Aesthetic mappings

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

https://r4ds.hadley.nz/ Chapter 10 Layers

CSCI 297b, Spring 2023

May 7, 2023

tidyverse

```
library(tidyverse}
```

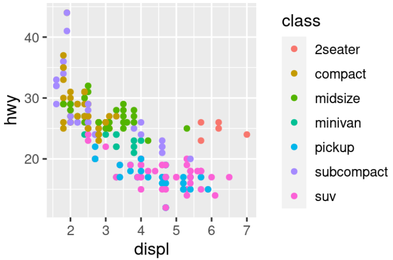
The mpg dataset

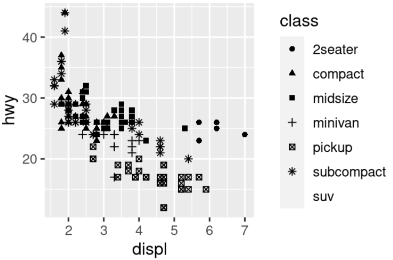
```
mpg
#> # A tibble: 234 × 11
#>   manufacturer model displ  year   cyl trans      drv     cty   hwy fl
#>   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
#> 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p
#> 4 audi         a4      2    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
#> # i 228 more rows
#> # i 1 more variable: class <chr>
```

- displ: A car's engine size, in liters. A numerical variable.
- hwy: A car's fuel efficiency on the highway, in miles per gallon (mpg). A car with a low fuel efficiency consumes more fuel than a car with a high fuel efficiency when they travel the same distance. A numerical variable.
- class: Type of car. A categorical variable.

Groups can go unplotted

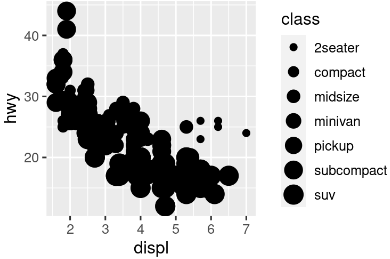
```
# Left
ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point()

# Right
ggplot(mpg, aes(x = displ, y = hwy, shape = class)) +
  geom_point()
#> Warning: The shape palette can deal with a maximum of 6 discrete values
#> because more than 6 becomes difficult to discriminate; you have 7.
#> Consider specifying shapes manually if you must have them.
#> Warning: Removed 62 rows containing missing values (‘geom_point()‘).
```

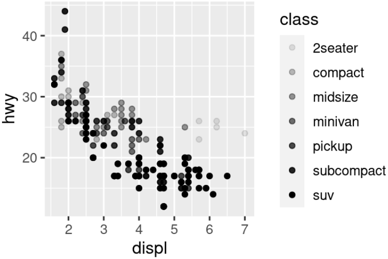




not advised

```
# Left
ggplot(mpg, aes(x = displ, y = hwy, size = class)) +  geom_point()
#> Warning: Using size for a discrete variable is not advised.

# Right
ggplot(mpg, aes(x = displ, y = hwy, alpha = class)) +  geom_point()
#> Warning: Using alpha for a discrete variable is not advised.
```





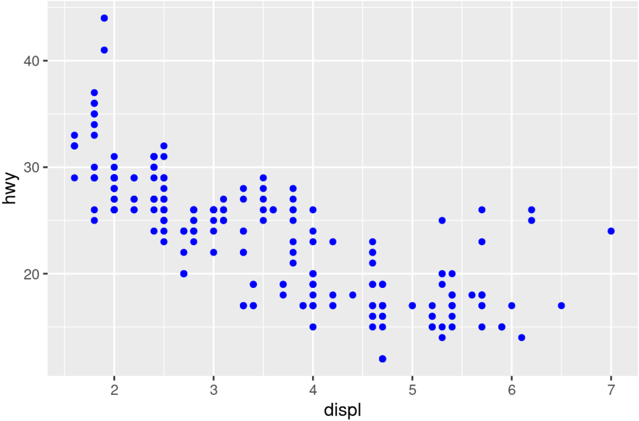
- Implies an order that does not exist.

ggplot2 defaults

- It selects a reasonable scale to use with the aesthetic.
- It constructs a legend that explains the mapping between levels and values.
- For x and y aesthetics, ggplot2 does not create a legend.
- But it creates an axis line with tick marks and a label.
- The axis line provides the same information as a legend.
- It explains the mapping between locations and values.

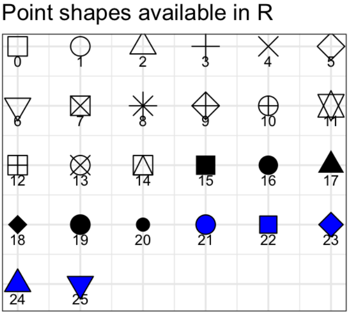
Set visual properties manually

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



Properties of points

- Color as character string, e.g. `color = "blue"`
- Size in mm, e.g. `size = 1`
- Shape as integer, e.g. `shape = 1`

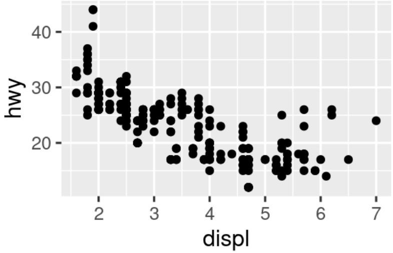


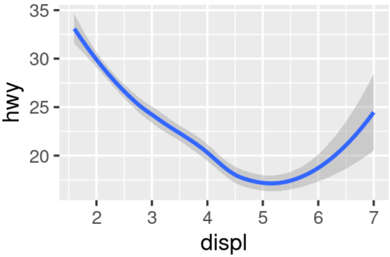
- R has 25 built-in shapes that are identified by numbers.
- There are some seeming duplicates: for example, 0, 15, and 22 are all squares.
- The difference comes from the interaction of the `color` and `fill` aesthetics.
- The hollow shapes (0–14) have a border determined by `color`
- The solid shapes (15–20) are filled with `color`
- The filled shapes (21–24) have a border of `color` and are filled with `fill`.

`https://ggplot2.tidyverse.org/articles/ggplot2-specs.html`

Do exercise 8

How are these plots similar?





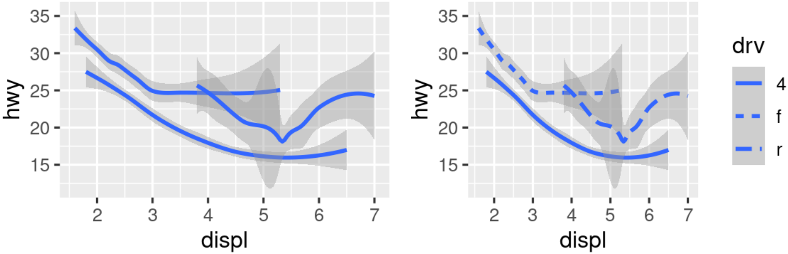
```
# Left
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point()

# Right
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth()
#> ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’
```

Mapping arguments

- Every geom function in `ggplot2` takes a mapping argument.
- It is either defined locally in the geom layer or globally in the `ggplot()` layer.
- Not every aesthetic works with every geom.
- You could set the shape of a point, but you couldn't set the "shape" of a line.
- If you try, `ggplot2` will silently ignore that aesthetic mapping.
- On the other hand, you could set the linetype of a line.
- `geom_smooth()` will draw a different line, with a different linetype, for each unique value of the variable that you map to linetype.

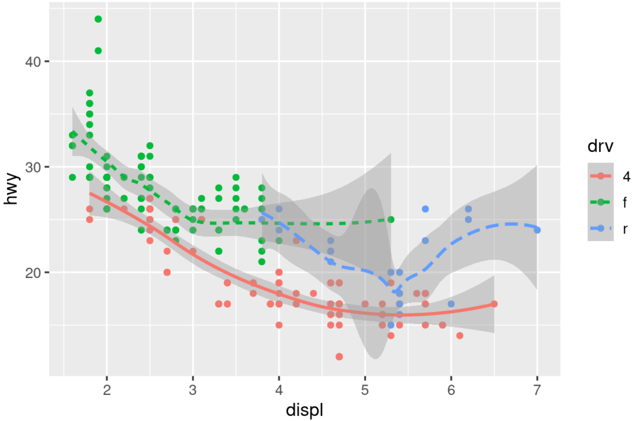
Mapping arguments

```
# Left
ggplot(mpg, aes(x = displ, y = hwy, shape = drv)) +
  geom_smooth()

# Right
ggplot(mpg, aes(x = displ, y = hwy, linetype = drv)) +
  geom_smooth()
```



Mapping arguments

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

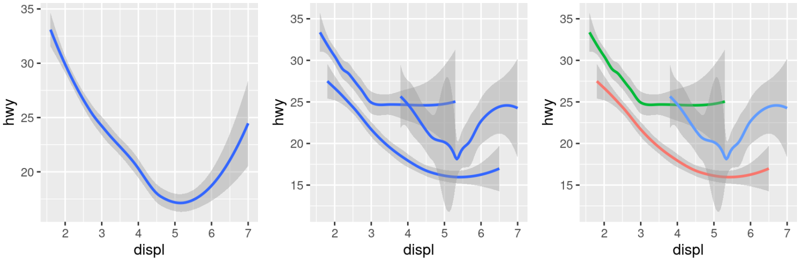


group aesthetic

- Many geoms, like `geom_smooth()`, use a single geometric object to display multiple rows of data.
- For these geoms, you can set the group aesthetic to a categorical variable to draw multiple objects.
- `ggplot2` will draw a separate object for each unique value of the grouping variable.
- In practice, `ggplot2` will automatically group the data for these geoms whenever you map an aesthetic to a discrete variable (as in the linetype example).
- It is convenient to rely on this feature because the group aesthetic by itself does not add a legend or distinguishing features to the geoms.

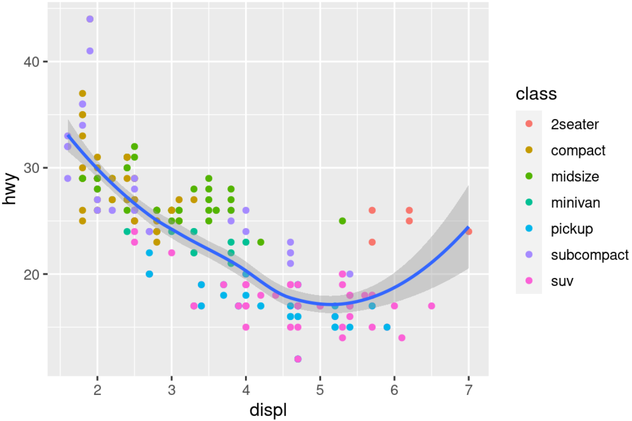
group aesthetic

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

# Middle
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(aes(group = drv))

# Right
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(aes(color = drv), show.legend = FALSE)
```



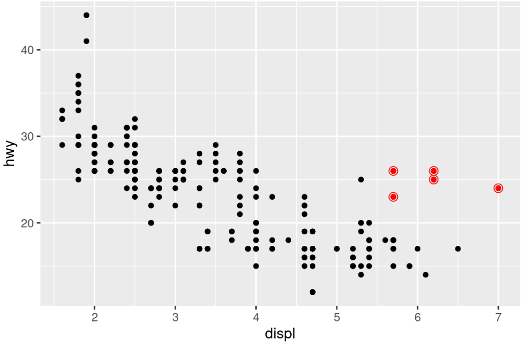
Different aesthetics in different layers

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



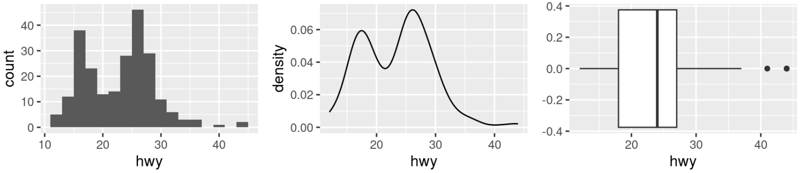
Different data in different layers

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  geom_point(
    data = mpg |> filter(class == "2seater"),
    color = "red"
  ) +
  geom_point(
    data = mpg |> filter(class == "2seater"),
    shape = "circle open", size = 3, color = "red"
  )
```



geoms change everything

```
# Left
ggplot(mpg, aes(x = hwy)) +
  geom_histogram(binwidth = 2)

# Middle
ggplot(mpg, aes(x = hwy)) +
  geom_density()

# Right
ggplot(mpg, aes(x = hwy)) +
  geom_boxplot()
```

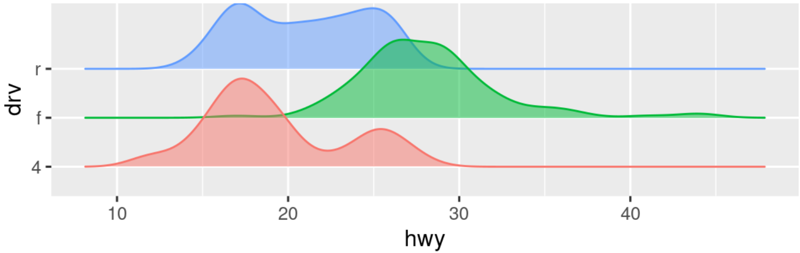


Extension packages

- ggplot2 provides more than 40 geoms.
- But these don't cover all possible plots one could make.
- If you need a different geom, we recommend looking into extension packages first to see if someone else has already implemented it.
- https://exts.ggplot2.tidyverse.org/gallery/
- For example, the ggridges package https://wilkelab.org/ggridges is useful for making ridgeline plots, which can be useful for visualizing the density of a numerical variable for different levels of a categorical variable.

Ridges example

```
library(ggridges)

ggplot(mpg, aes(x = hwy, y = drv, fill = drv, color = drv)) +
  geom_density_ridges(alpha = 0.5, show.legend = FALSE)
#> Picking joint bandwidth of 1.28
```



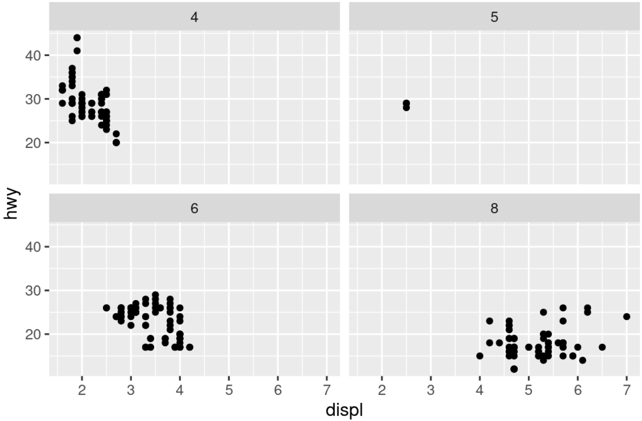
The tidyverse reference

https://ggplot2.tidyverse.org/reference

Do exercise 9

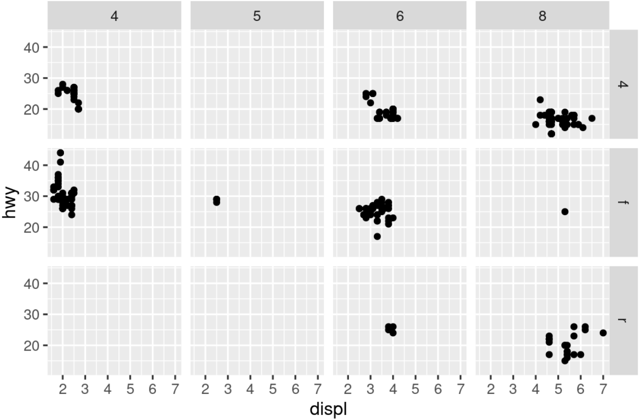
facet_wrap

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



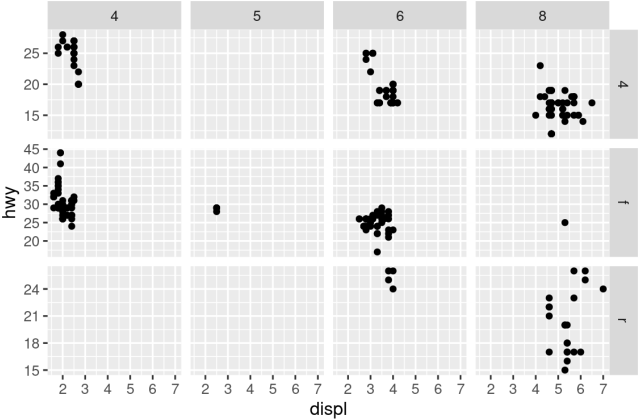
facet_grid

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



Free the scales!

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



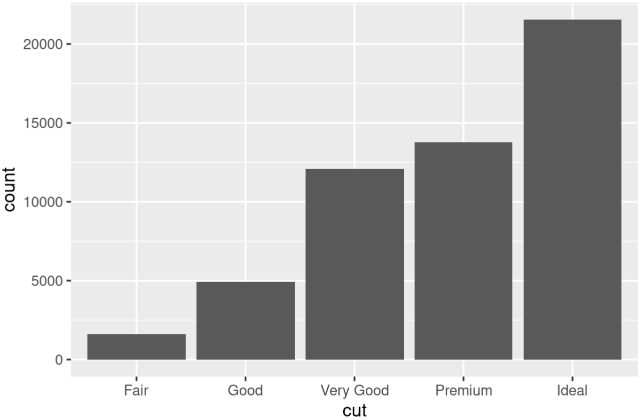
Do exercise 10

diamonds dataset

- The diamonds dataset is in the `ggplot2` package.
- It contains information on $\approx 54,000$ diamonds.
- It includes carat, color, clarity, and cut of each diamond.

Bar chart

```
ggplot(diamonds, aes(x = cut)) +
  geom_bar()
```



- count is not in the dataset!

Statistical transformations

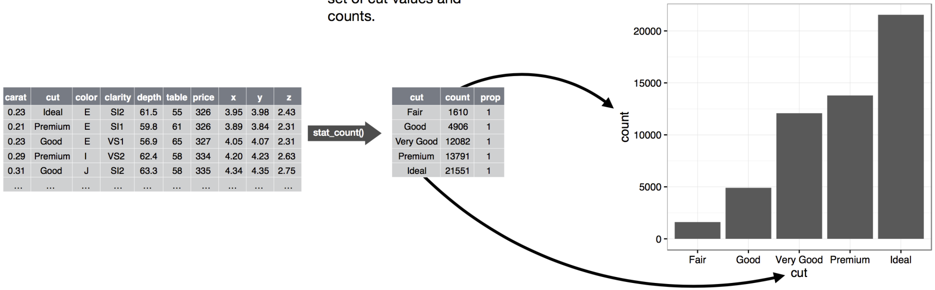
- Many graphs, like scatterplots, plot the raw values of your dataset.
- Other graphs, like bar charts, calculate new values to plot:
  - Bar charts, histograms, and frequency polygons bin your data and then plot bin counts, the number of points that fall in each bin.
  - Smoothers fit a model to your data and then plot predictions from the model.
  - Boxplots compute the five-number summary of the distribution and then display that summary as a specially formatted box.
- The algorithm used to calculate new values for a graph is called a **stat**, short for statistical transformation.

Computing stats

1. **geom_bar()** begins with the **diamonds** data set

2. **geom_bar()** transforms the data with the "count" stat, which returns a data set of cut values and counts.

3. **geom_bar()** uses the transformed data to build the plot. cut is mapped to the x axis, count is mapped to the y axis.



carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

stat_count()

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

Stats

- You can learn which stat a geom uses by inspecting the default value for the `stat` argument.
- For example, `?geom_bar` shows that the default value for stat is “count”, which means that `geom_bar()` uses `stat_count()`.
- `stat_count()` is documented on the same page as `geom_bar()`.
- If you scroll down, the section called “Computed variables” explains that it computes two new variables: `count` and `prop`.

Stats

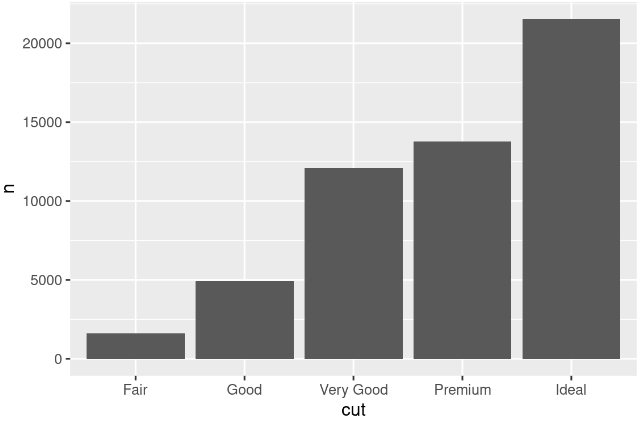
- Every geom has a default stat; and every stat has a default geom.
- This means that you can typically use geoms without worrying about the underlying statistical transformation.
- However, there are reasons why you might need to use a stat explicitly.

We might already have the value computed

```
> diamonds |>
+   count(cut)
# A tibble: 5 × 2
  cut           n
  <ord>     <int>
1 Fair       1610
2 Good       4906
3 Very Good 12082
4 Premium   13791
5 Ideal     21551
```

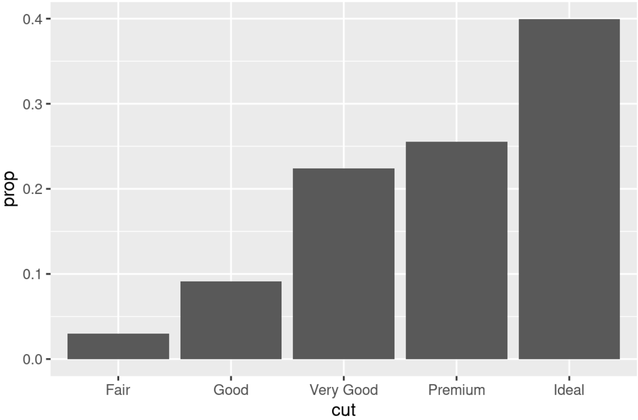
We might already have the value computed

```
diamonds |>
  count(cut) |>
  ggplot(aes(x = cut, y = n)) +
  geom_bar(stat = "identity")
```



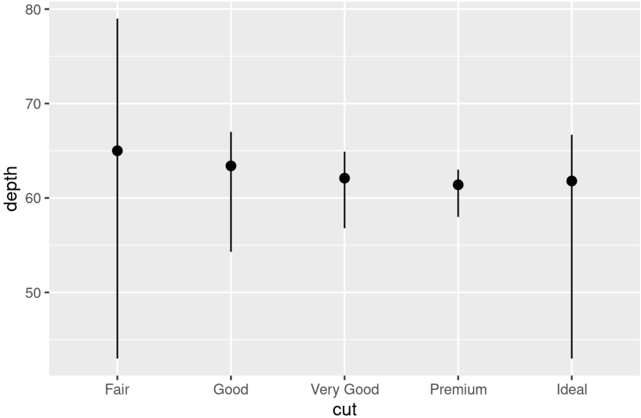
We might want to show proportion instead of count

```
ggplot(diamonds, aes(x = cut, y = after_stat(prop), group = 1)) +
  geom_bar()
```



We might want to use stat_summary

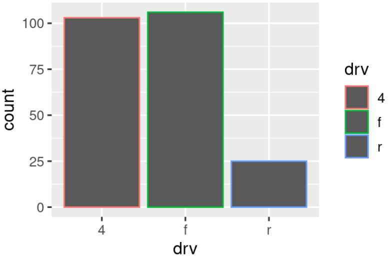
```
ggplot(diamonds) +
  stat_summary(
    aes(x = cut, y = depth),
    fun.min = min,
    fun.max = max,
    fun = median
  )
```

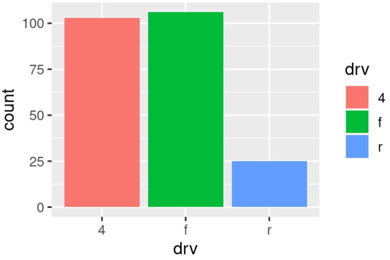


Do exercise 11

Bar charts can use either color or fill

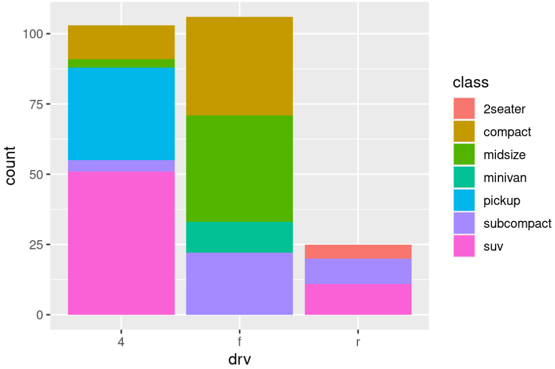
```
# Left
ggplot(mpg, aes(x = drv, color = drv)) +
  geom_bar()

# Right
ggplot(mpg, aes(x = drv, fill = drv)) +
  geom_bar()
```





Map `fill` to another variable

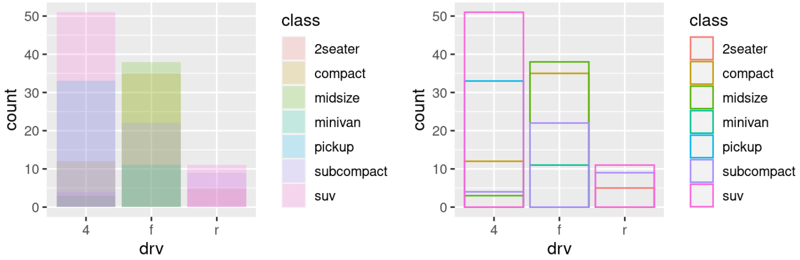
```
ggplot(mpg, aes(x = drv, fill = class)) +
  geom_bar()
```



- The stacking is performed automatically using the **position adjustment** specified by the `position` argument.
- Can be "identity", "dodge", or "fill".

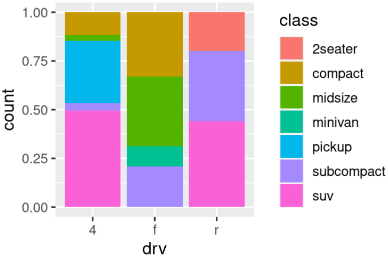
identity not useful for bars

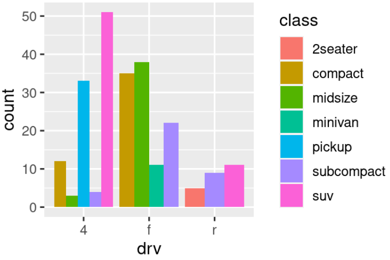
```
# Left
ggplot(mpg, aes(x = drv, fill = class)) +
  geom_bar(alpha = 1/5, position = "identity")

# Right
ggplot(mpg, aes(x = drv, color = class)) +
  geom_bar(fill = NA, position = "identity")
```



fill and dodge

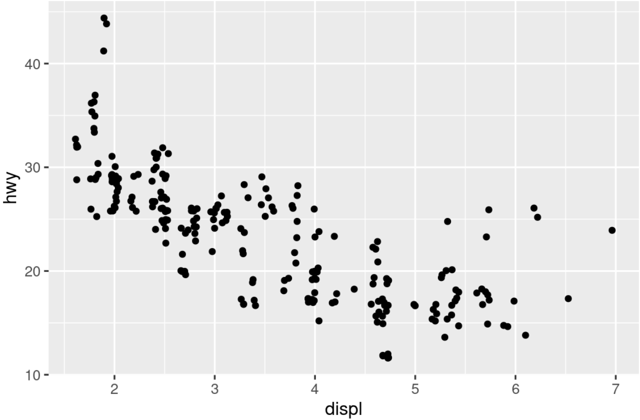
```
# Left
ggplot(mpg, aes(x = drv, fill = class)) +
  geom_bar(position = "fill")

# Right
ggplot(mpg, aes(x = drv, fill = class)) +
  geom_bar(position = "dodge")
```





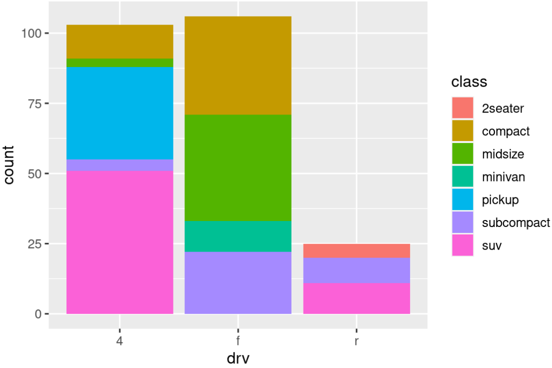
`position = jitter` useful for scatterplots

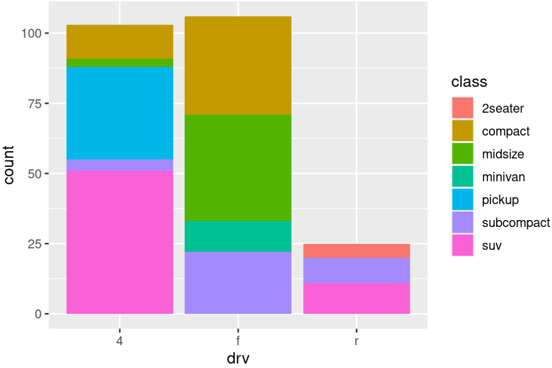
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(position = "jitter")
```

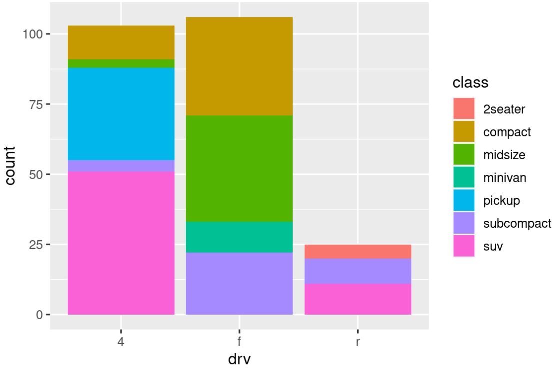


- `geom_jitter()` is shorthand for `geom_point(position = "jitter")`

Do exercise 12


count
100
75
50
25
0
4
f
r
drv
class
2seater
compact
midsize
minivan
pickup
subcompact
suv


100
75
50
25
0
count
4
f
r
drv
class
2seater
compact
midsize
minivan
pickup
subcompact
suv


100
75
50
25
0
count
4
f
r
drv
class
2seater
compact
midsize
minivan
pickup
subcompact
suv