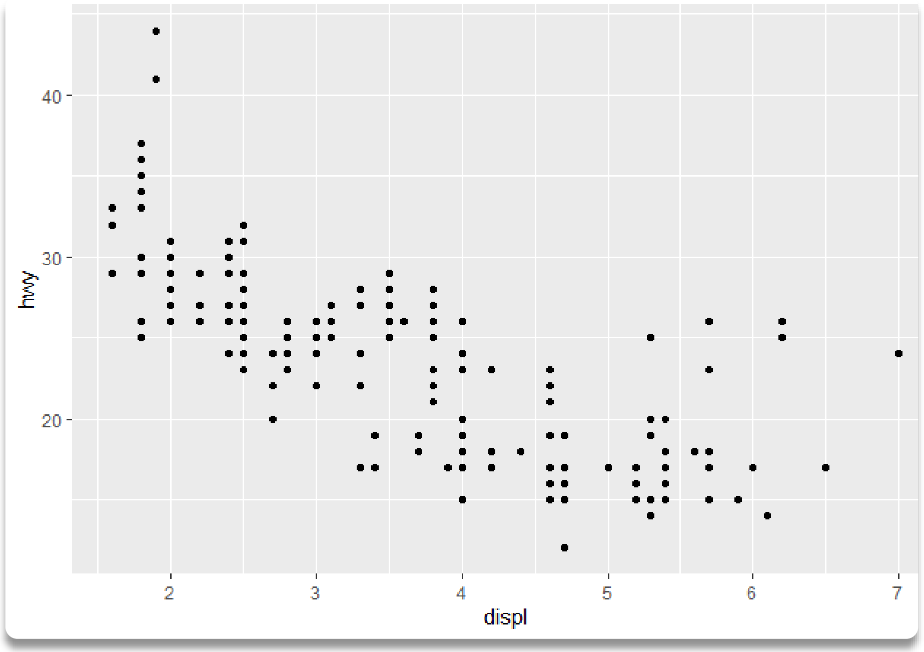
Data visualization

Data visualization



- data = as.data.frame(mpg)
- ## for inbuilt data, "as.data.frame" is not needed. But for other data, it is needed.
- ## displ, a car's engine size, in litres
- ## hwy, a car's fuel efficiency on the highway, in miles per gallon (mpg).
- ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy))

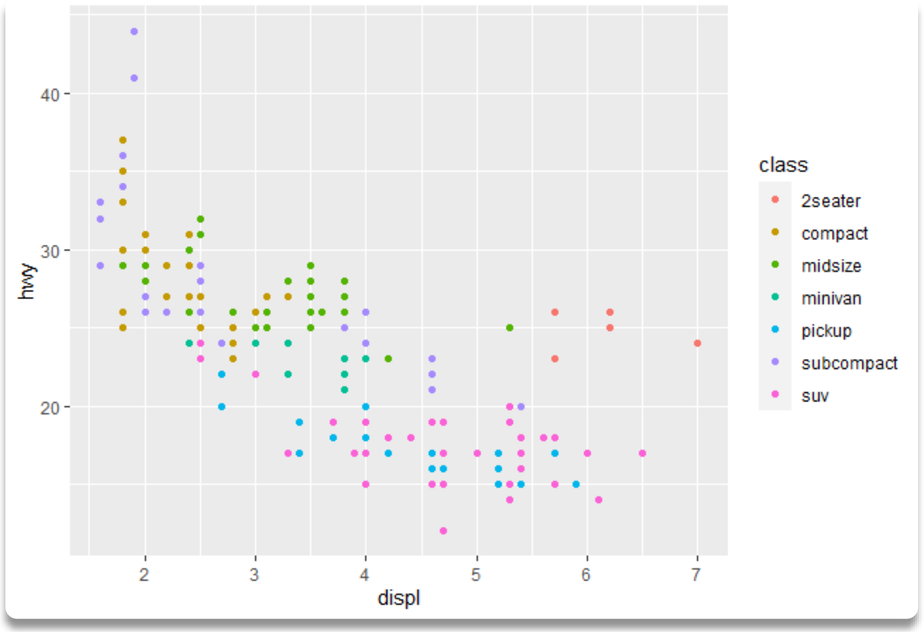
Data visualization

- A graphic template:

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

- To make a graph, replace the bracketed sections in the code below with a dataset, a geom function, or a collection of mappings.

Data visualization

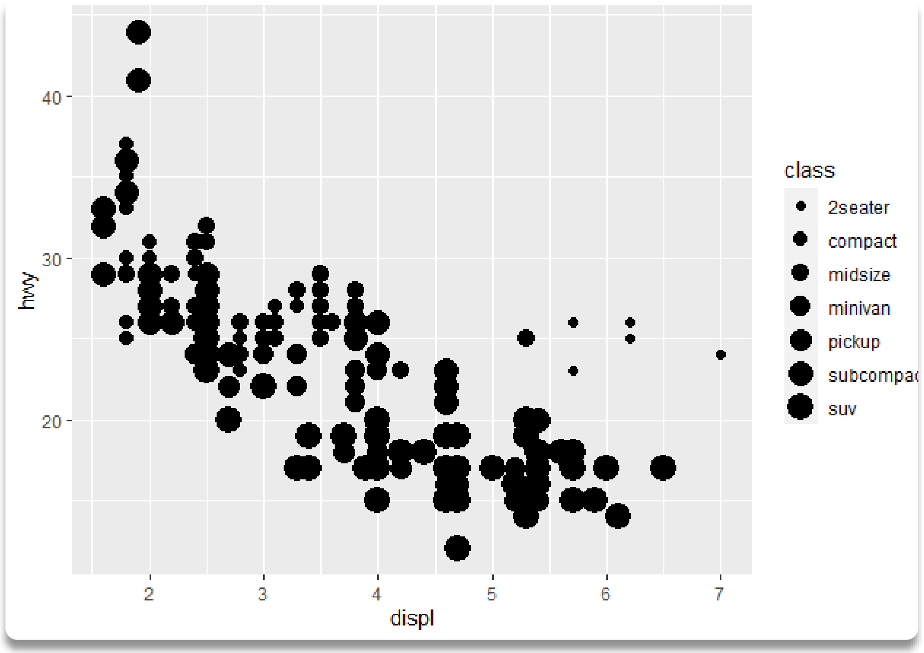


- ## you can map the colors of your points to the class variable to reveal the class of each car.

- ggplot(data = mpg) +

 geom_point(mapping = aes(x = displ, y = hwy, color = class))

Data visualization

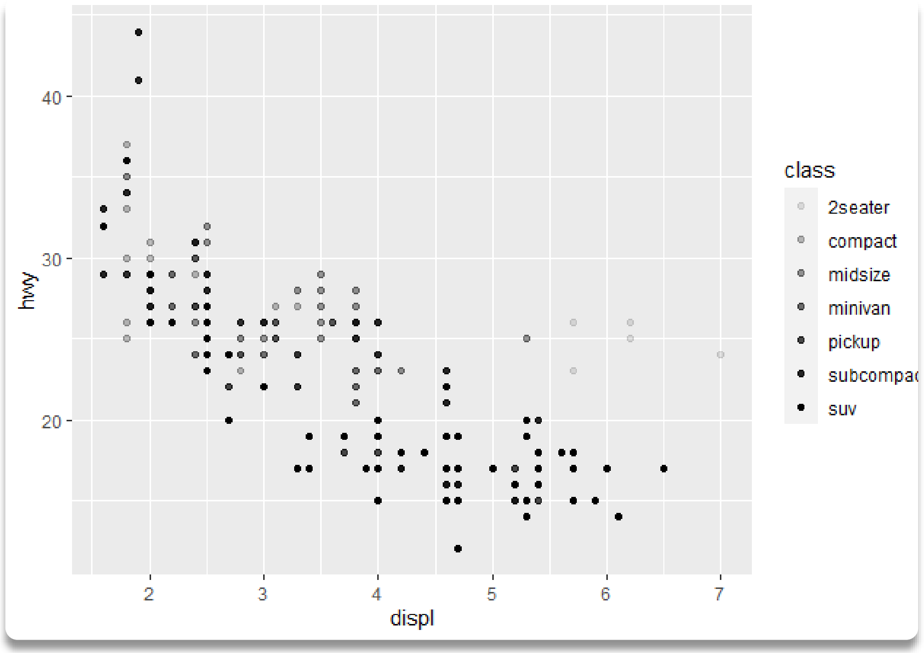


- ggplot(data = mpg) +

  geom_point(mapping = aes(x = displ, y = hwy, size = class))

#> Warning: Using size for a discrete variable is not advised.

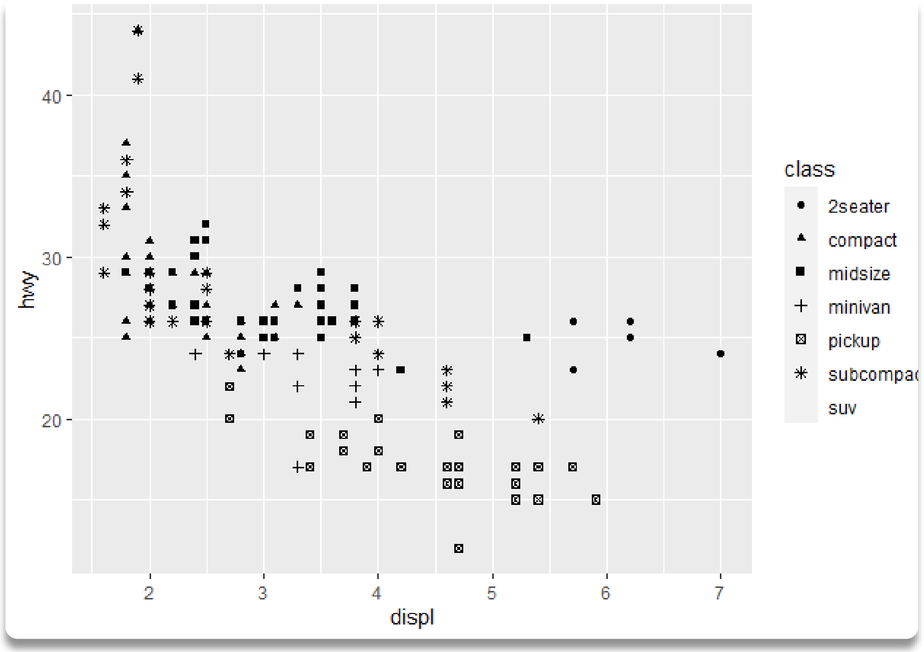
Data visualization



- ggplot(data = mpg) +

 geom_point(mapping = aes(x = displ, y = hwy, alpha = class))

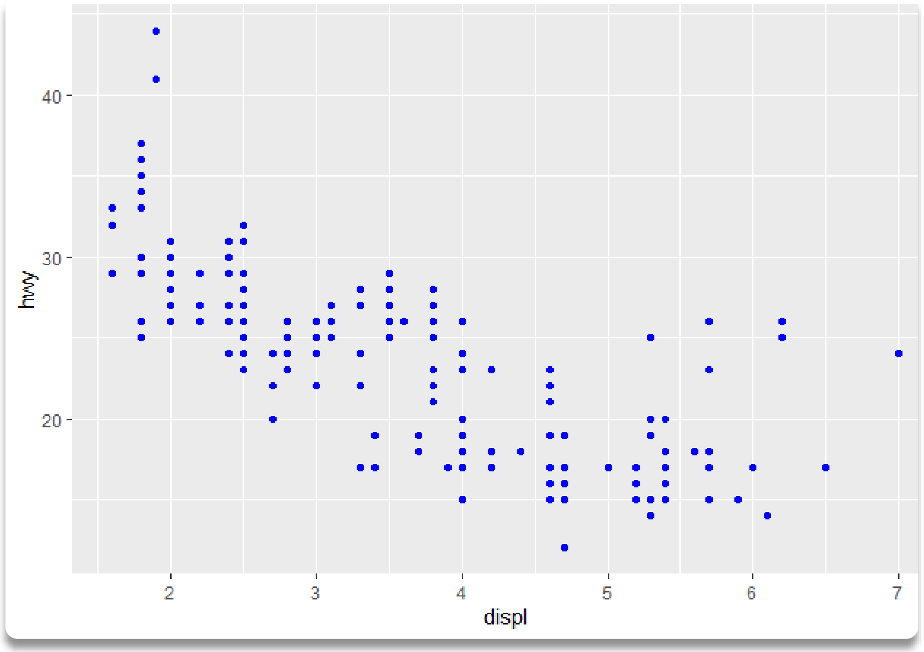
Data visualization



- ggplot(data = mpg) +

  geom_point(mapping = aes(x = displ, y = hwy, shape = class))

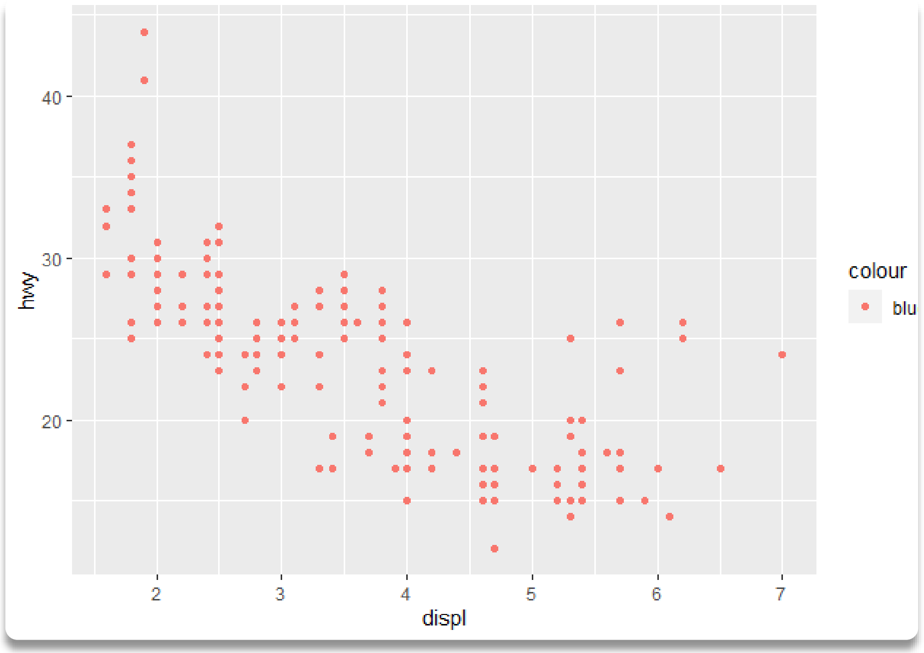
Data visualization



- If you want to set the aesthetic properties manually, you need to set aesthetic as an argument of the geom() function, outside aes() function.
- Example:
- ggplot(data = mpg) +

  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")

Data visualization



- If you set the aesthetic manually inside the geom() function, it will not work.

- Example:
- ggplot(data = mpg) +

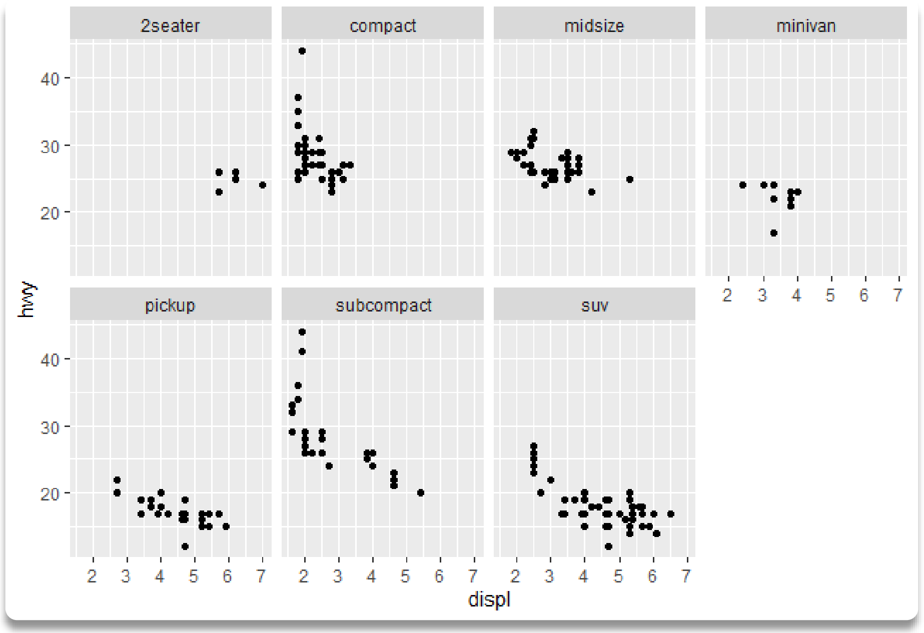
  geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))

Data visualization

- One common problem when creating ggplot2 graphics is to put the + in the wrong place: it has to come at the end of the line, not the start. In other words, make sure you haven’t accidentally written code like this:
- ggplot(data = mpg)

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

Data visualization

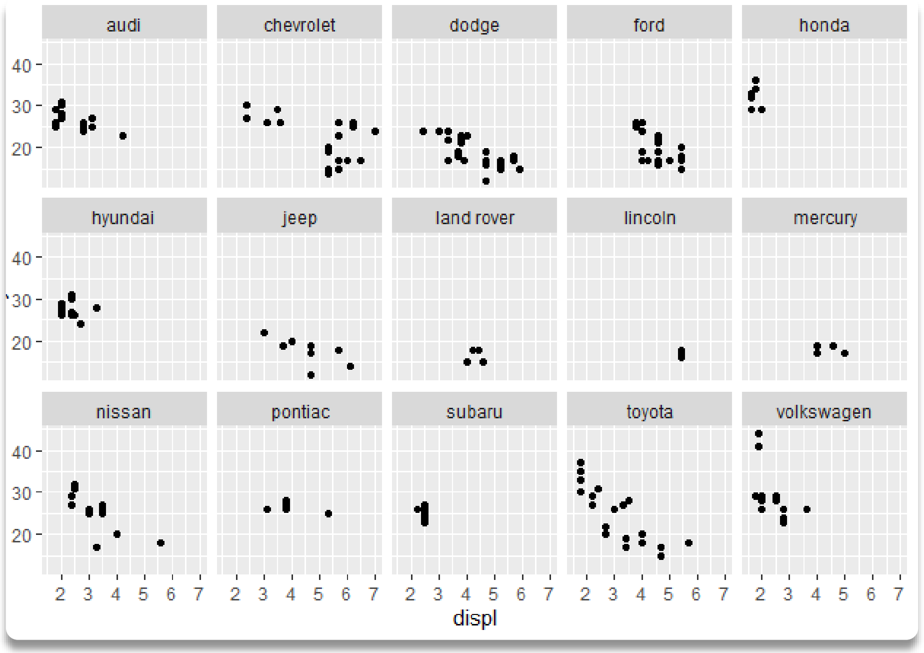


- #One way to add additional variables is with aesthetics. Another way, particularly useful for categorical variables, is to split your plot into facets, subplots that each display one subset of the data.
- ggplot(data = mpg) +

 geom_point(mapping = aes(x = displ, y = hwy)) +

facet_wrap(~ class, nrow = 2)

Data visualization

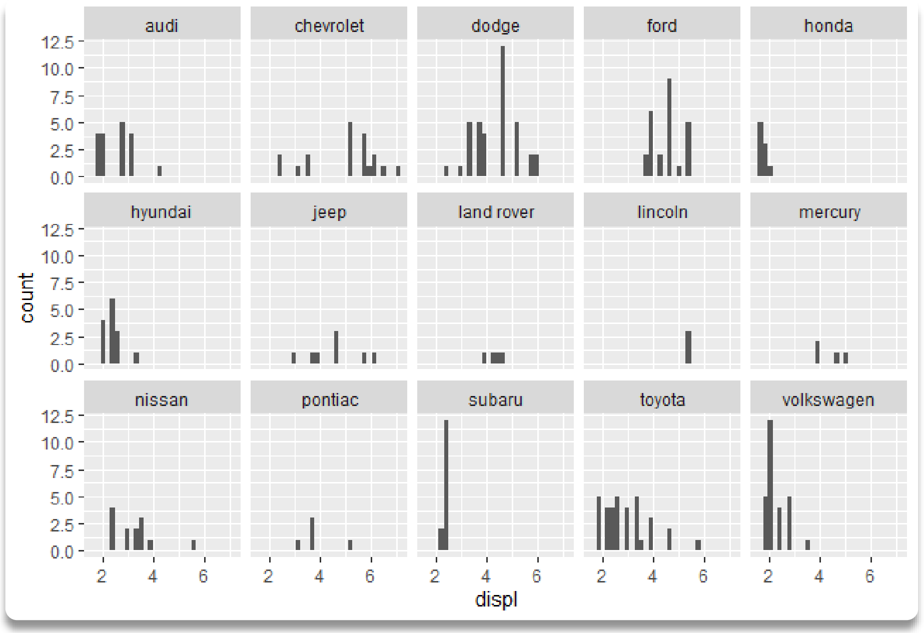


- ggplot(data = mpg) +

  geom_point(mapping = aes(x = displ, y = hwy)) +

  facet_wrap(~ manufacturer, nrow = 3)

Data visualization



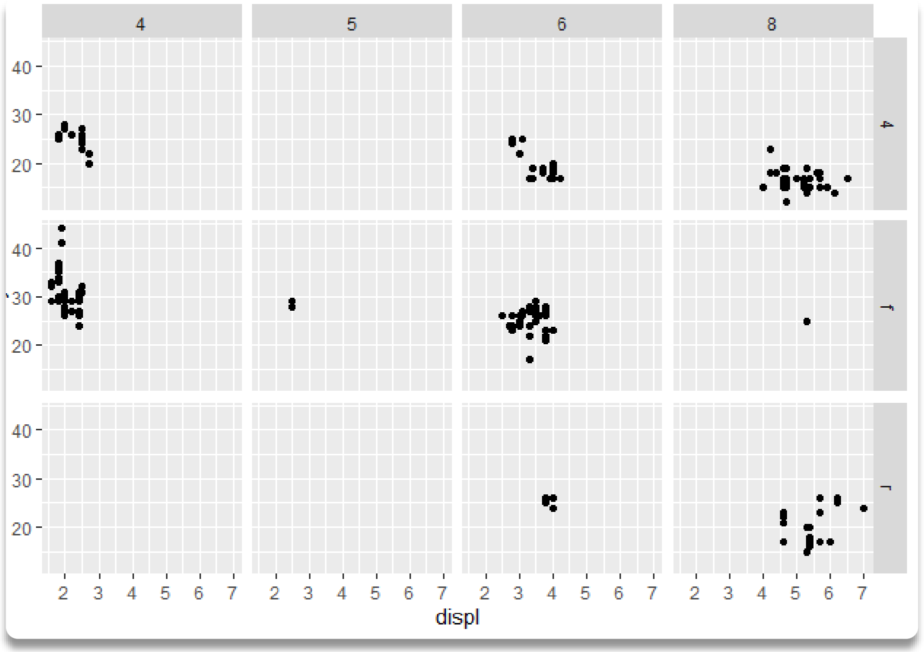
- You can create separate histogram for each manufacturer in a single plot with facet_wrap().

- ggplot(data = mpg) +

  geom_histogram(mapping = aes(x = displ)) +

  facet_wrap(~ manufacturer, nrow = 3)

Data visualization



- # To facet your plot on the combination of two variables, add facet_grid() to your plot call. The first argument of facet_grid() is also a formula. This time the formula should contain two variable names separated by a ~.

- ggplot(data = mpg) +

geom_point(mapping = aes(x = displ, y = hwy)) +

facet_grid(drv ~ cyl)

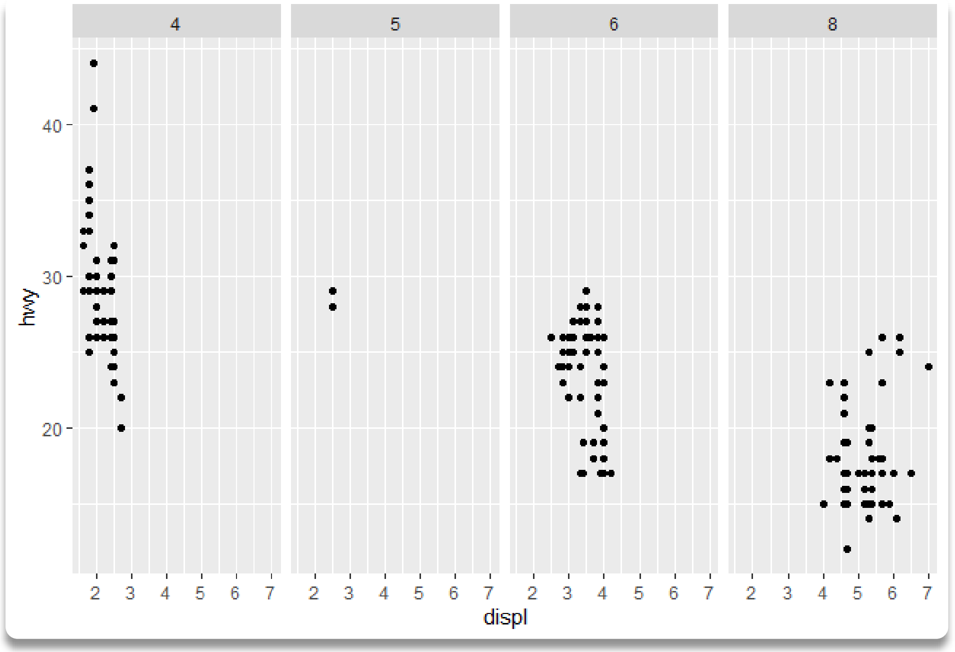
Data visualization

- #If you prefer to not facet in the rows or columns dimension, use a . instead of a variable name, e.g. + facet_grid(. ~ cyl).

- ggplot(data = mpg) +

geom_point(mapping = aes(x = displ, y = hwy)) +

facet_grid(. ~ cyl)

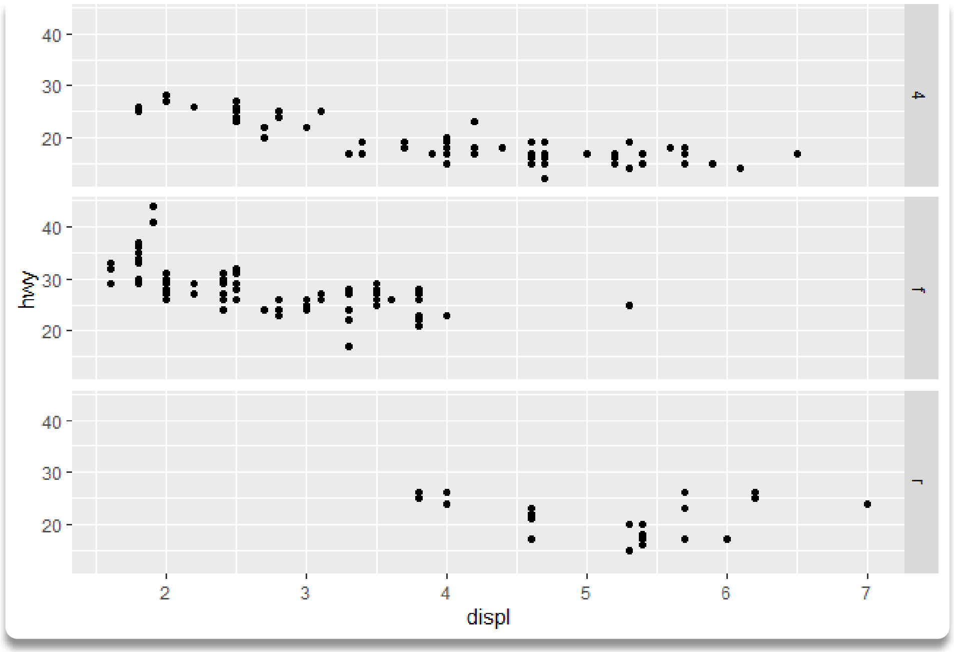


Data visualization

- ggplot(data = mpg) +

  geom_point(mapping = aes(x = displ, y = hwy)) +

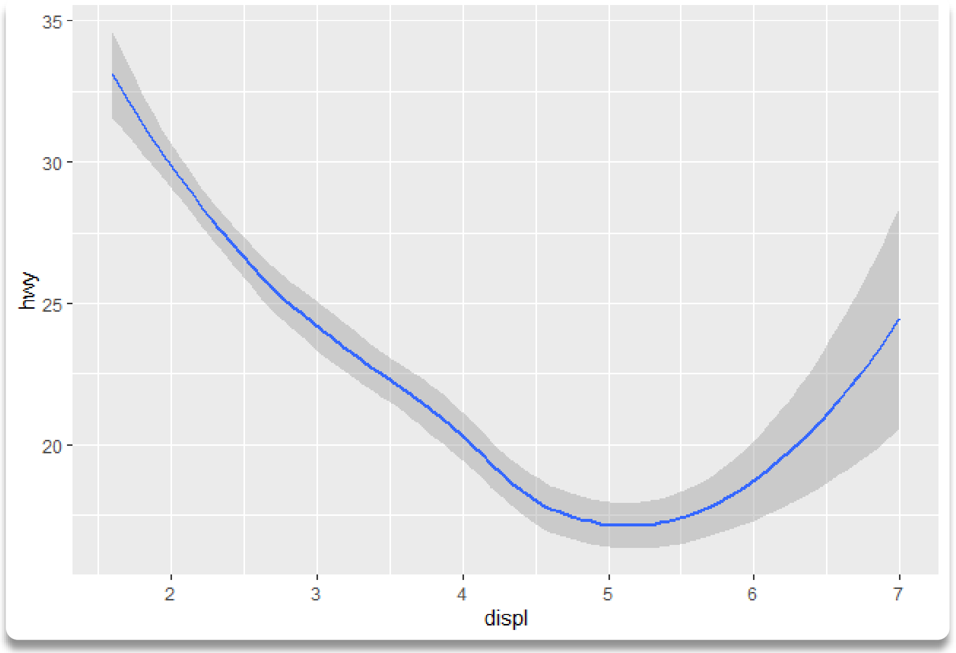
  facet_grid(drv ~ .)



Data visualization

- ggplot(data = mpg) +

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

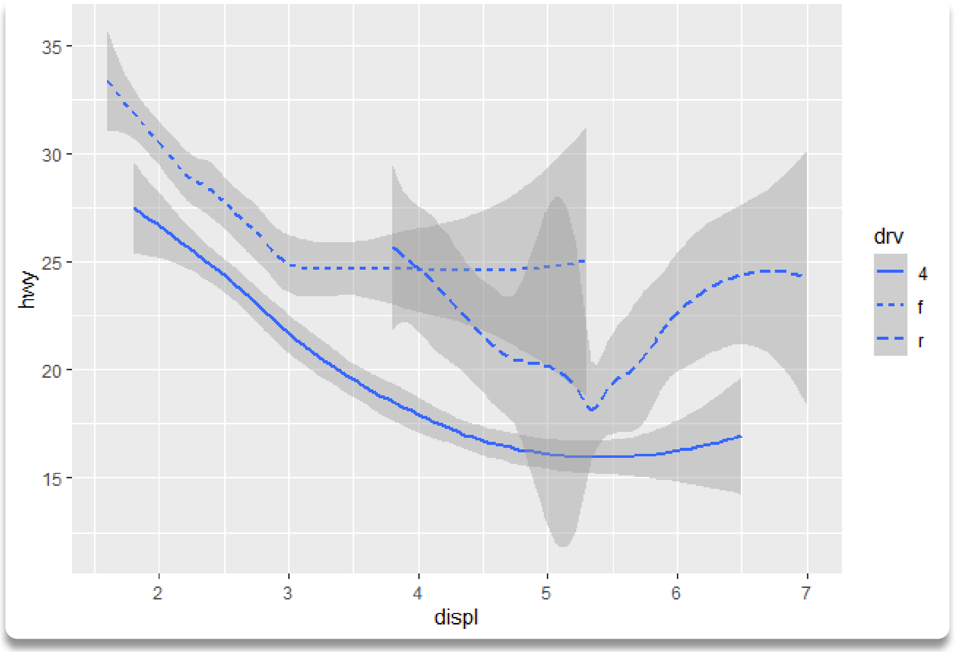


Data visualization

- It is possible to draw different lines with geom_smooth(), with a different linetype, for each unique value of the variable that you map to linetype.

- ggplot(data = mpg) +

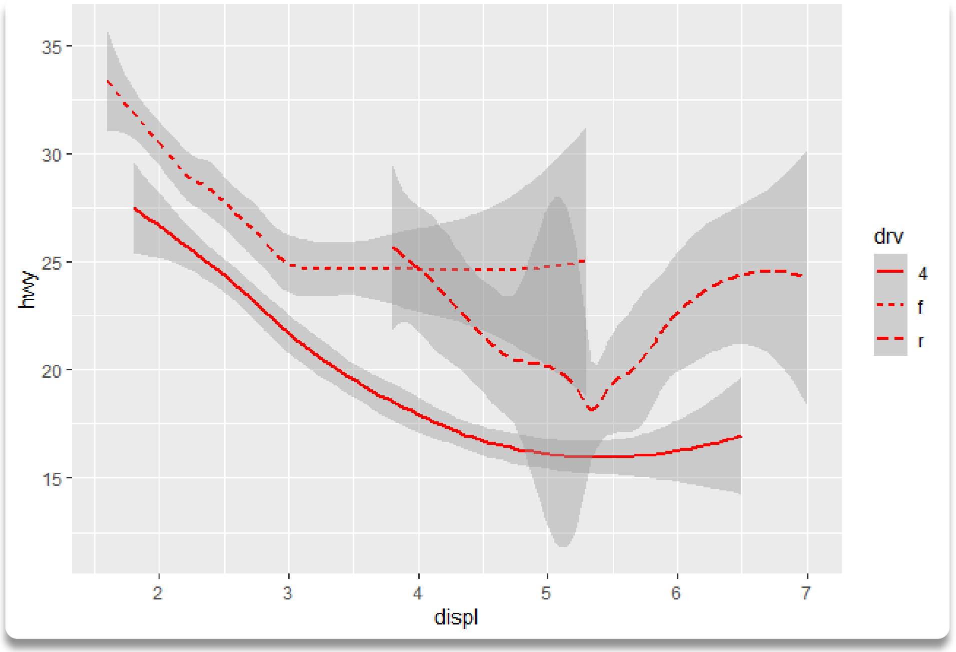
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))



Data visualization

- ggplot(data = mpg) +

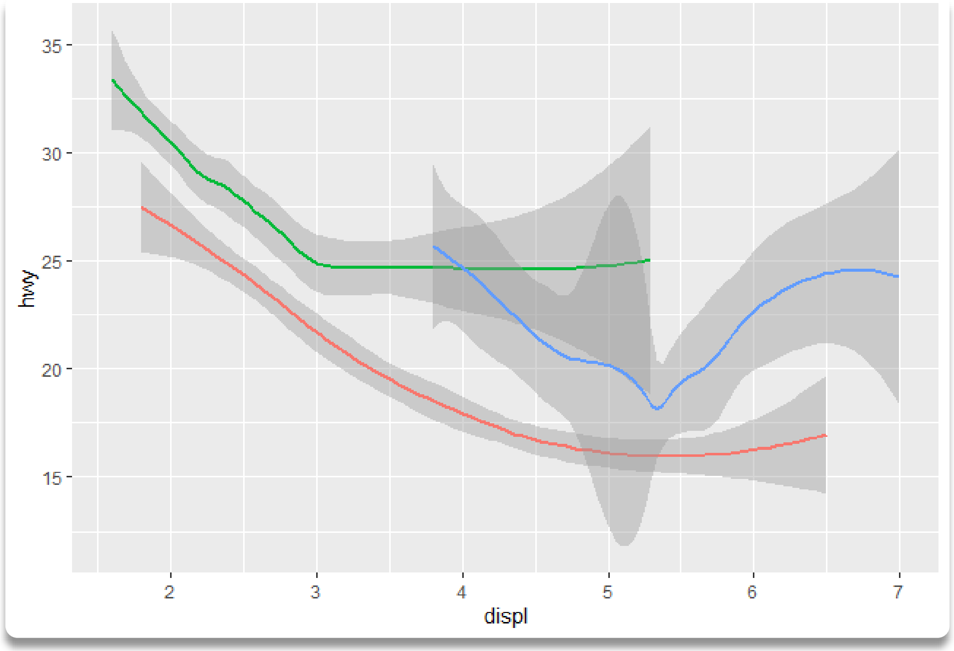
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv),color = "red")



Data visualization

- If you want to change the color for each category:
- ggplot(data = mpg) +

```
geom_smooth(
  mapping = aes(x = displ, y = hwy, color = drv),
  show.legend = FALSE
)
```



Data visualization

- Now we will do the same plot for different manufacturer.
- ggplot(data = mpg) +

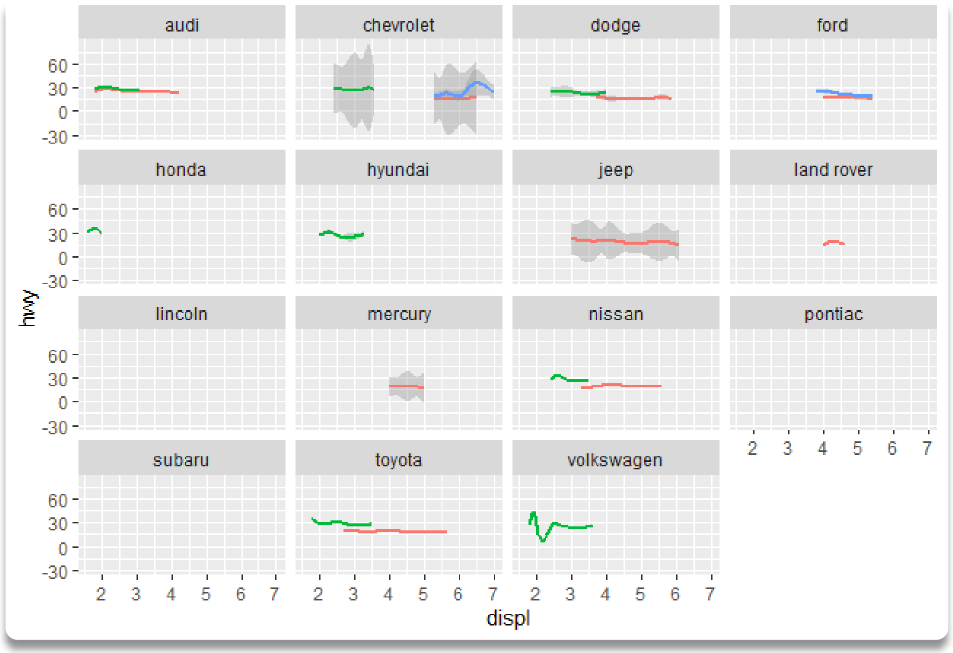
geom_smooth(

mapping = aes(x = displ, y = hwy, color = drv),

show.legend = FALSE

)+

facet_wrap(~manufacturer)

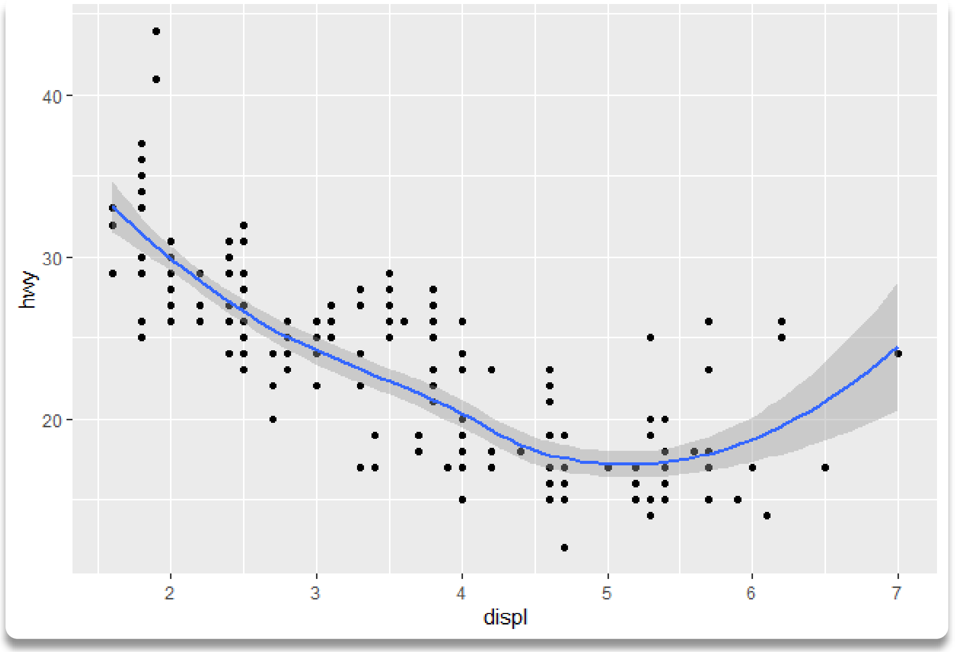


Data visualization

- To display multiple geoms in the same plot, add multiple geom functions to ggplot():
- ggplot(data = mpg) +

  geom_point(mapping = aes(x = displ, y = hwy)) +

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

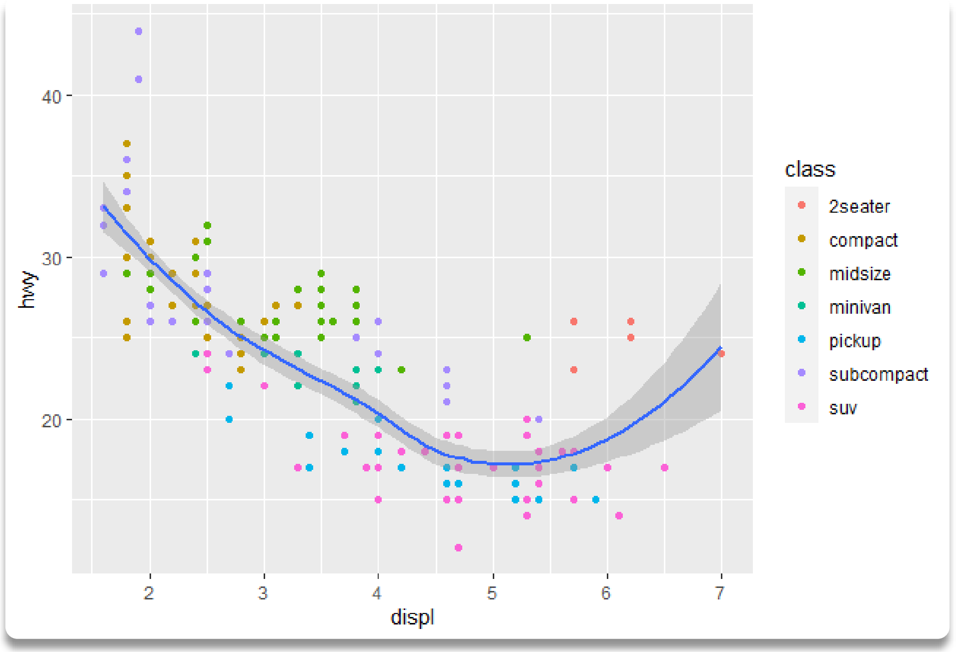


Data visualization

- If you place mappings in a geom function, ggplot2 will treat them as local mappings for the layer.

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

```
  geom_point(mapping = aes(color = class)) +
  geom_smooth()
```



Data visualization

- Bar chart:

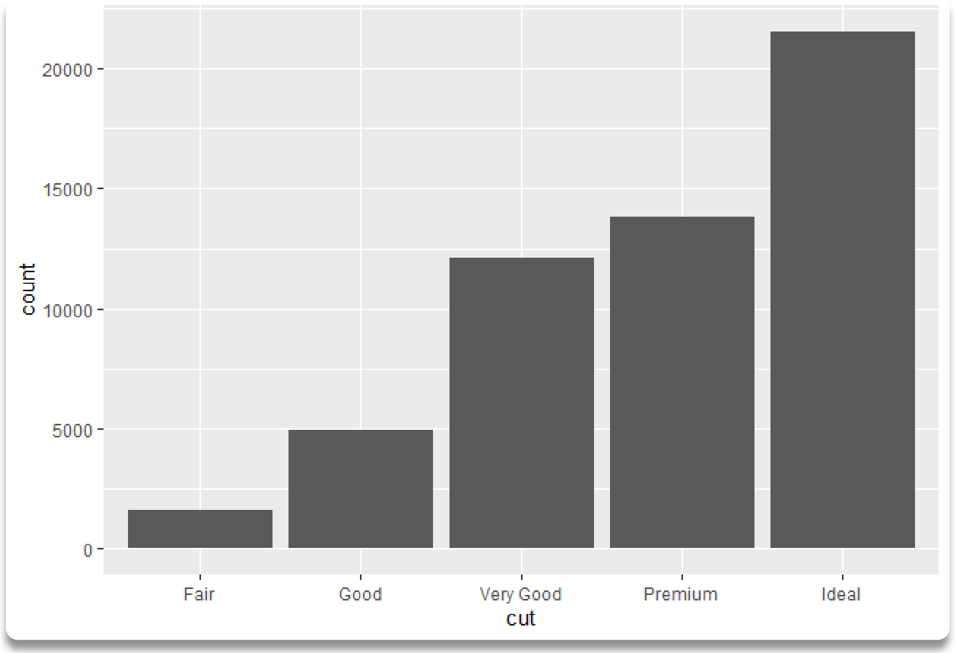
- The following chart displays the total number of diamonds in the diamonds dataset, grouped by cut

Data visualization

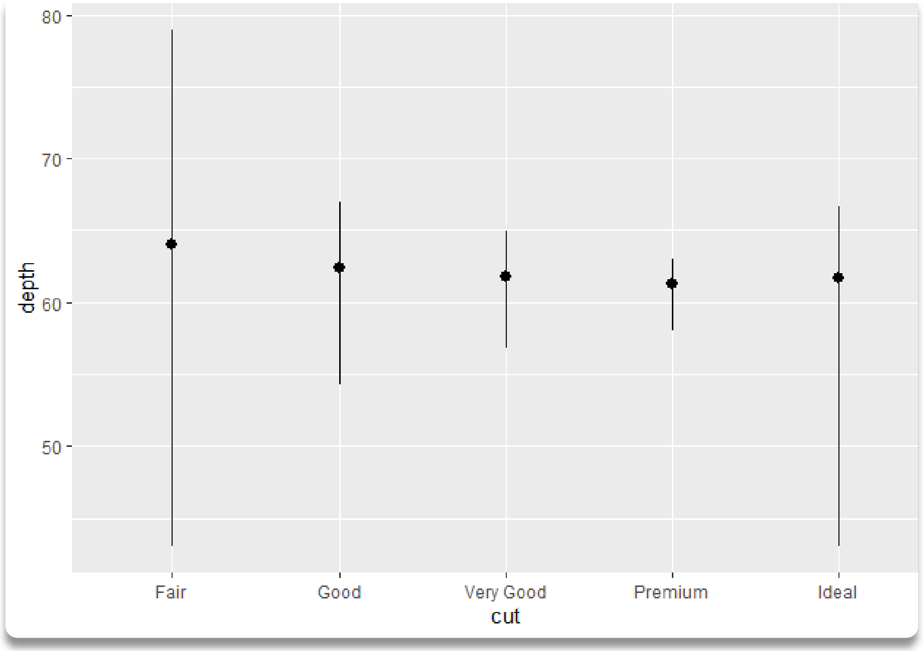
- ggplot(data = diamonds) +

  geom_bar(mapping = aes(x = cut))

- The chart shows more diamonds are available with higher quality cuts.



Data visualization

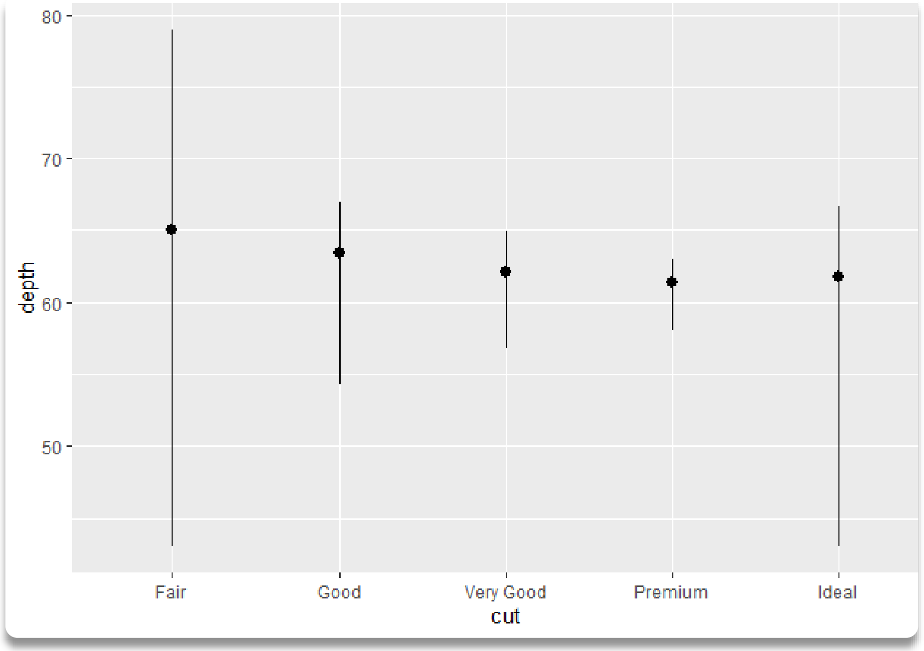


- You might want to draw greater attention to the statistical transformation in your code. For example, you might use stat_summary(), which summarises the y values for each unique x value,

- ggplot(data = diamonds) +

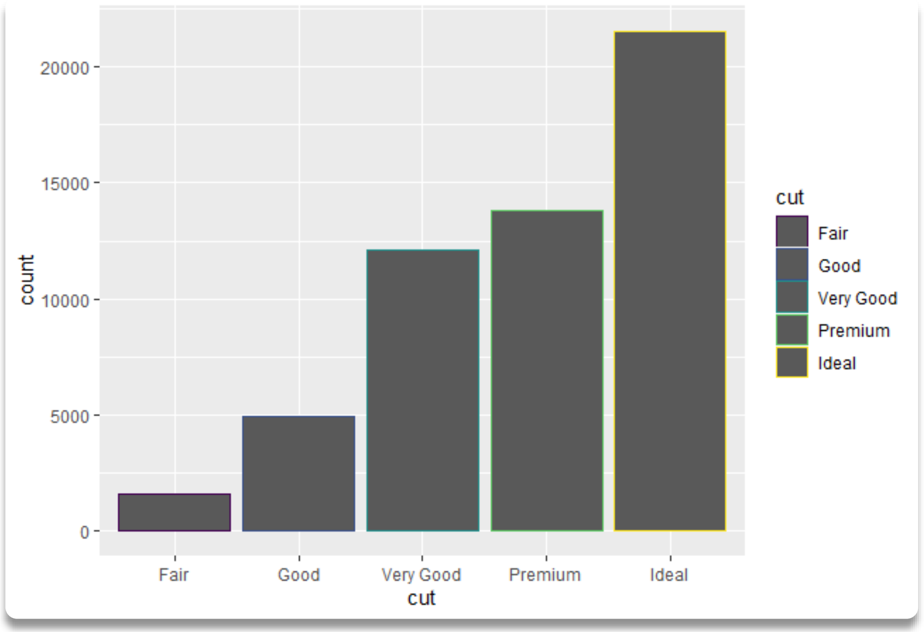
```
stat_summary(
  mapping = aes(x = cut, y = depth),
  fun.min = min,
  fun.max = max,
  fun = mean
)
```

Data visualization



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

Data visualization

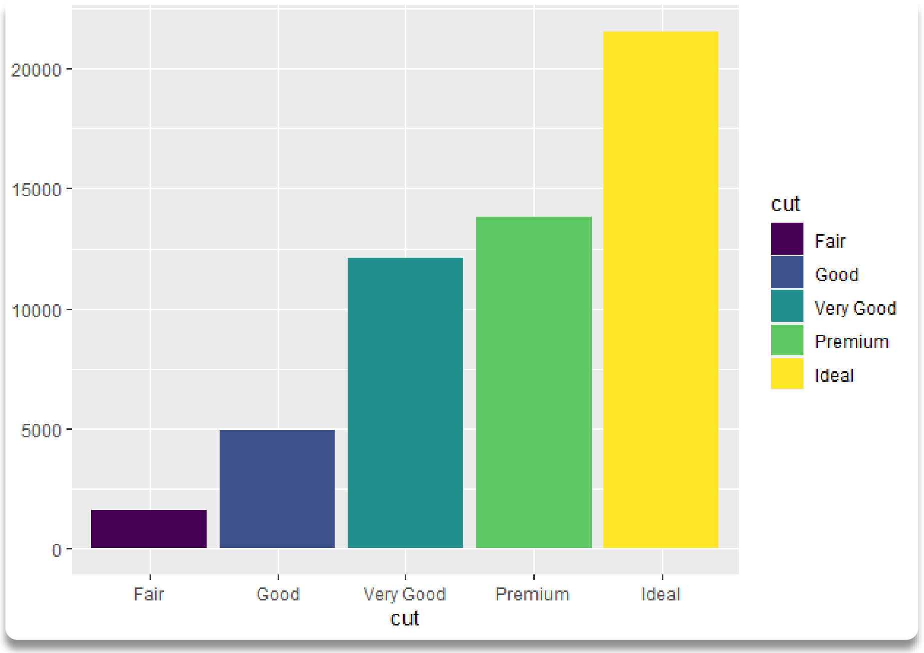


- You can colour a bar chart using either the colour aesthetic, or, more usefully, fill:

- ggplot(data = diamonds) +

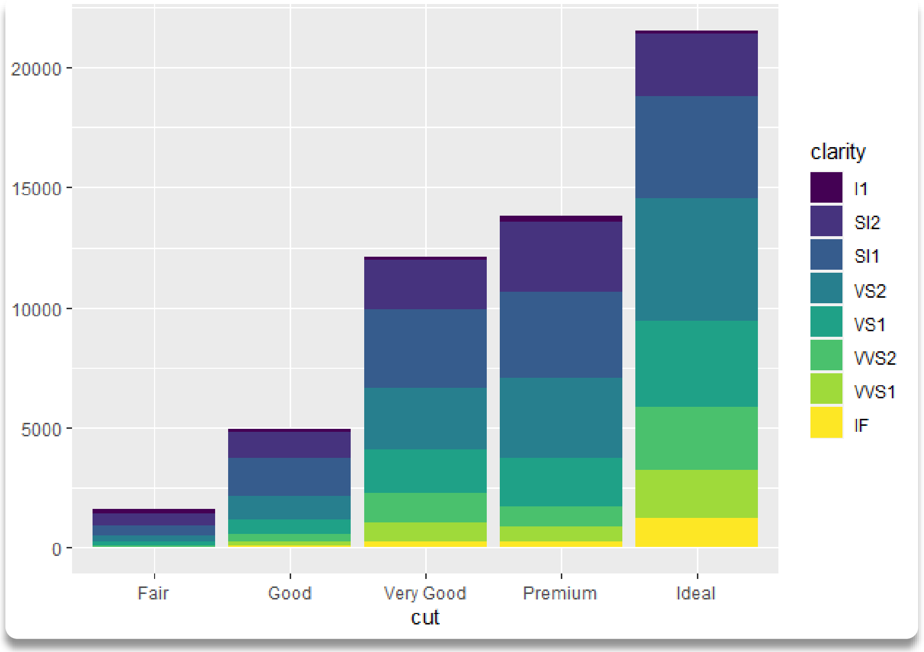
 geom_bar(mapping = aes(x = cut, colour = cut))

Data visualization



- ggplot(data = diamonds) + geom_bar(mapping = aes(x = cut, fill = cut))

Data visualization

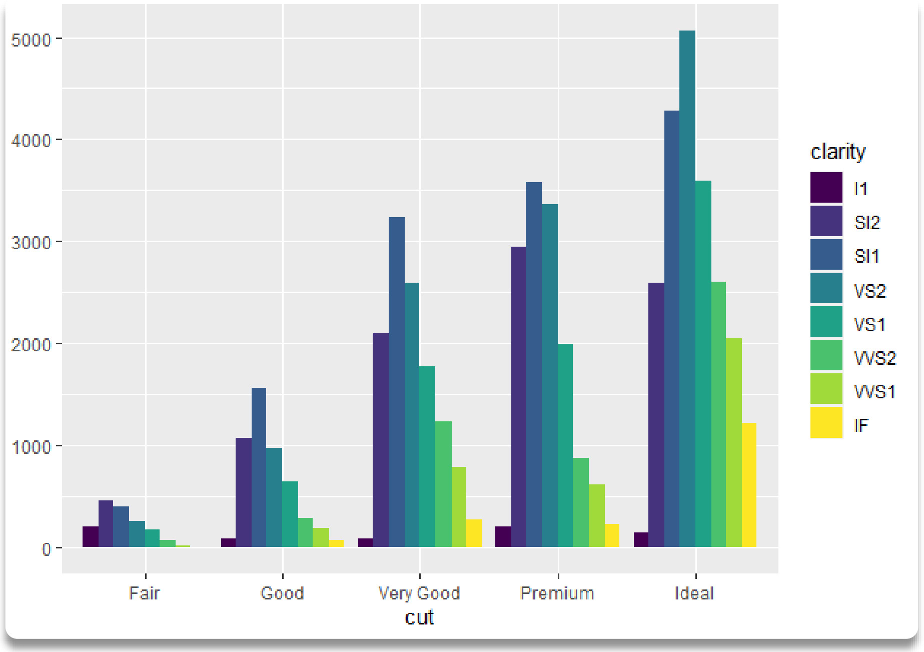


- what happens if you map the fill aesthetic to another variable, like clarity: the bars are automatically stacked. Each colored rectangle represents a combination of cut and clarity.

- ggplot(data = diamonds) +

  geom_bar(mapping = aes(x = cut, fill = clarity))

Data visualization



- position = "dodge" places overlapping objects directly beside one another. This makes it easier to compare individual values.
- ggplot(data = diamonds) +

geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")