Data Visualization

Kylie Ariel Bemis

9/11/2018

What is a graphic?

What is a graphic?

Common statistical plots:

- Scatter plot
- Line plot
- Box plot
- Histogram
- Bar plot

How are these plots related?

Common statistical plots:

- Scatter plot
- Line plot
- Box plot
- Histogram
- Bar plot

What are their basic building blocks?

How are these plots related?

- Scatter plot
  - Maps variables to x- and y-axes
  - Uses points to represent each observation
- Line plot
  - Maps variables to x- and y-axes
  - Uses lines to connect each observation
- Box plot
  - Maps 5-number summary (min, lower-hinge, median, upper-hinge, max) to y-axis
  - Uses shapes to (boxes and whiskers) to represent these
- Histogram
  - Maps bins to x-axis and frequencies to y-axis
  - Uses bars to represent these
- Bar plot
  - Maps categorical variable to x-axis and counts (usually) to y-axis
  - Uses bars to represent these

What are the common elements?

What are the common elements?

- Some kind of data
- Mappings from data to aspects of the plot ("aesthetics")
- Geometric objects
- Potentially, statistical transformation
- Coordinate systems

We're on to something...

Can we break a plot into its basic components?

Consider a simple dataset:

A	*B*	*C*	*D*
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

Figure 1: http://vita.had.co.nz/papers/layered-grammar.pdf

We wish to create a scatter plot of A versus C.

Mapping aesthetics

We map variable A to x, variable C to y, and shape to D.

x	y	Shape
2	4	circle
1	1	circle
4	15	square
9	80	square

Figure 2: http://vita.had.co.nz/papers/layered-grammar.pdf

Building a plot

We have (1) geometric objects, (2) scales and coordinate system, and (3) plot annotations:

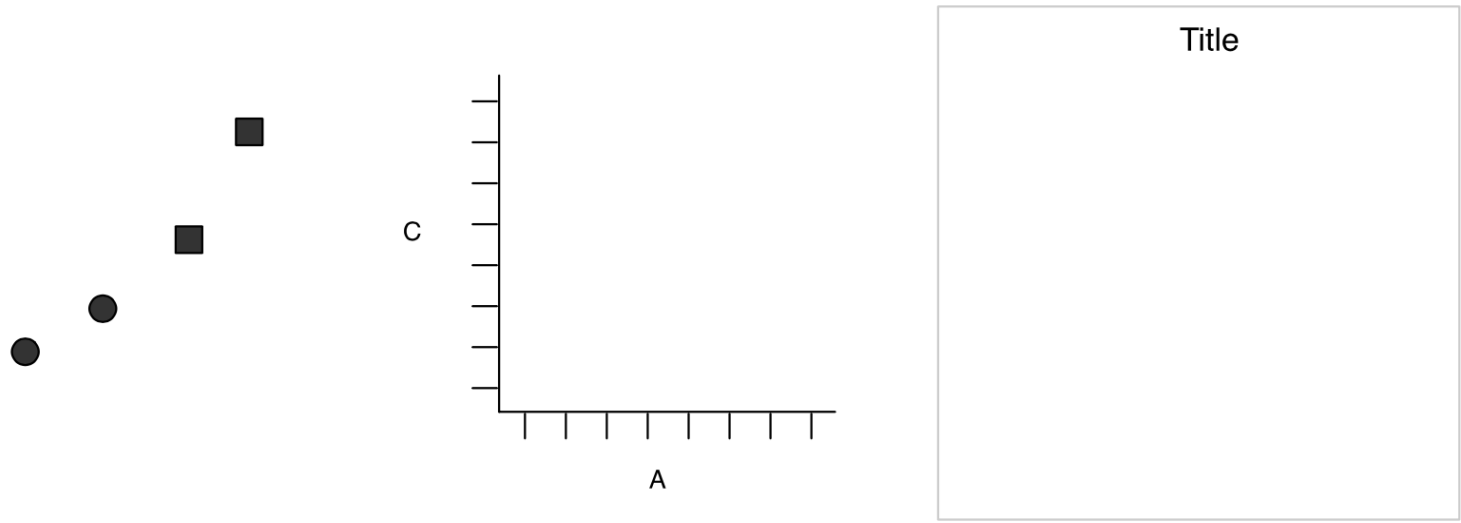


Figure 3: http://vita.had.co.nz/papers/layered-grammar.pdf

Building a plot (1)

And we have a plot:

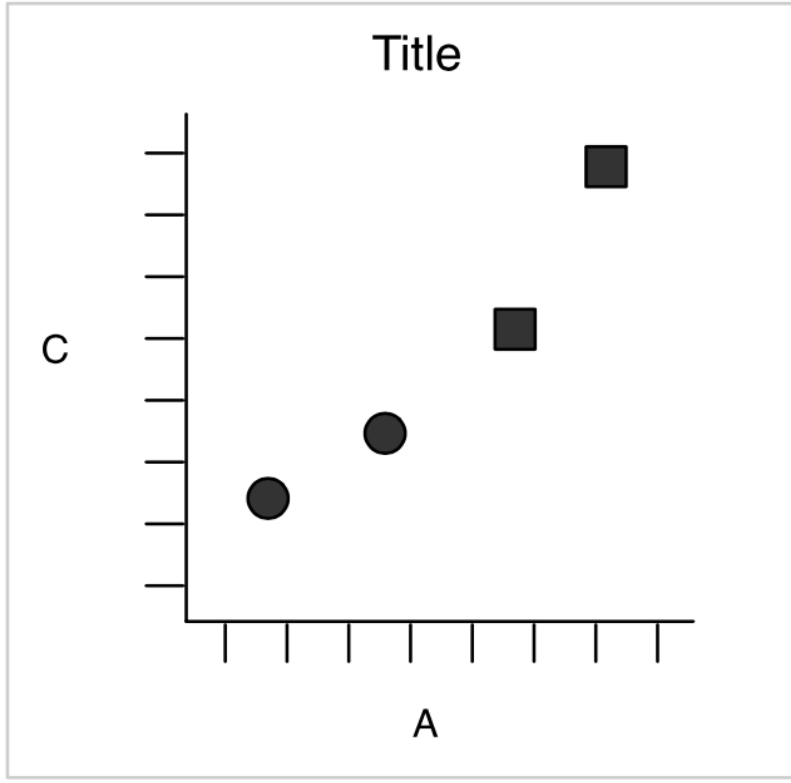


Figure 4: http://vita.had.co.nz/papers/layered-grammar.pdf

Building a plot (2)

We can create a more complicated plot by *faceting* on a variable:

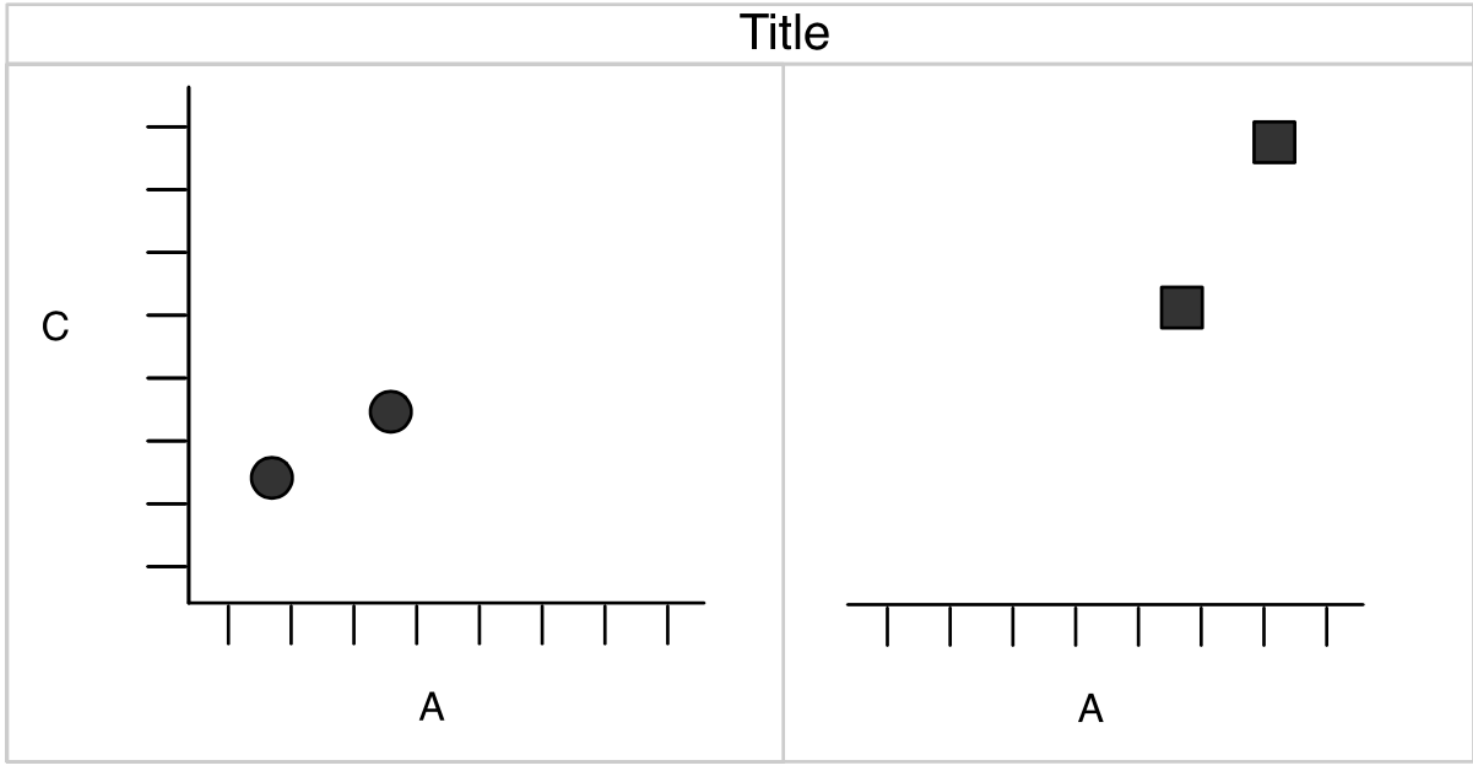


Figure 5: http://vita.had.co.nz/papers/layered-grammar.pdf

Faceting splits the data into subsets and creates subplots for each subset.

What about more complicated plots?

- Overlaying plots on top of each other
- Different datasets on the same plot
- Etc.

A Layered "Grammer of Graphics"

Components of a plot:

- Default dataset and set of mappings from variables to aesthetics
- One or more layers, each having:
  - A geometric object
  - A statistical transformation
  - A position adjustment
  - (Optional) A dataset
  - (Optional) A set of aesthetic mappings
- A scale for each mapped aesthetic
- A coordinate system
- A facet specification

A simple example (revisited)

```
simple = data.frame(A=c(2, 1, 4, 9),
                    B=c(3, 2, 5, 10),
                    C=c(4, 1, 15, 80),
                    D=c('a', 'a', 'b', 'b'))
simple
```

```
##   A  B  C D
## 1 2  3  4 a
## 2 1  2  1 a
## 3 4  5 15 b
## 4 9 10 80 b
```

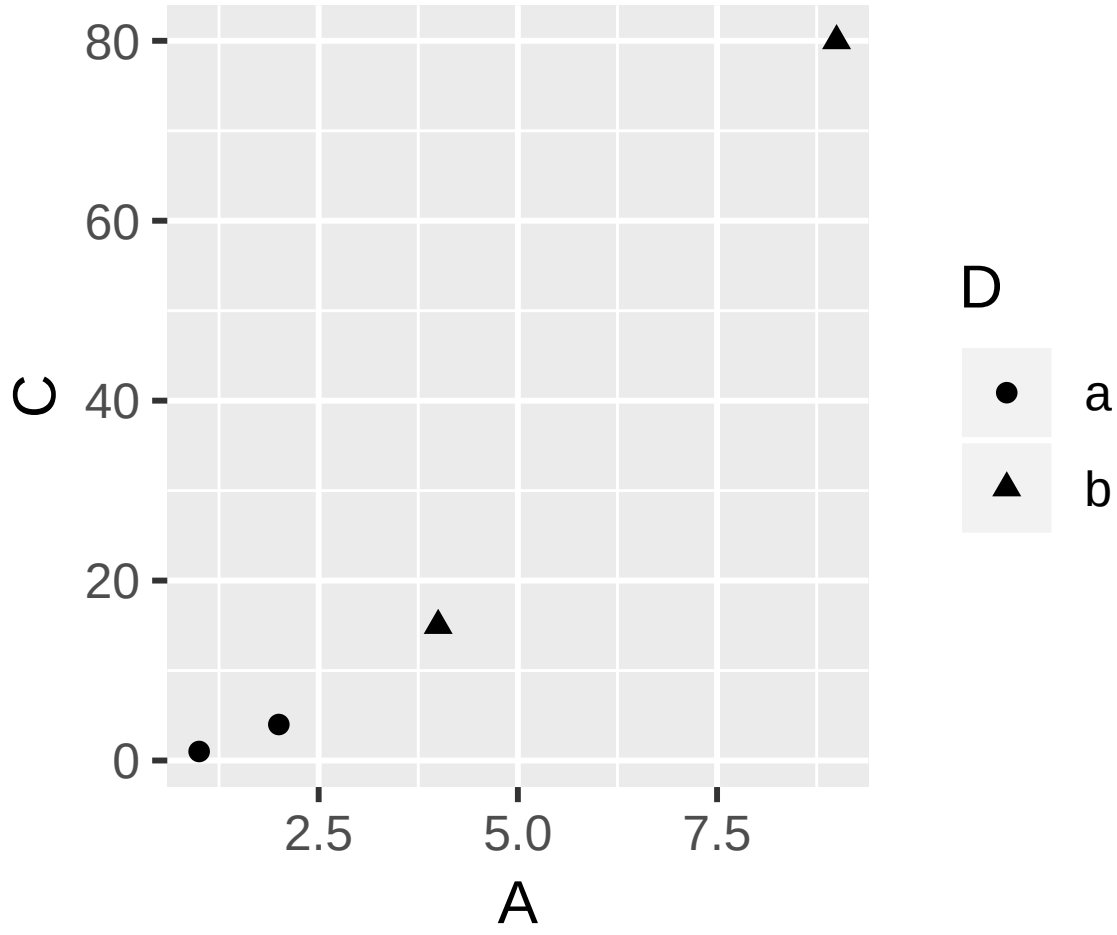
A simple example (revisited)

We wish to create a scatter plot of A versus C, using shape for D.

What are:

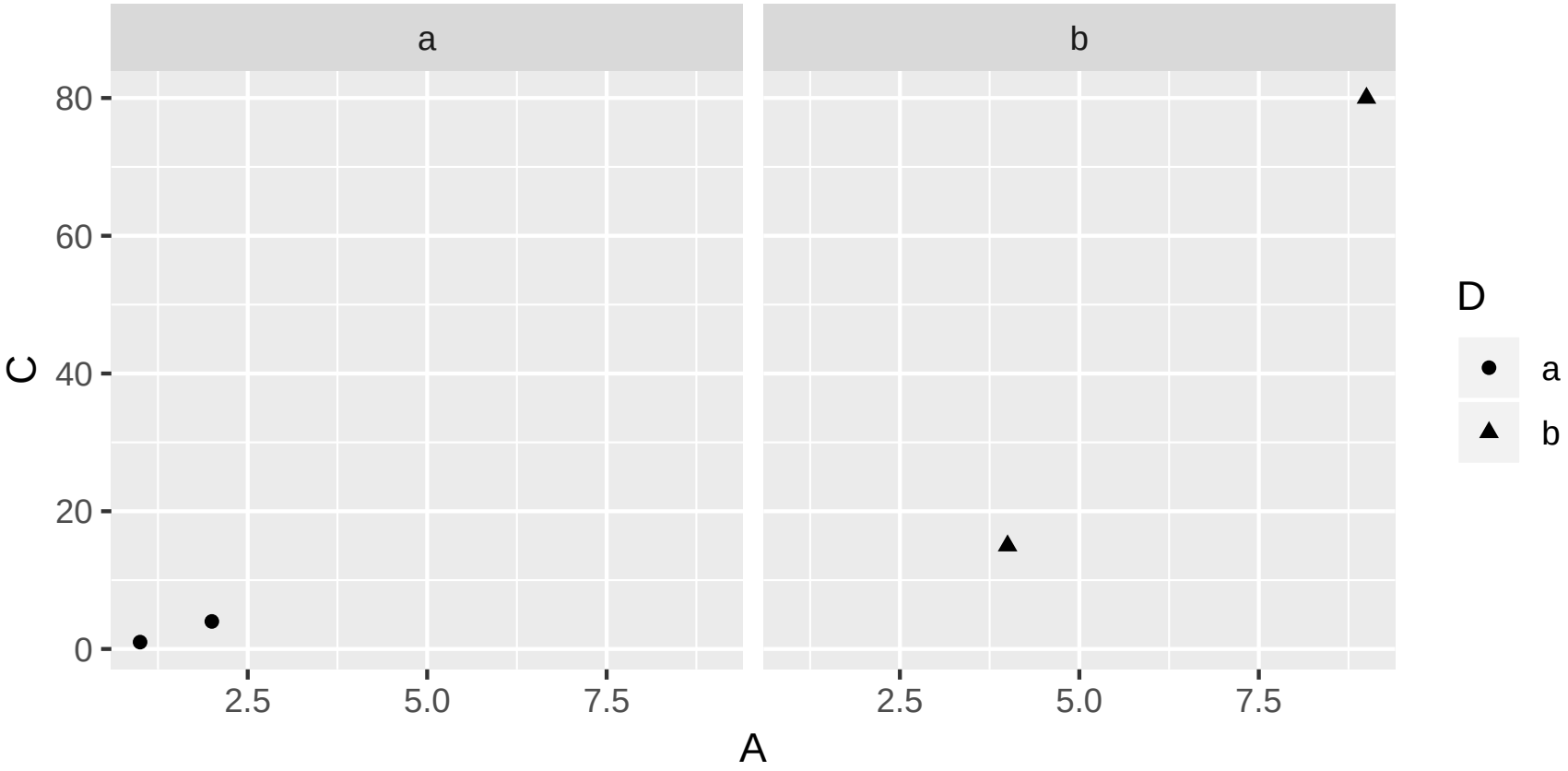
- The default dataset and aesthetic mappins?
- The layers?
  - The geometric object?
  - The statistical transformation?
  - The position adjustment?
- The scales for the mapped aesthetics?
- The coordinate system?
- The facet specification?

```
library(ggplot2)
ggplot(data=simple,
       mapping=aes(x=A, y=C, shape=D)) +
  layer(geom="point",
        stat="identity",
        position="identity") +
  scale_x_continuous() +
  scale_y_continuous() +
  coord_cartesian() +
  facet_null()
```


80
60
40
20
0
C
2.5
5.0
7.5
A
D
a
b

Faceting

```
ggplot(data=simple,
       mapping=aes(x=A, y=C, shape=D)) +
  layer(geom="point",
        stat="identity",
        position="identity") +
  scale_x_continuous() +
  scale_y_continuous() +
  coord_cartesian() +
  facet_wrap(~D)
```


a
b
80
60
40
20
0
C
2.5
5.0
7.5
2.5
5.0
7.5
A
D
a
b

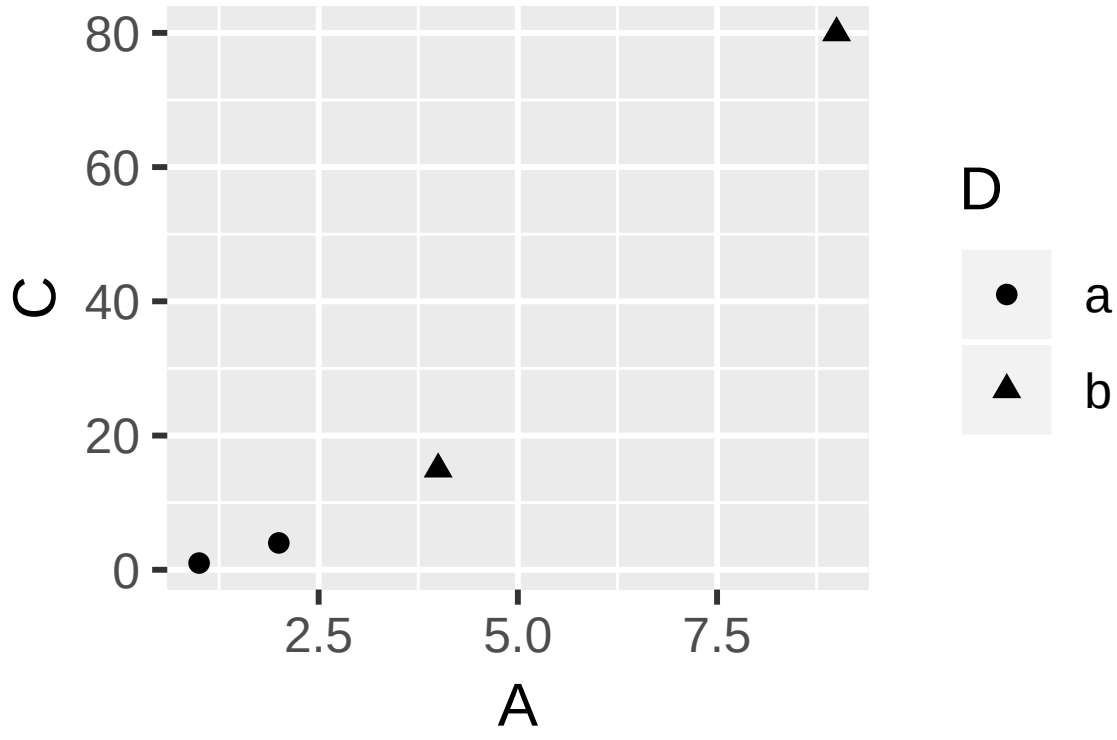
Sensible defaults

A number of these can be considered sensible defaults:

- For `geom="point"`, use `stat="identity"` unless otherwise specified
- For `geom="point"`, use `position="identity"` unless otherwise specified
- Cartesian coordinate system
- No facets unless explicitly specified

The usual way

```
ggplot(data=simple, mapping=aes(x=A, y=C, shape=D)) +
  geom_point()
```



Every `geom` has a default `stat`

- Scatter plot
- Line plot
- Box plot
- Histogram
- Bar plot

What are their default statistical transformations?

Every geom has a default stat

- Scatter plot - `geom_point`
  - Identity - `stat_identity`
- Line plot - `geom_line`
  - Identity - `stat_identity`
- Box plot - `geom_boxplot`
  - Boxplot (five summary statistics + outliers) - `stat_boxplot`
- Histogram - `geom_histogram`
  - Binning - `stat_bin`
- Bar plot - `geom_bar`
  - Count - `stat_count`

These can always be changed!

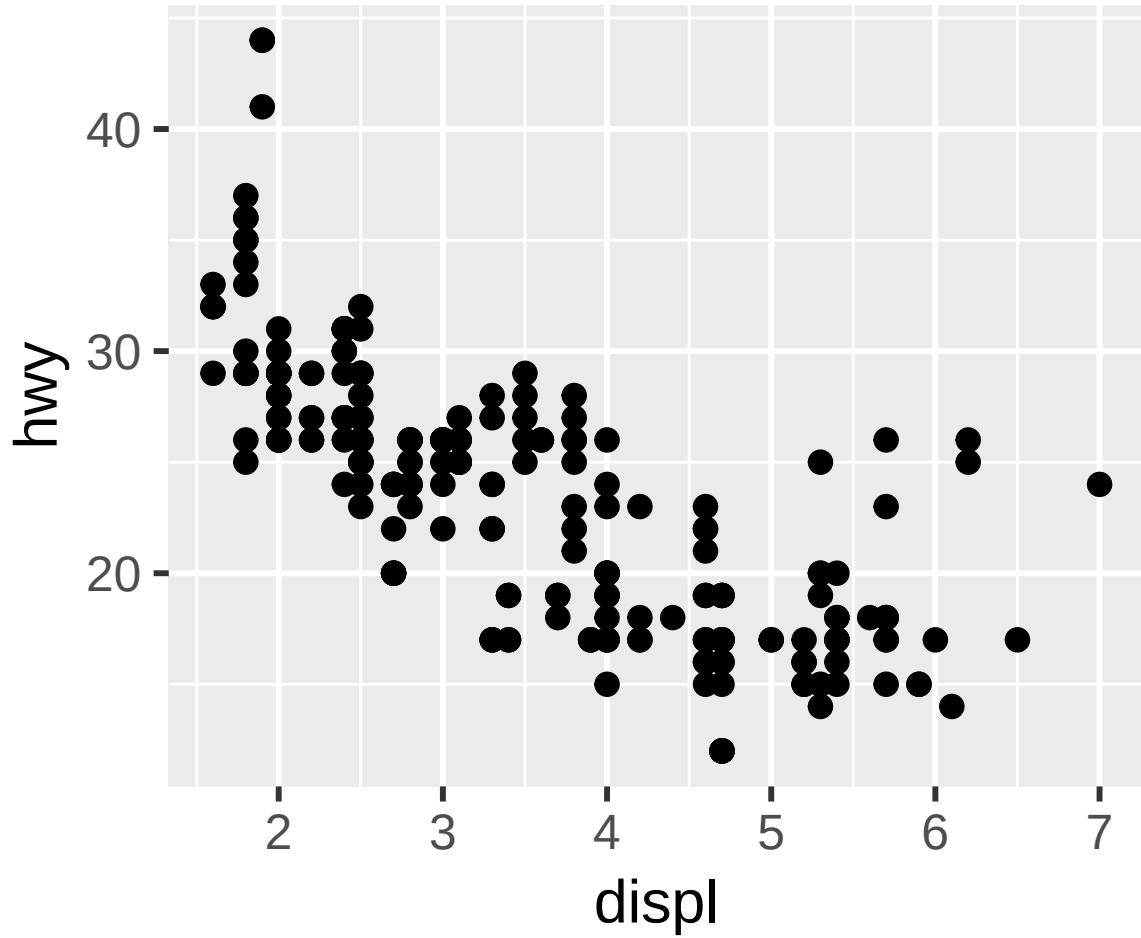
Example: Fuel Economy in Cars

```
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model displ  year   cyl trans drv      ct
##    <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int
##  1 audi         a4      1.8  1999     4 auto~ f         1
##  2 audi         a4      1.8  1999     4 manu~ f         2
##  3 audi         a4      2    2008     4 manu~ f         2
##  4 audi         a4      2    2008     4 auto~ f         2
##  5 audi         a4      2.8  1999     6 auto~ f         1
##  6 audi         a4      2.8  1999     6 manu~ f         1
##  7 audi         a4      3.1  2008     6 auto~ f         1
##  8 audi         a4 q~   1.8  1999     4 manu~ 4         1
##  9 audi         a4 q~   1.8  1999     4 auto~ 4         1
## 10 audi         a4 q~   2    2008     4 manu~ 4         2
## # ... with 224 more rows
```

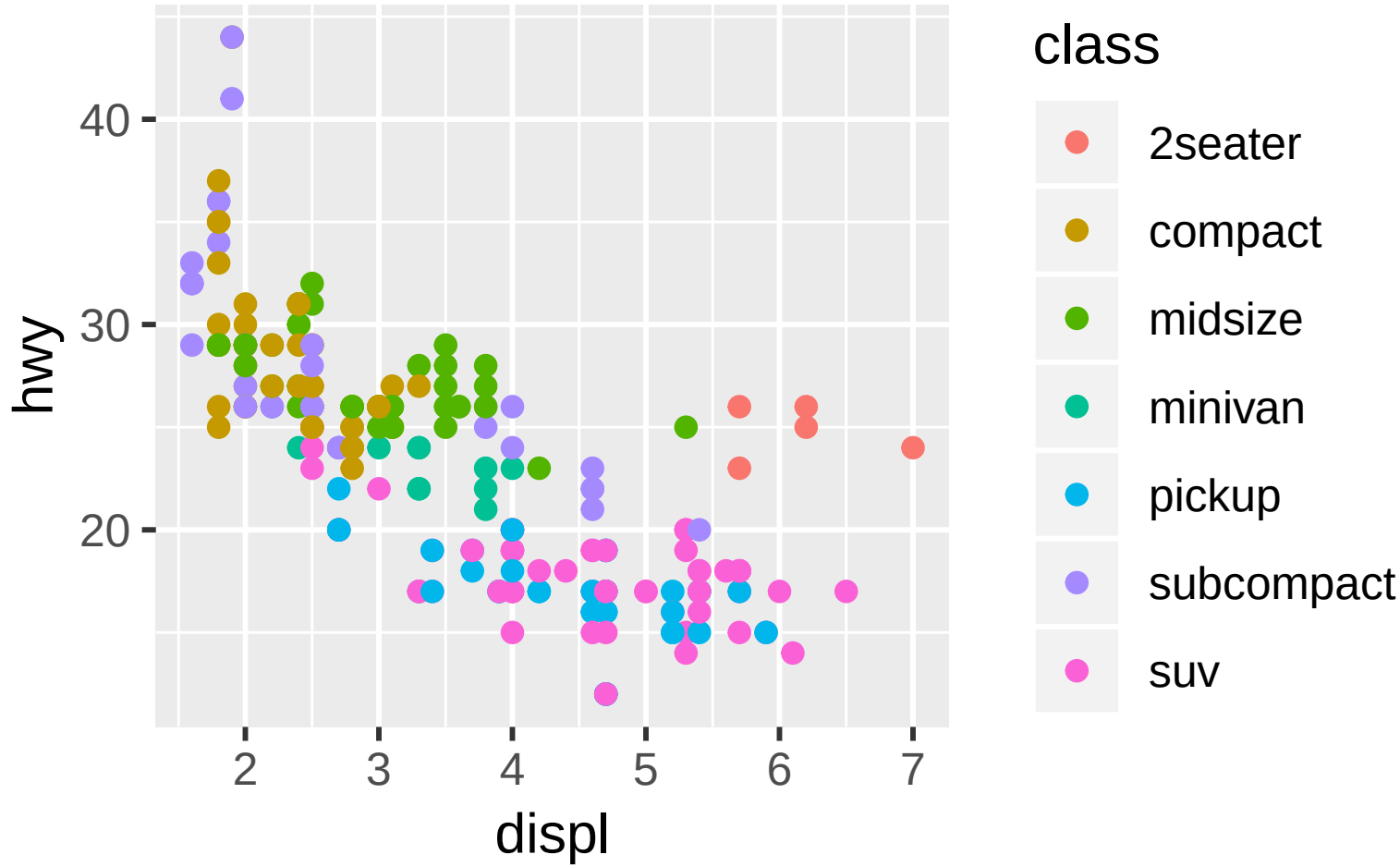
Plot engine size versus highway miles per gallon

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


hwy
40
30
20
2
3
4
5
6
7
displ

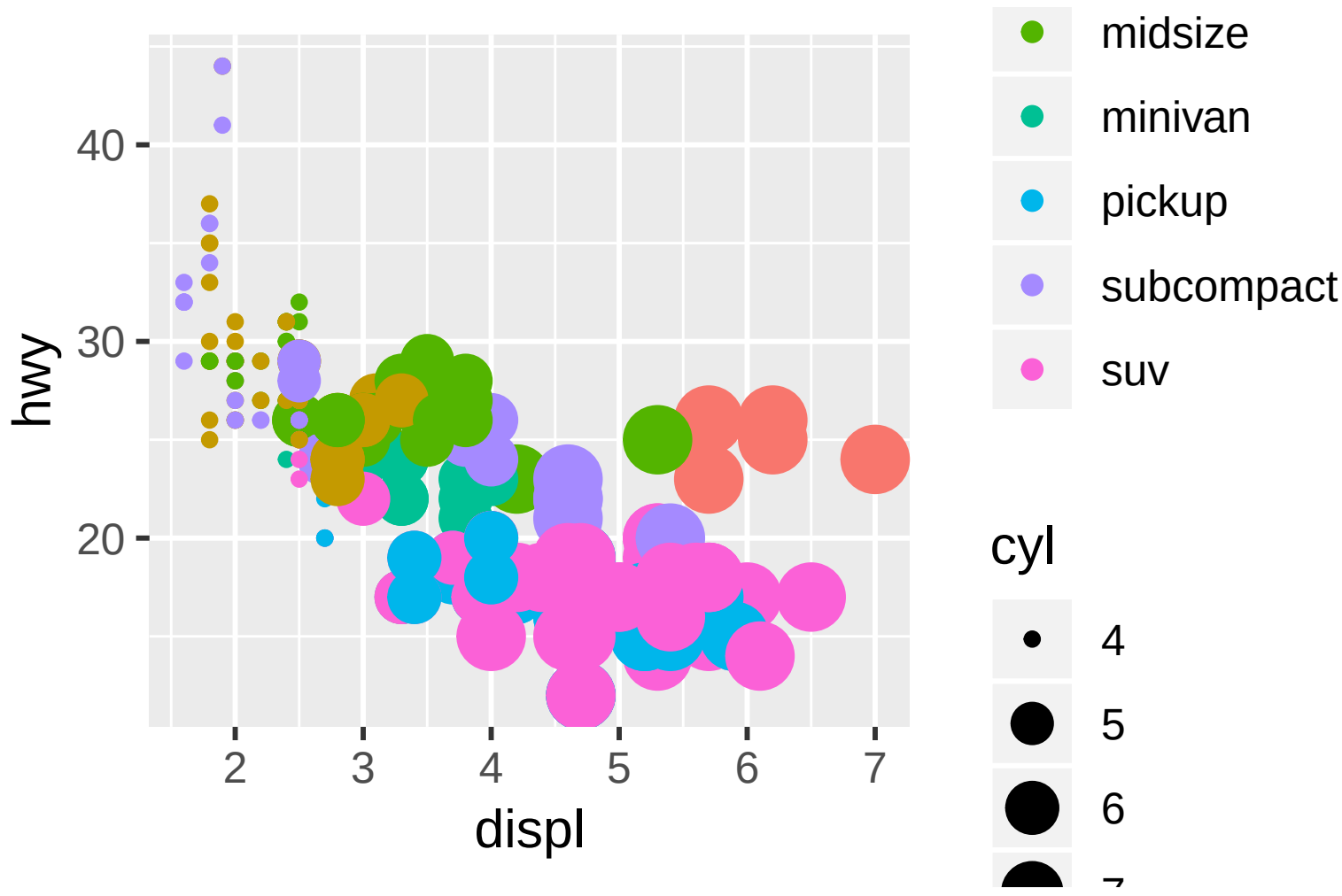
Map class to color

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


hwy
40
30
20
2
3
4
5
6
7
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

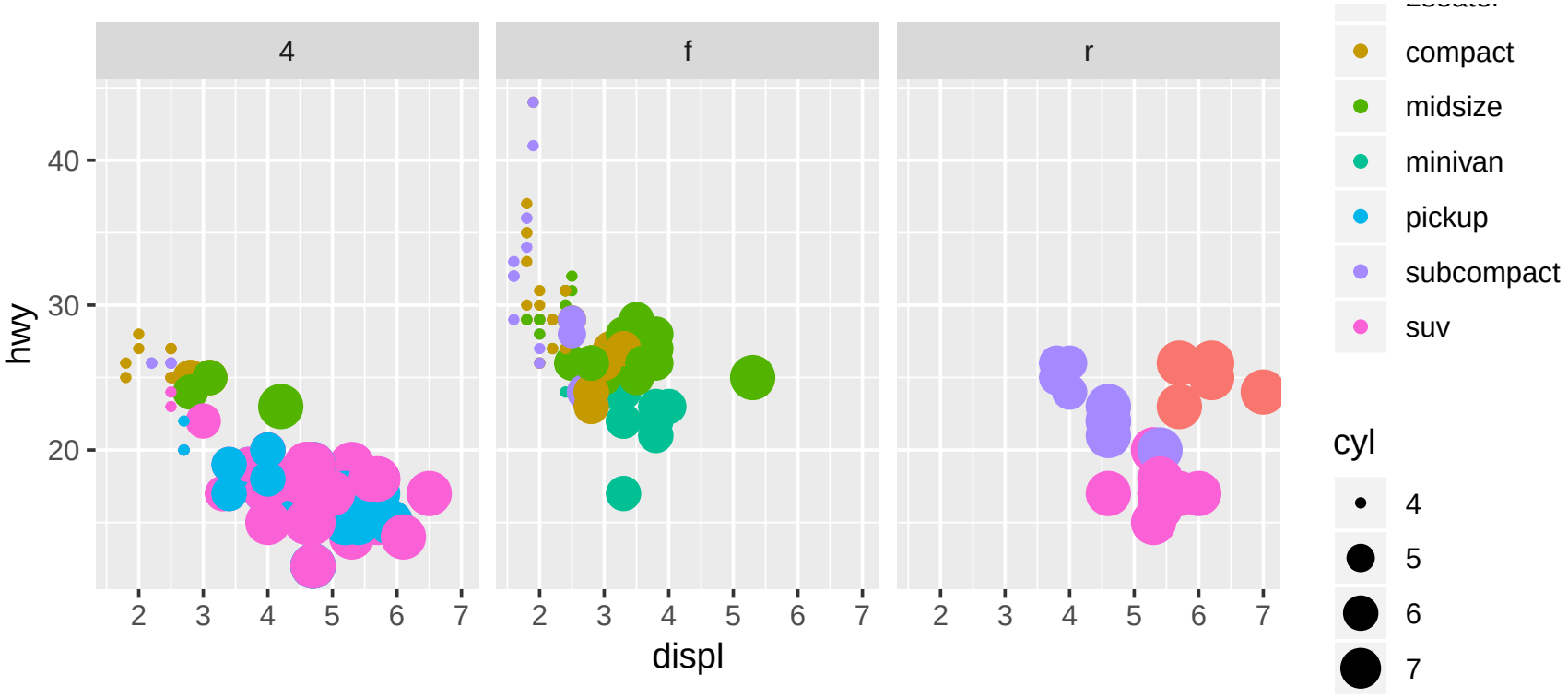
Map number of cylinders to size

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


hwy
40
30
20
2
3
4
5
6
7
displ
midsize
minivan
pickup
subcompact
suv
cyl
4
5
6

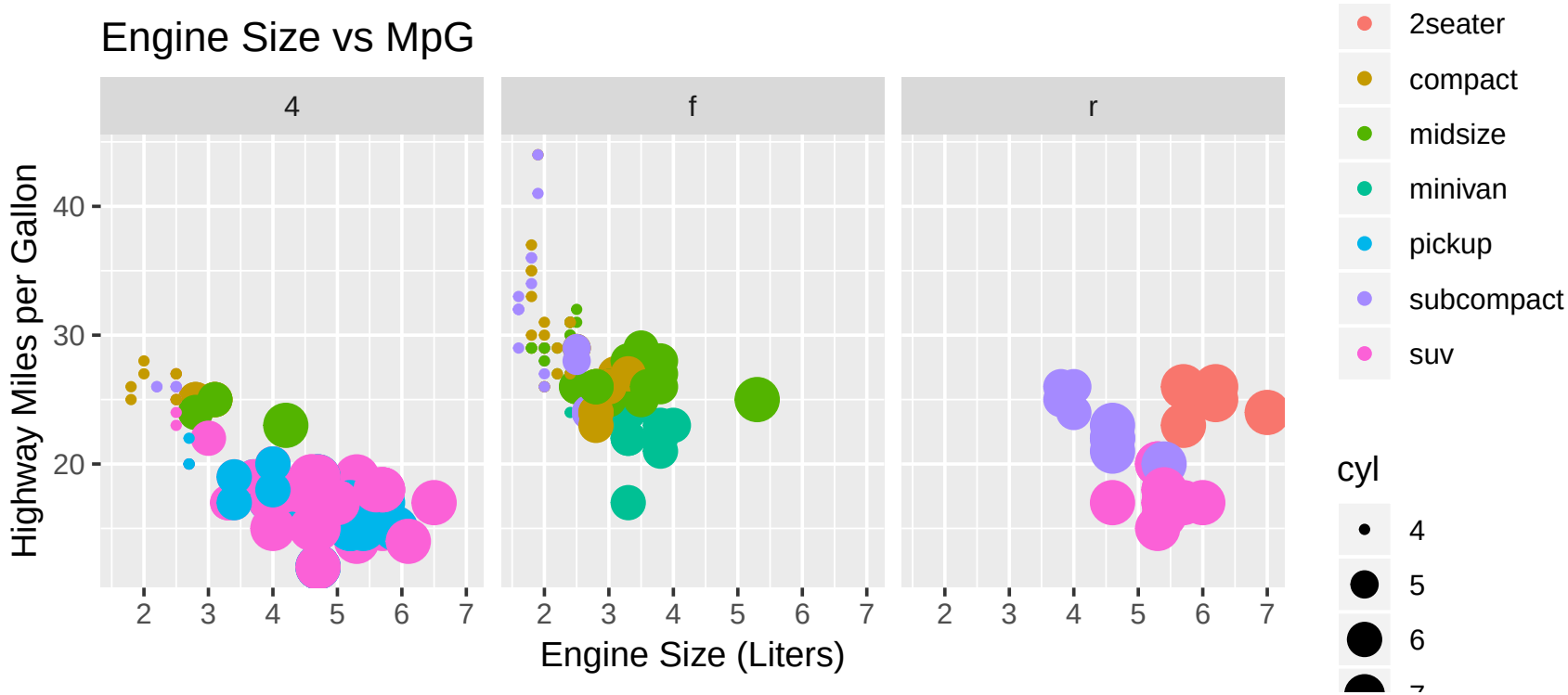
Facet by drive type (front/rear/4-wheel)

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


4
f
r
hwy
40
30
20
2
3
4
5
6
7
displ
compact
midsize
minivan
pickup
subcompact
suv
cyl
4
5
6
7

Add plot annotations (axis labels, title, etc.)

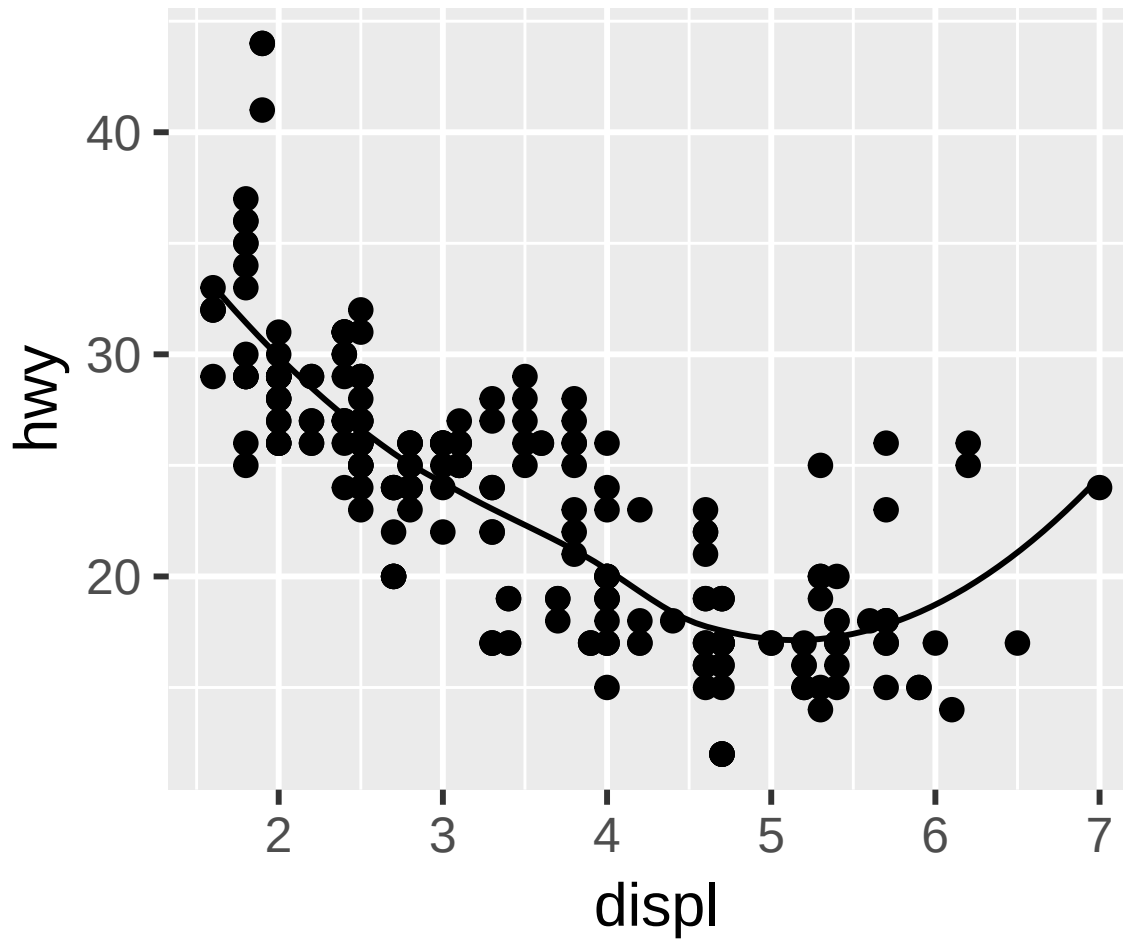
```
ggplot(data = mpg,
       mapping = aes(x=displ,
                     y=hwy,
                     color=class,
                     size=cyl)) +
  geom_point() + facet_wrap(~drv) +
  labs(x="Engine Size (Liters)",
       y="Highway Miles per Gallon",
       title="Engine Size vs MpG")
```


Engine Size vs MpG
4
f
r
Highway Miles per Gallon
40
30
20
2
3
4
5
6
7
Engine Size (Liters)
2seater
compact
midsize
minivan
pickup
subcompact
suv
cyl
4
5
6

What if we want to add a smoothed line??

```
ggplot(data = mpg,
       mapping = aes(x=displ, y=hwy)) +
  layer(geom="point",
        stat="identity",
        position="identity") +
  layer(geom="line",
        stat="smooth",
        position="identity",
        param=list(method="auto", formula=y~x, se=TRUE))
```

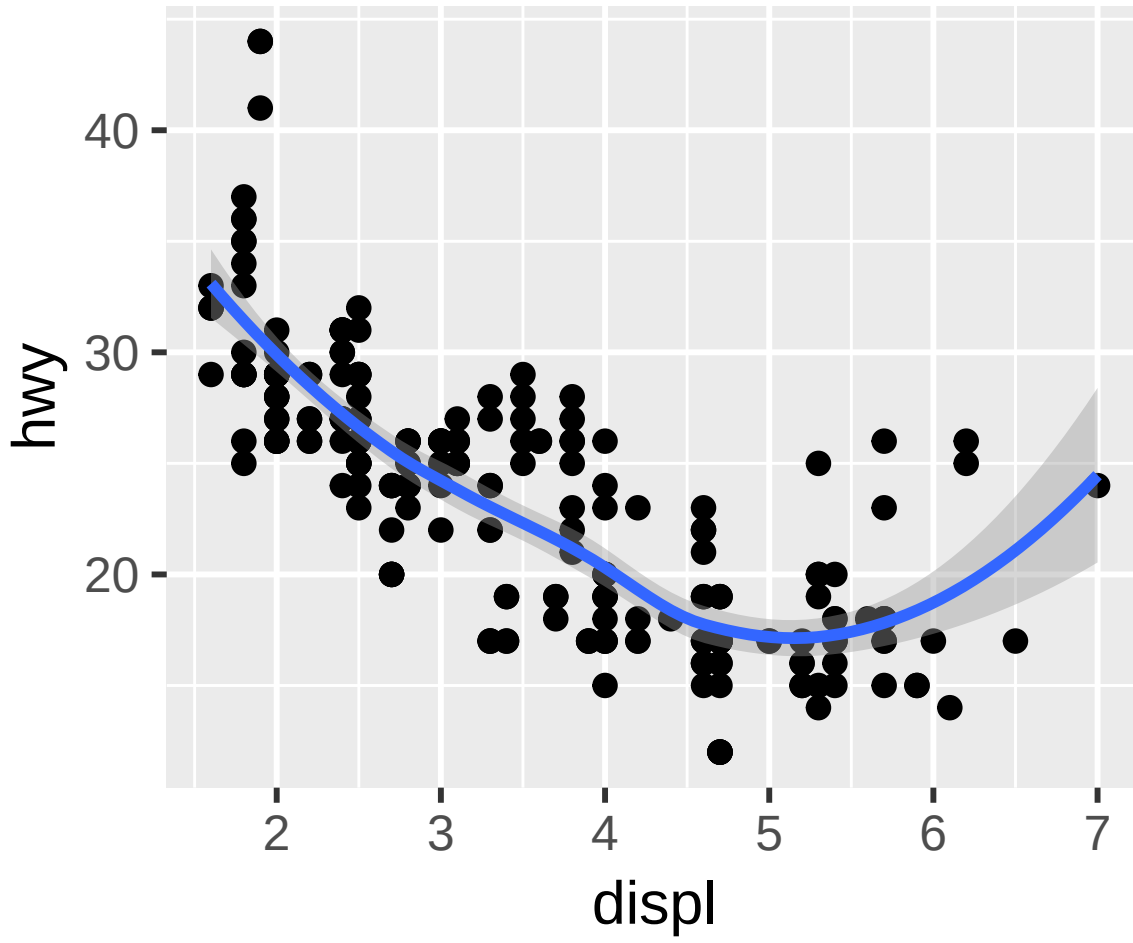
```
## `geom_smooth()` using method = 'loess' and formula 'y ~
```



Use geom_smooth

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

```
## `geom_smooth()` using method = 'loess' and formula 'y ~
```



geoms in ggplot2

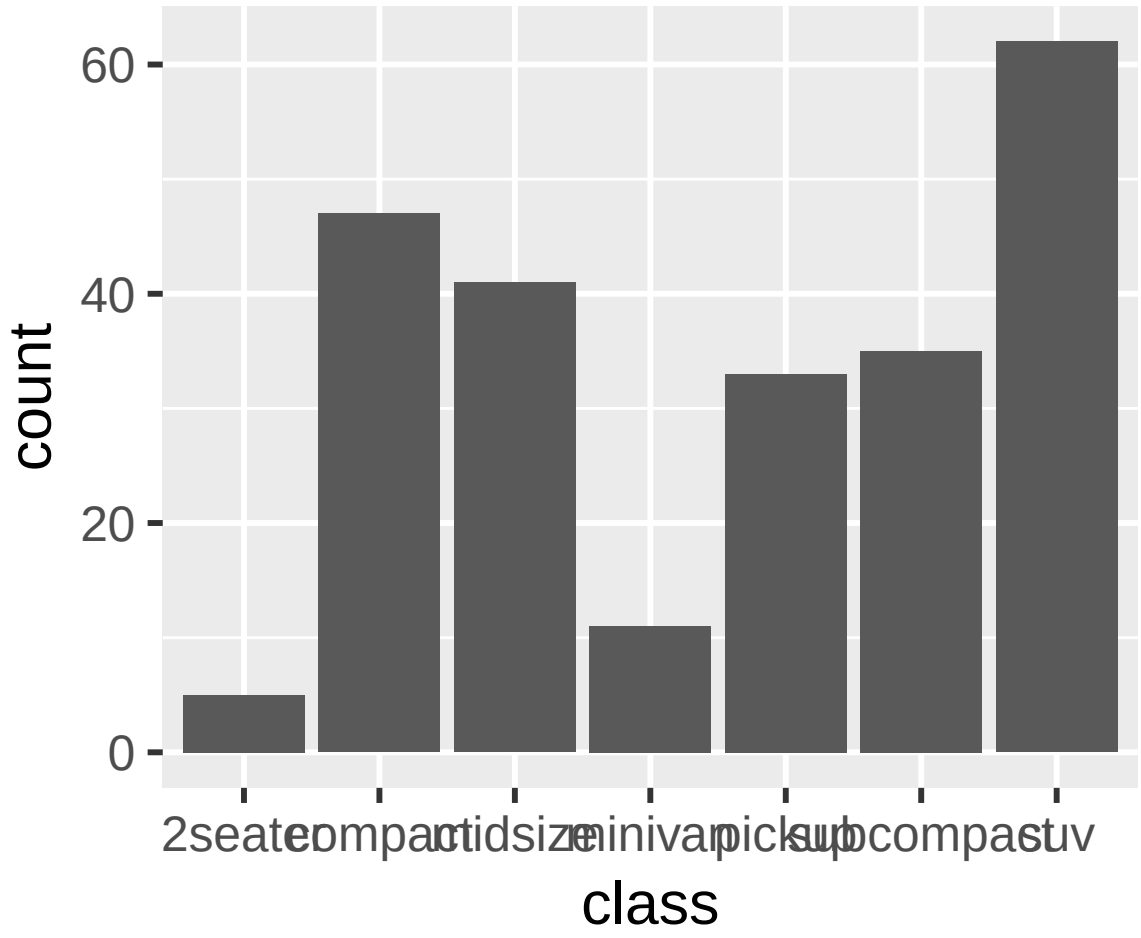
- Geoms in ggplot2 represent a layer with a set of defaults
  - Geometric object
  - Statistical transformation
  - Position adjustment
- Geoms in ggplot2 are shortcuts for potentially complex layers
- Geoms in ggplot2 are sometimes redundant with other geoms

Consider a histograms and bar plots

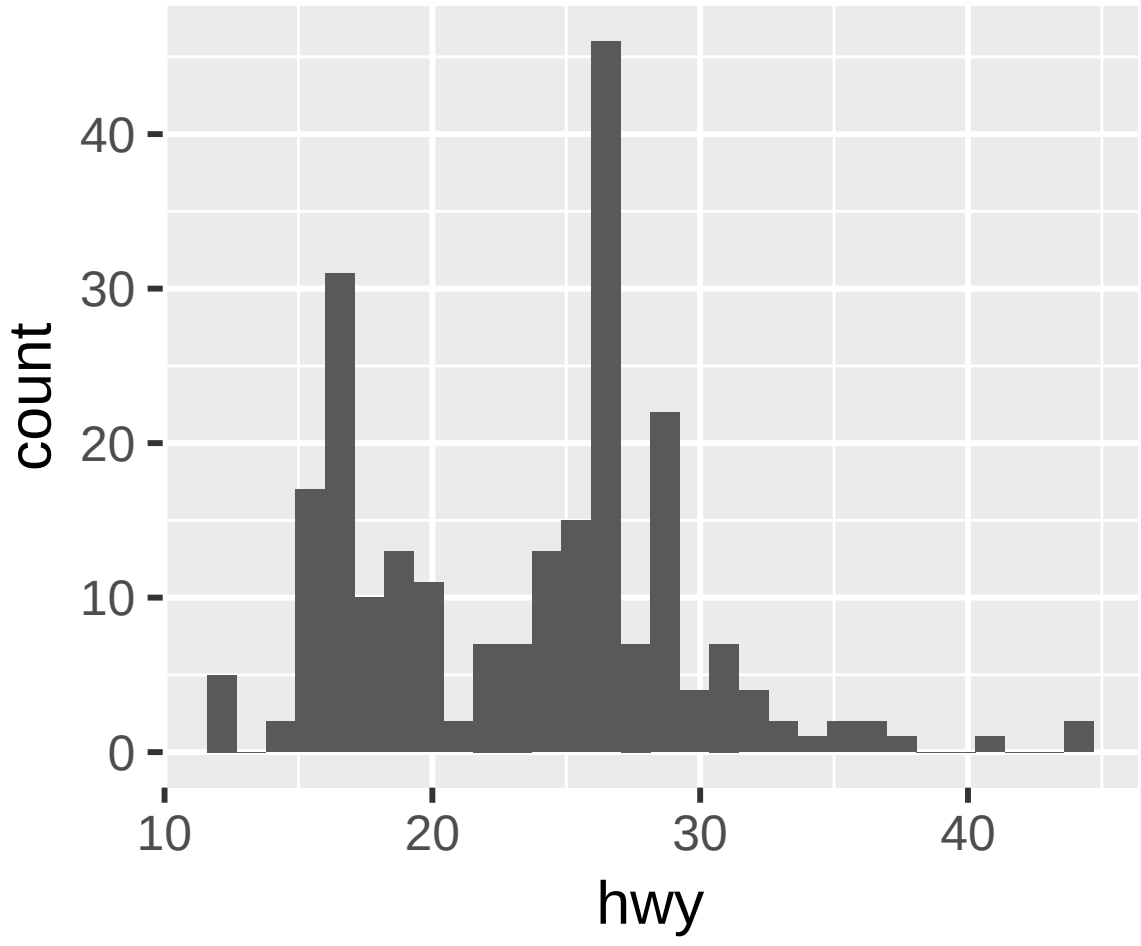
```
ggplot(data = mpg, mapping = aes(x=class)) +
  geom_bar()
```

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

Why doesn't histogram use the "bar" geom?


60
40
20
0
count
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```

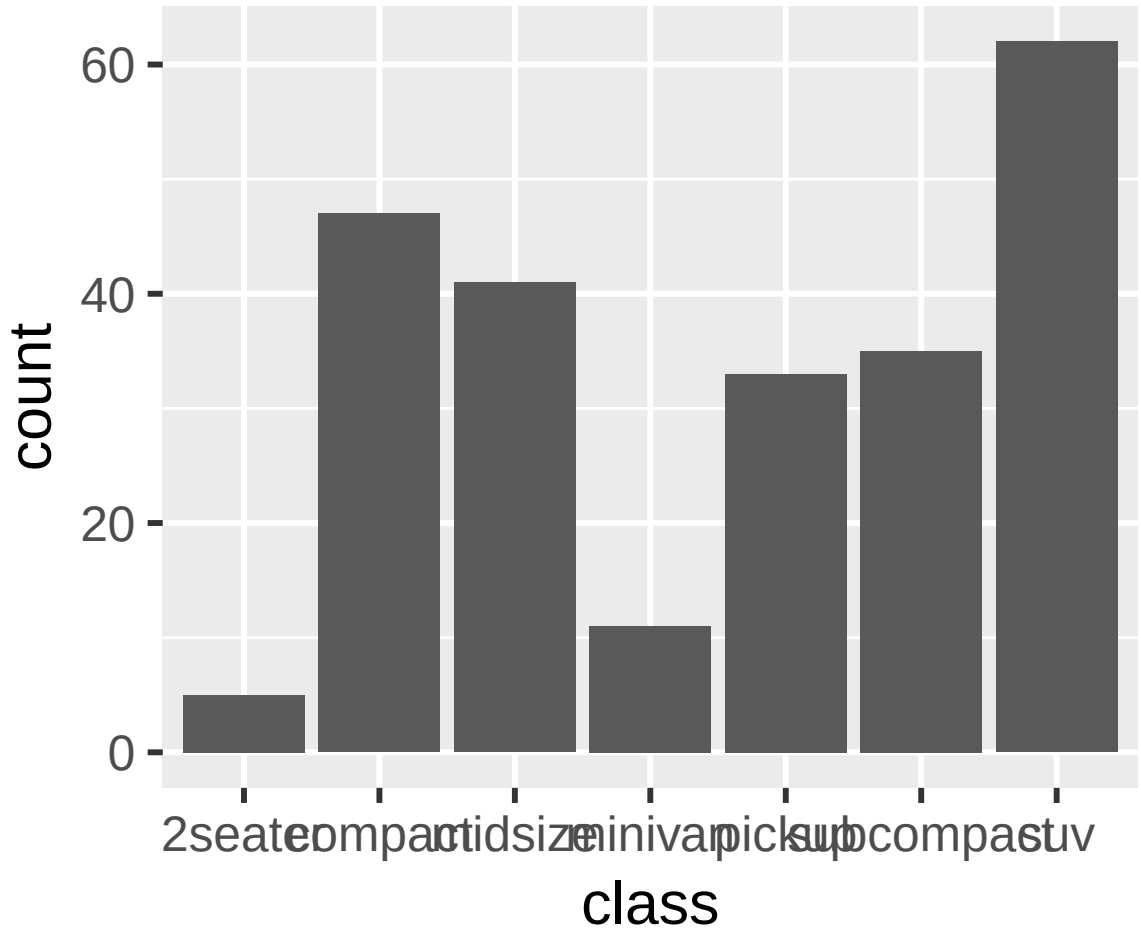


Histograms and bar plots

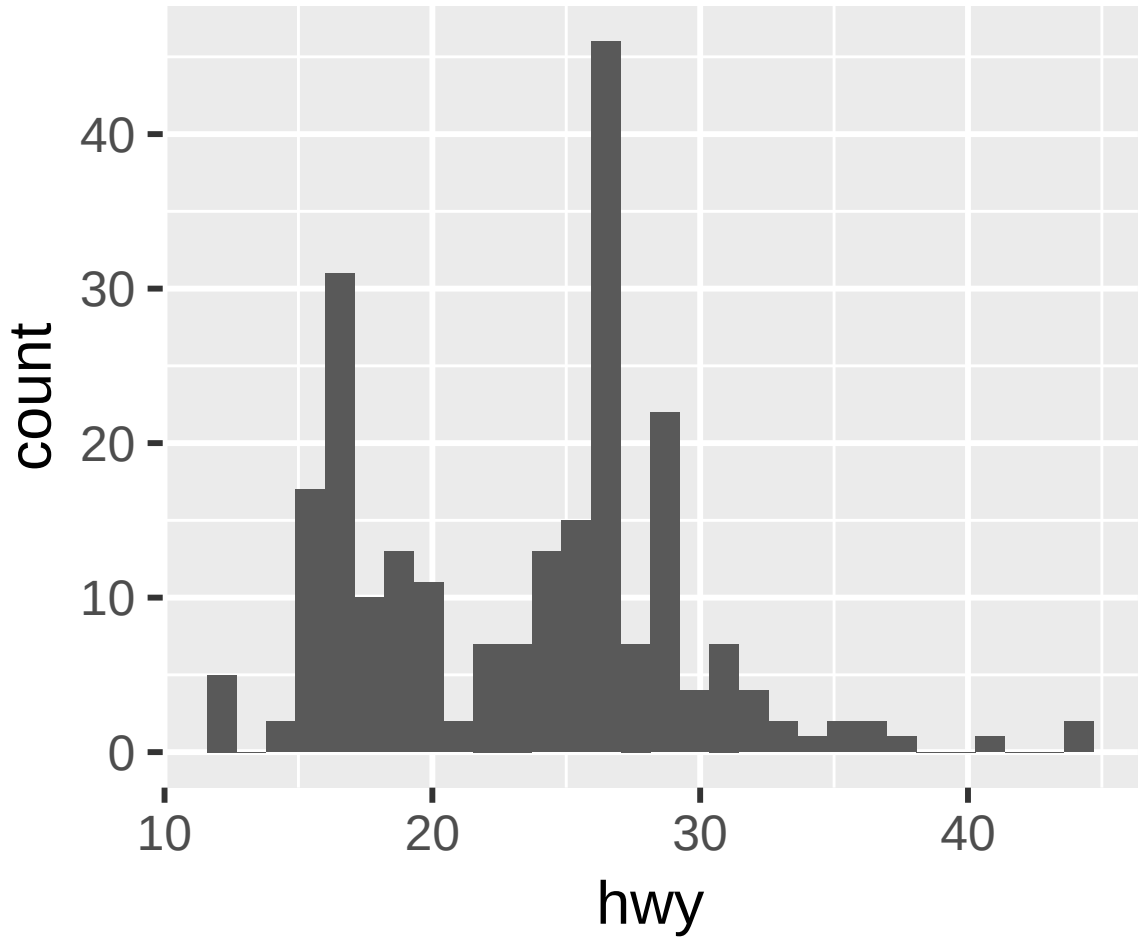
We can rewrite both explicitly to use `geom="bar"` and different stats.

```
ggplot(data = mpg, mapping = aes(x=class)) +
  layer(geom="bar",
        stat="count",
        position="identity")
```

```
ggplot(data = mpg, mapping = aes(x=hwy)) +
  layer(geom="bar",
        stat="bin",
        position="identity")
```


60
40
20
0
count
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



We could also use geom_bar for both

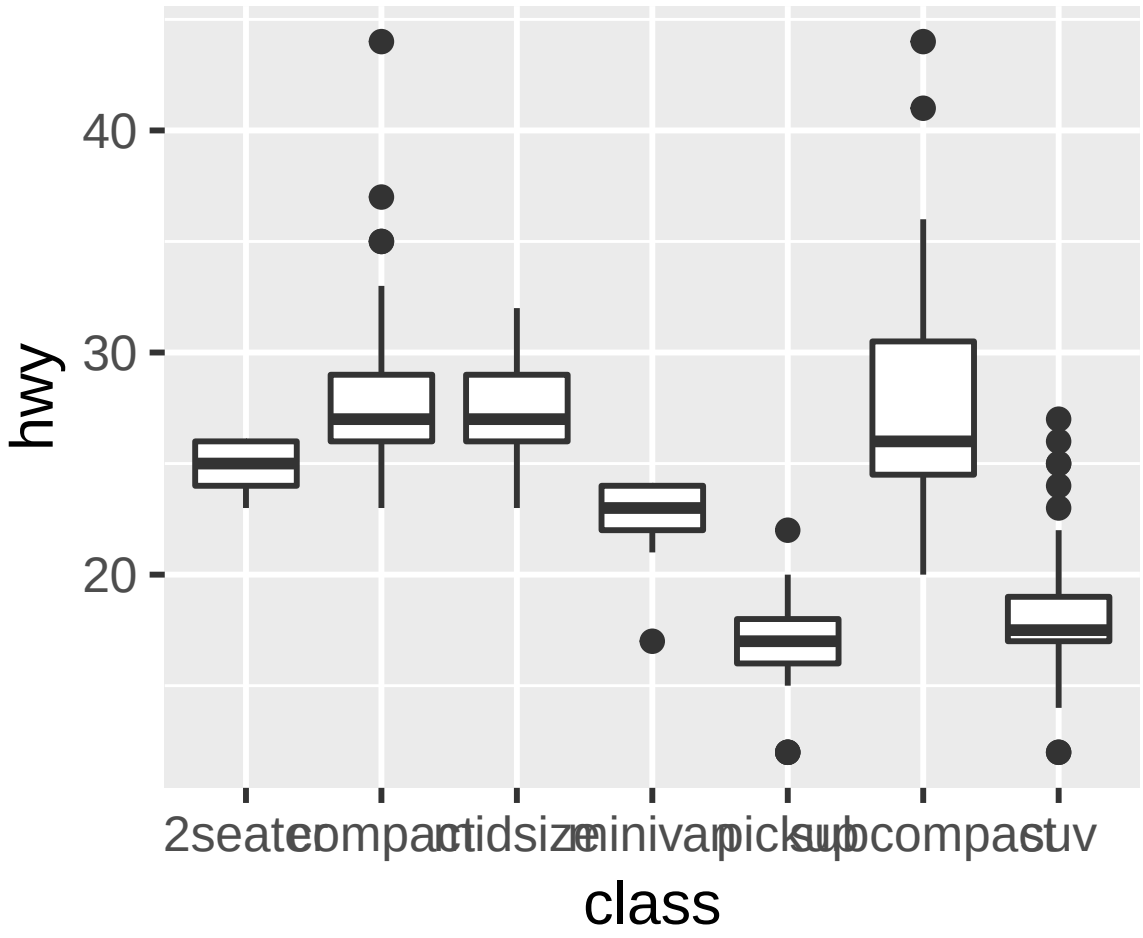
We can make a histogram with geom_bar by overwriting the default stat:

```
ggplot(data = mpg, mapping = aes(x=hwy)) +
  geom_bar(stat="bin")
```

Boxplots

- Boxplots are unique `geom` with a unique `stat`
- They plot the five-number summary + outliers
- Here we plot side-by-side boxplots for highway mpg by class

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


40
30
20
hwy
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

A template for plotting

We can develop a template for creating plots in `ggplot2`:

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

You will notice this is slightly different from the template that appears in the *R for Data Science* – how and why?

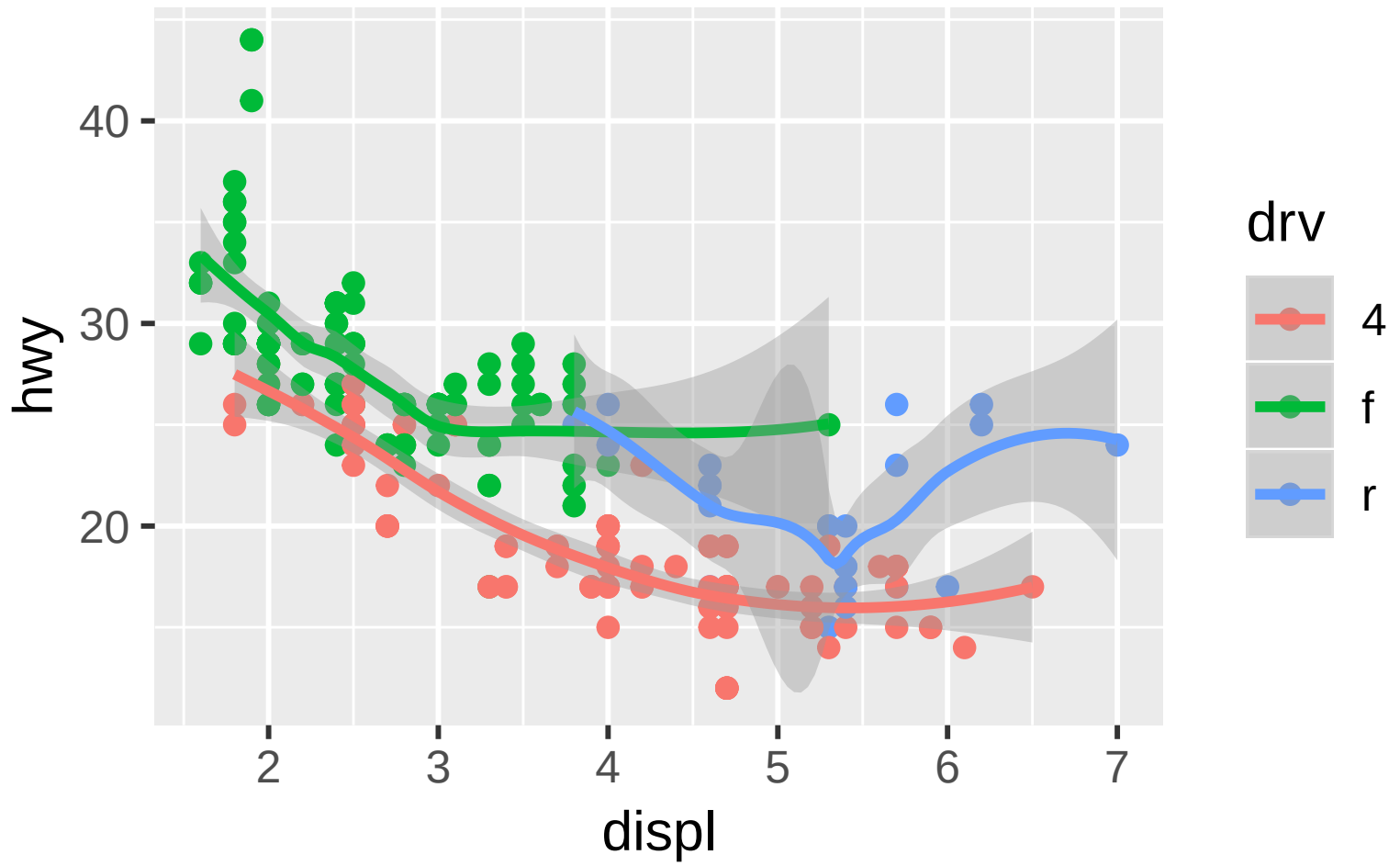
Overlaying geoms

As we've already seen, it is possible to overlay different geoms:

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

In this case, both geoms inherit the default data and aesthetic mappings.

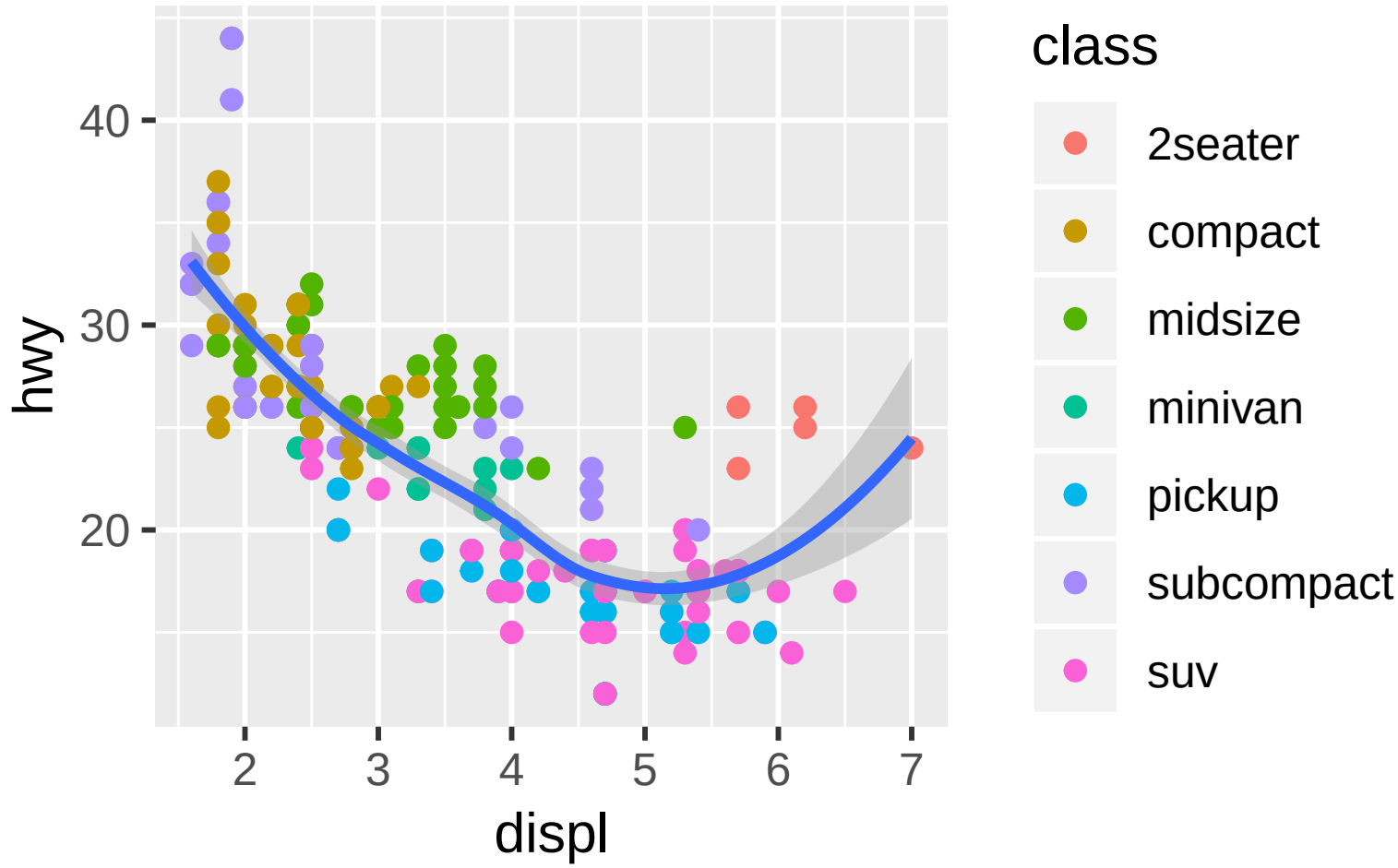
```
## `geom_smooth()` using method = 'loess' and formula 'y ~
```



Different aesthetics

However, suppose we want to use different aesthetics for each geom:

```
## `geom_smooth()` using method = 'loess' and formula 'y ~
```



Different aesthetics

We can either give each layer its own aesthetic mapping:

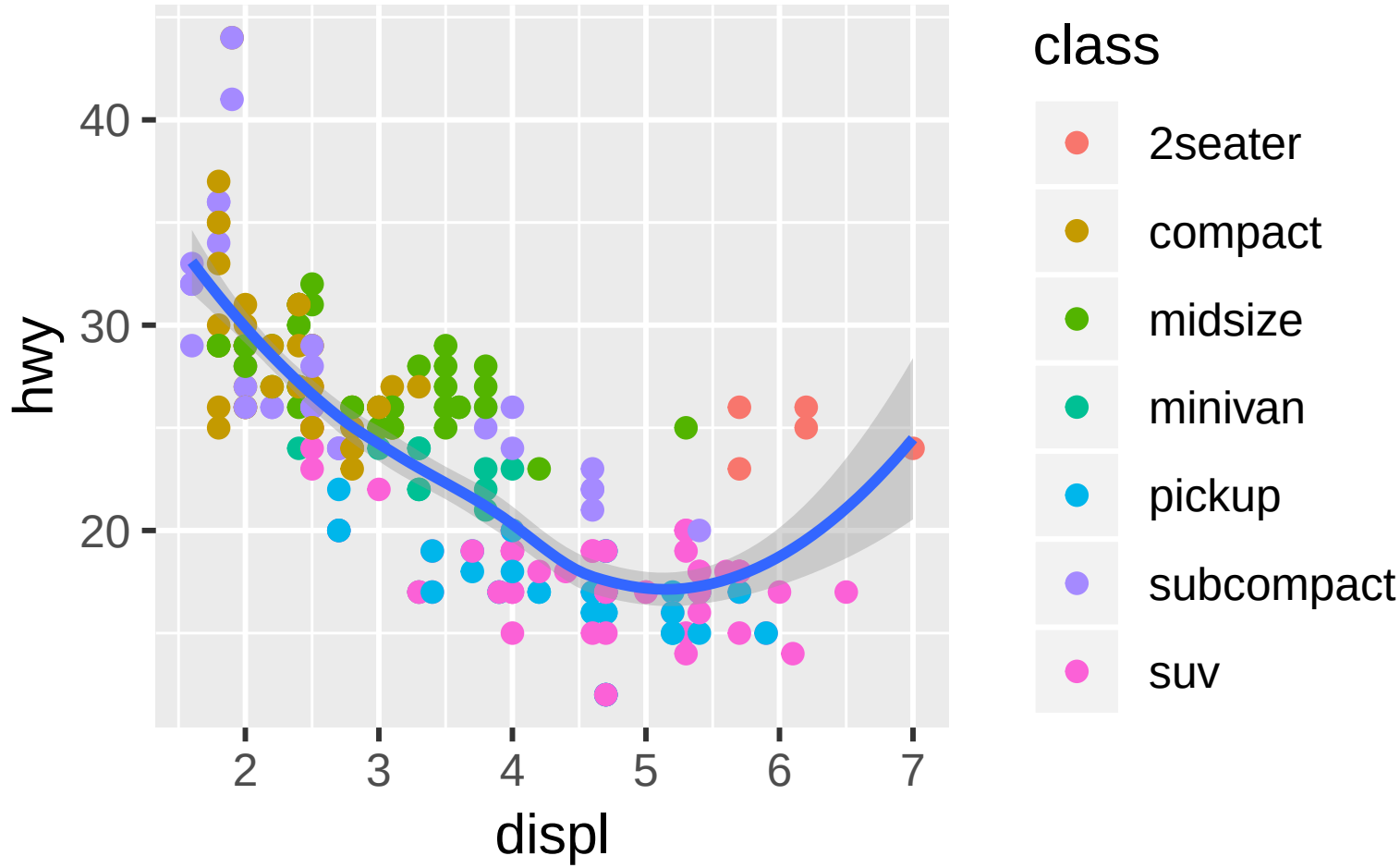
```
ggplot(data = mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, color=class)) +
  geom_smooth(mapping=aes(x=displ, y=hwy))
```

Or we can supply a default aesthetic and override it when necessary:

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

Both produce the same plot

```
## `geom_smooth()` using method = 'loess' and formula 'y ~
```



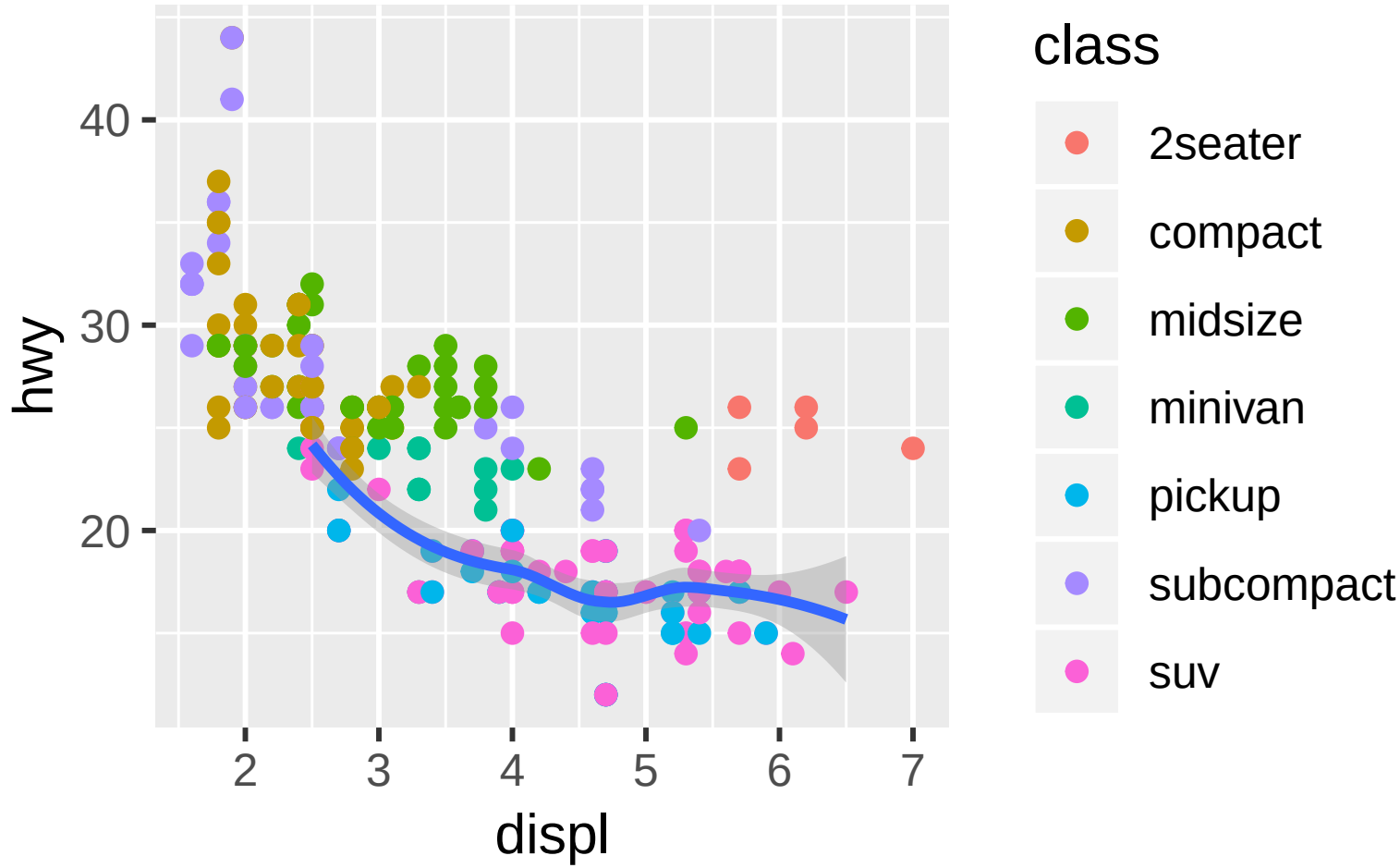
Different data

We can also specify different datasets for each layer, or allow them to inherit from the default dataset.

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth(data = mpg[mpg$class == "suv",])
```

Fitted line for SUVs only

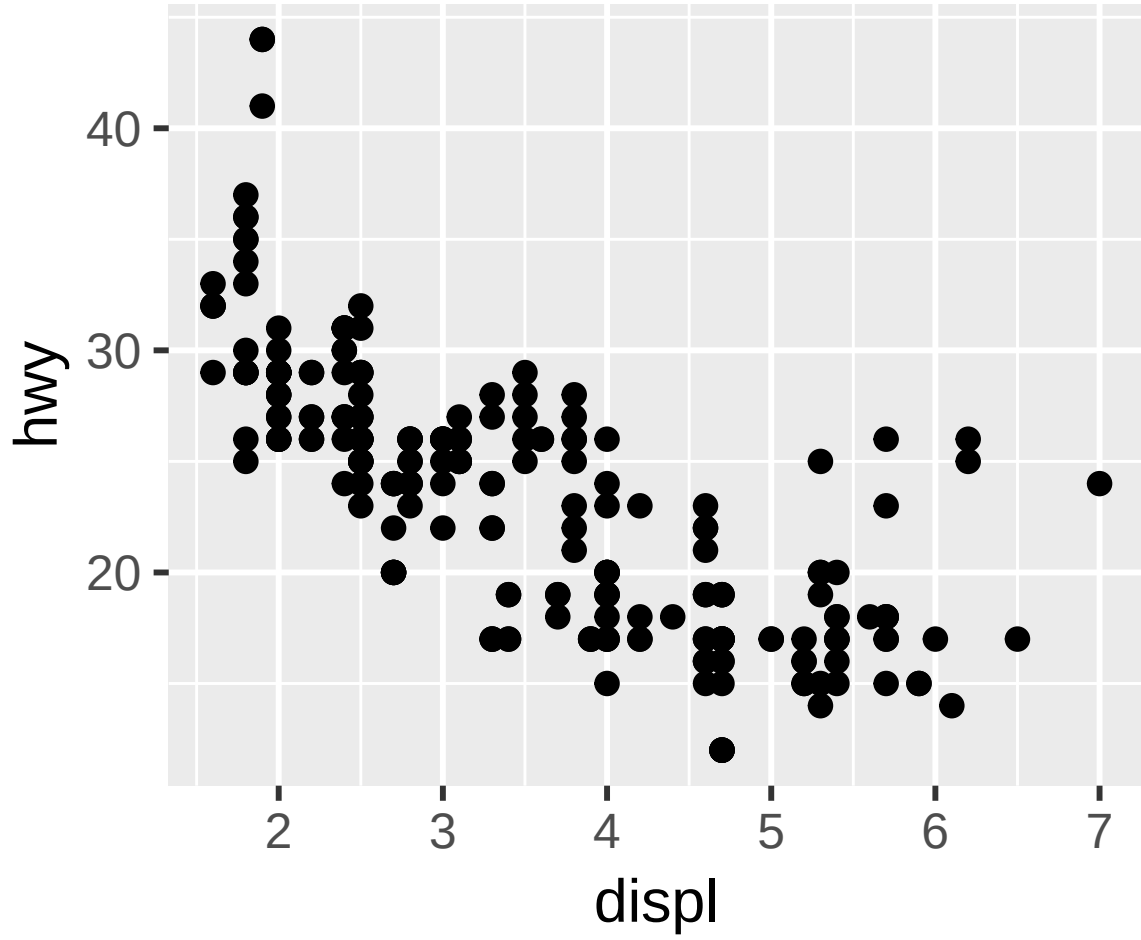
```
## `geom_smooth()` using method = 'loess' and formula 'y ~
```



Overplotting: rounded values

Consider the following plot:

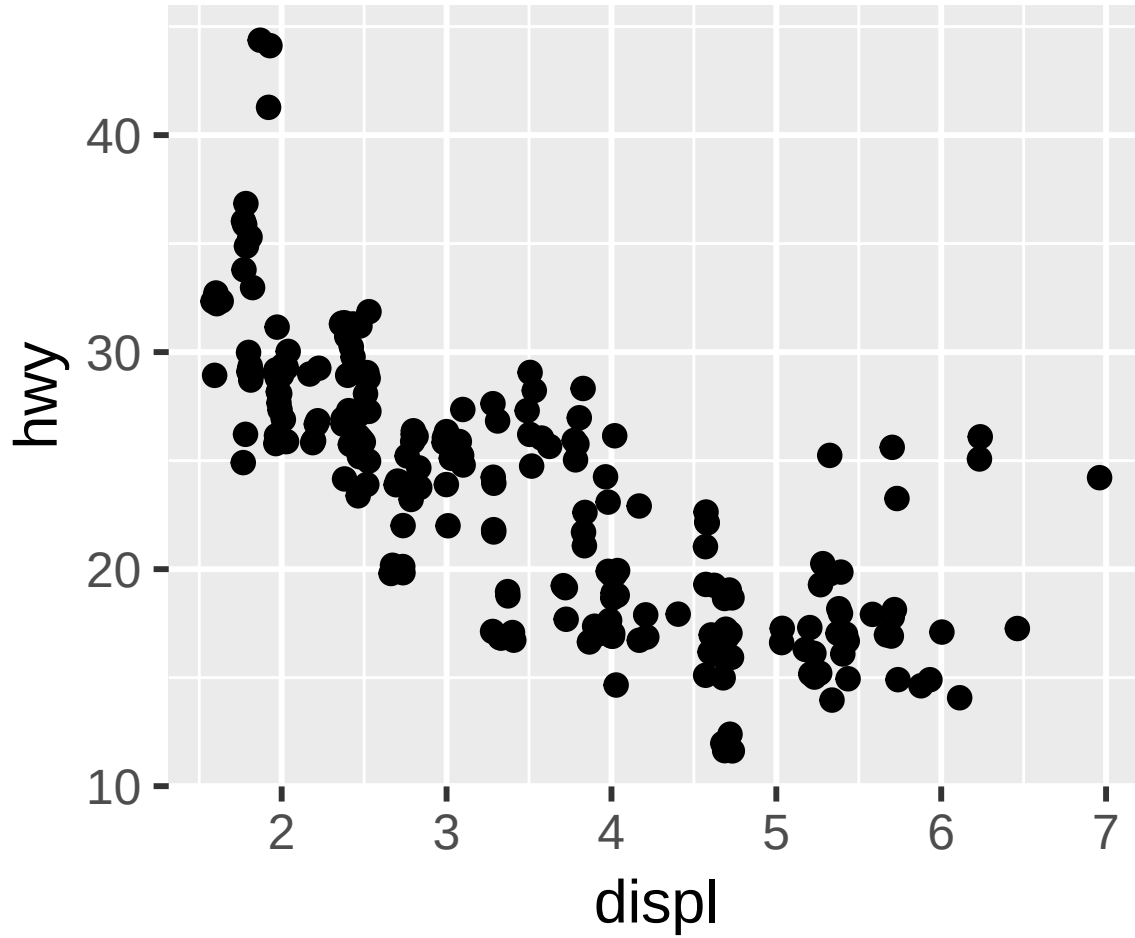
```
ggplot(data = mpg,
       mapping = aes(x=displ, y=hwy)) +
  geom_point()
```



Notice how all of the data points are on neat lines?

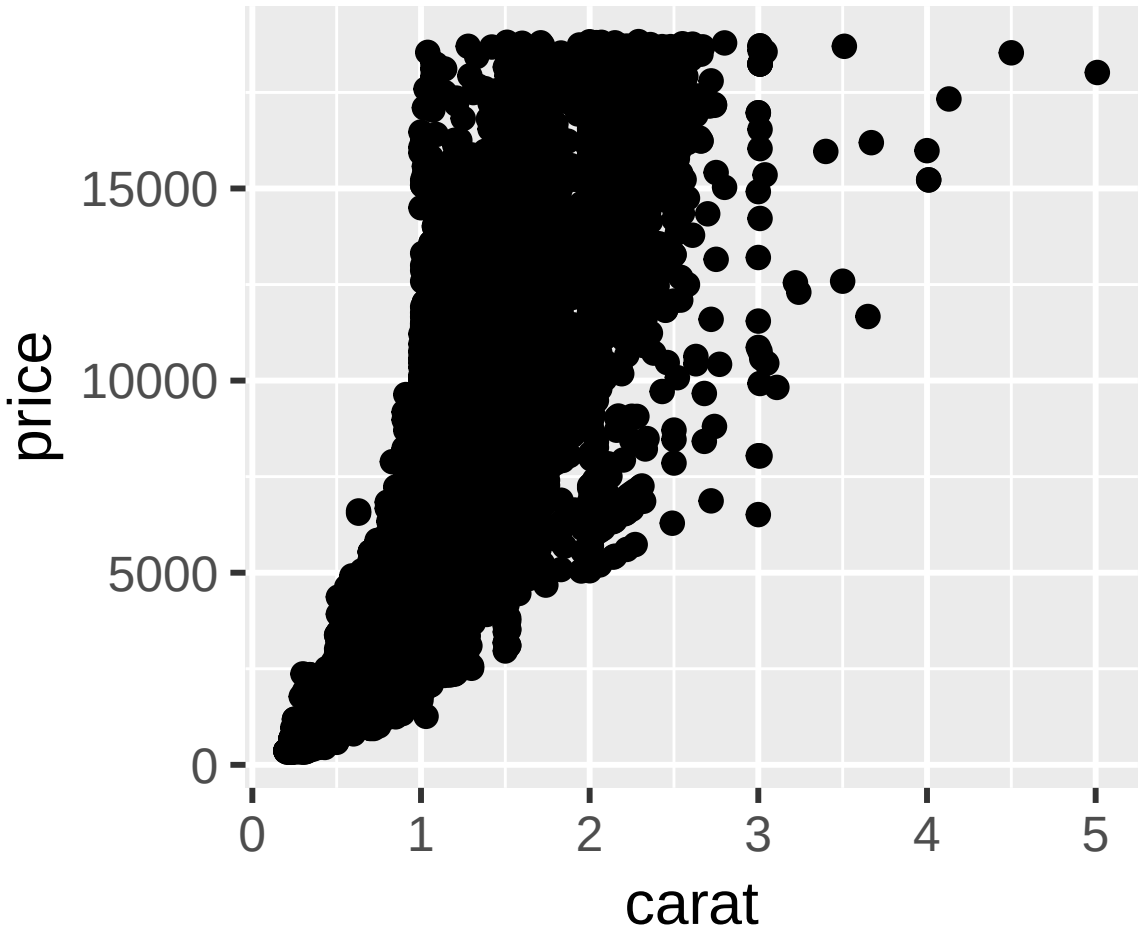
Solution: jitter

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


hwy
40
30
20
10
2
3
4
5
6
7
displ

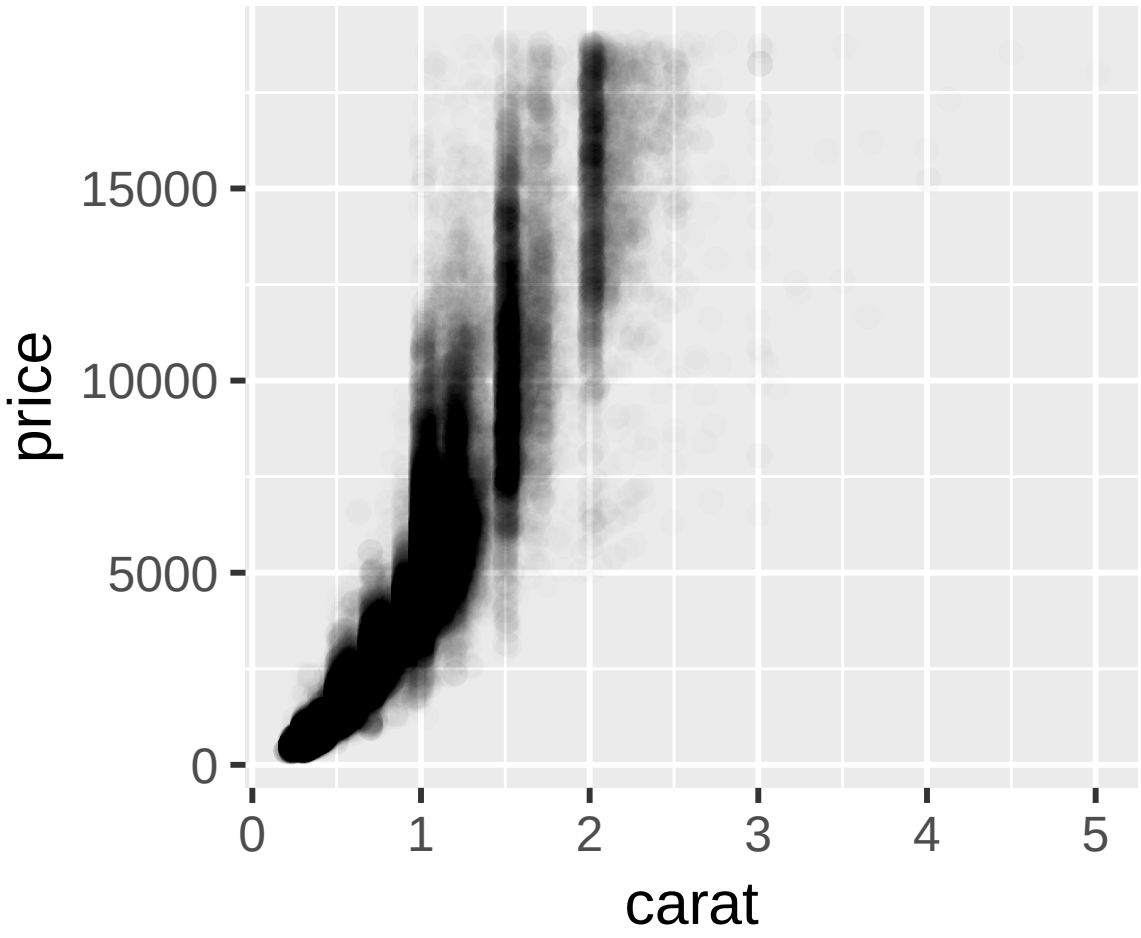
Overplotting: too much data

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


price
15000
10000
5000
0
0
1
2
3
4
5
carat

Solution: transparency

```
ggplot(diamonds, mapping=aes(x=carat, y=price)) +
  geom_point(alpha=1/100)
```


price
15000
10000
5000
0
0
1
2
3
4
5
carat