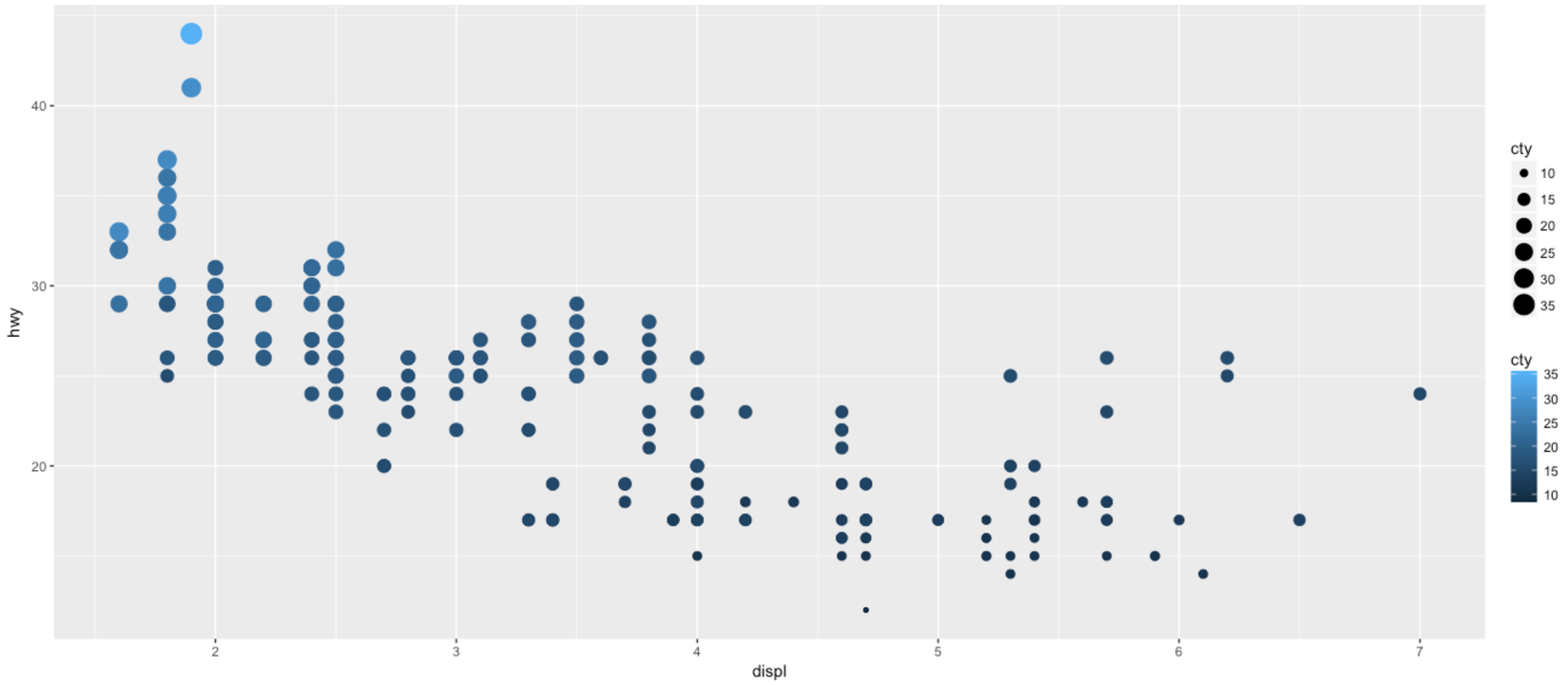
SOLUTION

```
# 1. which variables are continuous vs categorical
?mpg

mpg
# A tibble: 234 × 11
   manufacturer      model displ  year   cyl      trans   drv   cty   hwy    fl   class
          <chr>      <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>   <chr>
1         audi         a4   1.8  1999     4   auto(l5)     f    18    29     p compact
2         audi         a4   1.8  1999     4 manual(m5)     f    21    29     p compact
3         audi         a4   2.0  2008     4 manual(m6)     f    20    31     p compact
4         audi         a4   2.0  2008     4   auto(av)     f    21    30     p compact
5         audi         a4   2.8  1999     6   auto(l5)     f    16    26     p compact
6         audi         a4   2.8  1999     6 manual(m5)     f    18    26     p compact
7         audi         a4   3.1  2008     6   auto(av)     f    18    27     p compact
8         audi a4 quattro   1.8  1999     4 manual(m5)     4    18    26     p compact
9         audi a4 quattro   1.8  1999     4   auto(l5)     4    16    25     p compact
10        audi a4 quattro   2.0  2008     4 manual(m6)     4    20    28     p compact
# ... with 224 more rows
```

SOLUTION

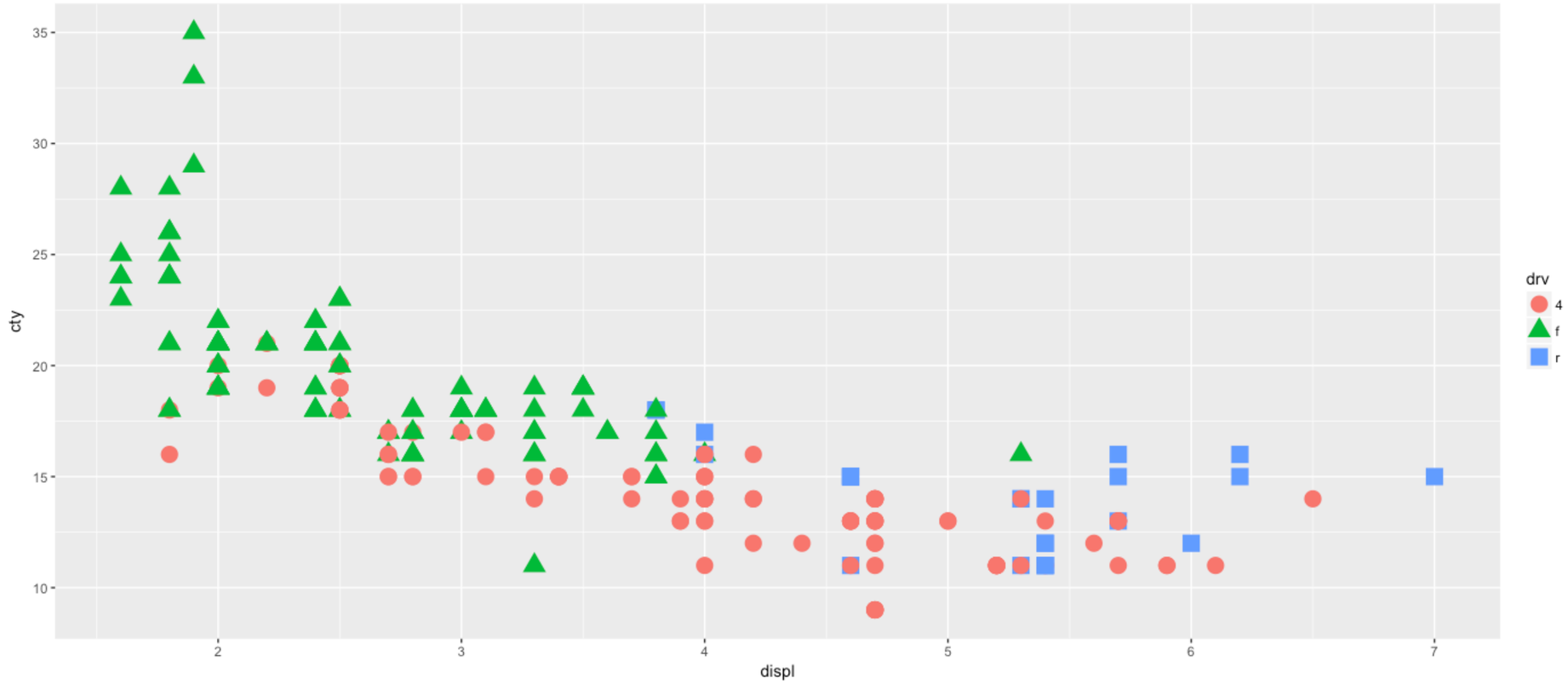
```
# 2. Map a continuous variable to color, size, and shape. How do these aesthetics behave
#    differently for categorical vs. continuous variables?
ggplot(mpg, aes(displ, hwy, color = cty, size = cty)) +
  geom_point()
```



Error: A continuous variable can not be mapped to shape

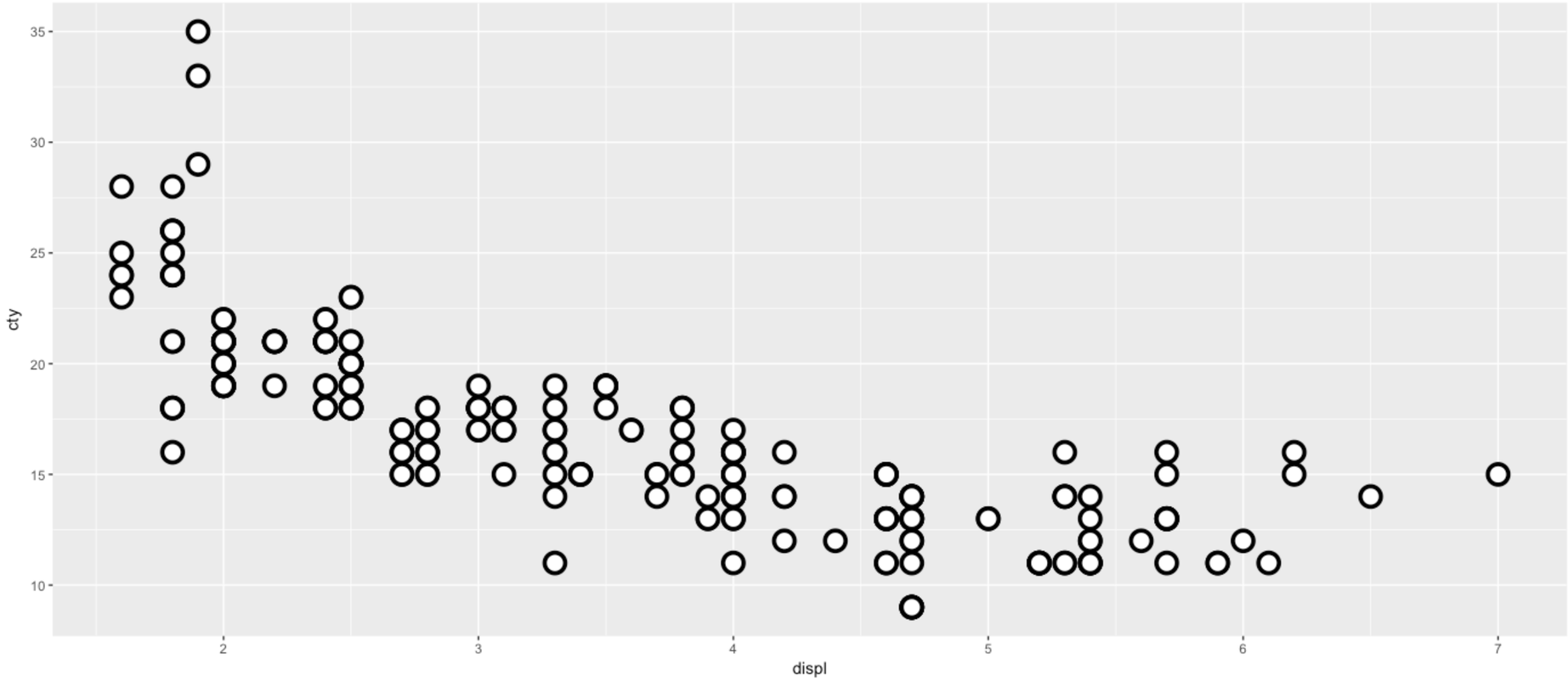
SOLUTION

```
# 3. What happens if you map the same variable to multiple aesthetics?
ggplot(mpg, aes(displ, cty, color = drv, shape = drv)) +
  geom_point()
```



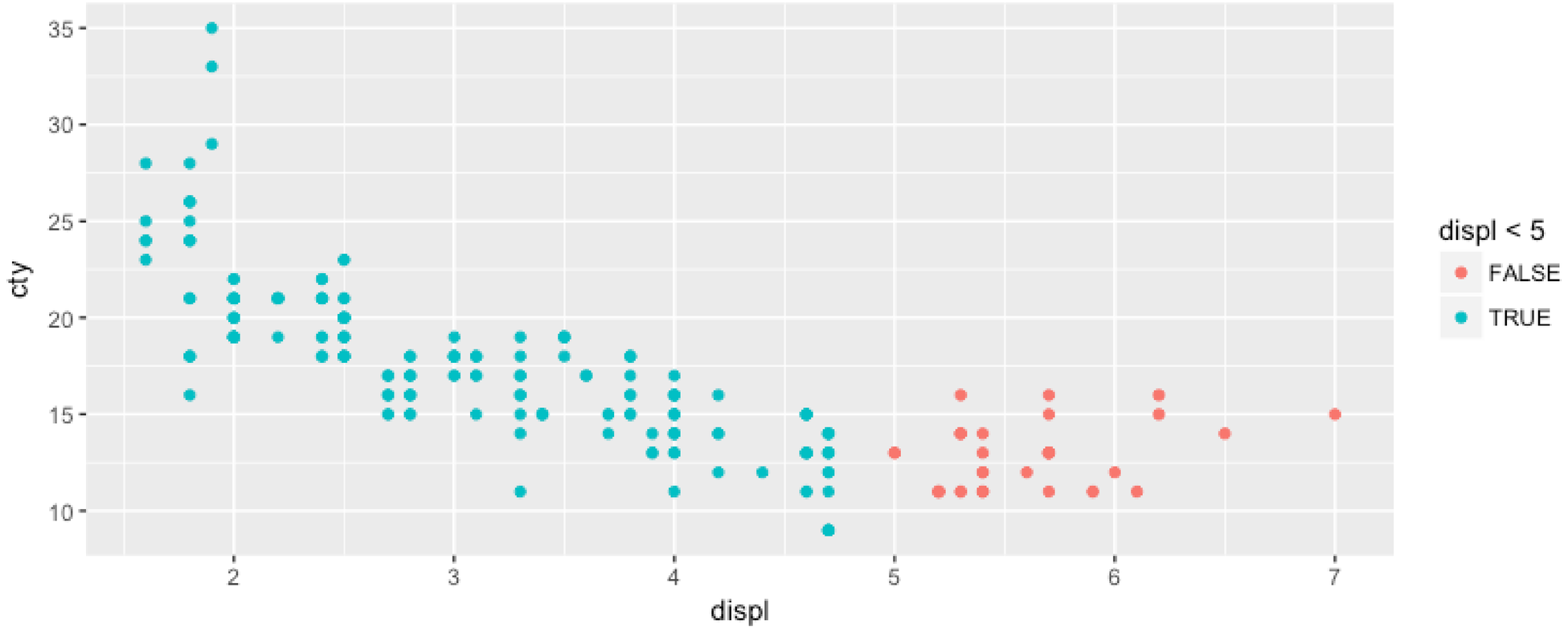
SOLUTION

```
# 4. What does the stroke aesthetic do? What shapes does it work with?
ggplot(mpg, aes(displ, cty)) +
  geom_point(shape = 21, colour = "black", fill = "white", size = 5, stroke = 2)
```



SOLUTION

```
# 5. What happens if you map an aesthetic to something other than a variable name, like
#    aes(colour = displ < 5)
ggplot(mpg, aes(displ, cty, color = displ < 5)) +
  geom_point()
```

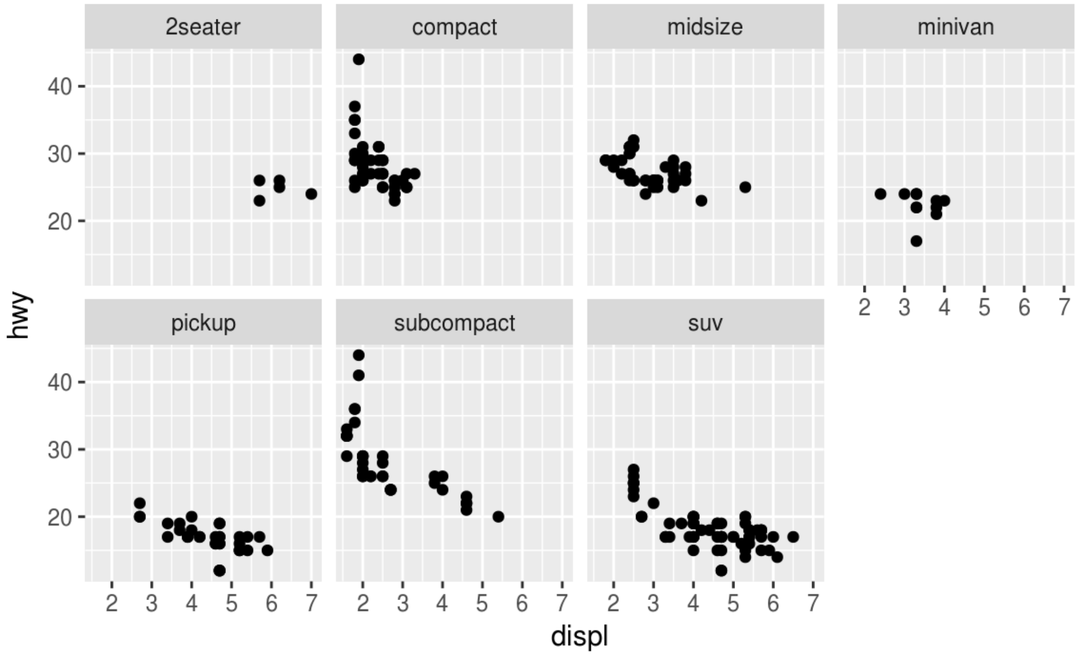


FACETS

FACETS = SMALL MULTIPLES

- The **facet** functions provide a simple way to create small multiples
  - **facet_wrap**: primarily used to create small multiples based on a single variable
  - **facet_grid**: primarily used to create a small multiples grid based on two variables

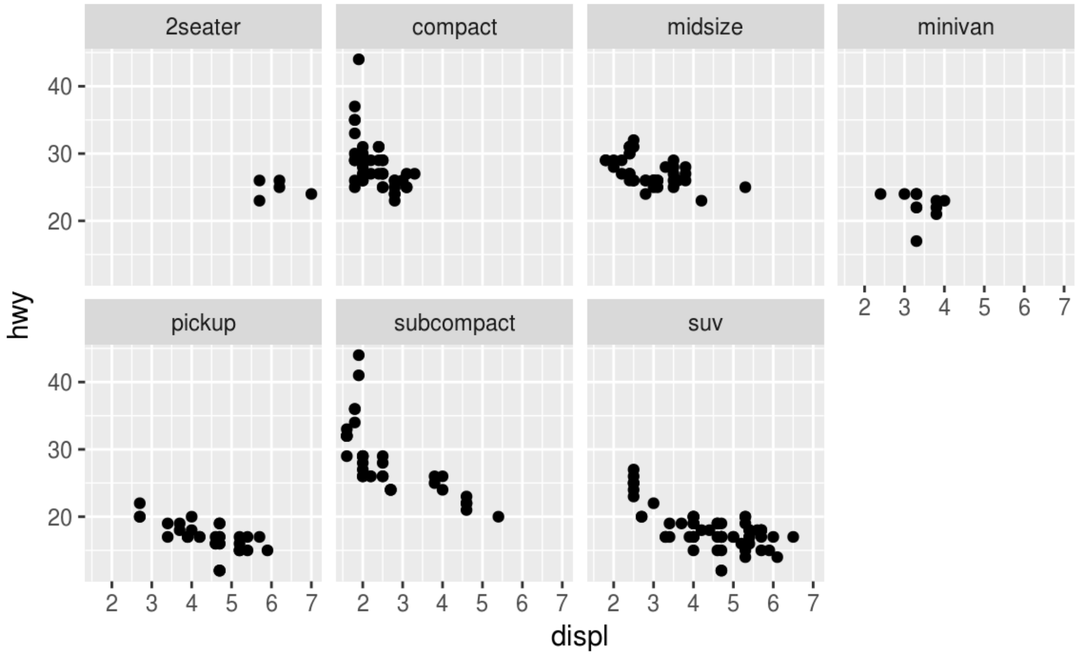
```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~ class, nrow = 2)
```



FACETS = SMALL MULTIPLES

- The **facet** functions provide a simple way to create small multiples
  - **facet_wrap**: primarily used to create small multiples based on a single variable
  - **facet_grid**: primarily used to create a small multiples grid based on two variables

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~ class, nrow = 2)
```

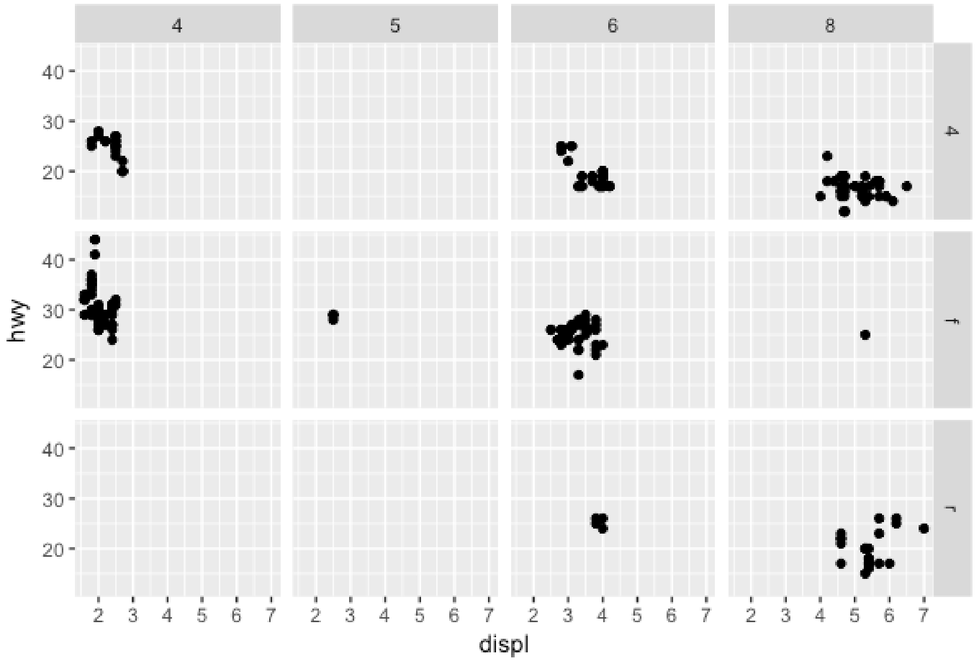


- use **nrow** or **ncol** to specify dimensions
- **?facet_wrap** to see other arguments to control the output

FACETS = SMALL MULTIPLES

- The **facet** functions provide a simple way to create small multiples
  - **facet_wrap**: primarily used to create small multiples based on a single variable
  - **facet_grid**: primarily used to create a small multiples grid based on two variables

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(drv ~ cyl)
```

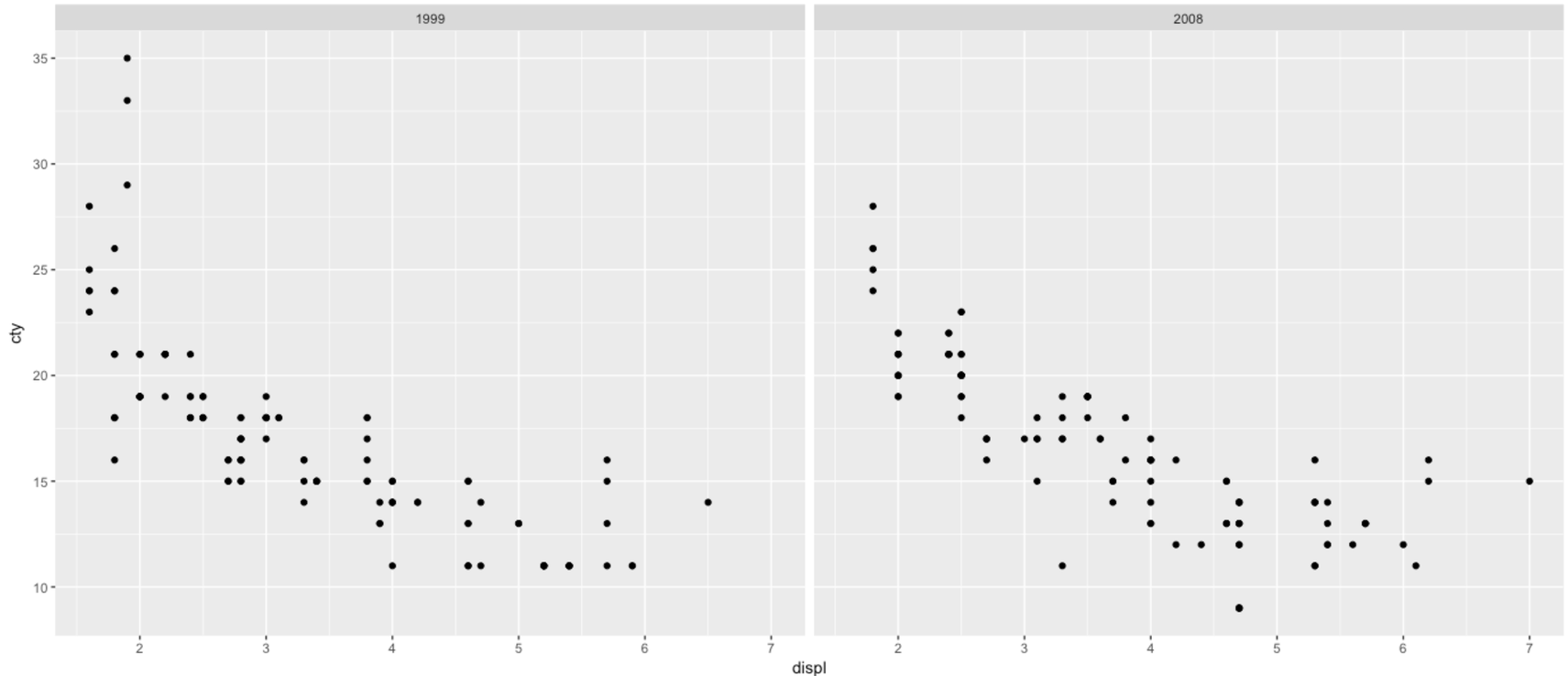


YOUR TURN!

1. Create a scatter plot of **displ** vs **cty** facetted by **year**.
2. Create a scatter plot of **displ** vs **cty** facetted by **year** and **cyl**.
3. How does placement within **facet_grid(cyl ~ year)** affect the output?
4. How does **facet_grid(cyl ~ year)** differ from **facet_grid(~ year + cyl)**?
5. What do the **`scales`** and **`space`** arguments do?

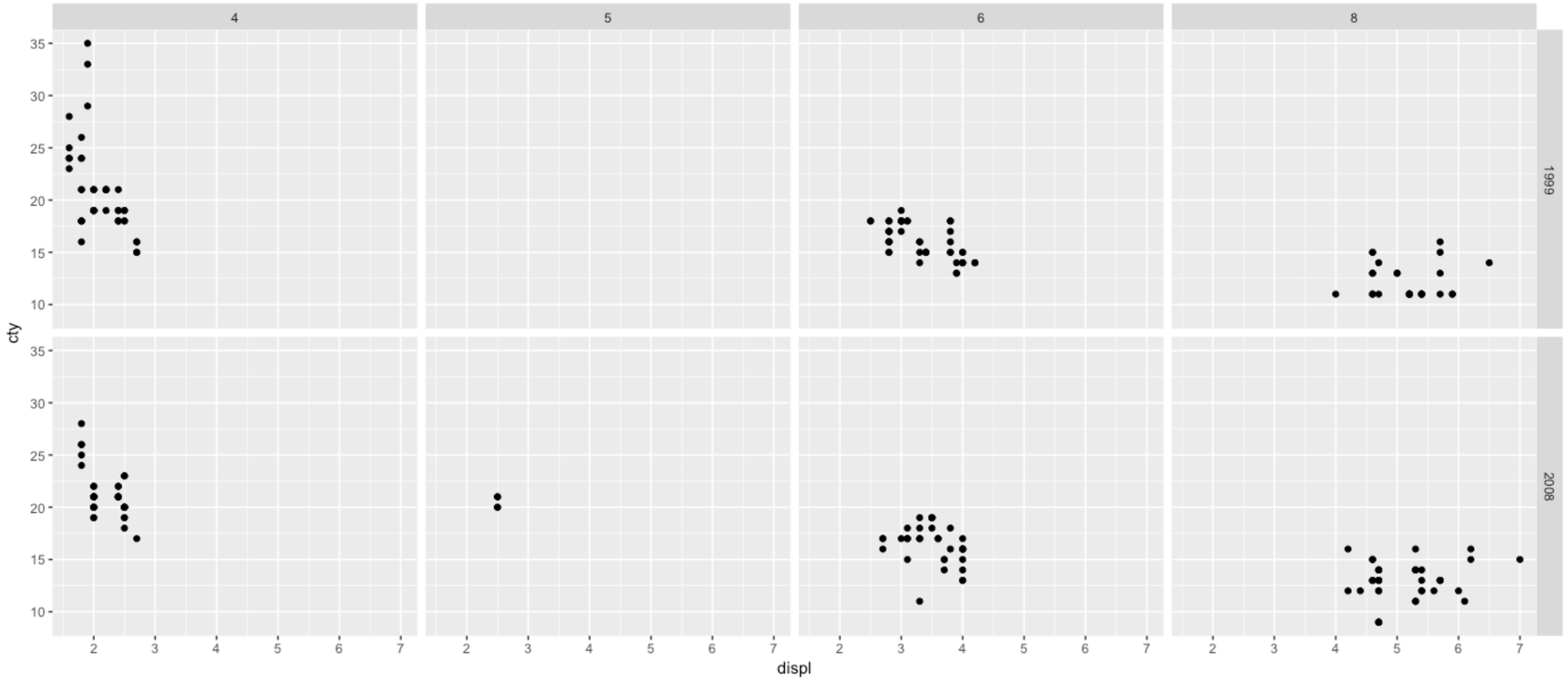
SOLUTION

```
# 1. Create a scatter plot of displ vs cty facetted by year.
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_wrap(~ year)
```



SOLUTION

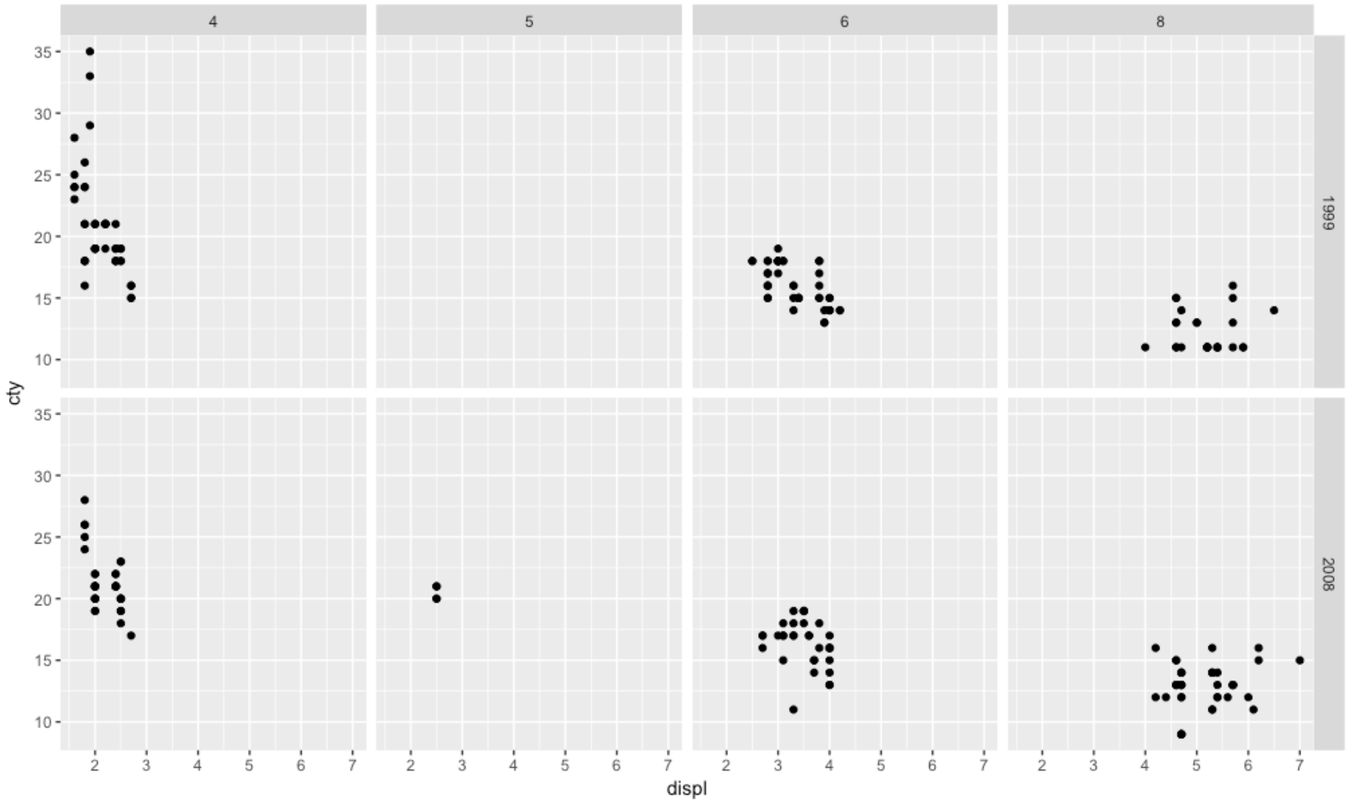
```
# 2. Create a scatter plot of displ vs cty facetted by year and cyl
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_grid(year ~ cyl)
```



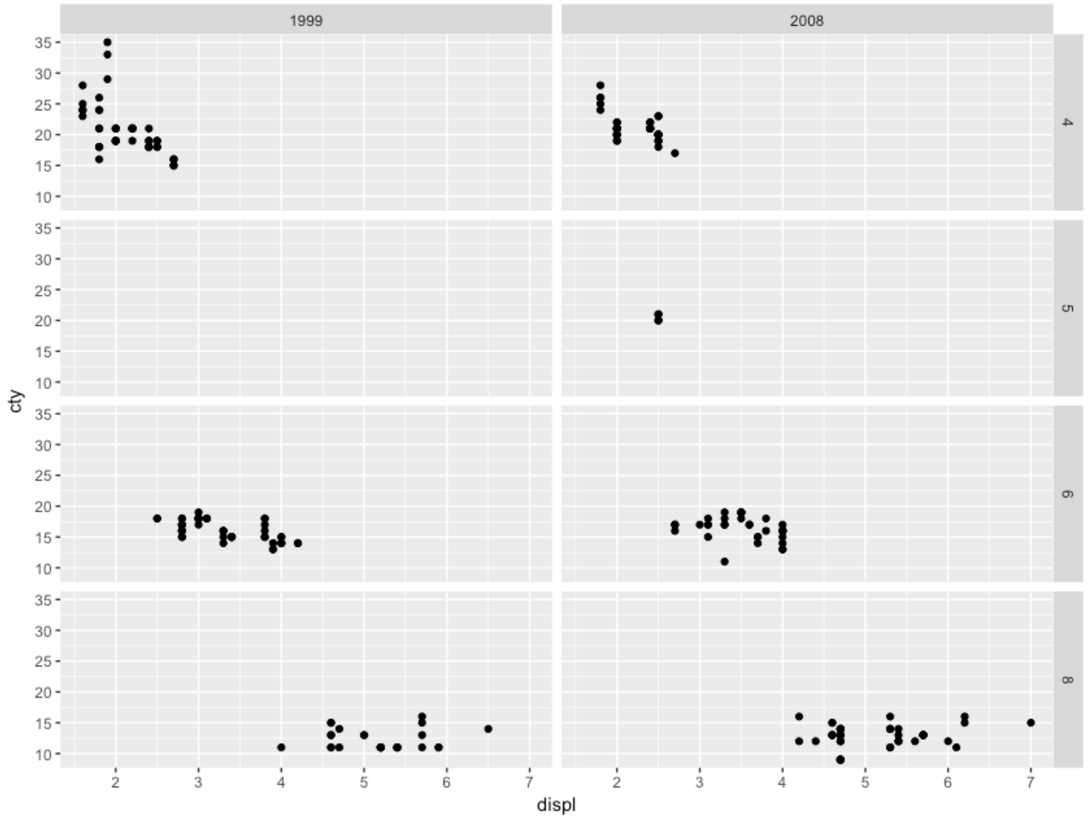
SOLUTION

3. How does placement within facet_grid(cyl ~ year) affect the output?

facet_grid(year ~ cyl)

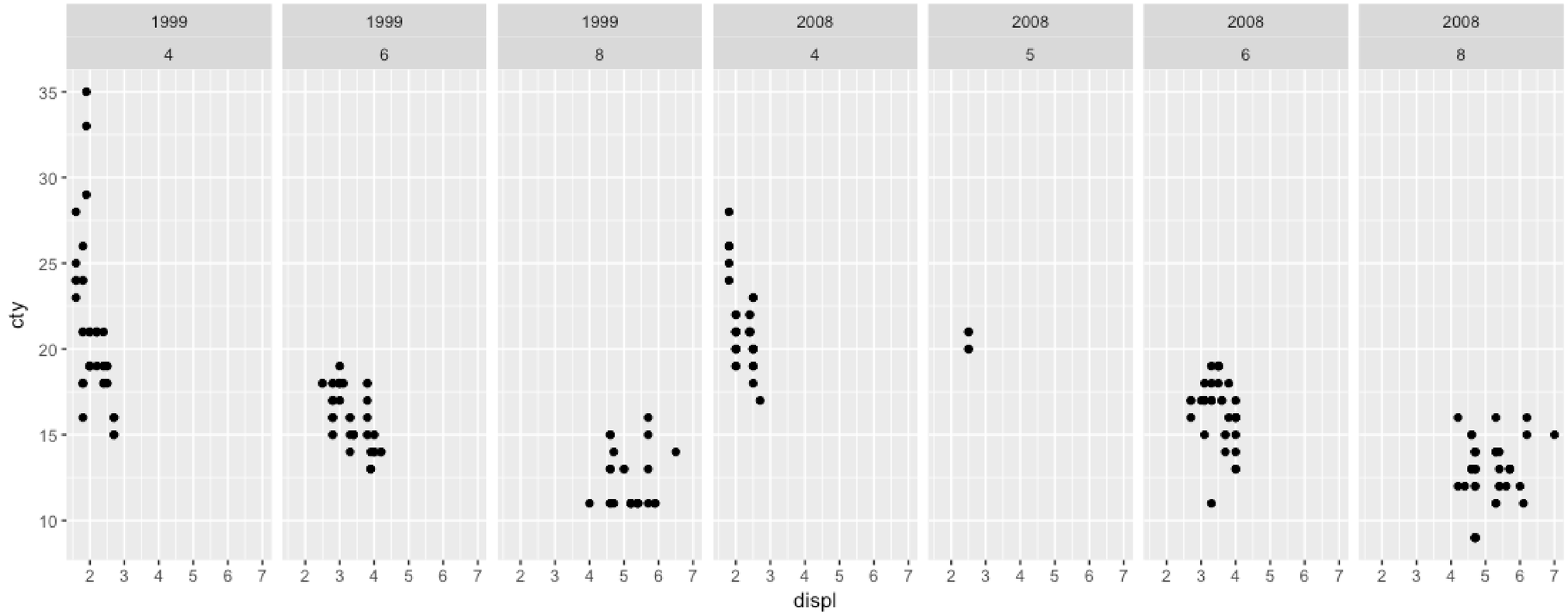


facet_grid(cyl ~ year)



SOLUTION

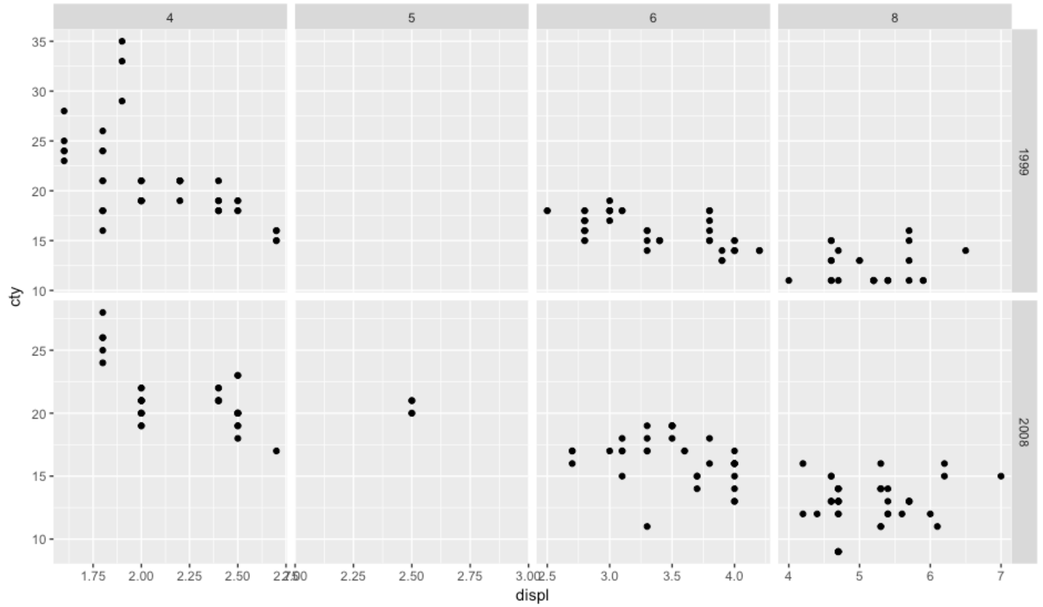
```
# 4. How does facet_grid(cyl ~ year) differ from facet_grid(~ year + cyl)?
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_grid(~ year + cyl)
```



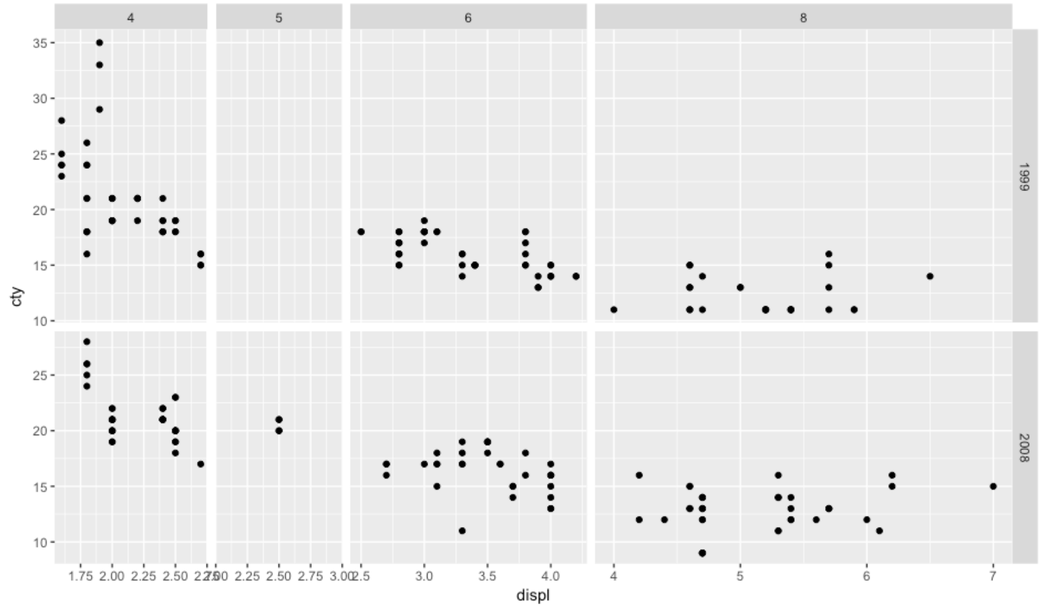
SOLUTION

5. What do the scales and space arguments do

```
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_grid(year ~ cyl, scales = "free")
```



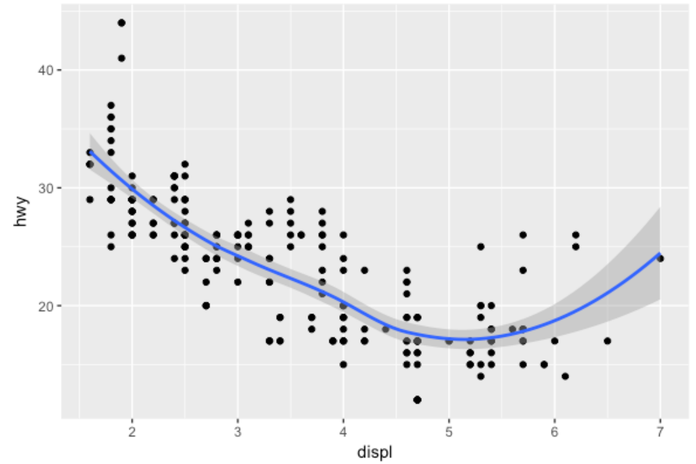

```
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_grid(year ~ cyl, scales = "free", space = "free")
```



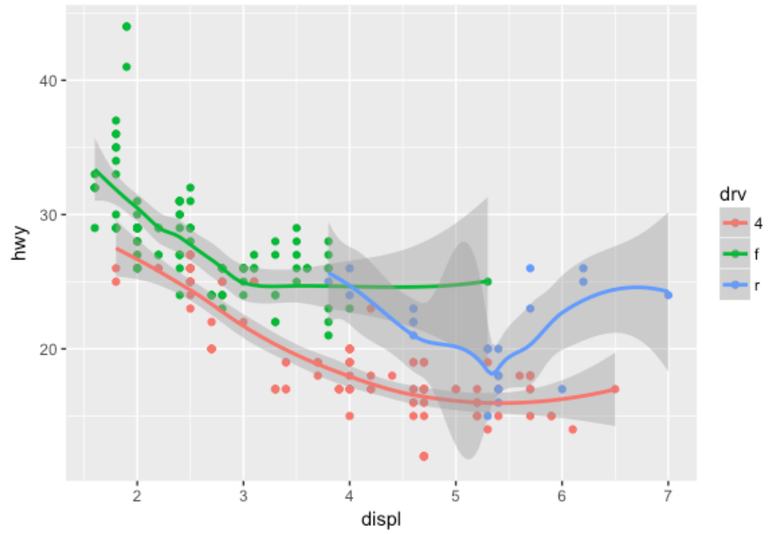
OVERPLOTTING

LAYERING HELPS DISPLAY PATTERNS

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

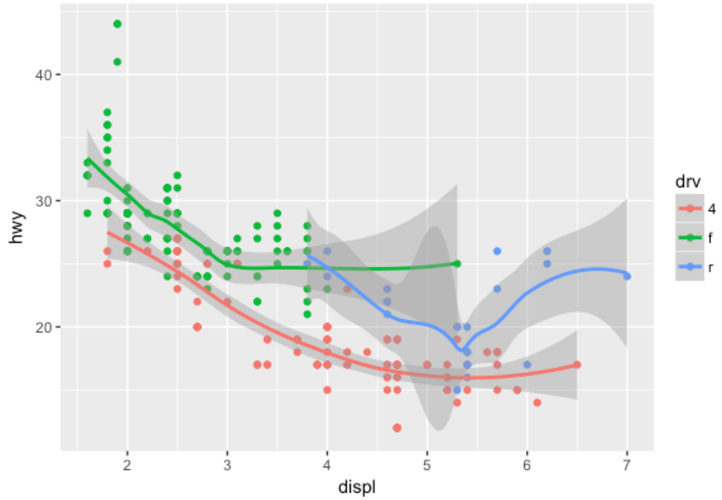



```
mpg %>%
  ggplot(aes(x = displ, y = hwy, color = drv)) +
    geom_point() +
    geom_smooth()
```

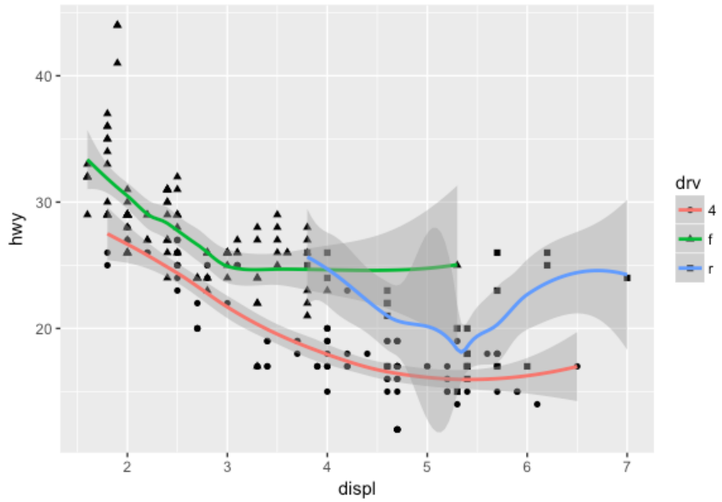


LAYERING HELPS DISPLAY PATTERNS

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



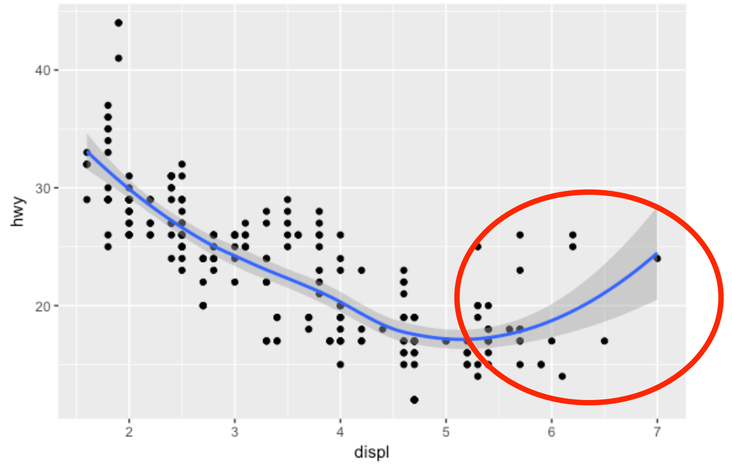

```
mpg %>%
   ggplot(aes(x = displ, y = hwy)) +
    geom_point(aes(shape = drv)) +
    geom_smooth(aes(color = drv))
```



aes mapping can be done in **ggplot()** or **geom_xx()**

LAYERING HELPS ID ABNORMALITIES

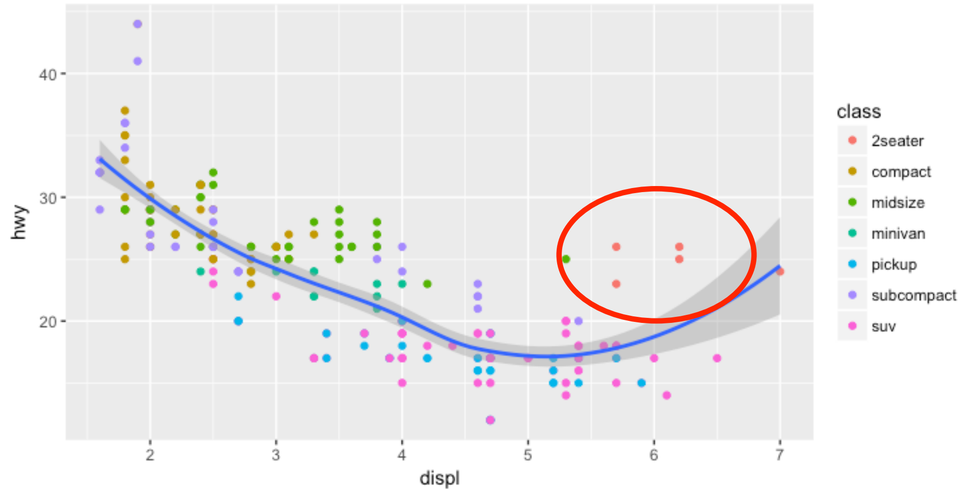
```
mpg %>%
    ggplot(aes(x = displ, y = hwy)) +
      geom_point() +
      geom_smooth()
```



What's driving this upward swing?

LAYERING HELPS ID ABNORMALITIES

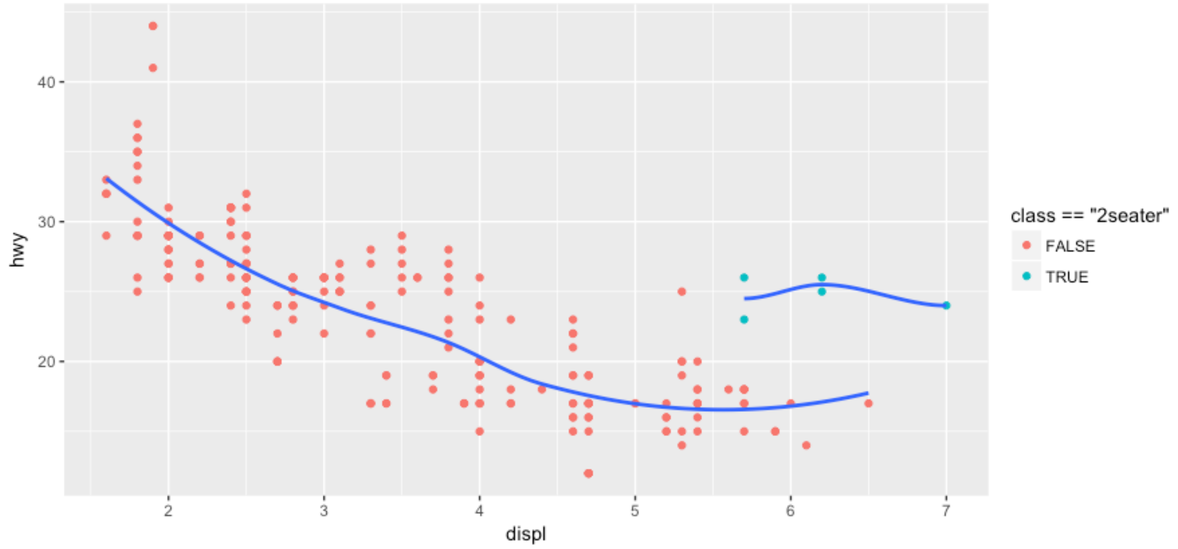
```
mpg %>%
    ggplot(aes(x = displ, y = hwy)) +
      geom_point(aes(color = class)) +
      geom_smooth()
```



Looks like it could be the 2 seaters but we need to verify

LAYERING HELPS ID ABNORMALITIES

```
mpg %>%
    ggplot(aes(x = displ, y = hwy)) +
        geom_point(aes(color = class == "2seater")) +
      geom_smooth(data = filter(mpg, class == "2seater"), se = FALSE) +
      geom_smooth(data = filter(mpg, class != "2seater"), se = FALSE)
```



YOUR TURN!

1. Over plot:

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

with `geom_jitter(width = .2, alpha = .5)`

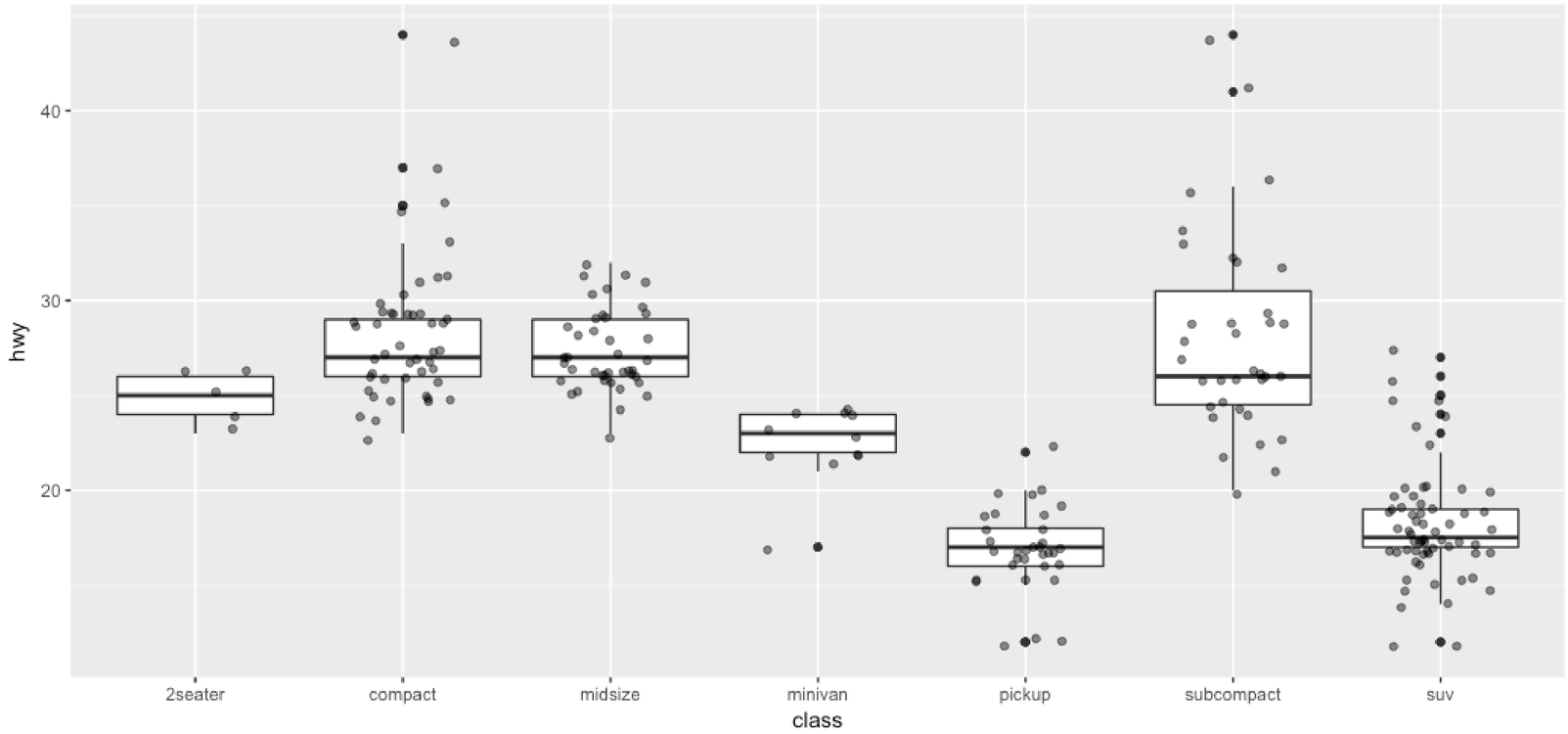
2. If you add `geom_rug()` to

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

what does this tell you?

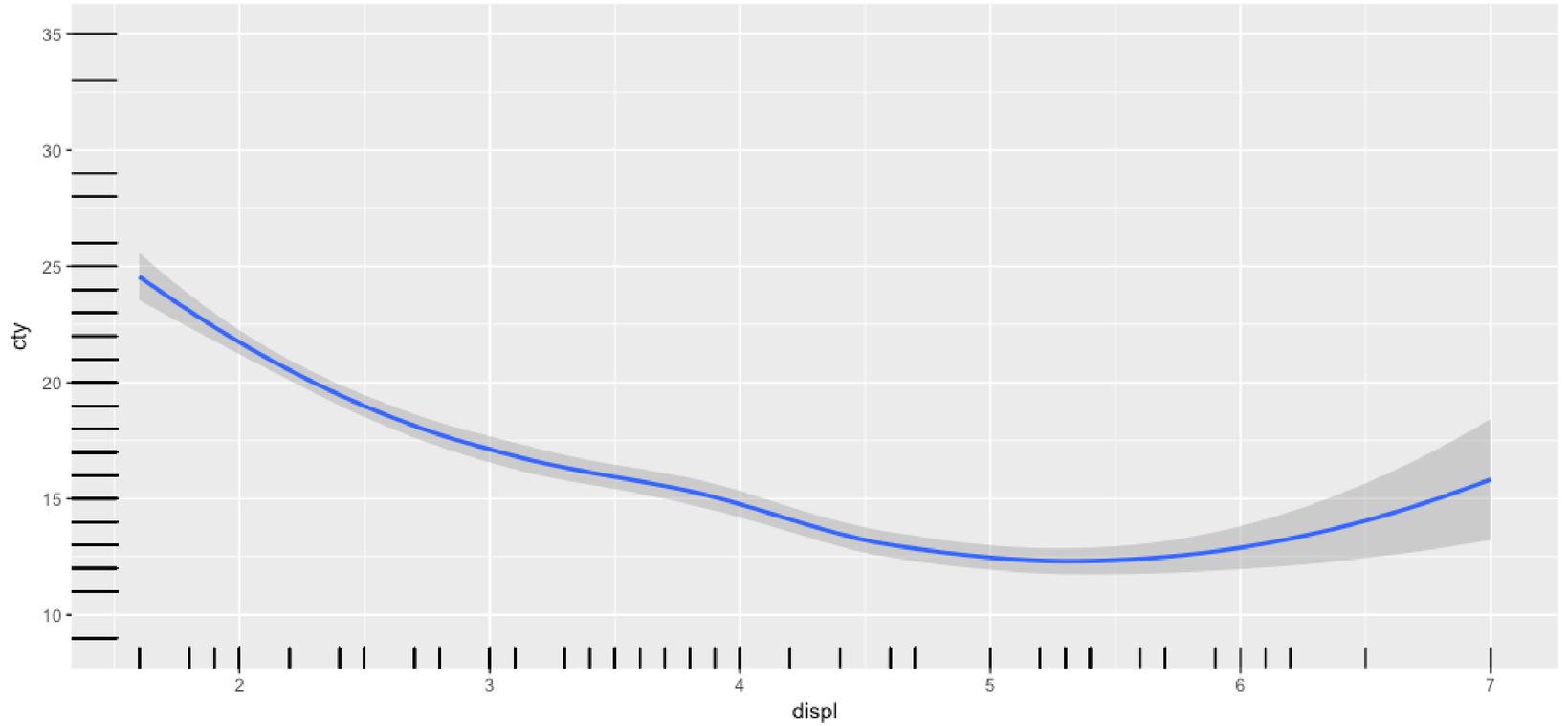
SOLUTION

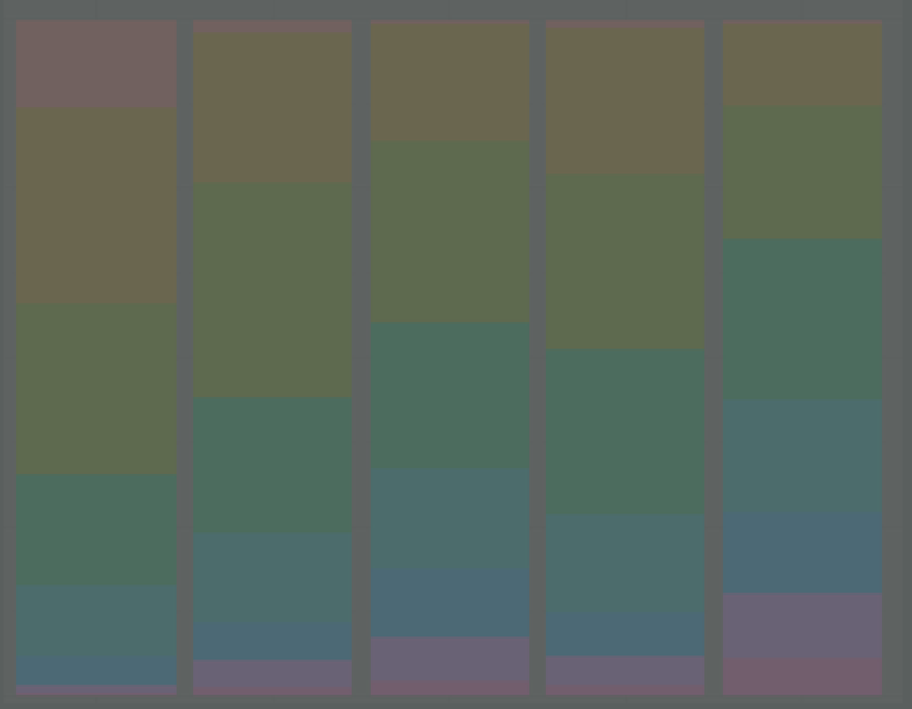
```
# 1. Overplot boxplot with jitter
ggplot(mpg, aes(class, hwy)) +
  geom_boxplot() +
  geom_jitter(width = .25, alpha = .5)
```



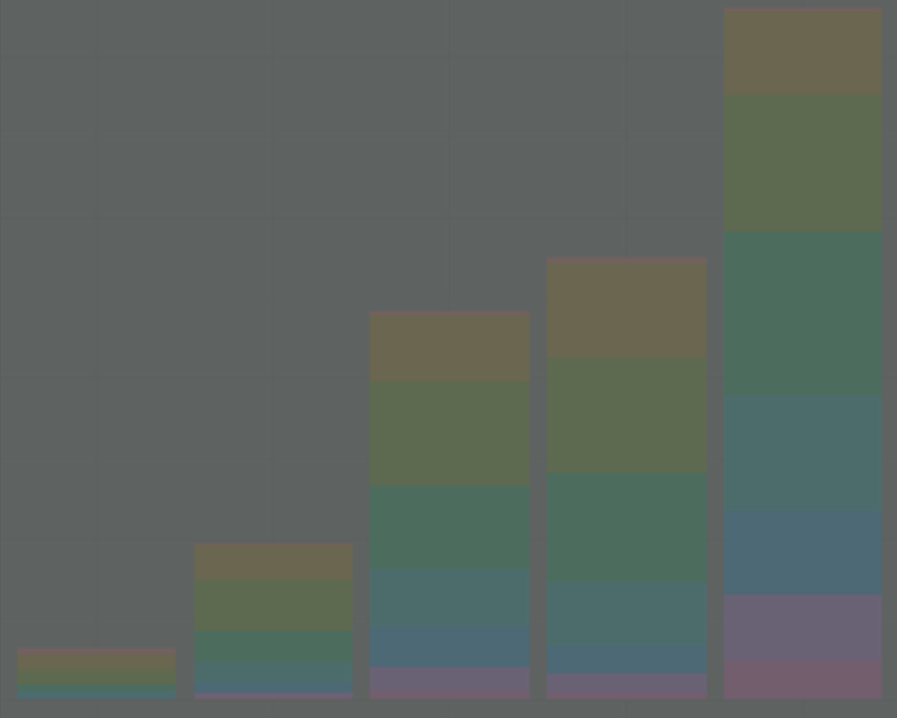
SOLUTION

```
# 2. add geom_rug to smoother
ggplot(mpg, aes(displ, cty)) +
  geom_smooth() +
  geom_rug()
```





POSITIONING

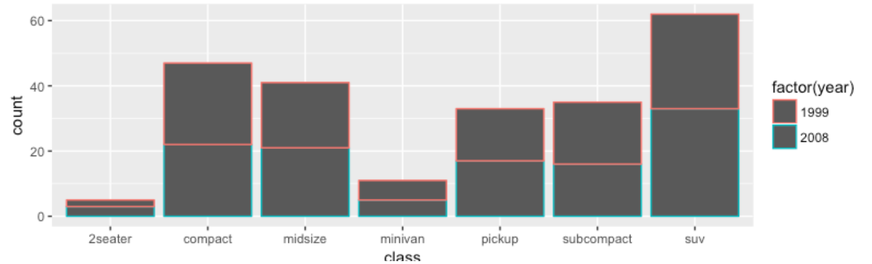


BAR CHARTS

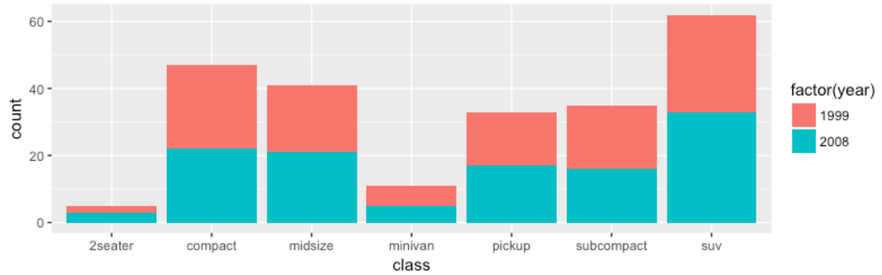
- All **geoms** have a position argument but rarely will you need to adjust it
- **geom_bar** and a few others benefit from its use though:
  - position = “stack” (default)
  - position = “fill”
  - position = “dodge”

BAR CHARTS

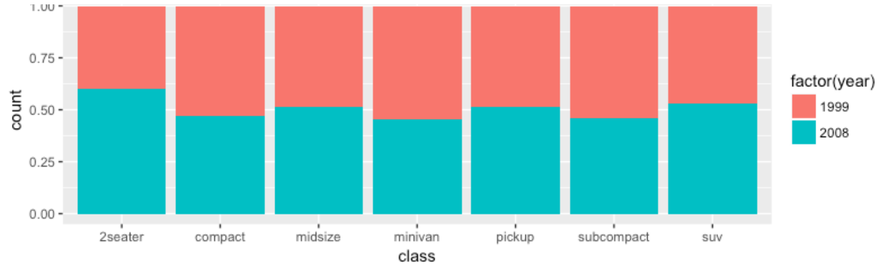
```
ggplot(data = mpg, aes(class, color = factor(year))) +
  geom_bar()
```



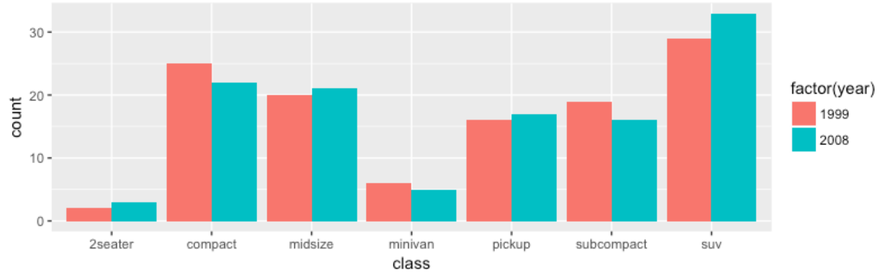

```
ggplot(data = mpg, aes(class, fill = factor(year))) +
  geom_bar()
```




```
ggplot(data = mpg, aes(class, fill = factor(year))) +
  geom_bar(position = “fill”)
```




```
ggplot(data = mpg, aes(class, fill = factor(year))) +
  geom_bar(position = “dodge”)
```

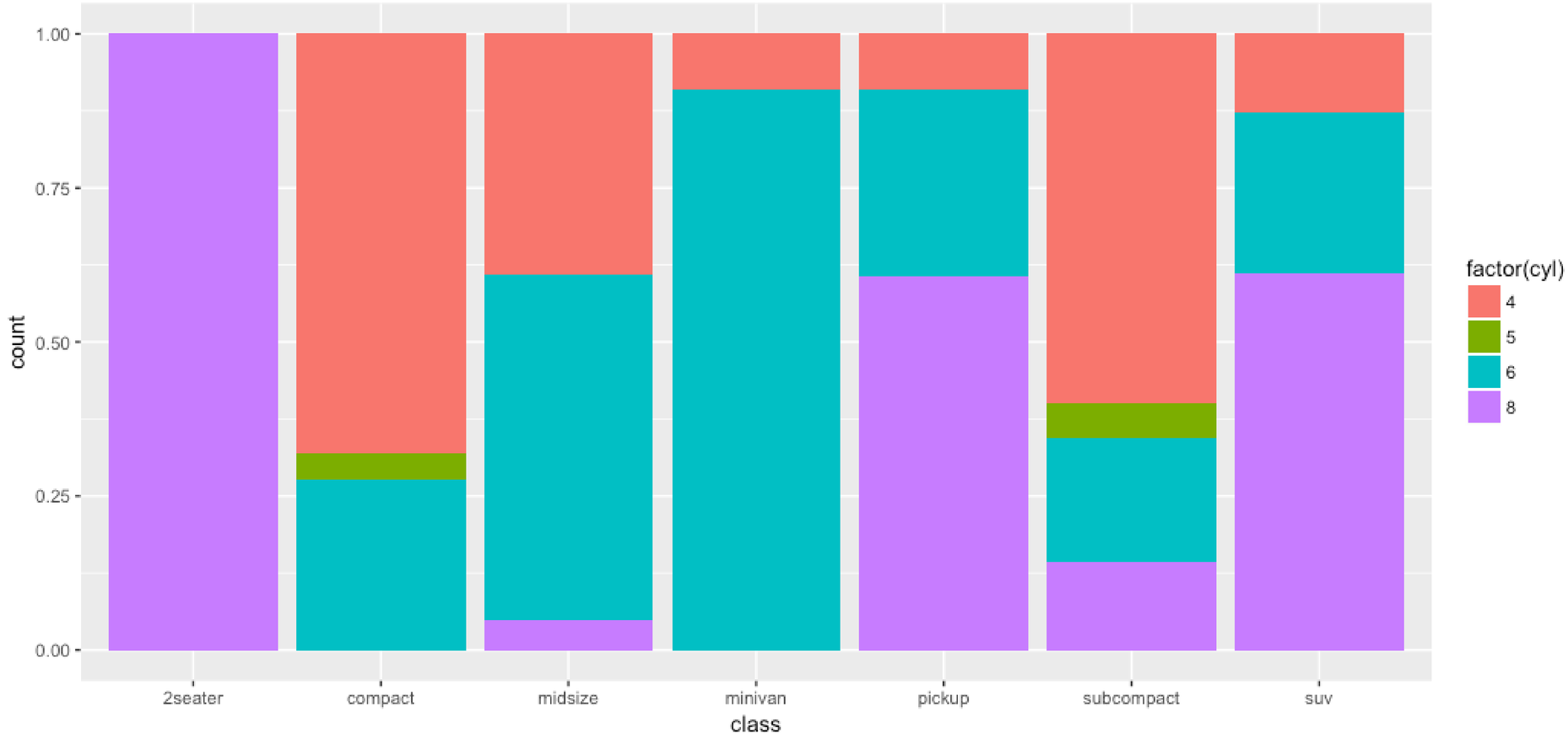


YOUR TURN!

Use **geom_bar** and the different **position** arguments to assess how the **cyl** variable is distributed within each **class**.

SOLUTION

```
# One potential solution
ggplot(mpg, aes(class, fill = factor(cyl))) +
  geom_bar(position = "fill")
```



COORDINATE SYSTEM

MANIPULATING YOUR COORDINATES

- There are ***many*** options to manipulate and adjust the coordinate system but some basic ones include:

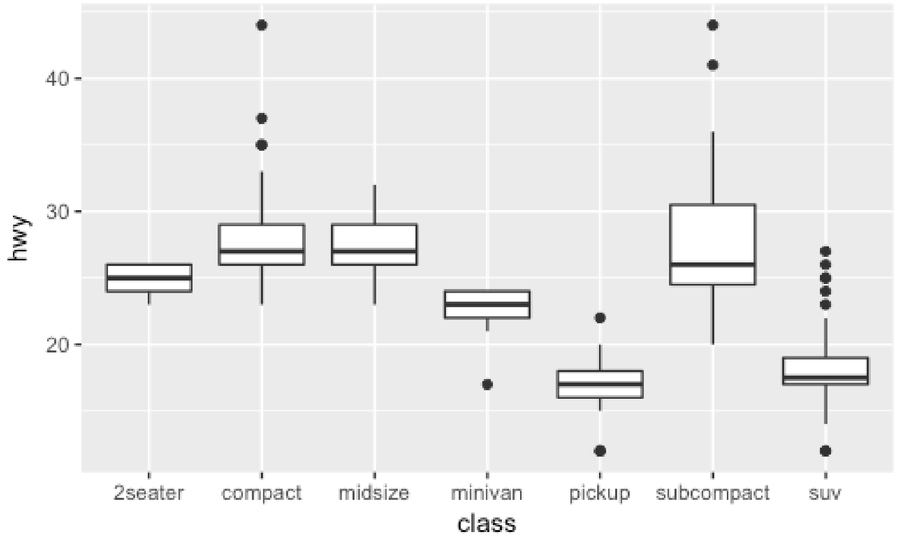
MANIPULATING YOUR COORDINATES

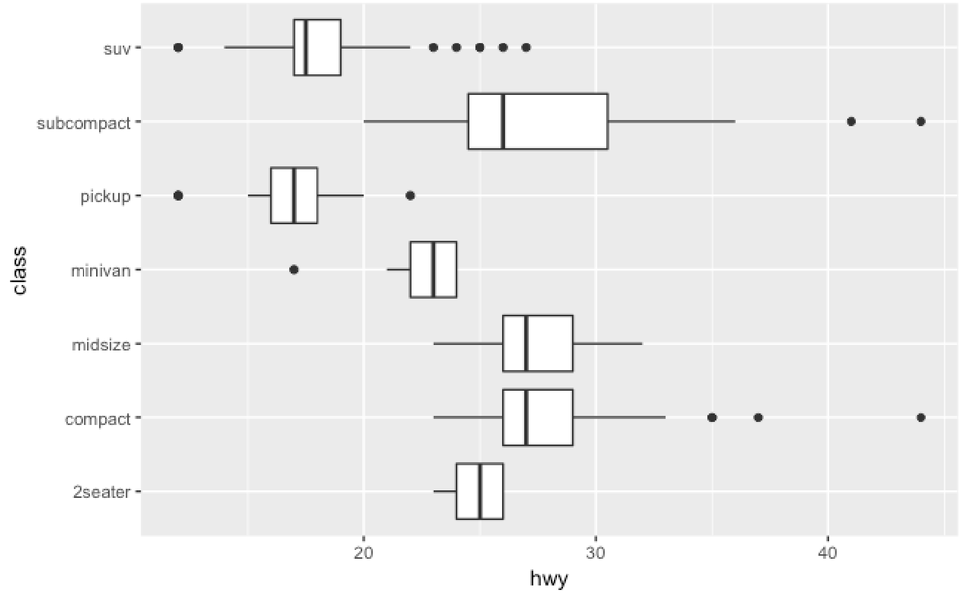
- There are ***many*** options to manipulate and adjust the coordinate system but some basic ones include:
  - flipping the coordinates

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

  geom_boxplot()

# bottom
ggplot(data = mpg, aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```

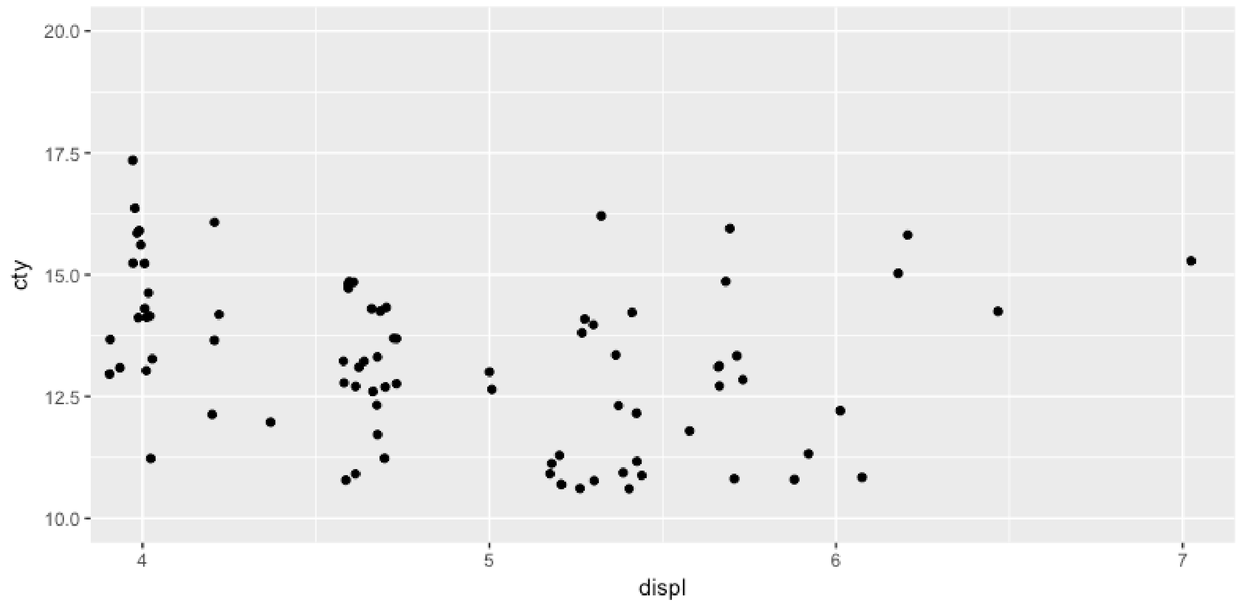




MANIPULATING YOUR COORDINATES

- There are *many* options to manipulate and adjust the coordinate system but some basic ones include:
    - zooming in or out

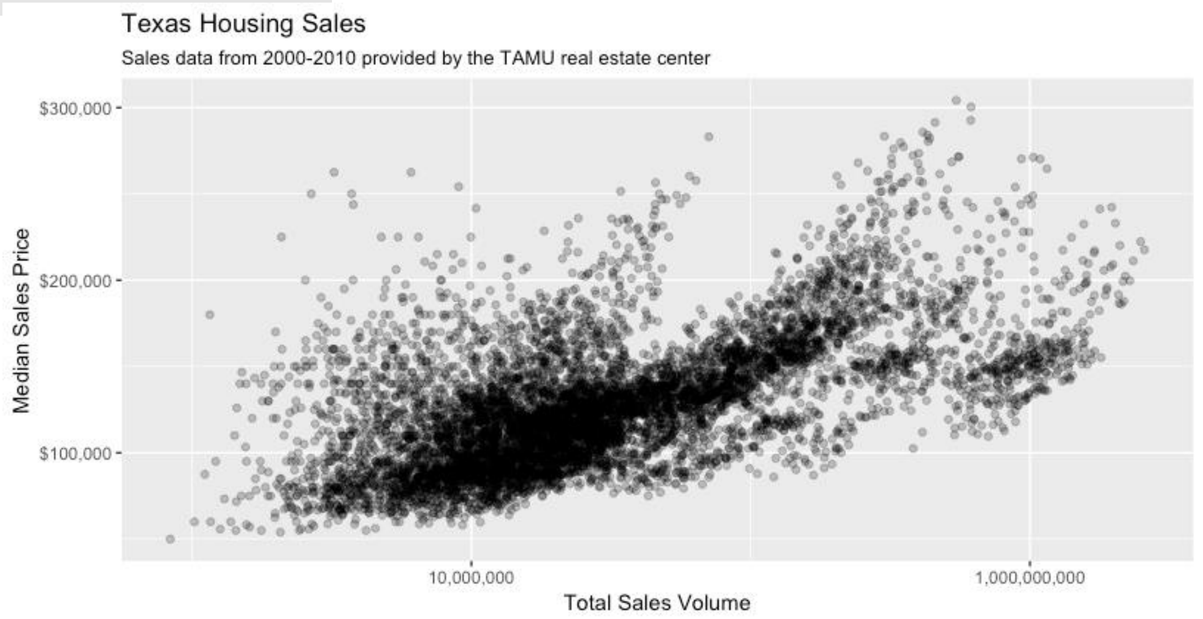
```
ggplot(data = mpg, aes(x = displ, y = cty)) +
  geom_jitter() +
  coord_cartesian(xlim = c(4, 7), ylim = c(10, 20))
```



MANIPULATING YOUR COORDINATES

- There are *many* options to manipulate and adjust the coordinate system but some basic ones include:
    - formatting axes and labels

```
ggplot(data = txhousing, aes(x = volume, y = median)) +
    geom_point(alpha = .25) +
    scale_y_continuous(name = "Median Sales Price", labels = scales::dollar) +
    scale_x_log10(name = "Total Sales Volume", labels = scales::comma) +
    ggtitle("Texas Housing Sales",
        subtitle = "Sales data from 2000-2010 provided by the TAMU real estate center")
```

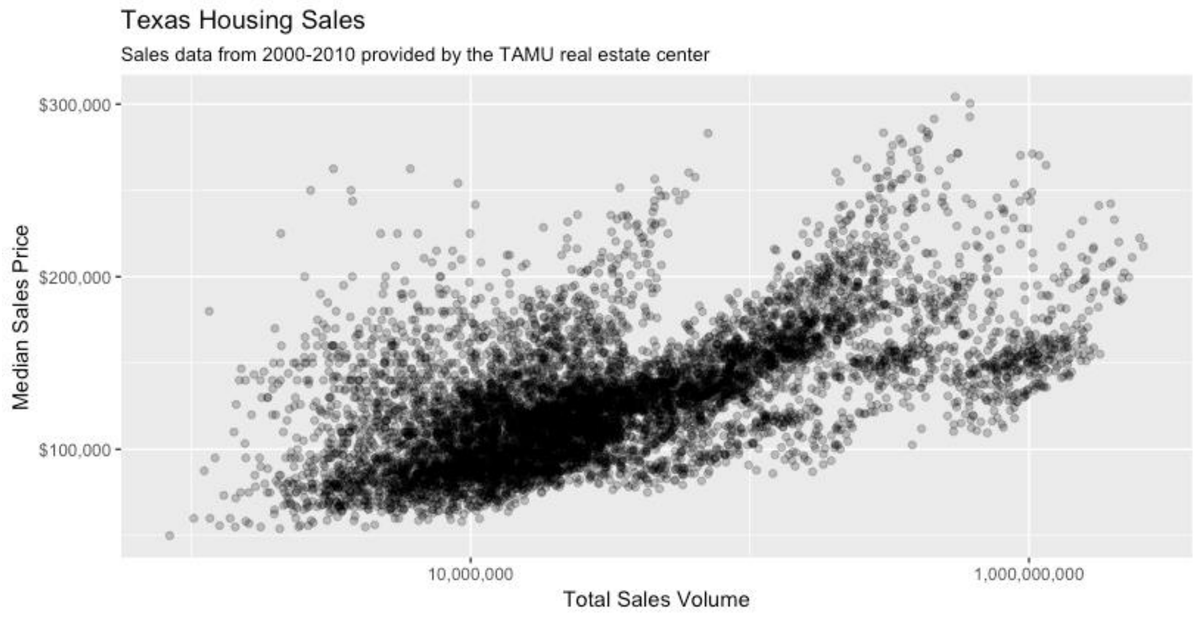


MANIPULATING YOUR COORDINATES

- There are *many* options to manipulate and adjust the coordinate system but some basic ones include:
  - formatting axes and labels

```
ggplot(data = txhousing, aes(x = volume, y = median)) +
    geom_point(alpha = .25) +
    scale_y_continuous(name = "Median Sales Price", labels = scales::dollar) +
    scale_x_log10(name = "Total Sales Volume", labels = scales::comma) +
    ggtitle("Texas Housing Sales",
        subtitle = "Sales data from 2000-2010 provided by the TAMU real estate center")
```

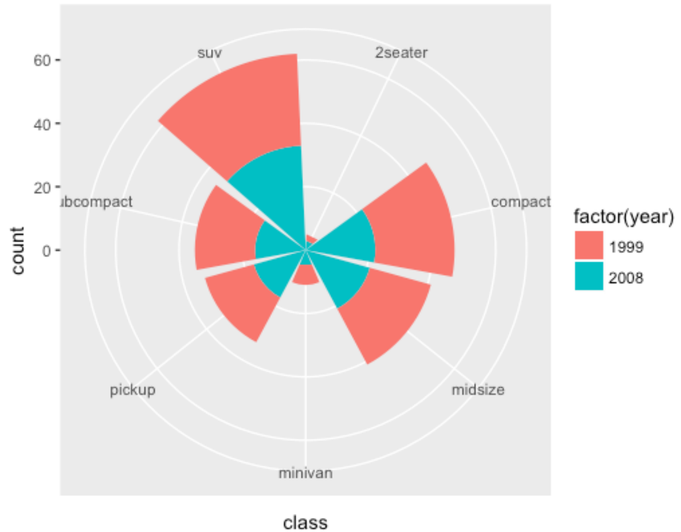
This is what I expect your project visualizations to look like!



MANIPULATING YOUR COORDINATES

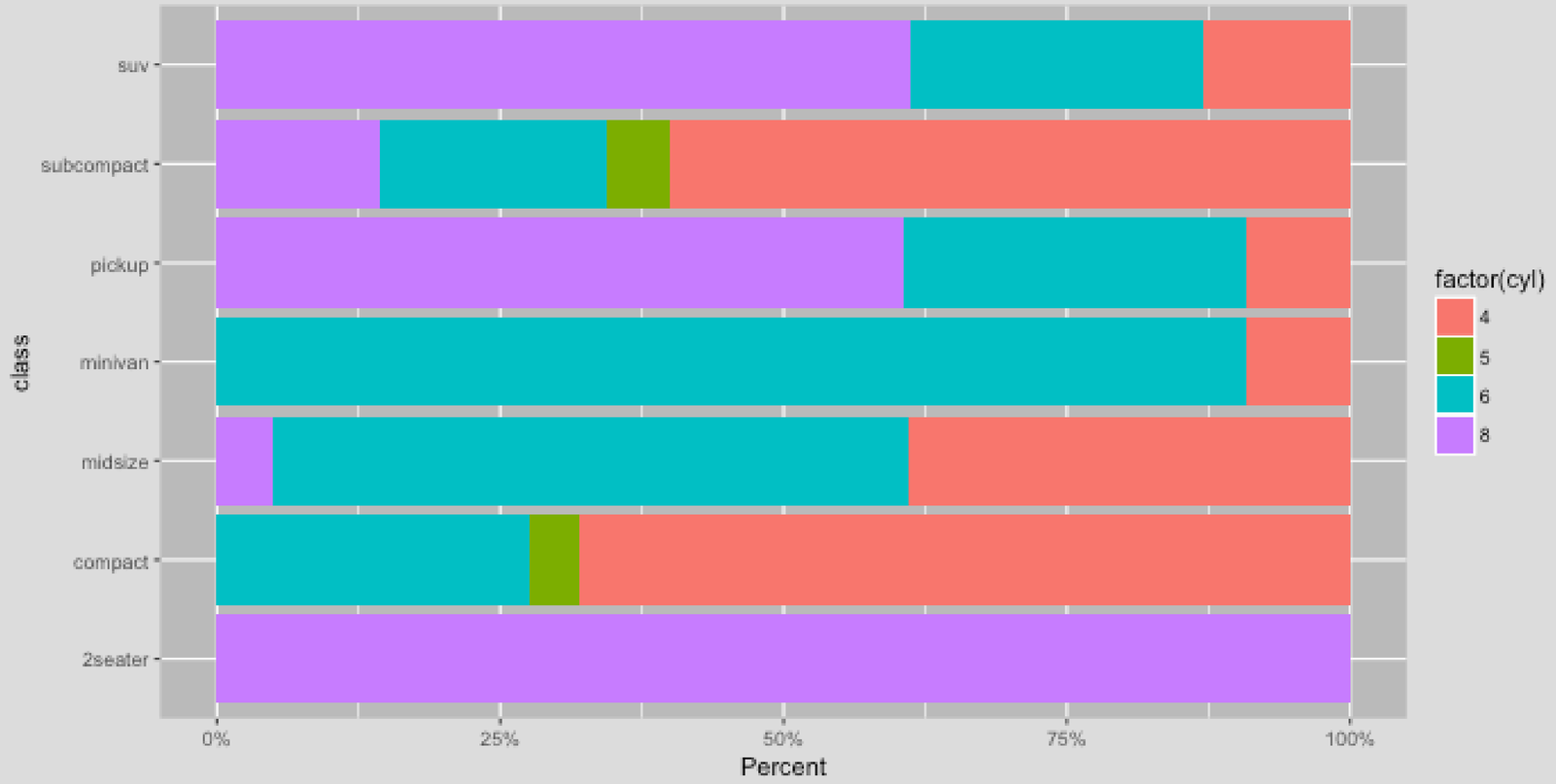
- There are *many* options to manipulate and adjust the coordinate system but some basic ones include:
  - ~~creating pie charts~~

```
ggplot(data = mpg, aes(class, fill = factor(year))) +
    geom_bar() +
    coord_polar()
```



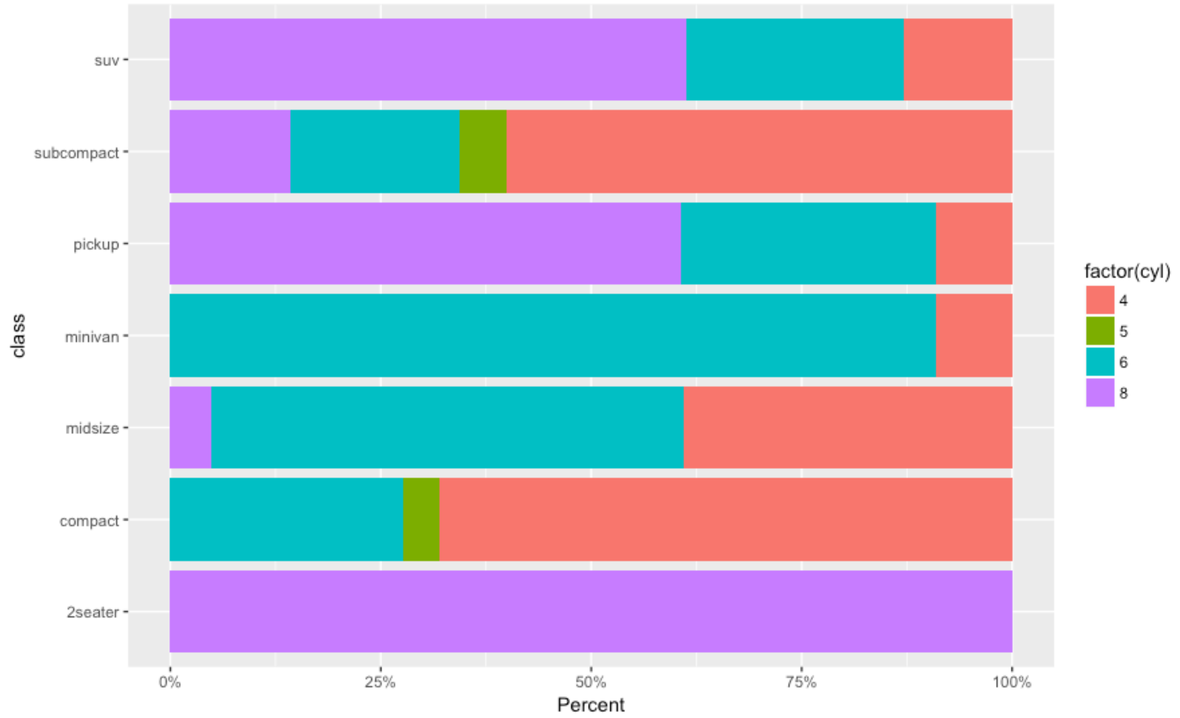
YOUR TURN!

How close can you get to re-creating this?



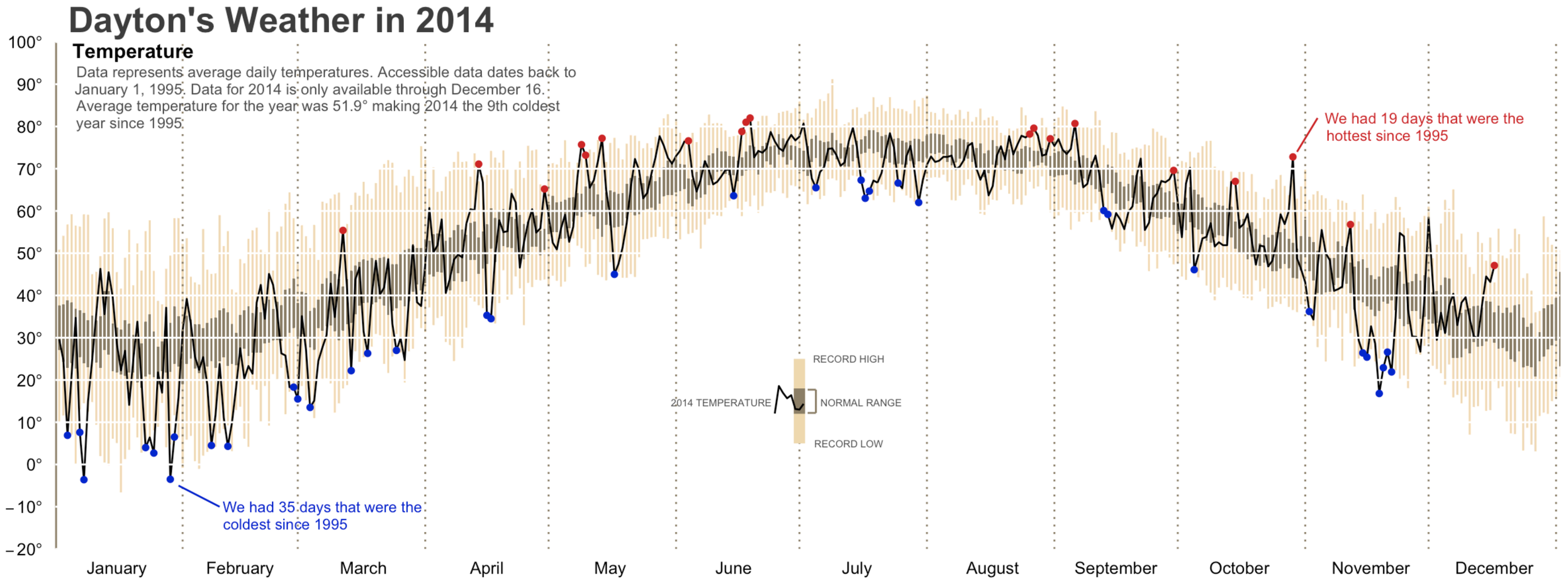
SOLUTION

```
mpg %>%
  ggplot(aes(x = class, fill = factor(cyl))) +
  geom_bar(position = "fill") +
  scale_y_continuous(name = "Percent", labels = scales::percent) +
  coord_flip()
```



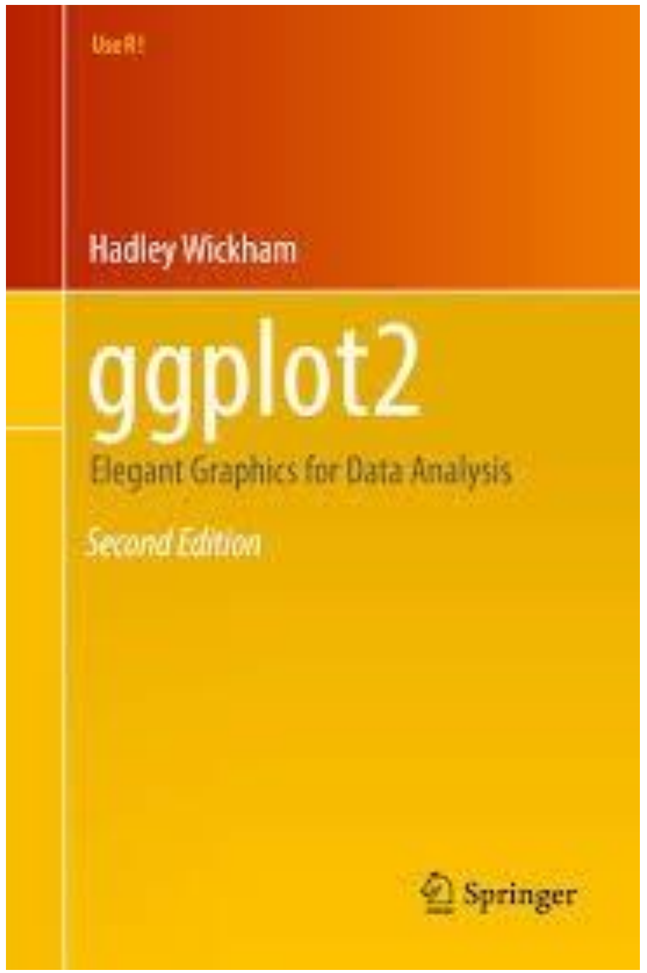

SO LITTLE TIME!

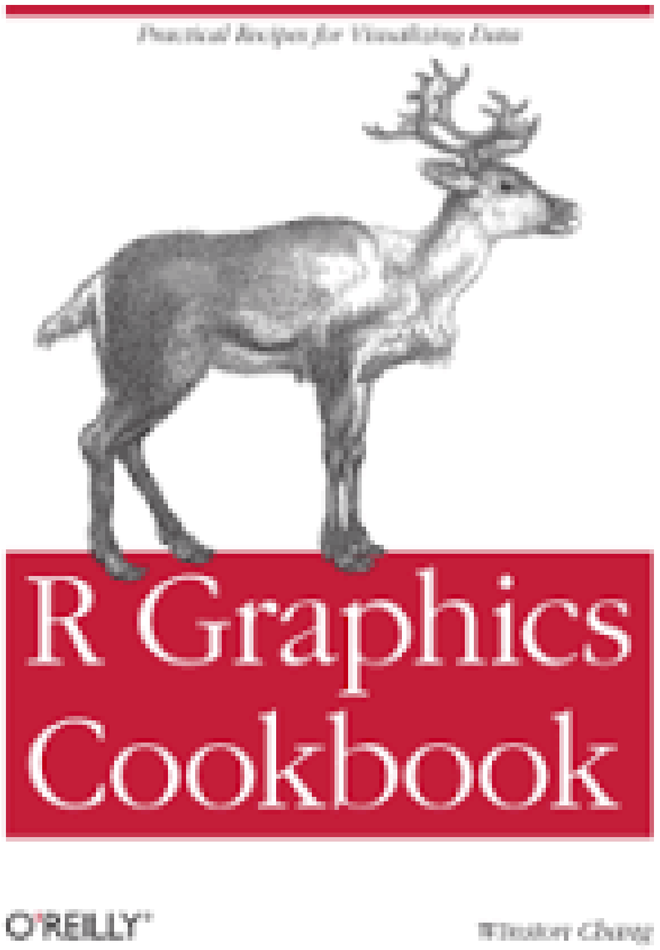
SEE LAYERING IN ACTION

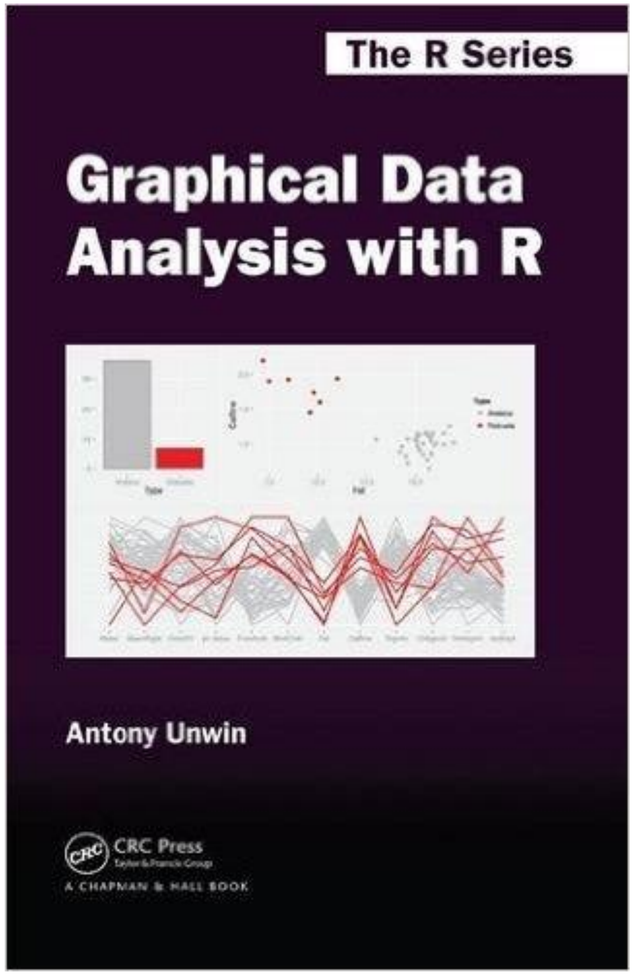


https://rpubs.com/bradleyboehmke/weather_graphic

LEARN MORE

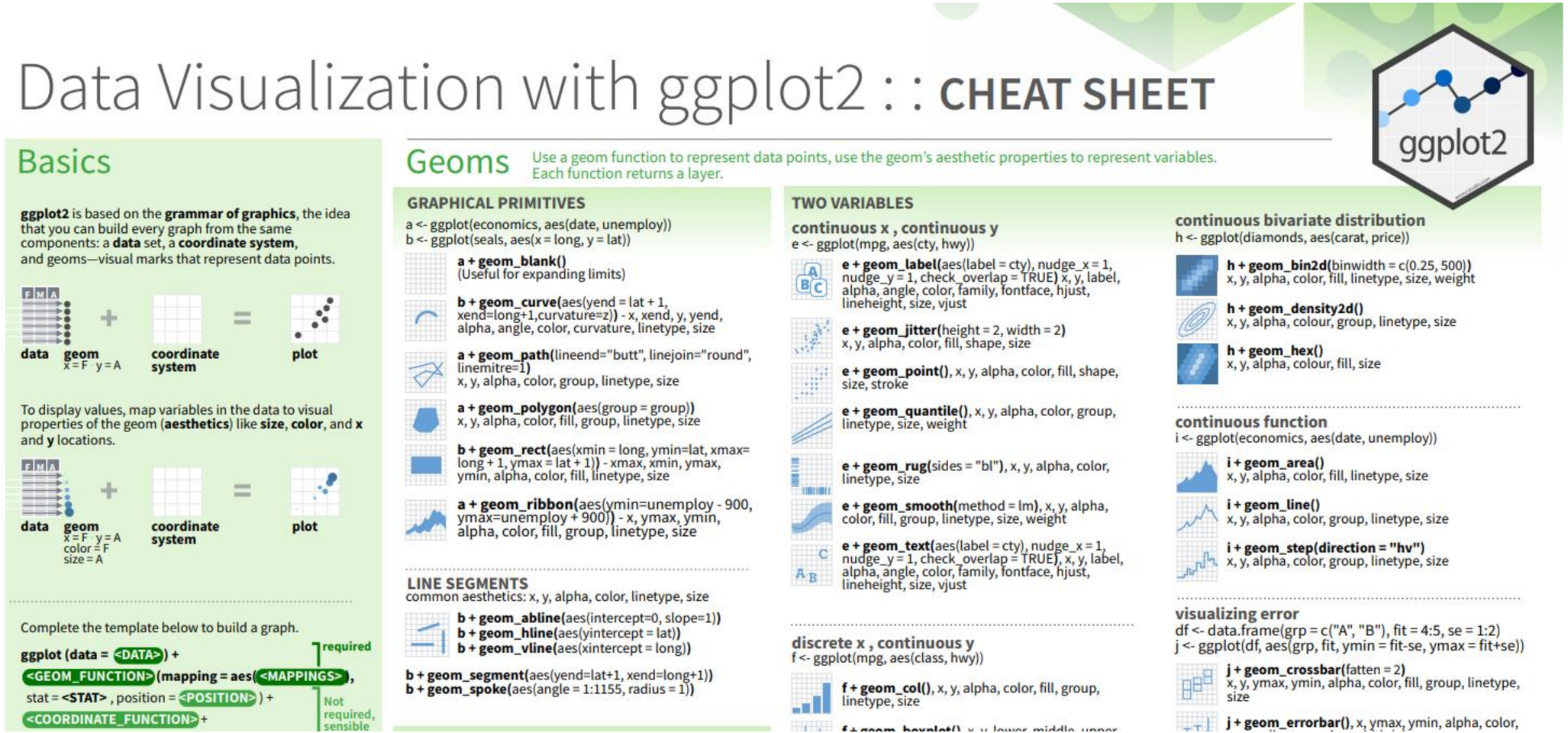

UseR!
Hadley Wickham
ggplot2
Elegant Graphics for Data Analysis
Second Edition
Springer


R Graphics Cookbook
O'REILLY


The R Series
Graphical Data Analysis with R
Antony Unwin
CRC Press
A CHAPMAN & HALL BOOK

LEVERAGE HELP AS YOU'RE LEARNING

Help >> Cheatsheets >> Data Visualization with ggplot2



Data Visualization with ggplot2 : : CHEAT SHEET

ggplot2

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

data + geom (x = F, y = A) = coordinate system = plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data + geom (x = F, y = A, color = F, size = A) = coordinate system = plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION> ) +                 Not required, sensible
<COORDINATE_FUNCTION> +                                    defaults supplied
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

Geoms

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

GRAPHICAL PRIMITIVES

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

a + geom_blank()
(Useful for expanding limits)

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

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

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

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

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

LINE SEGMENTS

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

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

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

TWO VARIABLES

continuous x , continuous y

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

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

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

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

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

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

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

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

discrete x , continuous y

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

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

continuous bivariate distribution

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

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

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

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

continuous function

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

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

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

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

visualizing error

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

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

j + geom_errorbar(), x, ymax, ymin, alpha, color,

WHAT TO REMEMBER

FUNCTIONS TO REMEMBER

Operator/Function	Description
ggplot()	Initializes a ggplot object (creates the blank canvas)
aes()	Creates aesthetic mappings
geom_xx	Geometric shapes to plot the data
color, shape, size, alpha, etc	Aesthetic parameters
facet_wrap, facet_grid	Create small multiples
position	Position argument (primarily used with bar charts)
coord_xx	Functions to adjust the coordinate system
scale_xx	Functions to adjust x and y axis

IMPORTANT

This aligns to standards 4.2, 4.3, 4.5.

10 MINUTE BREAK