ST537-G6-ggplot2-Week 2

Group6

4/12/2020

Contents

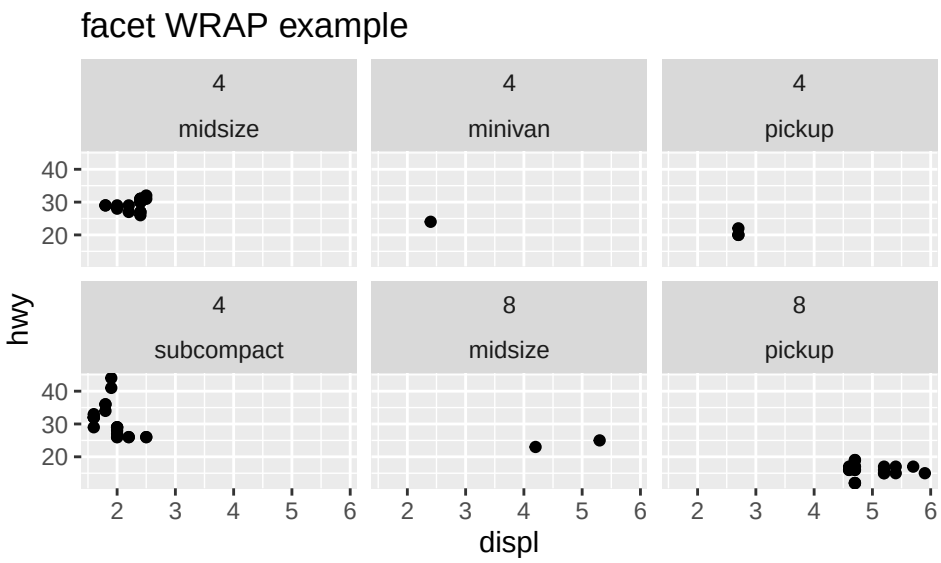
Section 3.5 - Julie Sampson
Section 3.6 - Josh Olsen
Section 3.7 - Jake Oetinger
Section 3.8 - Katie Tallan
Section 3.9 - Susan McNerney
Section 3.10 - David Cobertera
Compilation - Teresa Burlingame

3.5 Facets

For the facetting questions I subsetted the mpg data to a smaller set to make the plots simpler.

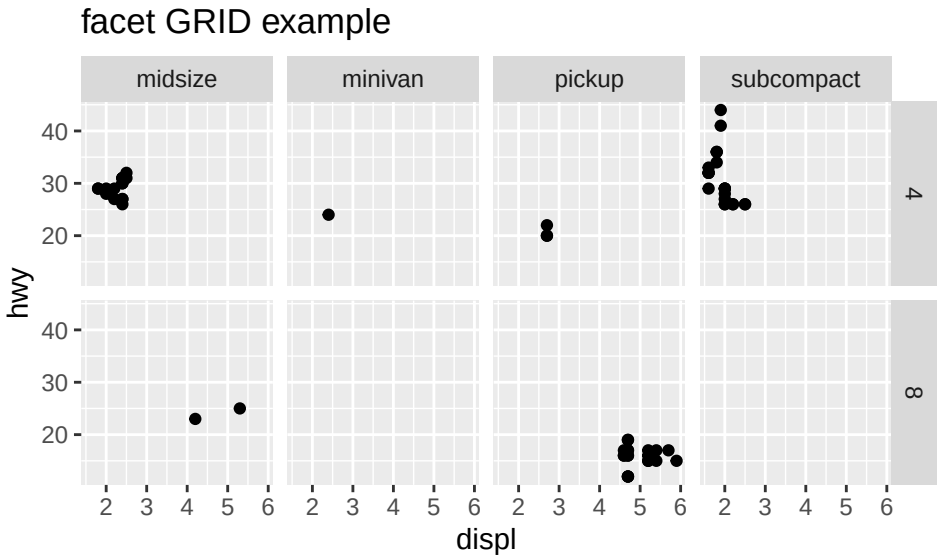
The following 2 plots are the same code except one call is to facet_grid and one is to facet_wrap. Facet_wrap lays out a 1D set of plots that slices the main x vs. y plot by the variables specified in the facet call. In this example, the main plot is displ vs. hwy and it is scoped to cyl on the vertical and class on the horizontal.

```
# use smaller data set for facets example
mpg_ex <- subset(mpg, cyl != 5 & cyl!=6 & drv %in% c("4", "f")
                 & class != "2seater" & class!="compact" & class!= "suv")

 ggplot(data = mpg_ex) +
    geom_point(mapping = aes(x = displ, y = hwy)) +
    facet_wrap(cyl~class)  +
  labs(title="facet WRAP example")
```



Facet_grid lays out a 2D set of plots as a matrix. It displays all combinations of the levels that are specified in the call to facet_grid(). It will display a blank plot if there are no observations for the combination of levels. Facet_wrap() will not display empty plots by default. You can display the empty plots by specifying drop=FALSE in method call.

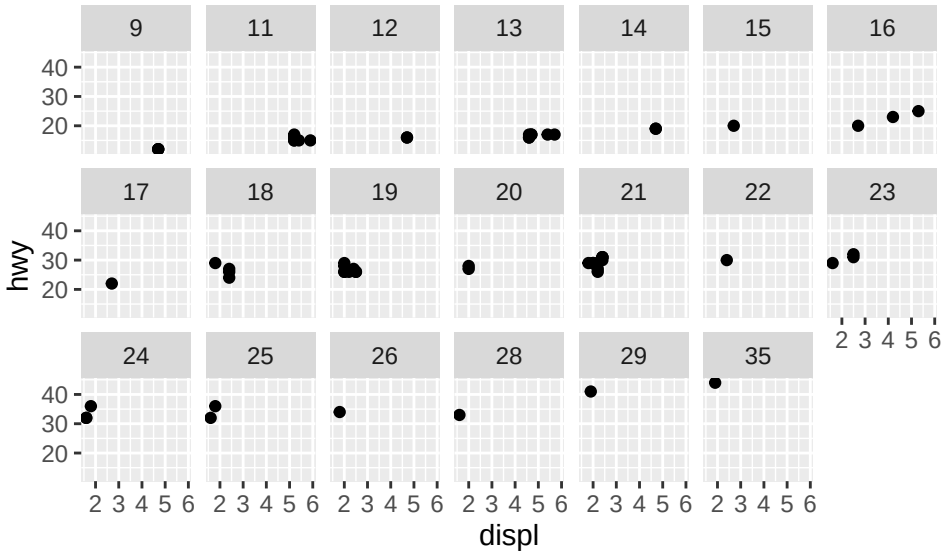
```
ggplot(data = mpg_ex) +
   geom_point(mapping = aes(x = displ, y = hwy)) +
   facet_grid(cyl~class) +
 labs(title="facet GRID example")
```



3.5.1 Questions

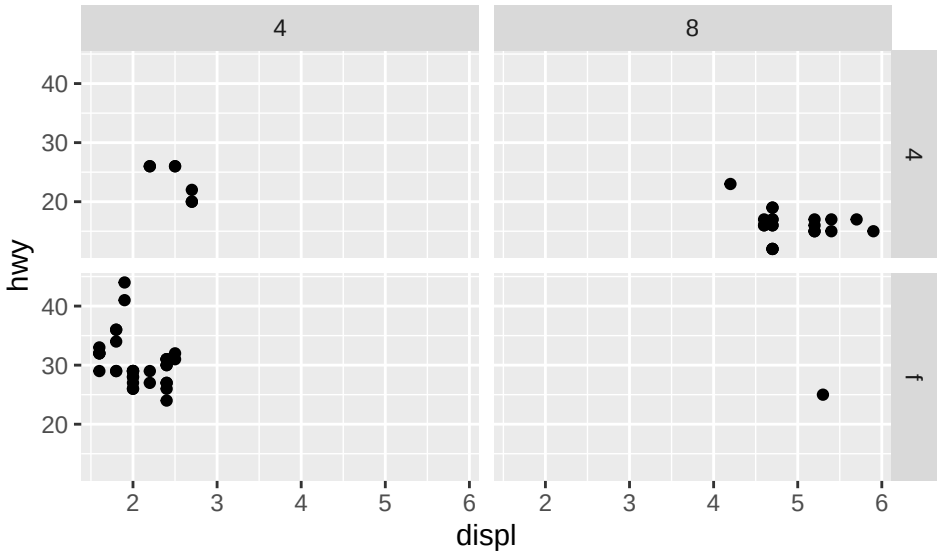
1.Faceting on a continuous variable will give you a plot for each value. There's no categorization or grouping done.

```
ggplot(data = mpg_ex) +
 geom_point(mapping = aes(x = displ, y = hwy)) +
 facet_wrap(. ~ cty, nrow=3)
```



2.What do empty cells in this plot mean?

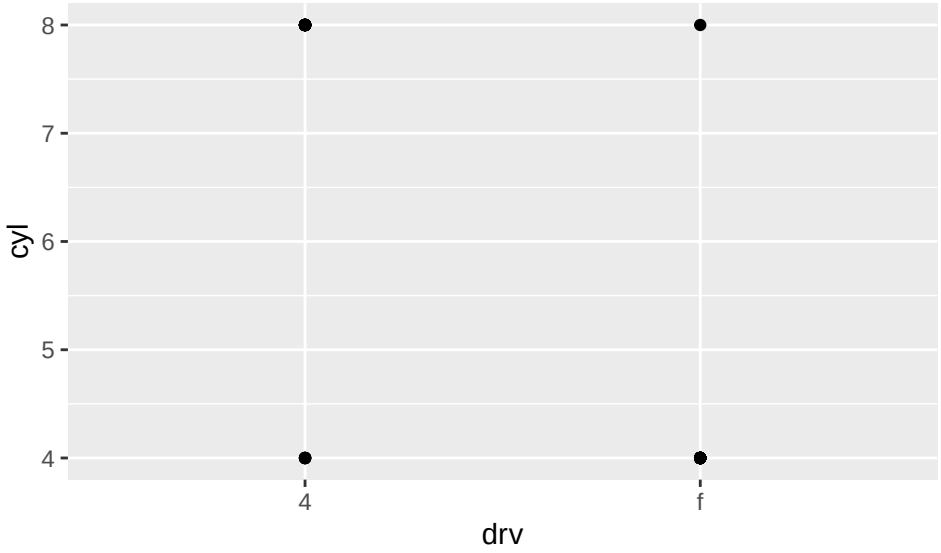
```
ggplot(data = mpg_ex) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
   facet_grid(drv~cyl)
```



There are no observations for those levels of the variables. For example, when drv = r and cyl = 5; a rear wheel drive with 5 cylinders is not in the data.

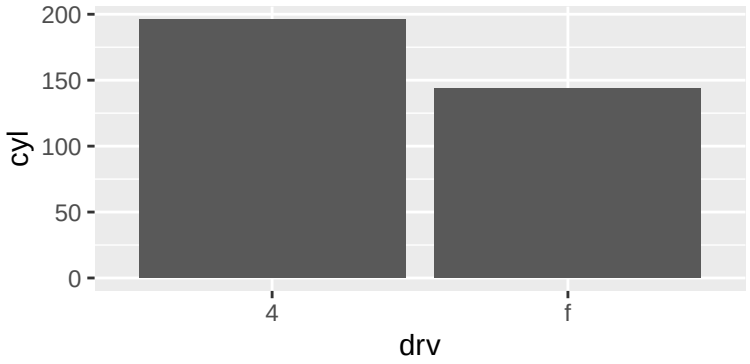
How does that relate to this plot:

```
ggplot(data = mpg_ex) +
  geom_point(mapping = aes(x = drv, y = cyl))
```



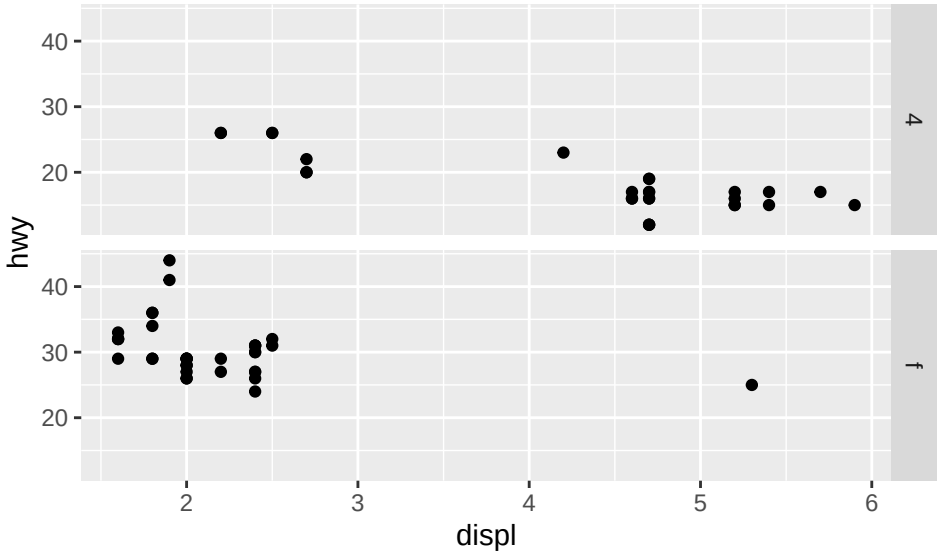
The facet displays the hwy miles per liter grouped into plots by drive and number of cylinders The single plot is just where drv and cyl intersect, it would be easier to see in a histogram format. Discovered geom_col. This is cool and useful!!

```
ggplot(data = mpg_ex) +
  geom_col(mapping = aes(x = drv, y=cyl))
```

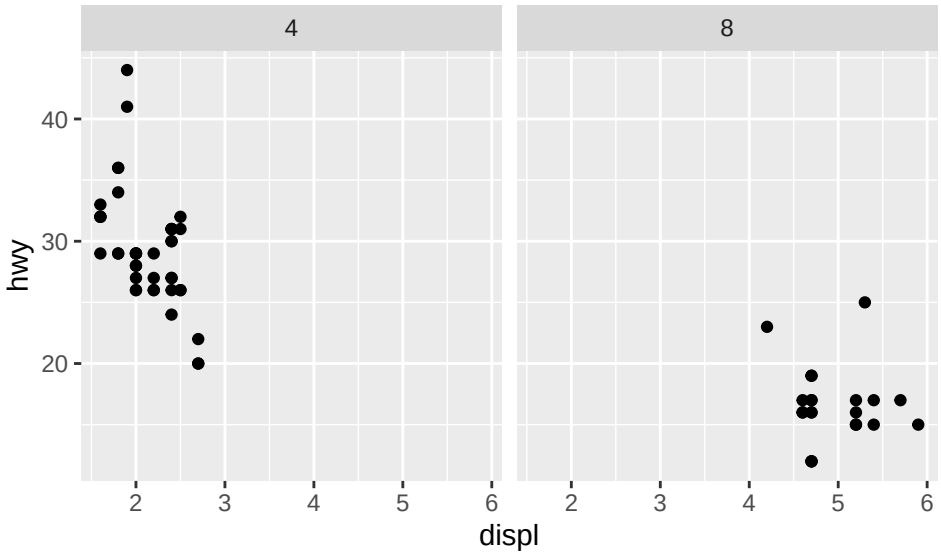


3. What does the "." in each of these plots do:

```
ggplot(data = mpg_ex) +
   geom_point(mapping = aes(x = displ, y = hwy)) +
   facet_grid(drv ~ .)
```



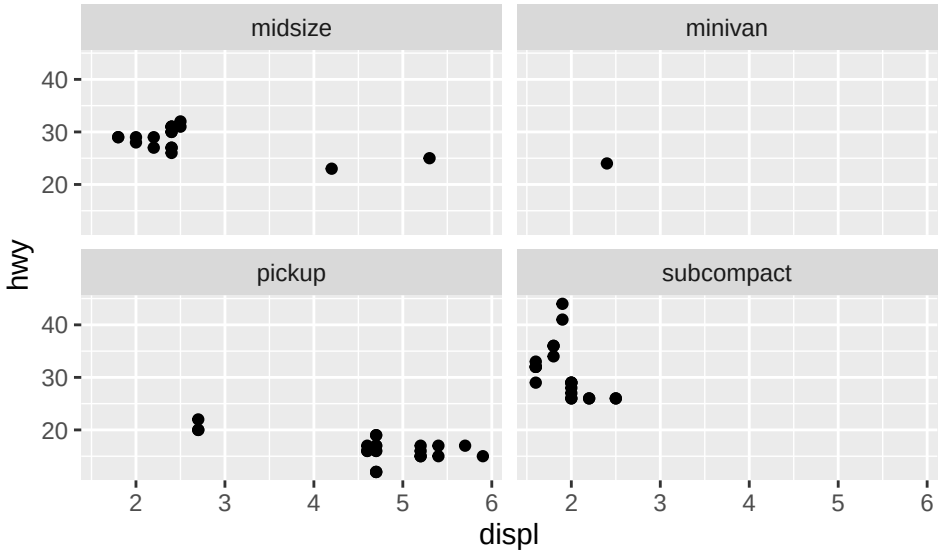

```
ggplot(data = mpg_ex) +
   geom_point(mapping = aes(x = displ, y = hwy)) +
   facet_grid(. ~ cyl)
```



The "." tells ggplot to omit that dimension (x,y), so the first plot omits the vertical and the second omits the horizontal

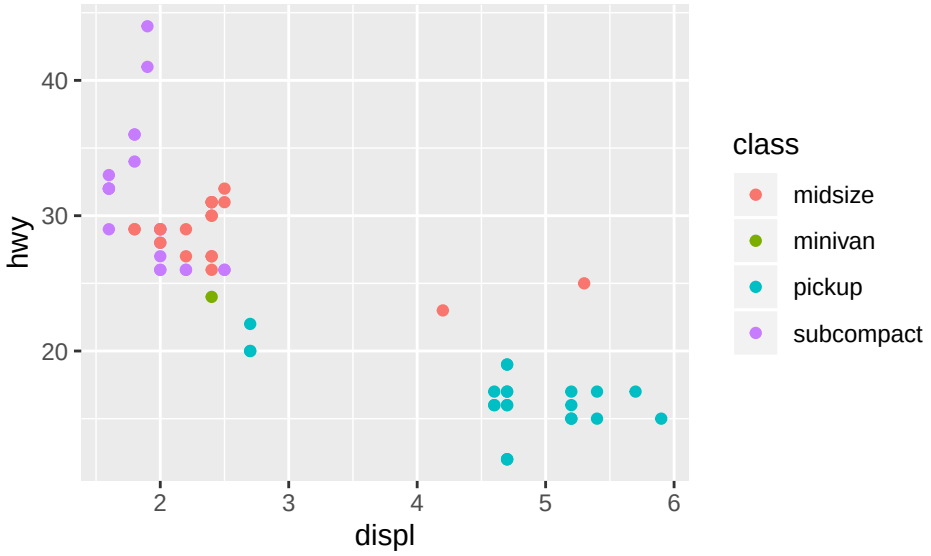
4. Consider the plot:

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



compare to using color for class

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



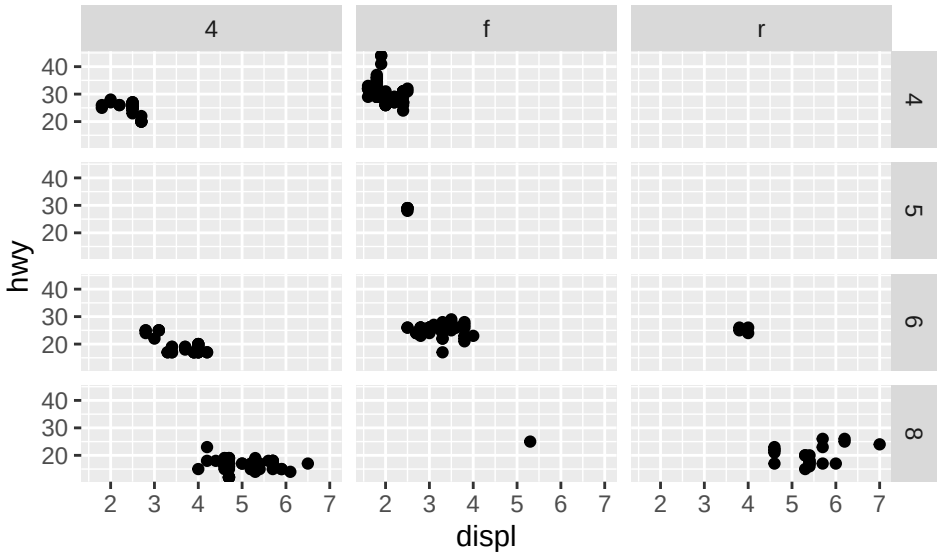
I think it's easier to look at this data set with class as a color then as a facet wrap or facet grid because you can easily see how different classes of cars compare to each other with respect to the whole set of cars (overview)

5. Read ?facet_wrap() nrow is number of rows ncol is number of columns why doesn't facet_grid() have nrow or ncol? because the number of rows and cols is determined by the number of levels in the variables you pass to facet_grid determines the horizontal and vertical dimensions of the the plots

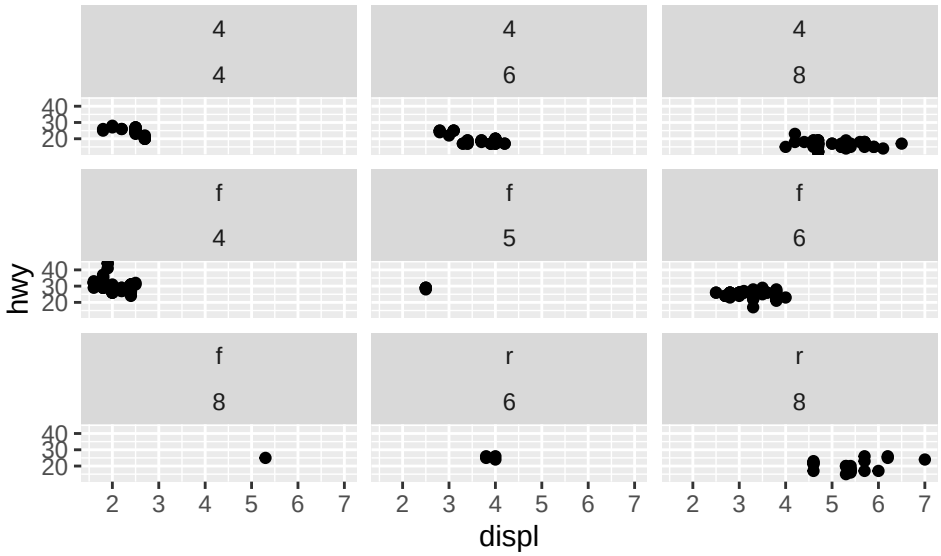
What are other options to control layout

scales - the default is to share scales across all facets - "fixed" is you want to vary the scales based on data then specify "free_x", "free_y", or both "free"

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



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap( drv~cyl, scales = "fixed")
```



shrink - TRUE by default, shrink scales to fit statistical output, if false it will use raw data **space** - default is for all panels to have same size "fixed" which is the scale of he y variable. Can change it to "free_x" or "free_y" **labeller** - complicated different behavior based on type of object you pass to it. See docs. **drop** - don't display facets that have no data (only valid with facet_wrap()) **switch** - changed position that labels are on

6.Why should you put the variable with more unique levels in the columns spot when using facet_grid()?

To avoid the y-axis shrinking so small that it becomes hard to see the plotted values.

Here are some good references for facets:

https://ggplot2-book.org/facet.html
https://ggplot2.tidyverse.org/reference/facet_grid.html
https://ggplot2.tidyverse.org/reference/facet_wrap.html

3.6 Geometric Objects

3.6.1 Exercises

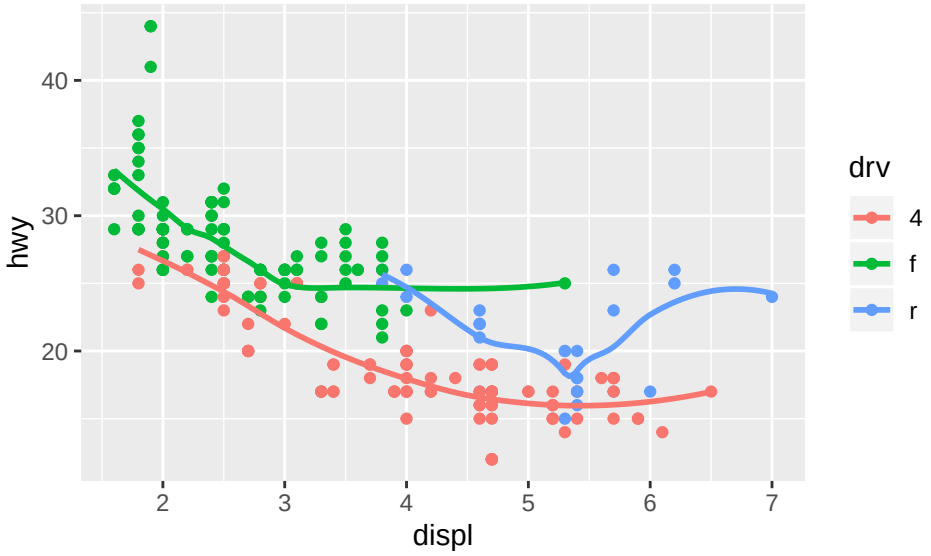
1. What geom would you use to draw a line chart? A boxplot? A histogram? An area chart?

geom_line for Line Chart, geom_boxplot for boxplot, geom_histogram for historgram, and geom_area for area chart.

2. Run this code in your head and predict what the output will look like. Then, run the code in R and check your predictions.

Looks like this code will add each point for displ and hwy variables grouped by drv variable. Also, the should see the mean.

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



The code adds each point for displ and hwy variables grouped by drv variable by color. Also, the mean grouped by drv by color. Pretty close to our prediction.

3. What does show.legend = FALSE do? What happens if you remove it? Why do you think I used it earlier in the chapter?

The show.legend = FALSE means R won't show the legend of the smooth fuction. If you remove it, there is only one legend based on the point function. It was used earlier in the charter because it shows the two different options with two different functions.

4. What does the se argument to geom_smooth() do?

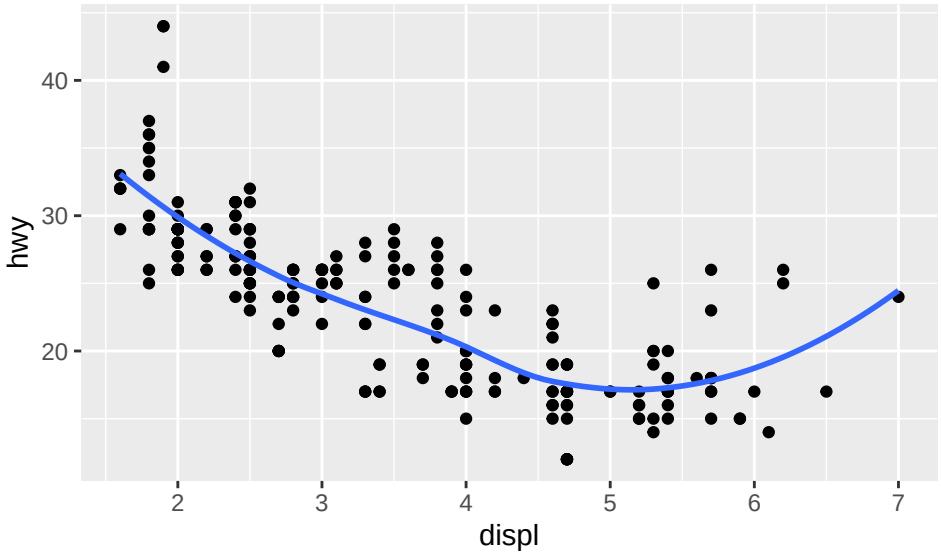
The se argument to geom_smooth displays confidence interval around smooth.

5. Will these two graphs look different? Why/why not?

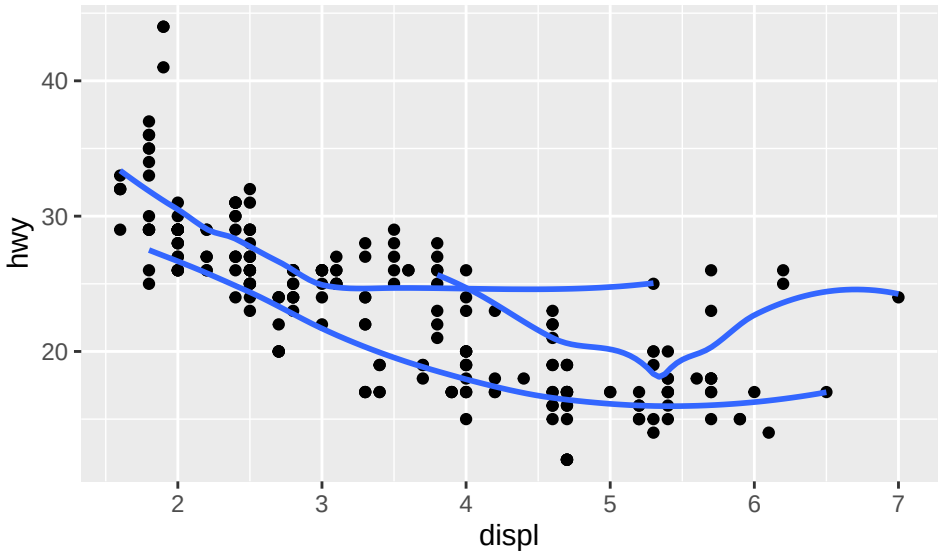
These two graphs will look the same because the mapping is the same in geom_point and geom_smooth.

6. Recreate the R code necessary to generate the following graphs.

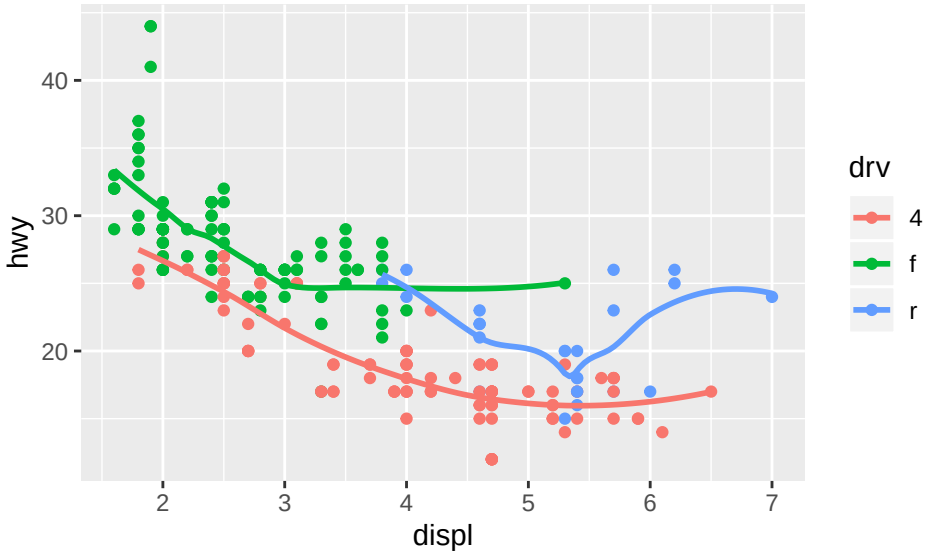
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



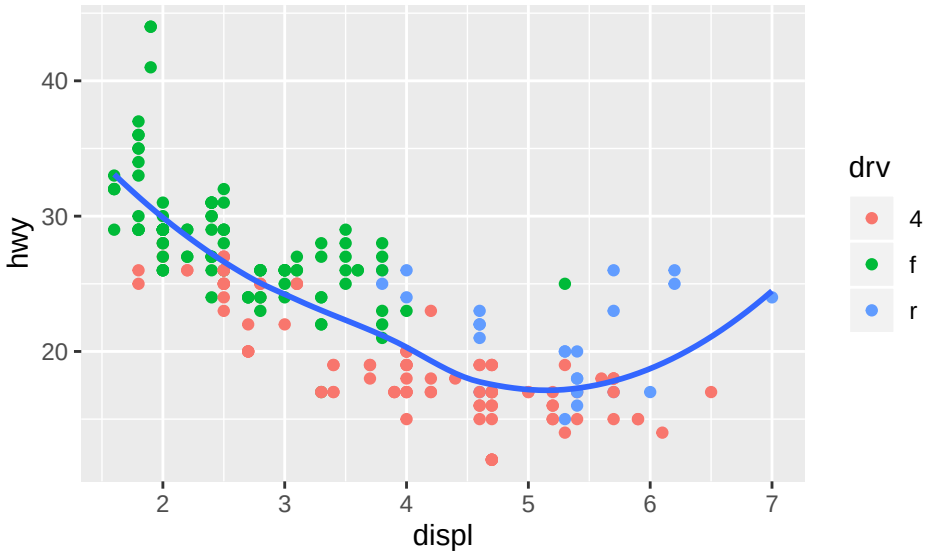

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



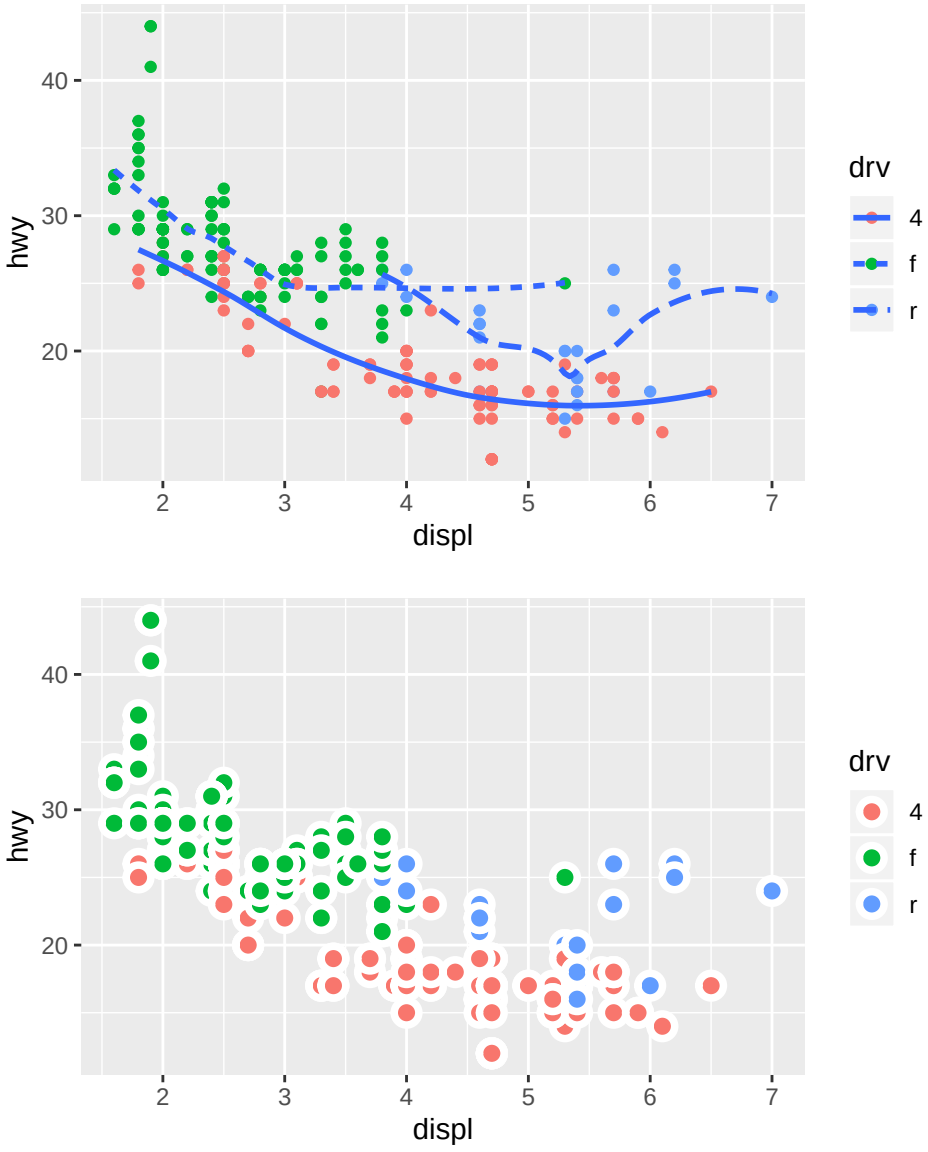

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




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




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



3.7 Statistical transformations

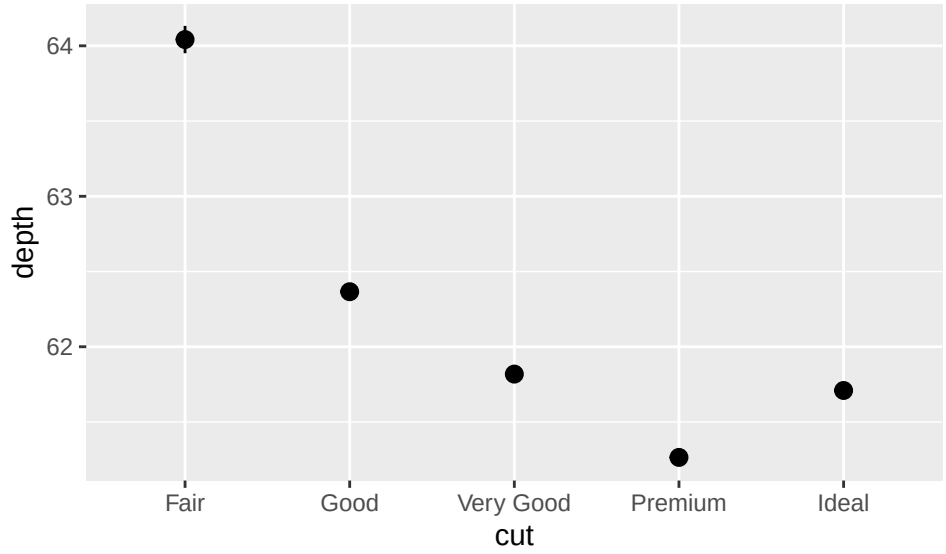
Question 1. What is the default geom associated with `stat_summary()`? How could you rewrite the previous plot to use that geom function instead of the stat function?

First, the original plot.

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

```
## Warning: Ignoring unknown parameters: fun.min, fun.max, fun
```

```
## No summary function supplied, defaulting to `mean_se()
```



The default geom associated with `stat_summary()` is `geom_pointrange()`. To recreate the previous plot using `geom_pointrange()` we can summarize the relevant information before plotting.

```
# d <- diamonds %>%
#   group_by(cut) %>%
#    summarise(min = min(depth), max = max(depth), median = median(depth))
#
# ggplot(data = d) +
#    geom_pointrange(aes(x =cut, y = median, ymin = min, ymax = max))
```

2.What does `geom_col()` do? How is it different to `geom_bar()`?

`geom_col` differs from `geom_bar` in the default stat transformation that it uses. `geom_bar` uses `stat_count` by default, so you can create a bar chart without grouping observations and finding the total number in each group, `stat_count` will do it for you. If, for some reason, you have data which are already grouped so that the total number in each group (the height of each bar) is a column in the data frame, you can use `geom_col` since its default stat transformation is `stat_identity`, which leaves data as is.

3.Most geoms and stats come in pairs that are almost always used in concert. Read through the documentation and make a list of all the pairs. What do they have in common?

geom	stat
`geom_bar()`	`stat_count()`
`geom_bin2d()`	`stat_bin_2d()`
`geom_boxplot()`	`stat_boxplot()`
`geom_contour()`	`stat_contour()`
`geom_count()`	`stat_sum()`
`geom_density()`	`stat_density()`
`geom_density_2d()`	`stat_density_2d()`
`geom_hex()`	`stat_hex()`
`geom_freqpoly()`	`stat_bin()`
`geom_histogram()`	`stat_bin()`
`geom_qq_line()`	`stat_qq_line()`

geom	stat
`geom_qq()`	`stat_qq()`
`geom_quantile()`	`stat_quantile()`
`geom_smooth()`	`stat_smooth()`
`geom_violin()`	`stat_violin()`
`geom_sf()`	`stat_sf()`

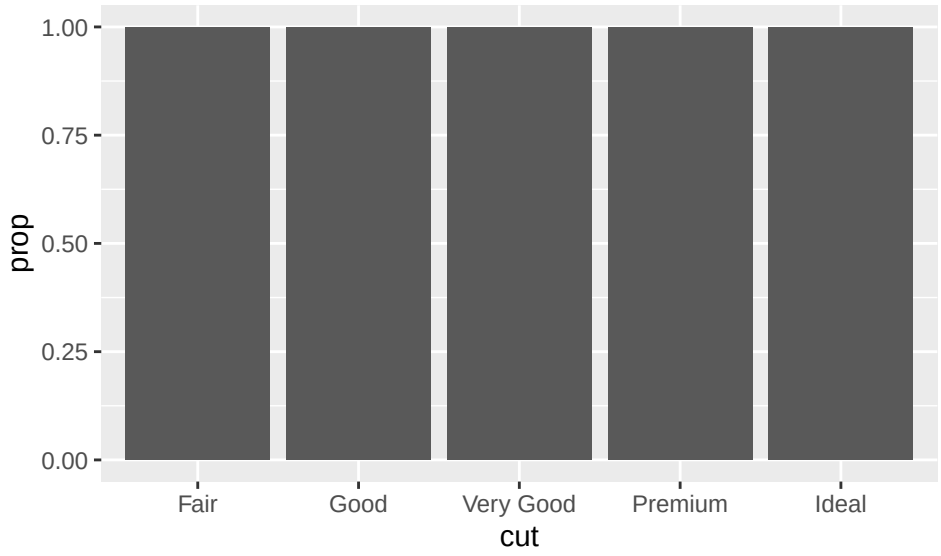
These geom and stat pairs are almost always used together. Generally, corresponding geom-stat pairs will have matching names, although there are a few exceptions.

4.What variables does `stat_smooth()` compute? What parameters control its behaviour?

`stat_smooth` calculates a predicted y value, upper and lower bounds of a confidence interval, and standard error. The two most important arguments for controlling the behavior of stat_smooth are `method` and `formula`. The former establishes which statistical method you would like to use to find predictions (e.g. `lm`, `loess`, or `gam`), and the latter specifies the formula.

5. In our proportion bar chart, we need to set group = 1. Why? In other words what is the problem with these two graphs?

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop..))
```

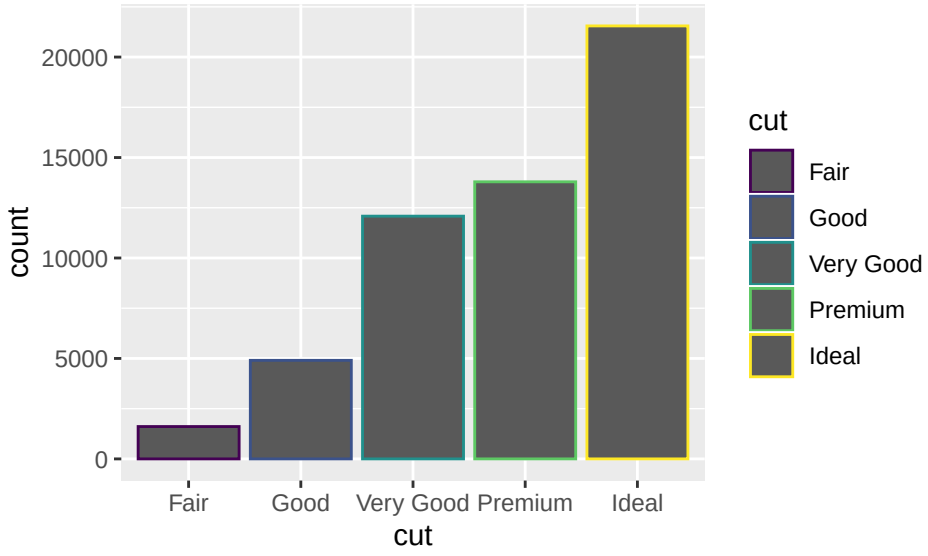


By default, `geom_bar()` groups according to the grouping variable assigned to `x` in the aesthetic mapping. So when you try to calculate proportions, it will consider each group as a whole. This gives every group a proportion of 1. To specify that you want proportions calculated using the grand total, you need to consider all the groups as one big group by setting `group = 1`.

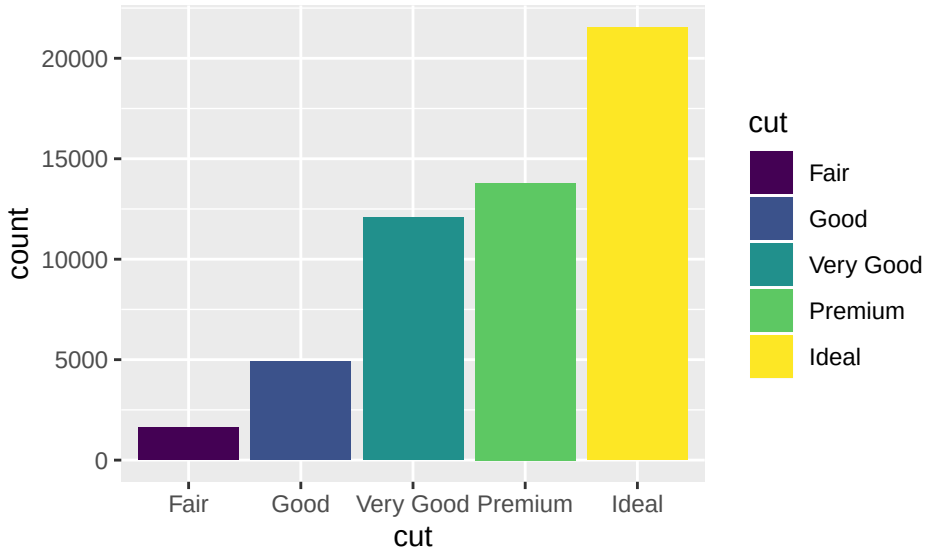
3.8 Position Adjustments

You can color a bar chart using either the colour or fill aesthetic:

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

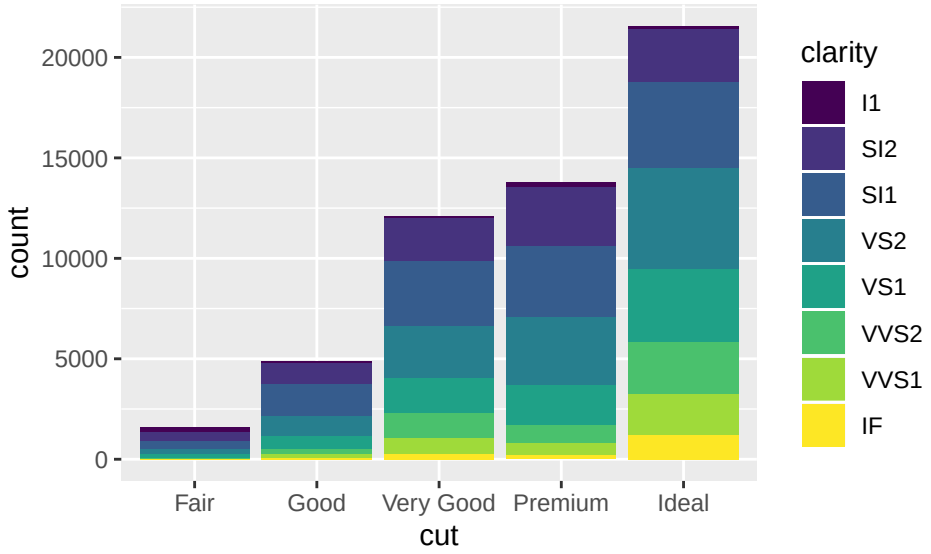



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = cut))
```



Mapping the fill aesthetic to a different variable stacks the bars:

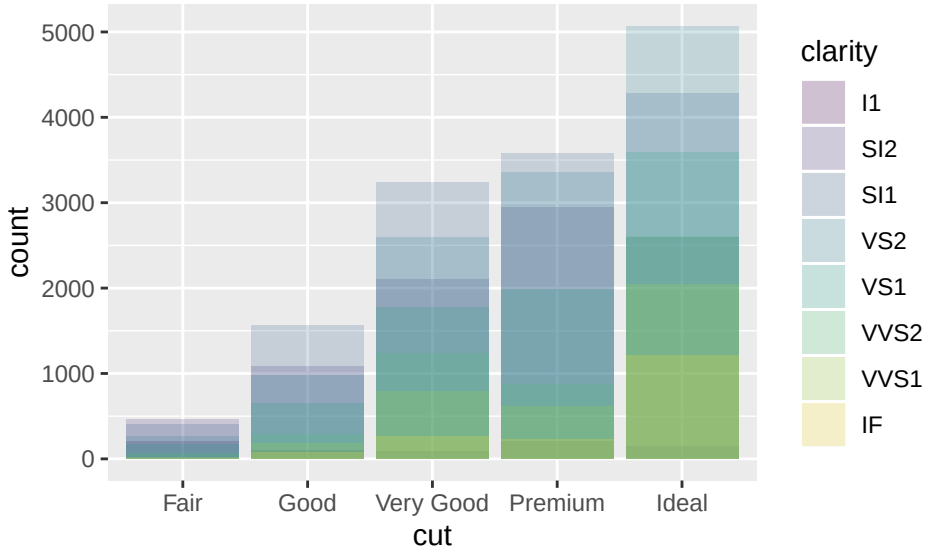
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```



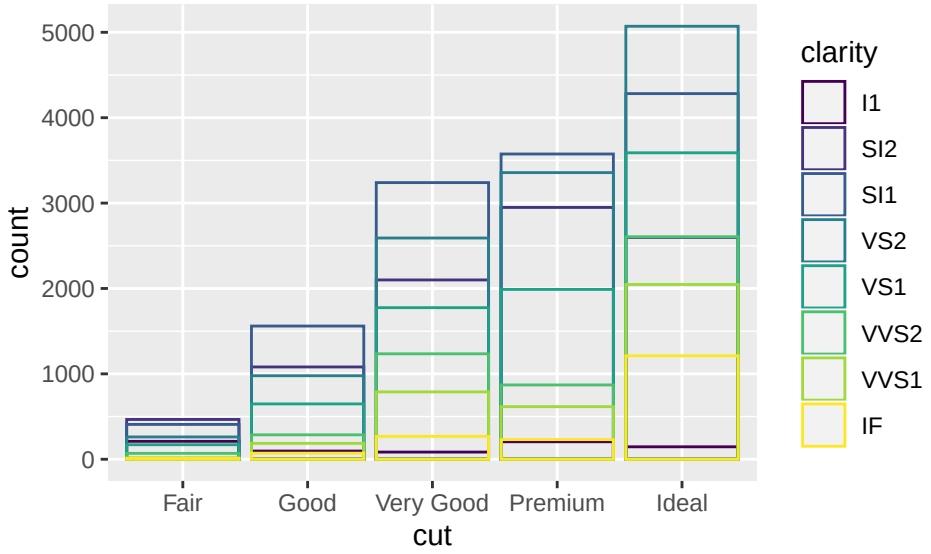
Stacking is performed automatically by the **position adjustment** specificed by the position argument. Other options for *position adjustment* are "identity", "dodge", or "fill".

position = "identity" will place each object exactly where it falls in the context of the graph (overlaps the bars, so we need to use alpha or fill = NA to see them).

```
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(alpha = 1/5, position = "identity")
```

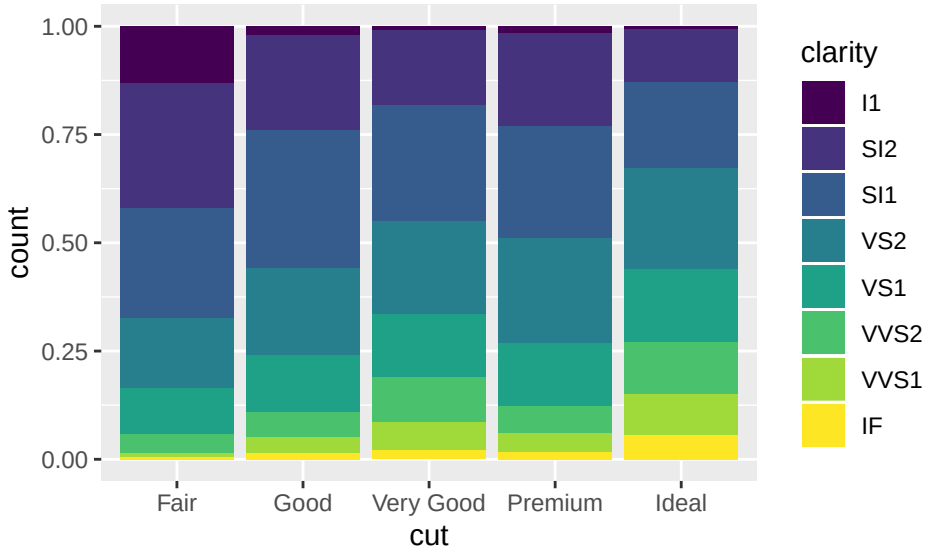



```
ggplot(data = diamonds, mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity")
```



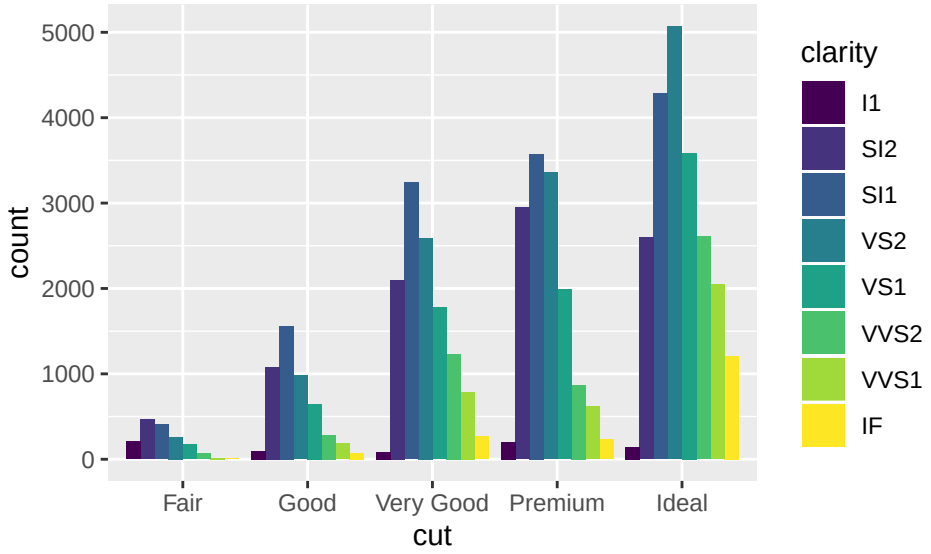
position = "fill" makes each set of stacked bars the same height, for proportion comparison.

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "fill")
```



position = "dodge" places overlapping objects directly *beside* one another.

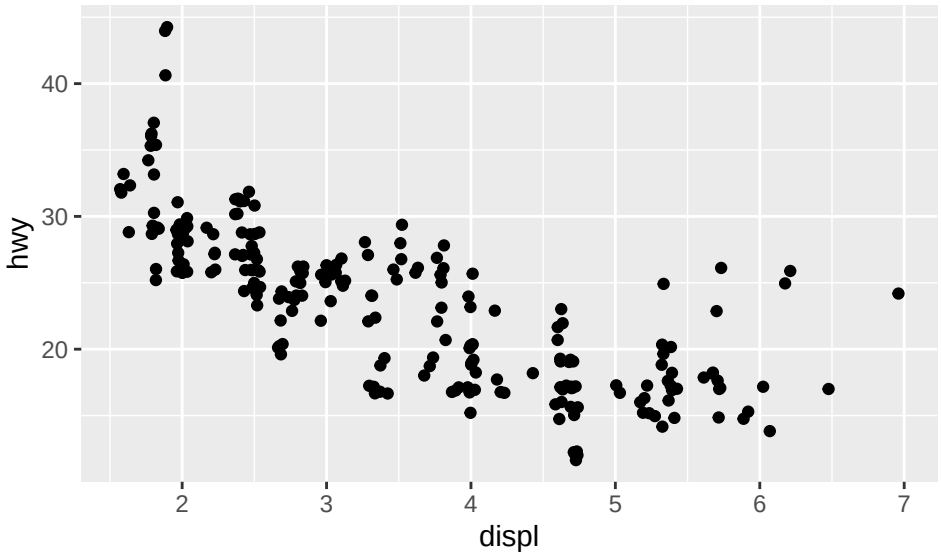
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



overplotting: many points overlapping on the grid and make it hard to see where the mass of the data is.

Setting position = "jitter" fixes this by adding a small amount of random noise to each point.

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



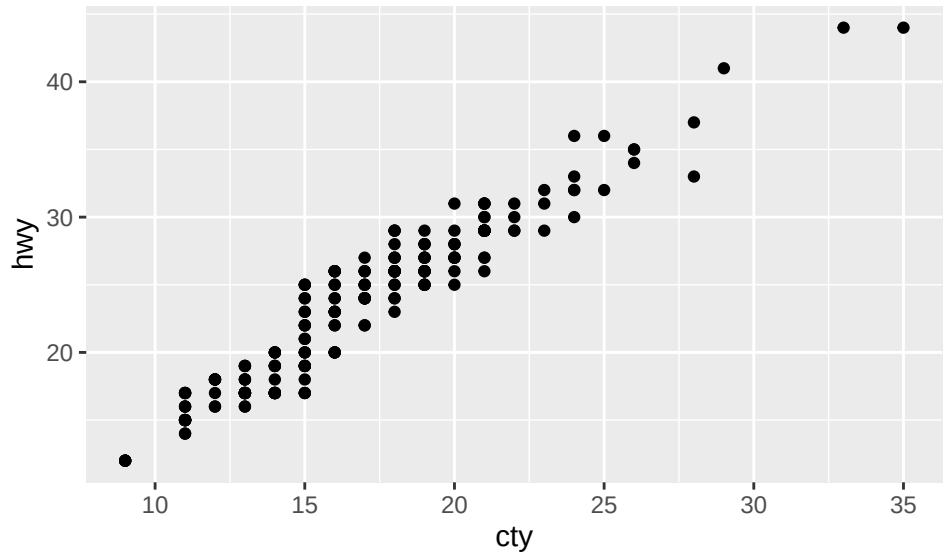
ggplot2 has a shorthand for geom_point(position = "jitter"): geom_jitter().

Other position adjustments: position_dodge, position_fill, position_identity, position_jitter, and position_stack (look up using ? in R for more info).

3.8.1 Exercises

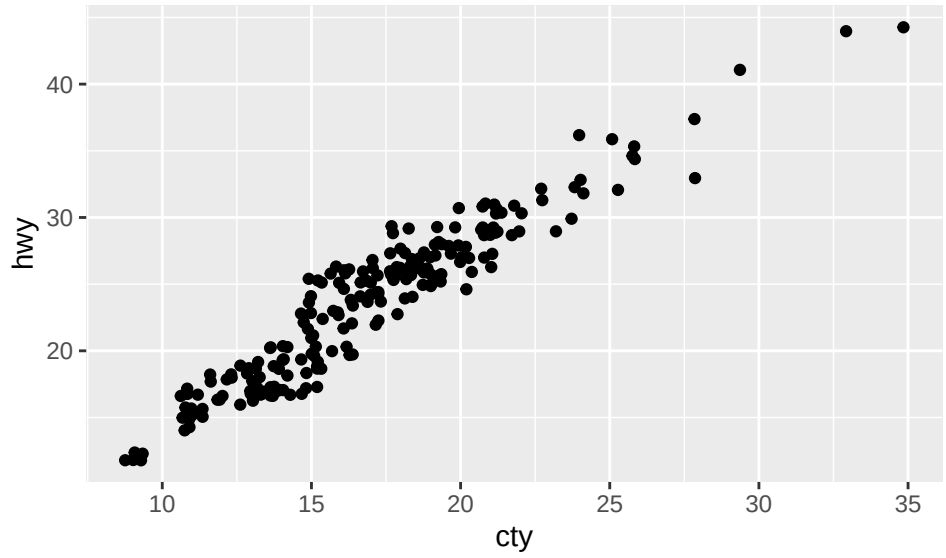
1.What is the problem with this plot? How could you improve it?

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



The problem with this plot is that it doesn't show all the points, so we are unable to tell where the concentration of observations lie. We can fix this by setting position = "jitter" to add a small amount of random noise:

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



2. What parameters to geom_jitter() control the amount of jittering?

```
#geom_jitter()
```

width and height control the amount of horizontal and vertical jitter added in both positive and negative directions. These default to 40% of the resolution of the data (doubled by being both in positive and negative directions so that jitter values are 80% of the bins).

3. Compare and contrast geom_jitter() with geom_count().

```
#?geom_count()
```

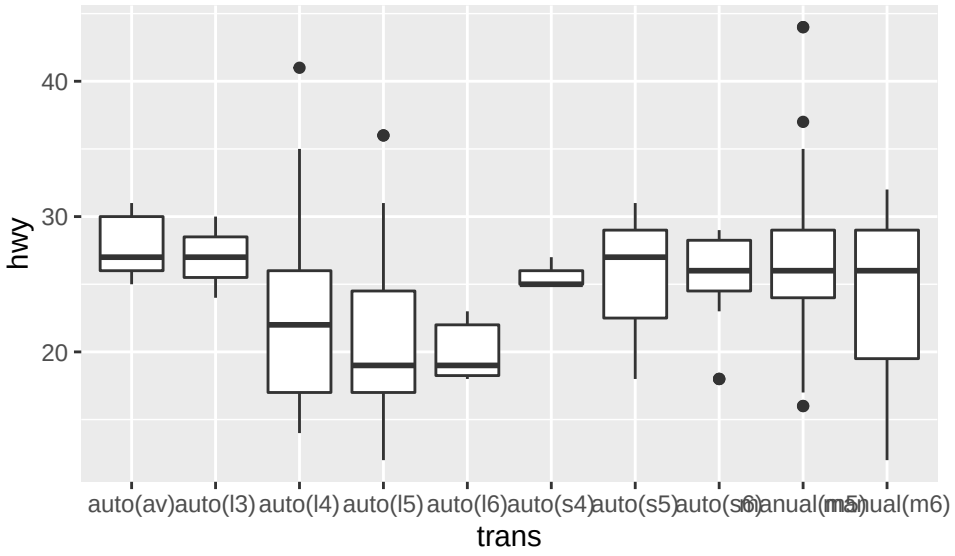
Both geom_jitter() and geom_count() serve the purpose of better displaying many many data points that may otherwise be obscured by binning and consolidation in the plotting defaults. While geom_jitter adds a small bit of random noise to move each of the points into view, geom_count() "counts the number of observations at each location, then maps the count to point area" so that the viewer can percieve the volume of points in a certain area based on the size of the point shown.

4. What's the default position adjustment for geom_boxplot()? Create a visualization of the mpg dataset that demonstrates it.

```
#?geom_boxplot()
```

The default position adjustment seems to be some level of dodging, at least enough that the boxes do not overlap. There is no natural order to the categorical trans variable, so I used it and hwy in the following example to show how geom_boxplot() dodges the categories:

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(trans,hwy))
```



3.9 Coordinate Systems

Coordinate systems provide the ability to specify any given point in a plot. The most commonly known is Cartesian, which is just the standard X,Y format we are all used to. Each X and pair points to a specific place on the chart. But there are other features in R that can also be used to specify location in more complex ways.

–coord_flip() swaps the X and Y axes. –coord_quickmap() used for special mapping in plots, for example on a map of Oregon. –coord_polar()

Coordinate Systems exercises

1. Create a polar plot: see below

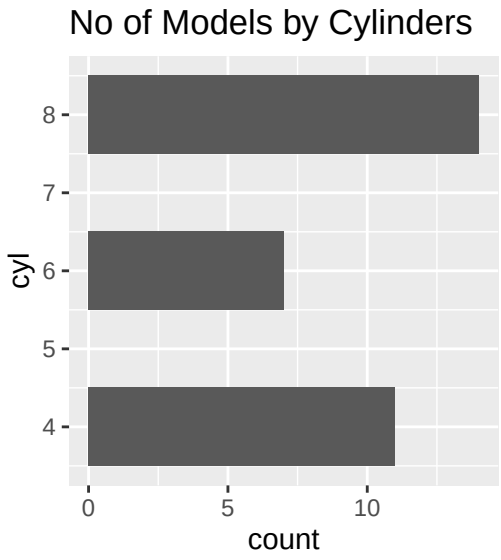
2. What does labs() do? It allows adding titles, subtitles, captions, tags, and labels to the charts. Title was added to the charts below this way.

3. What is the difference between coord_map and coord_quickmap? Coord_map attempts to represent geography on a spherical plane, whereas coord_quickmap assumes the plane is flat. Coord_quickmap is best for smaller geographical areas where distortion won't cause problems with the data representation.

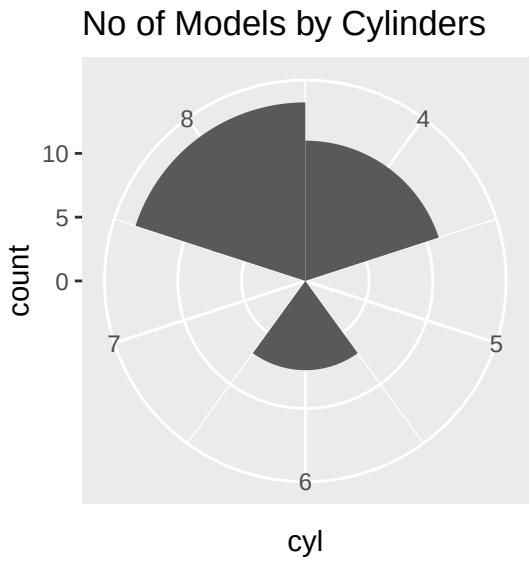
4. What does this plot tell you about the relationship between city and highway mpg? Higher city mileage is associated with higher highway mileage, but as hwy mileage goes up, there is a slight lag in city mileage increase (the overall slope of the data seems to tilt left of the neutral line)

—What is geom_abline? this puts a line of a particular slope on the graph. By default it puts a 1x1 slope on the graph. —What is coord_fixed? This enforces an "aspect ratio" so that the scales on the x and y axes are fixed in a ratio.

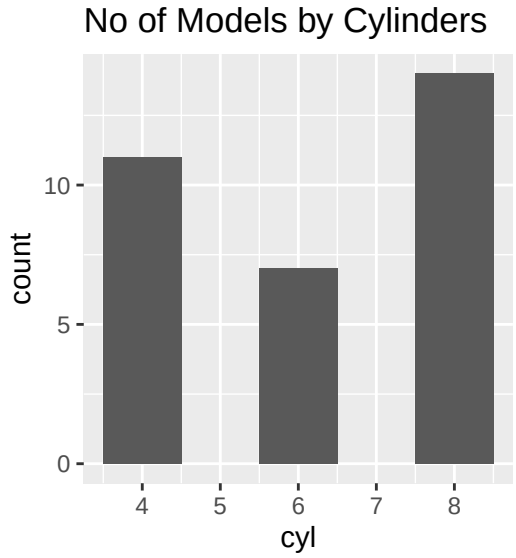
```
# Playing with bar and polar charts
bar <- ggplot(data = mtcars) +
  geom_bar(
    mapping = aes(x=cyl, fill=cyl),
    show.legend = FALSE,
    width = 1
  ) +
  theme(aspect.ratio = 1)

bar + coord_flip() + labs(title = "No of Models by Cylinders")
```



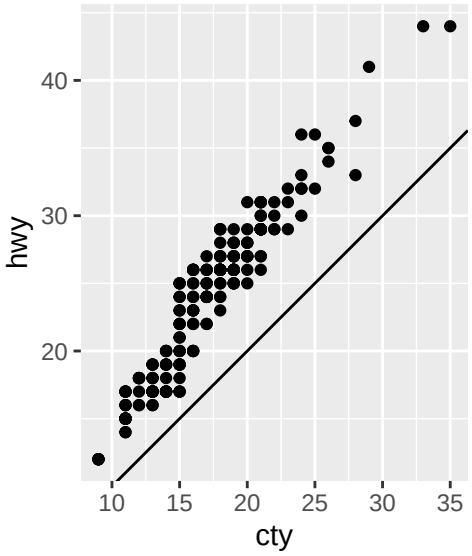

```
bar + coord_polar() + labs(title = "No of Models by Cylinders ")
```




```
bar + labs(title="No of Models by Cylinders")
```




```
# Trying out abline and fixed

ggplot(data=mpg, mapping = aes(x = cty, y=hwy)) +
  geom_point() +
  geom_abline() +
  coord_fixed()
```



3.10 The Layered grammar of graphics

Section 3.10 did not have exercises but gave a template for putting it all together. Below are some examples for visualizing the diamonds data with the various tools presented in chapter 3.

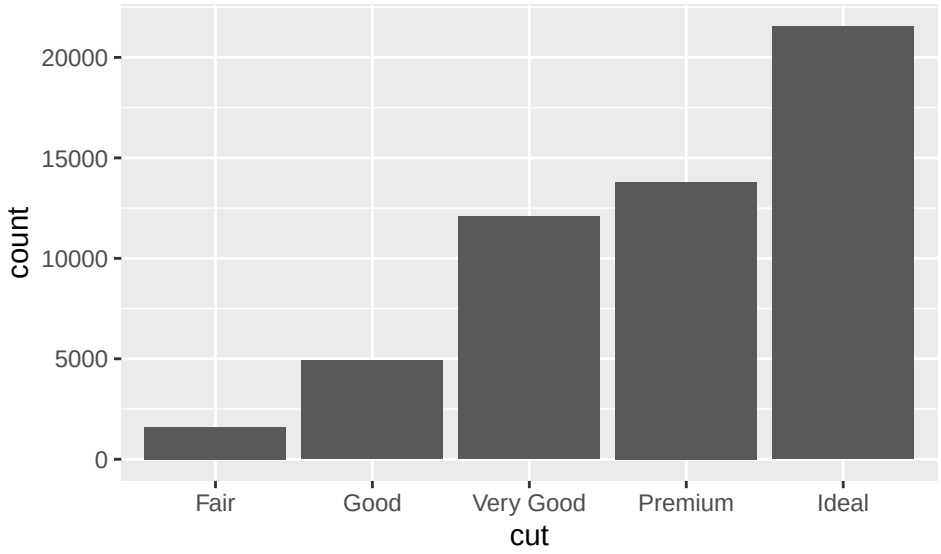
Layered grammar of graphic have seven parameters in the template:

1. combination of a dataset.
2. A geom
3. A set of mappings
4. A stat
5. A position adjustment
6. A coordinate system
7. A faceting scheme

Get data and Represent in a bar

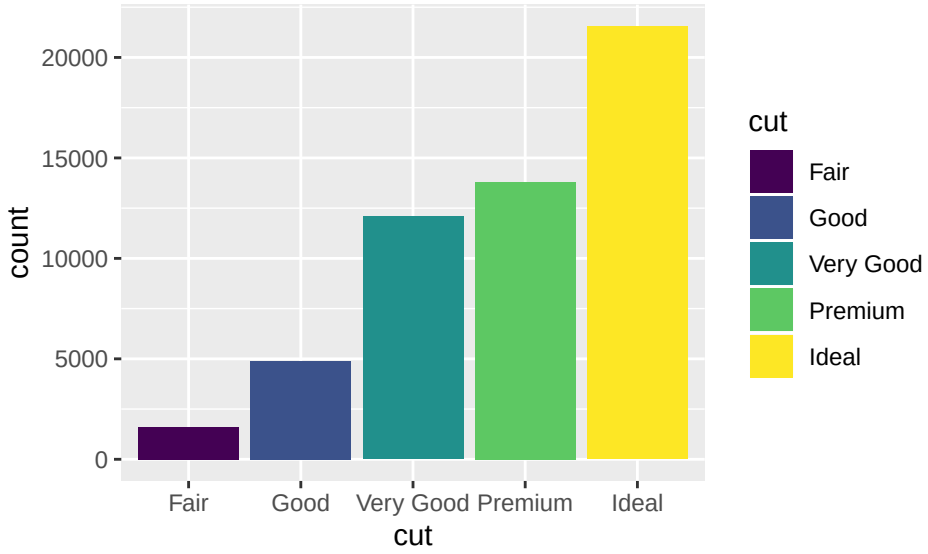
By choosing geometric object, like a bars.

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```



Map the fill of each bar to the count variable

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

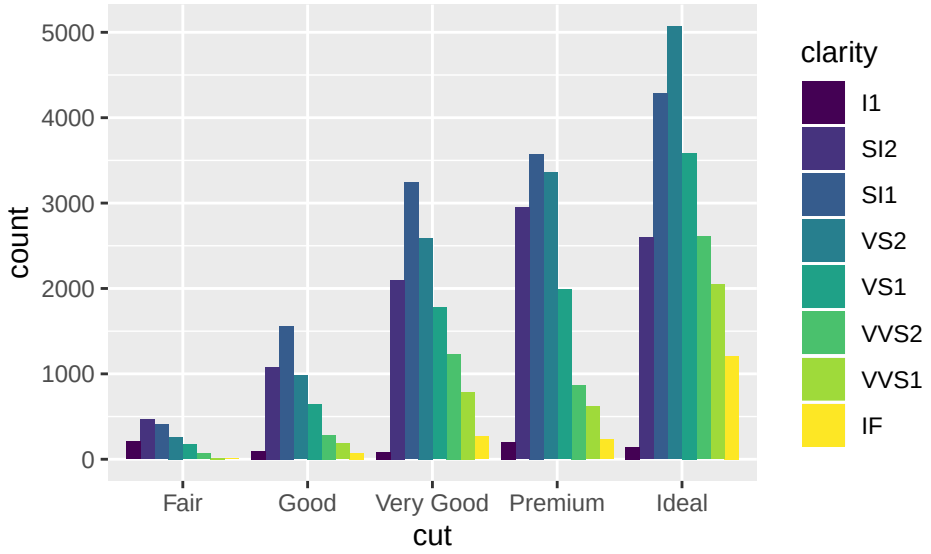


5. Place geoms in a cartesian coordiantes systems

Display the values of the x and y variables.

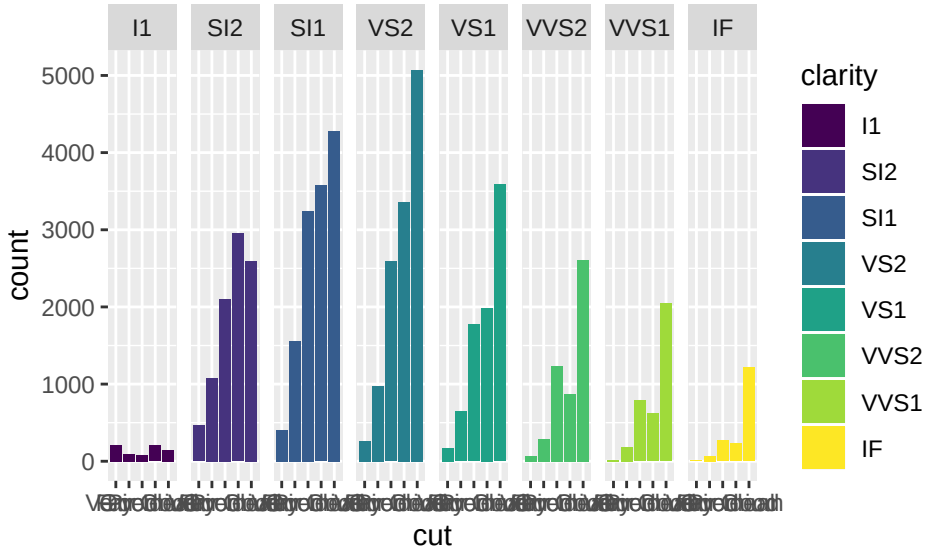
Adjust the postion by: *Coordinate system (a position adjustment)

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



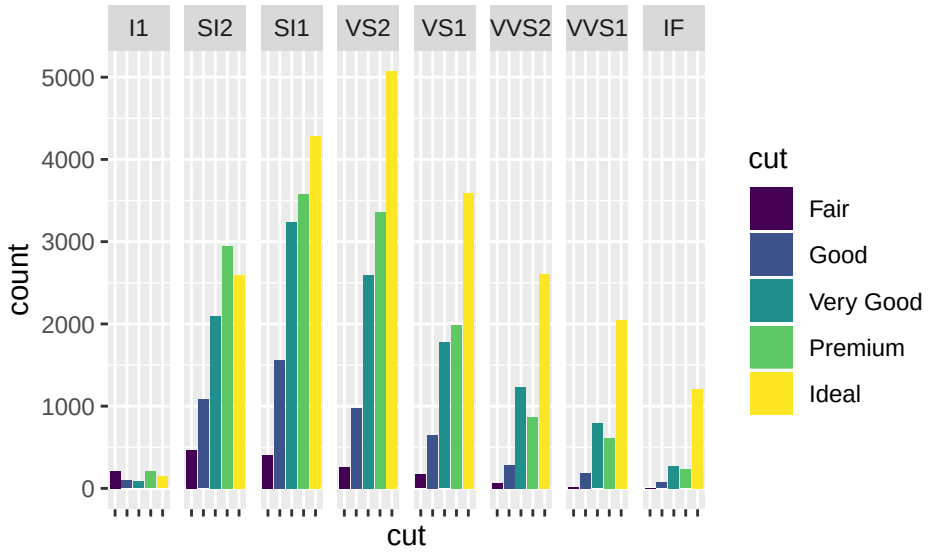
*Split the graph into subplots (faceting)

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill=clarity))+
  facet_grid(~clarity)
```



*Add more layer and modified the graph

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill=cut))+
  facet_grid(~clarity)+
  theme(axis.text.x = element_blank())
```


I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
5000
4000
3000
2000
1000
0
count
cut
cut
Fair
Good
Very Good
Premium
Ideal