Week7 R Code - Anly 506-51

Nitin

July 23, 2019

Data visualisation

Load required library and packages for this exercises.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ------ tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.0       v purrr   0.2.5
## v tibble  2.1.3       v dplyr   0.8.0.1
## v tidyr   0.8.1       v stringr 1.3.1
## v readr   1.1.1       v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'tibble' was built under R version 3.5.3
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
## -- Conflicts --------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Test mpg data frame for out put.

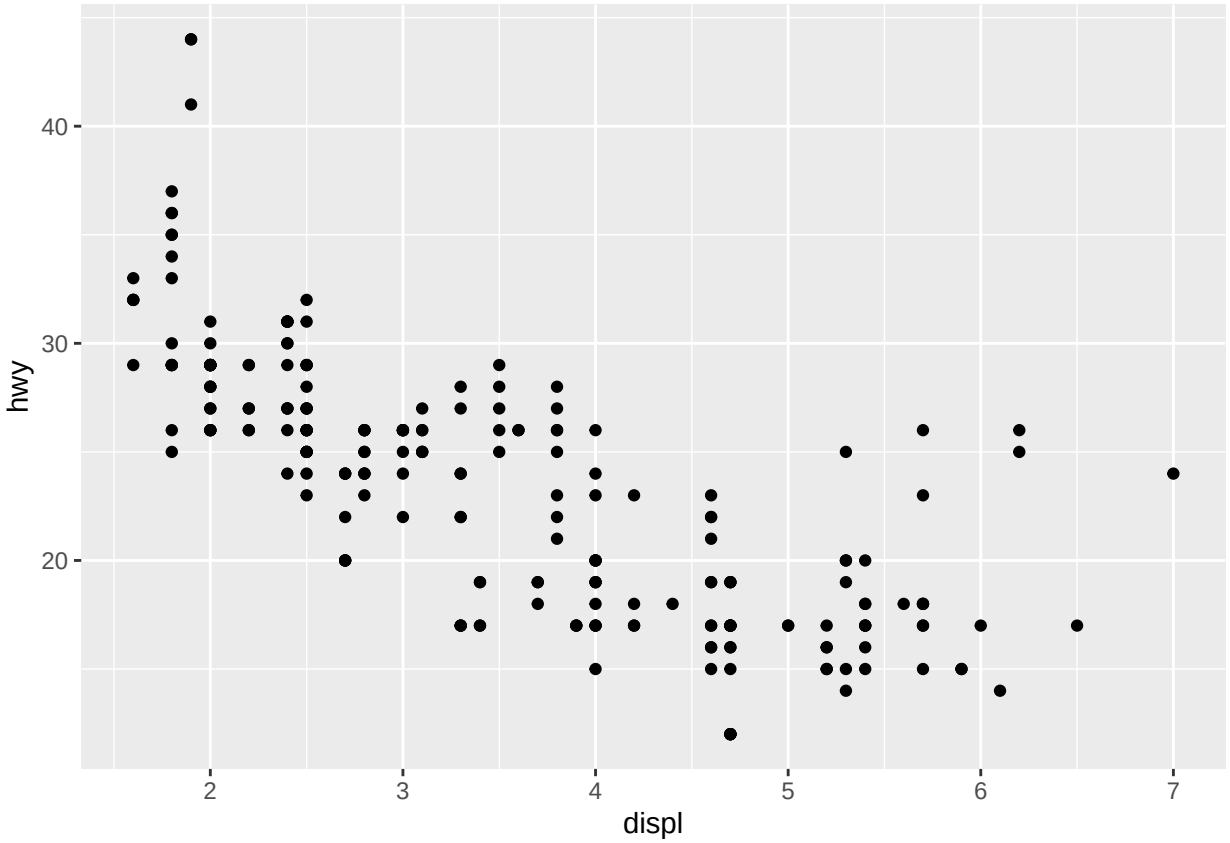
```
mpg
```

```
## # A tibble: 234 x 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~ f        18    29 p     comp~
##  2 audi         a4      1.8  1999     4 manu~ f        21    29 p     comp~
##  3 audi         a4      2    2008     4 manu~ f        20    31 p     comp~
##  4 audi         a4      2    2008     4 auto~ f        21    30 p     comp~
##  5 audi         a4      2.8  1999     6 auto~ f        16    26 p     comp~
##  6 audi         a4      2.8  1999     6 manu~ f        18    26 p     comp~
##  7 audi         a4      3.1  2008     6 auto~ f        18    27 p     comp~
##  8 audi         a4 q~   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 q~   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 q~   2    2008     4 manu~ 4        20    28 p     comp~
## # ... with 224 more rows
```

Creating a ggplot

Creating a ggplot for mpg data with x axis displ and y axis hwy.

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

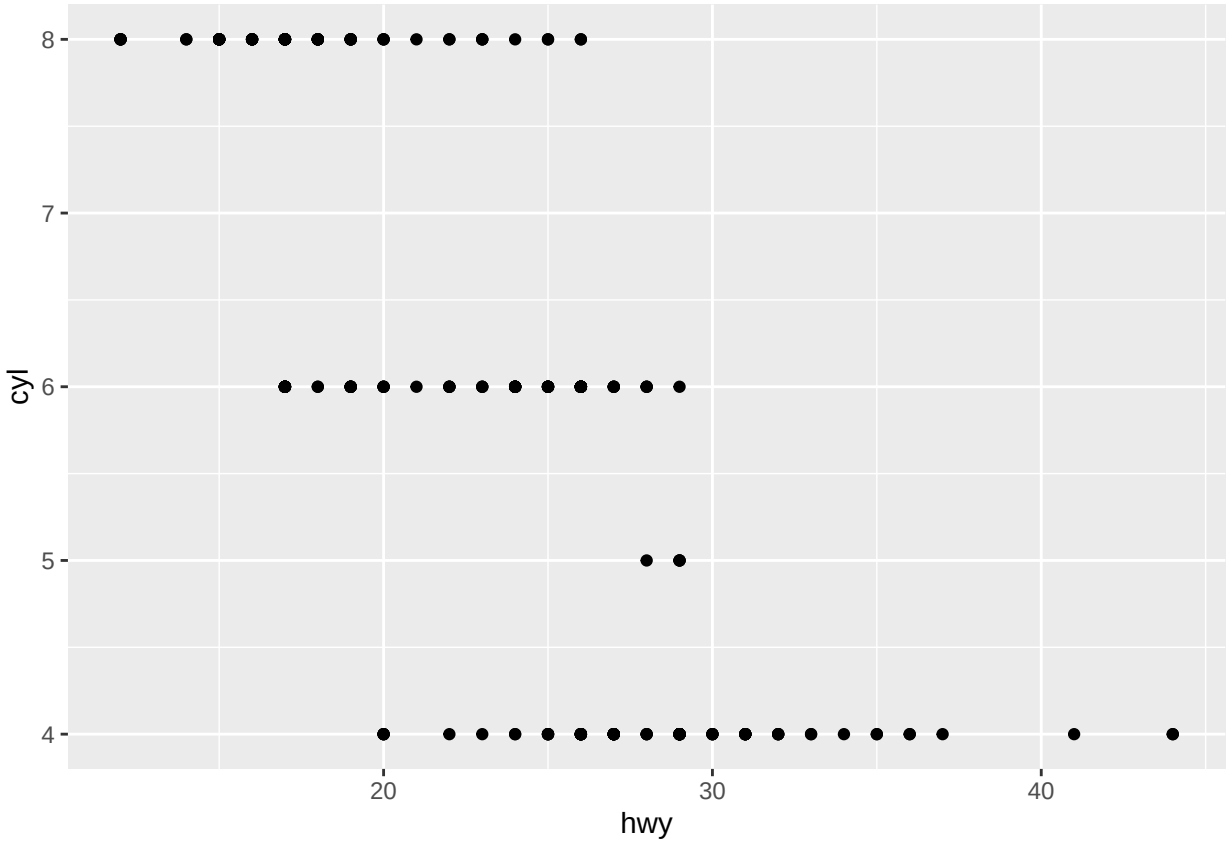


A graphing template

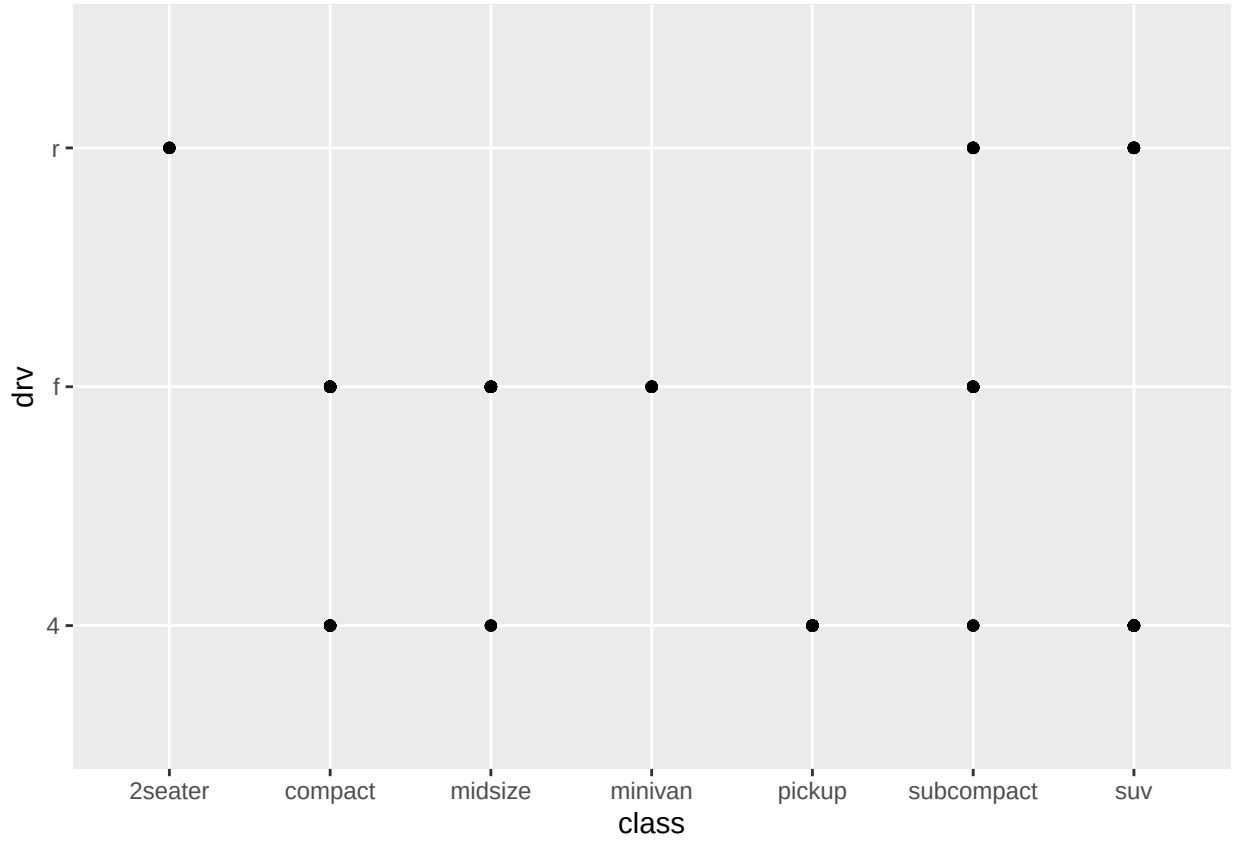
Following is a graphng templet used. ggplot(data =) + (mapping = aes())

Exercises

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



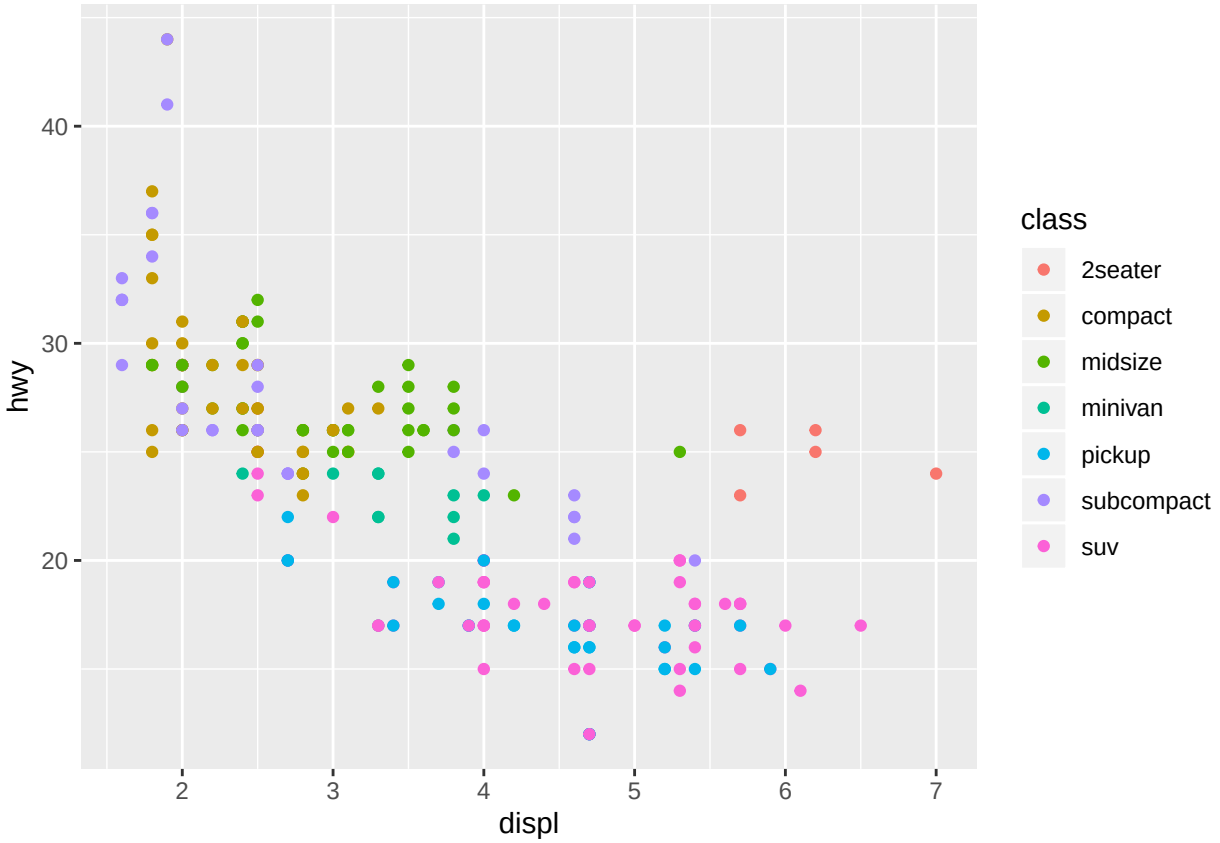

```
ggplot(data = mpg) + geom_point(mapping = aes(x = class, y = drv))
```



Aesthetic mappings

Good aesthrtic forces us to see and notice things we never noticed.

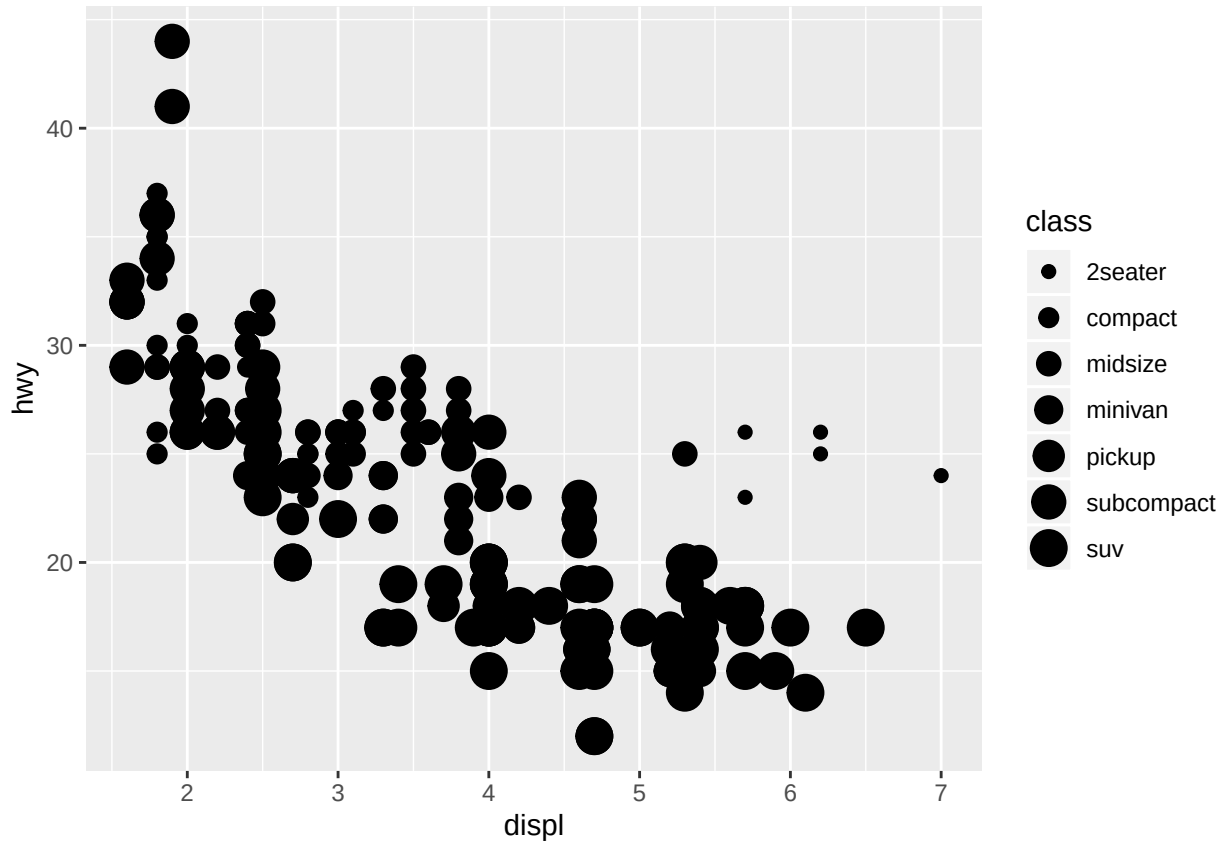
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



We get a warning here, because mapping an unordered variable (class) to an ordered aesthetic (size) is not a good idea.

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

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

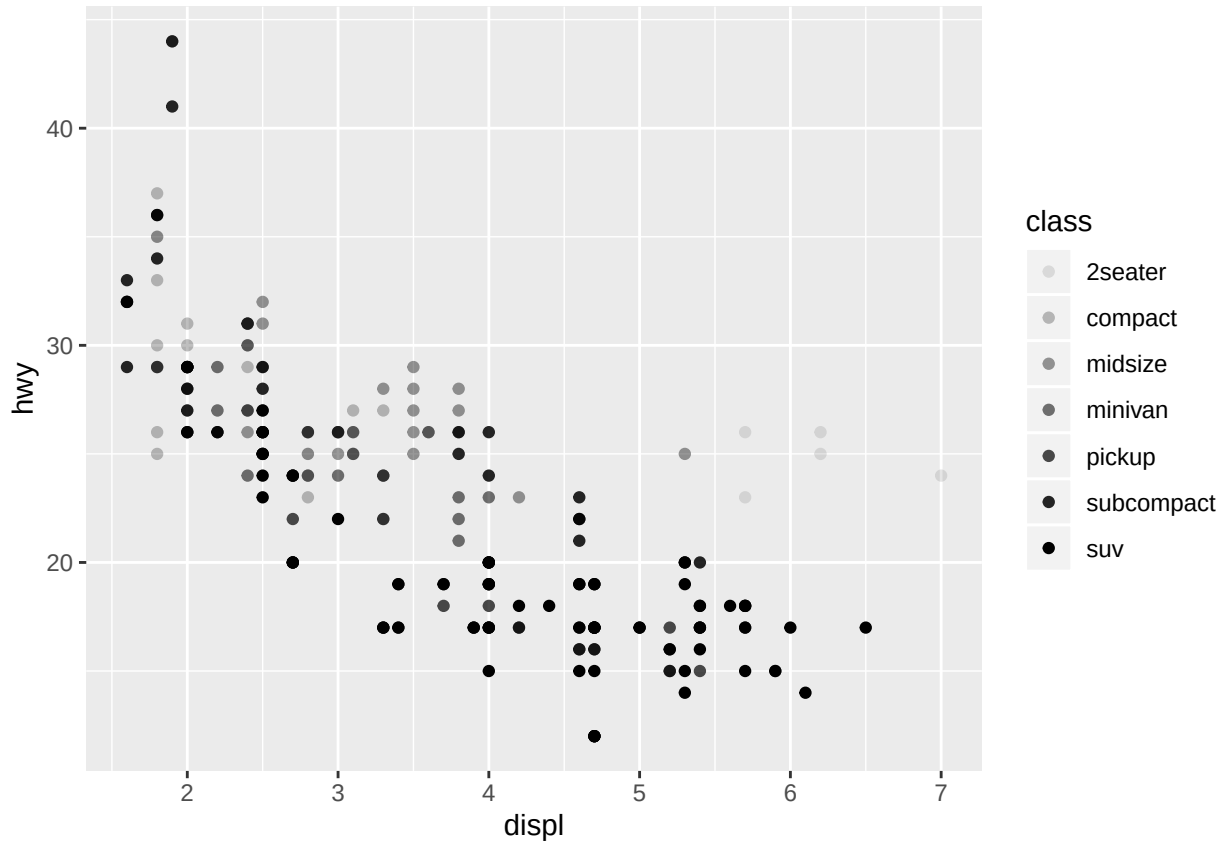



```
#> Warning: Using size for a discrete variable is not advised.
```

Recomonded way is to mapped class to the alpha aesthetic, which controls the transparency of the points. Ohter way is to mappe to the shape aesthetic, which controls the shape of the points

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

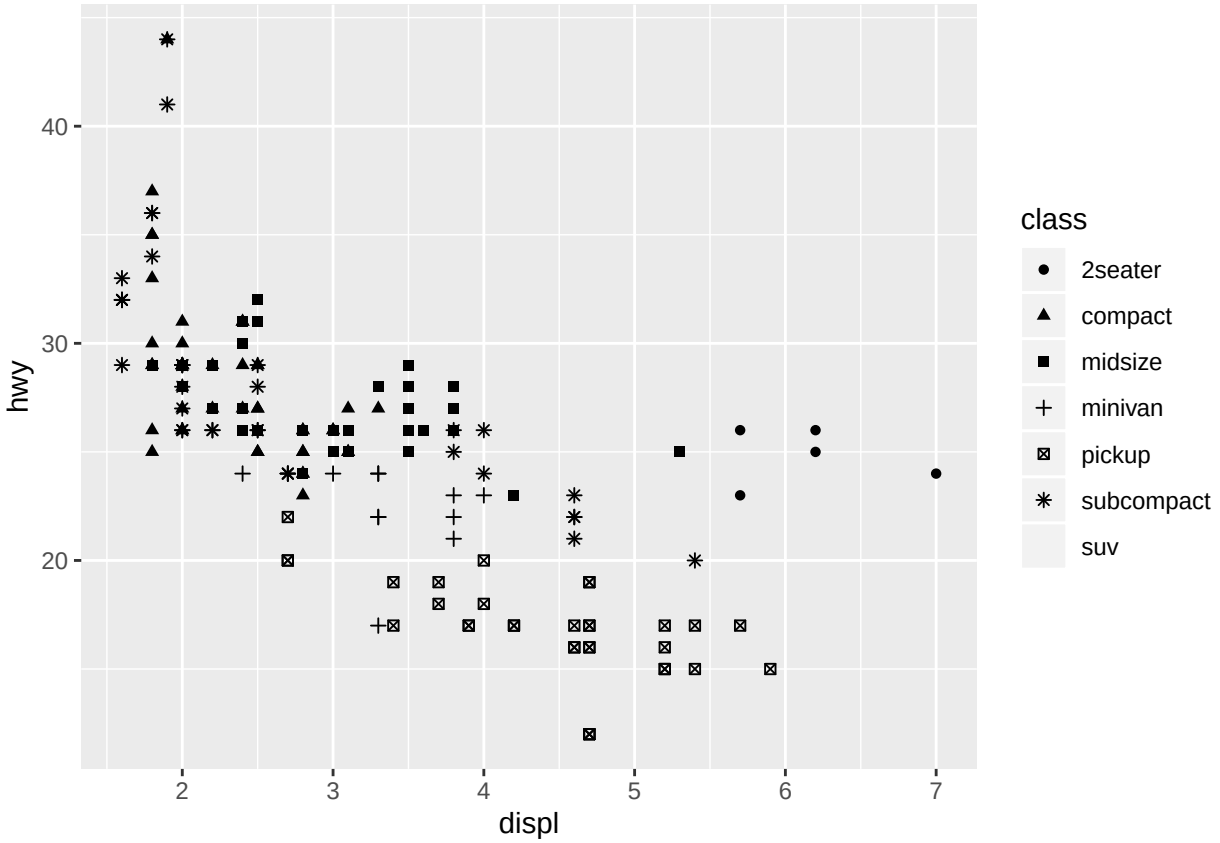
```
## Warning: Using alpha for a discrete variable is not advised.
```




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

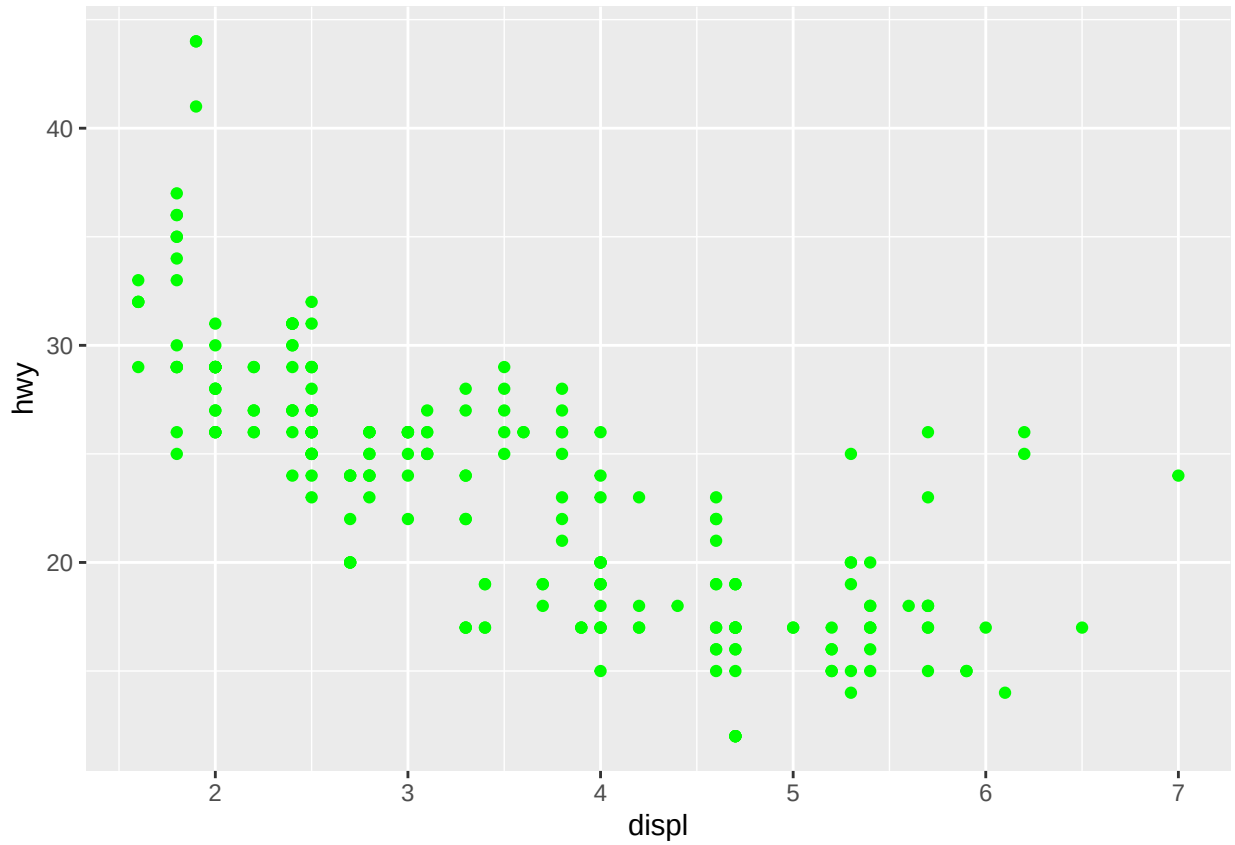
```
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have 7.
## Consider specifying shapes manually if you must have them.
```

```
## Warning: Removed 62 rows containing missing values (geom_point).
```



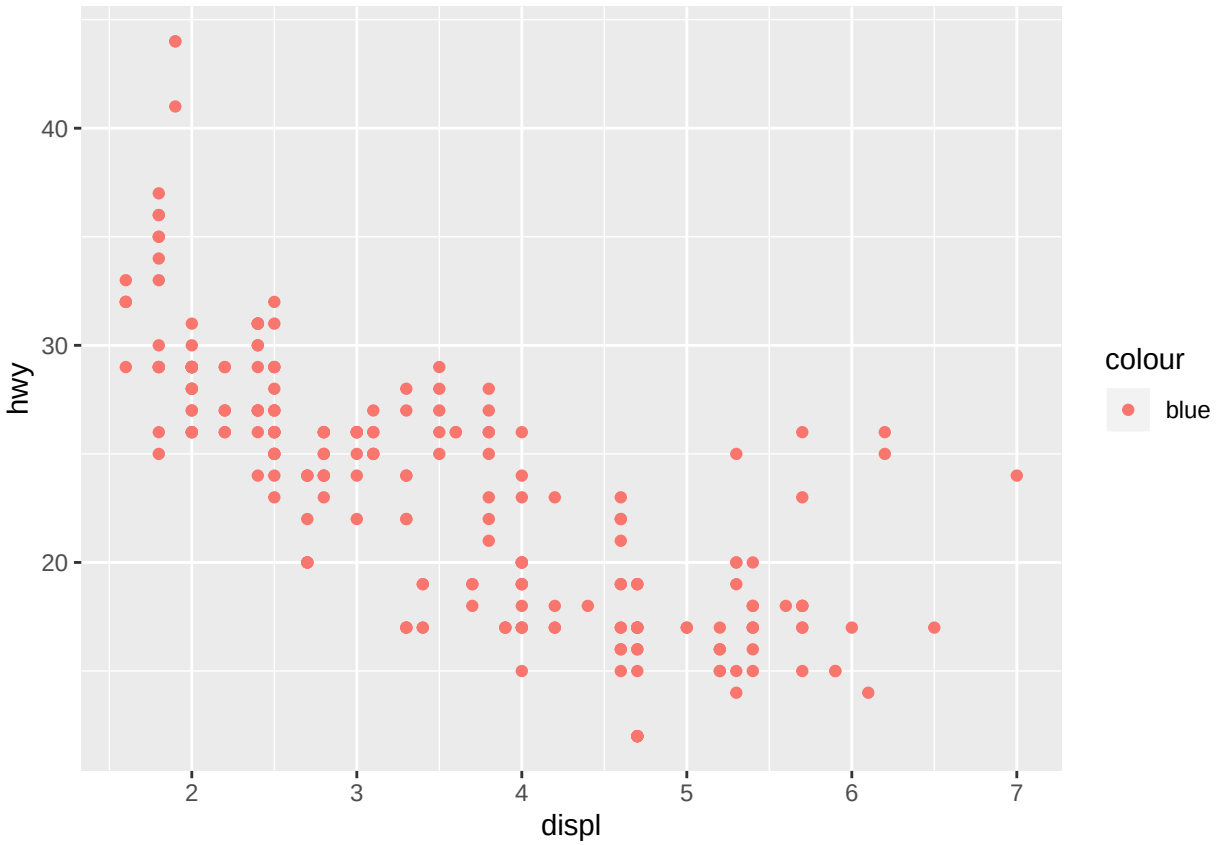
Set the aesthetic properties your choice like color to green.

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



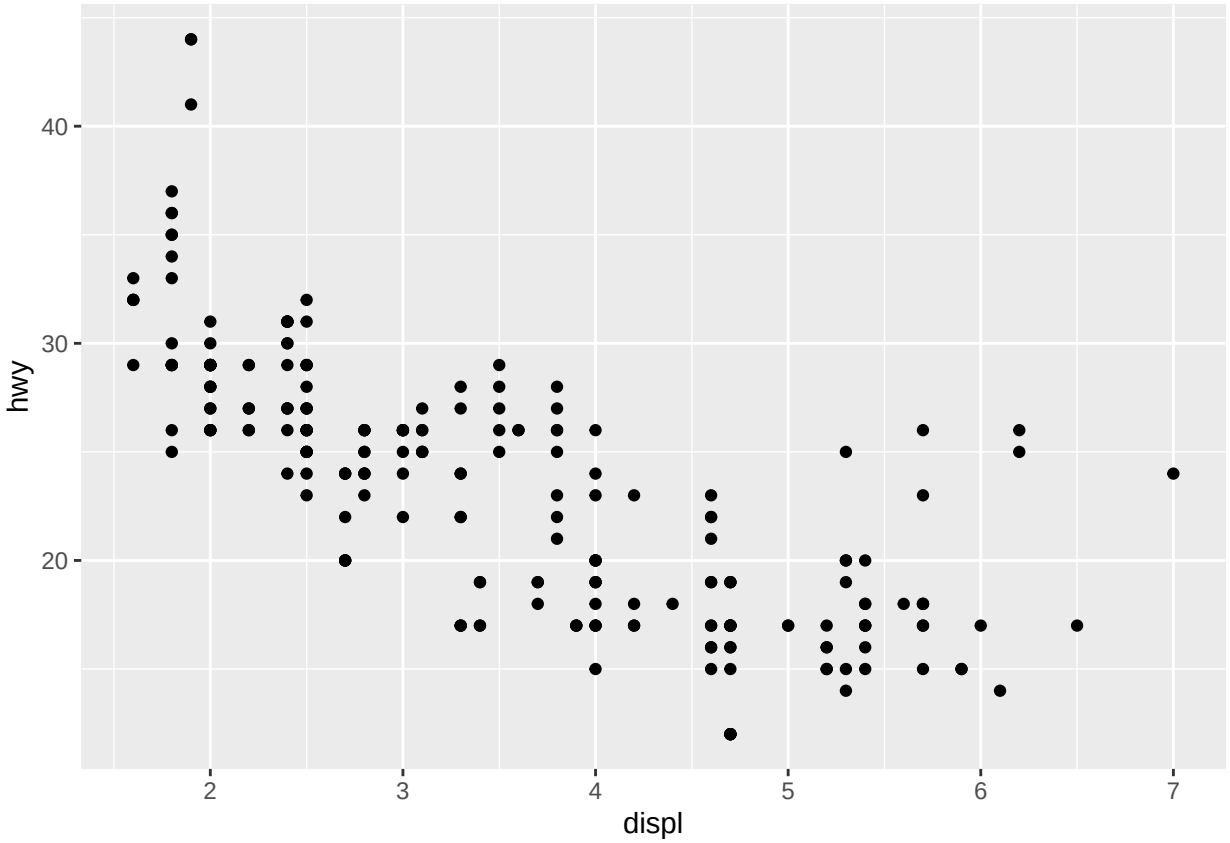
Exercises

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



Common problems

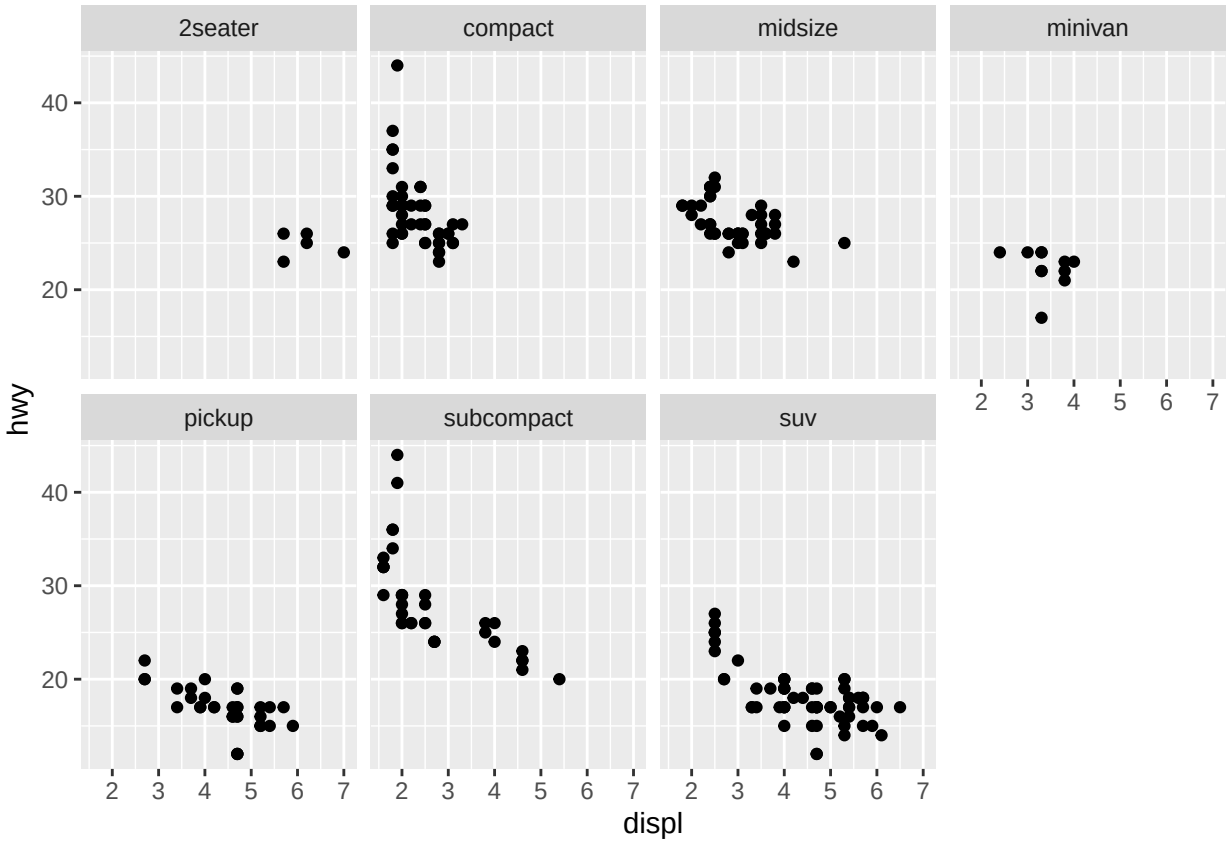
```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy))
```



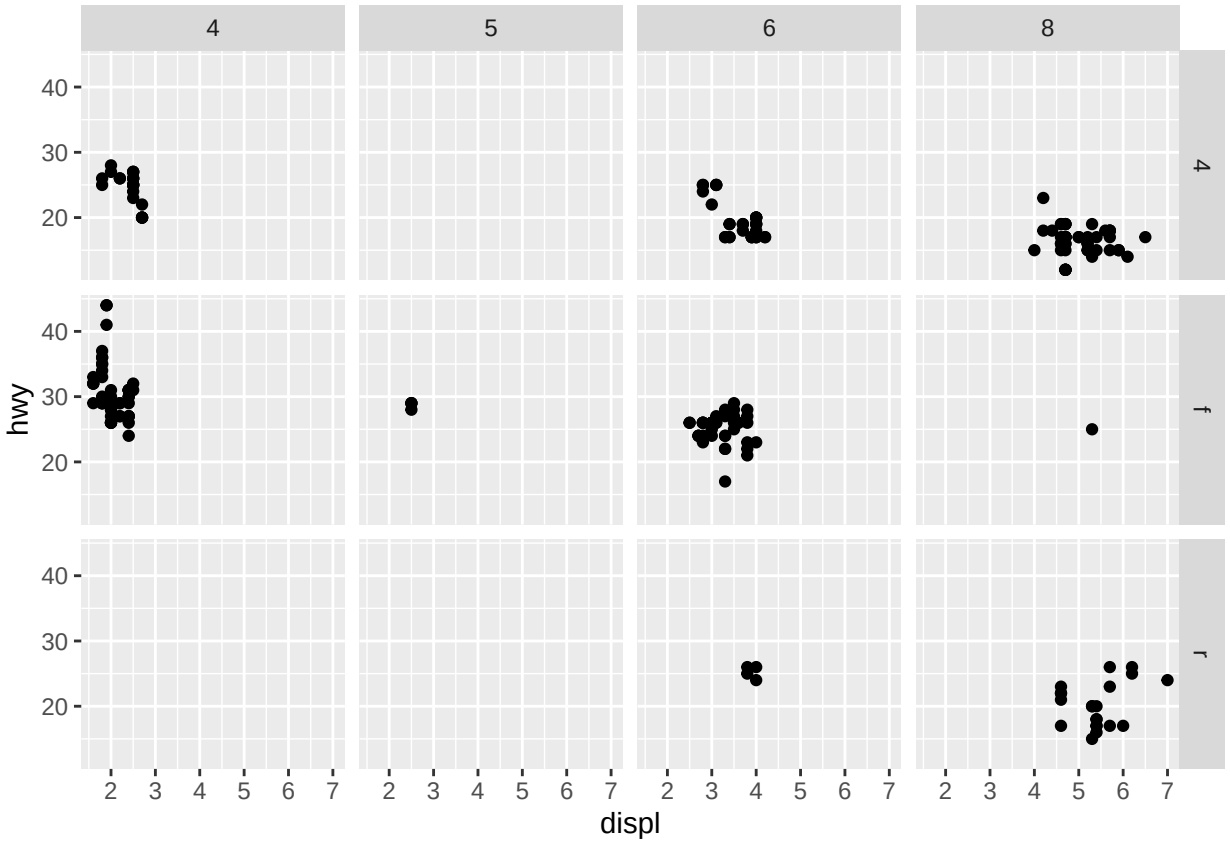
Facets

Split your plot into subplots using facets which display one subset of the data

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



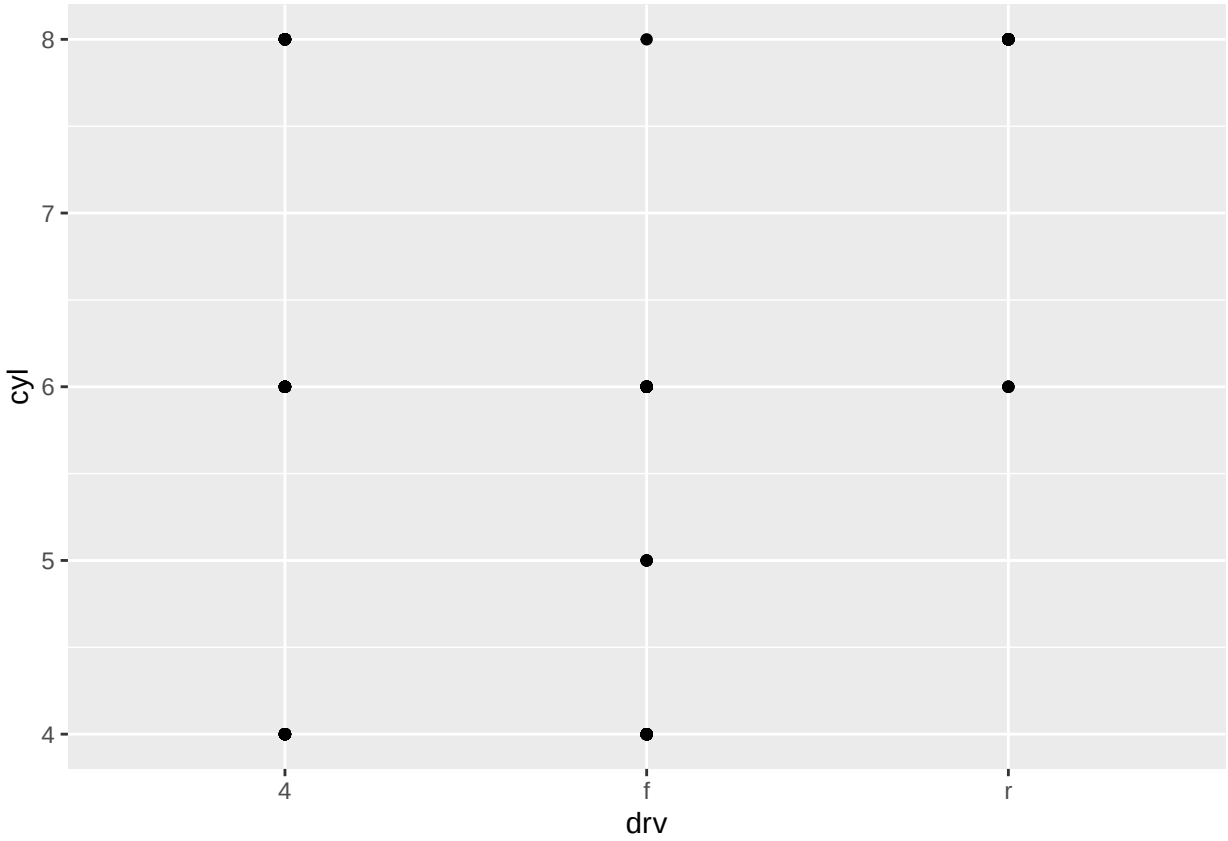

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



Exercises 1. What happens if you facet on a continuous variable?

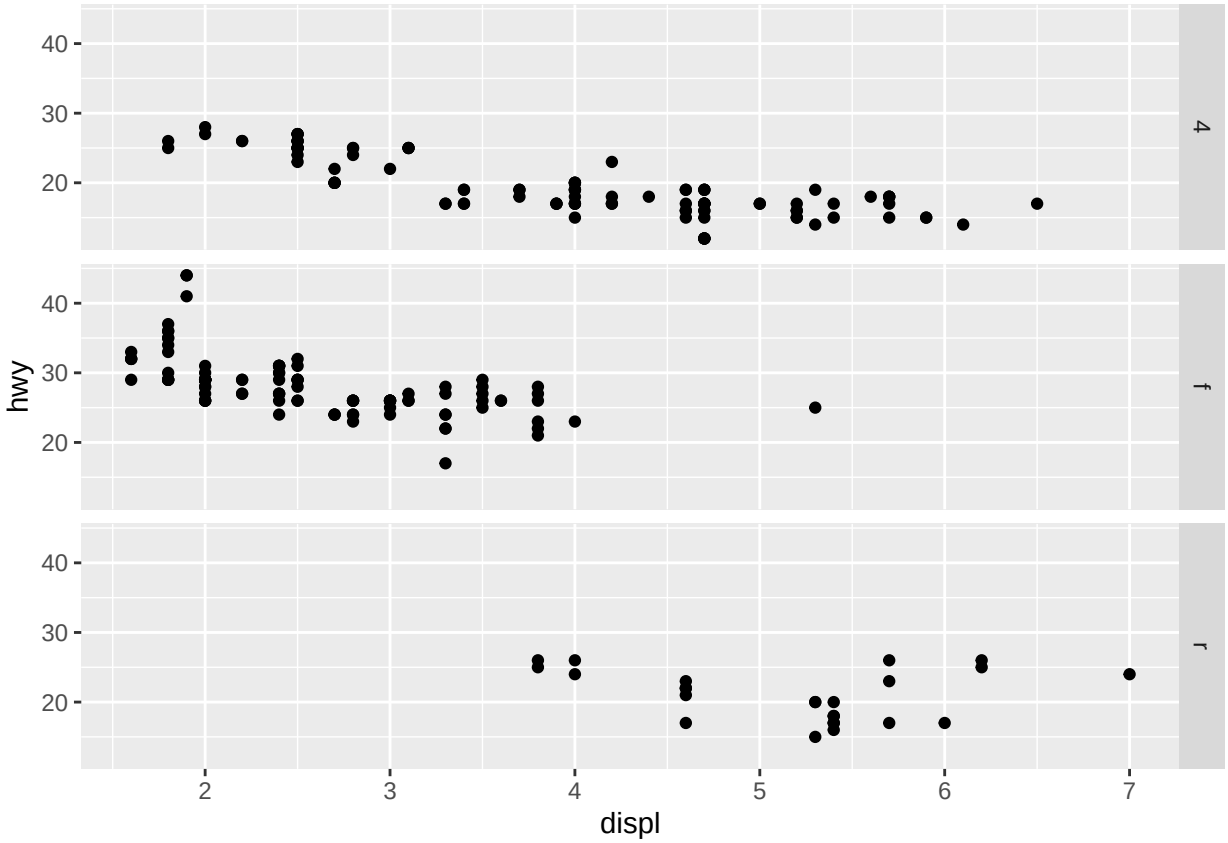
2. What do the empty cells in plot with facet_grid(drv ~ cyl) mean? How do they relate to this plot?

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

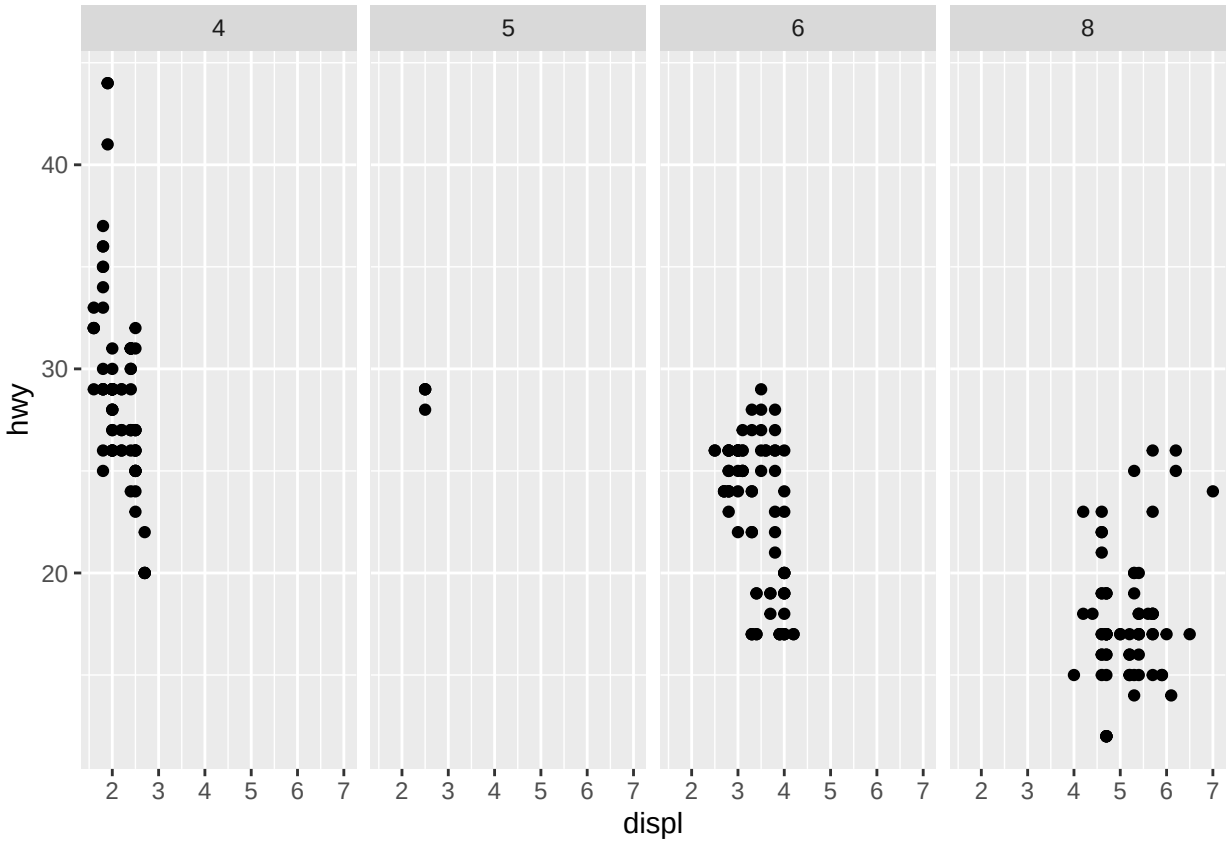


3. What plots does the following code make? What does . do?

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

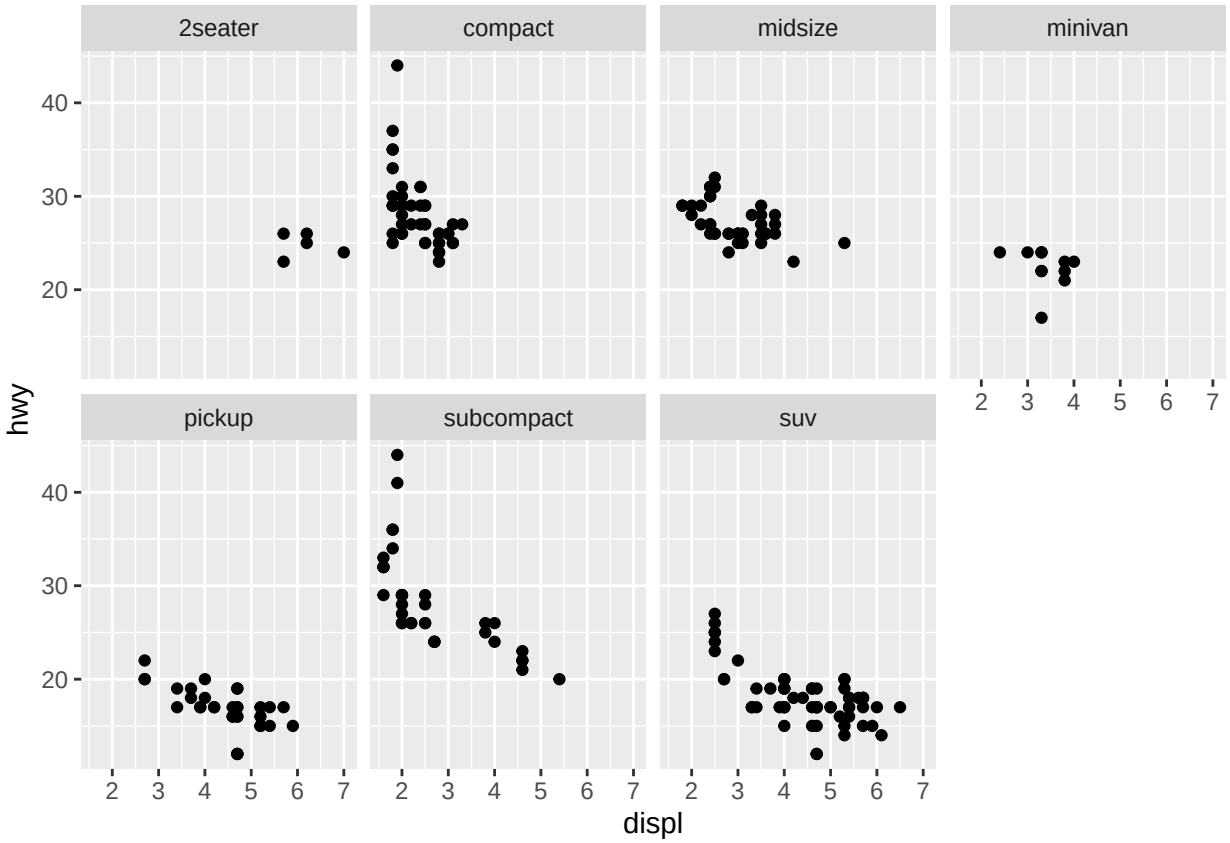



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(. ~ cyl)
```



4. Take the first faceted plot in this section:

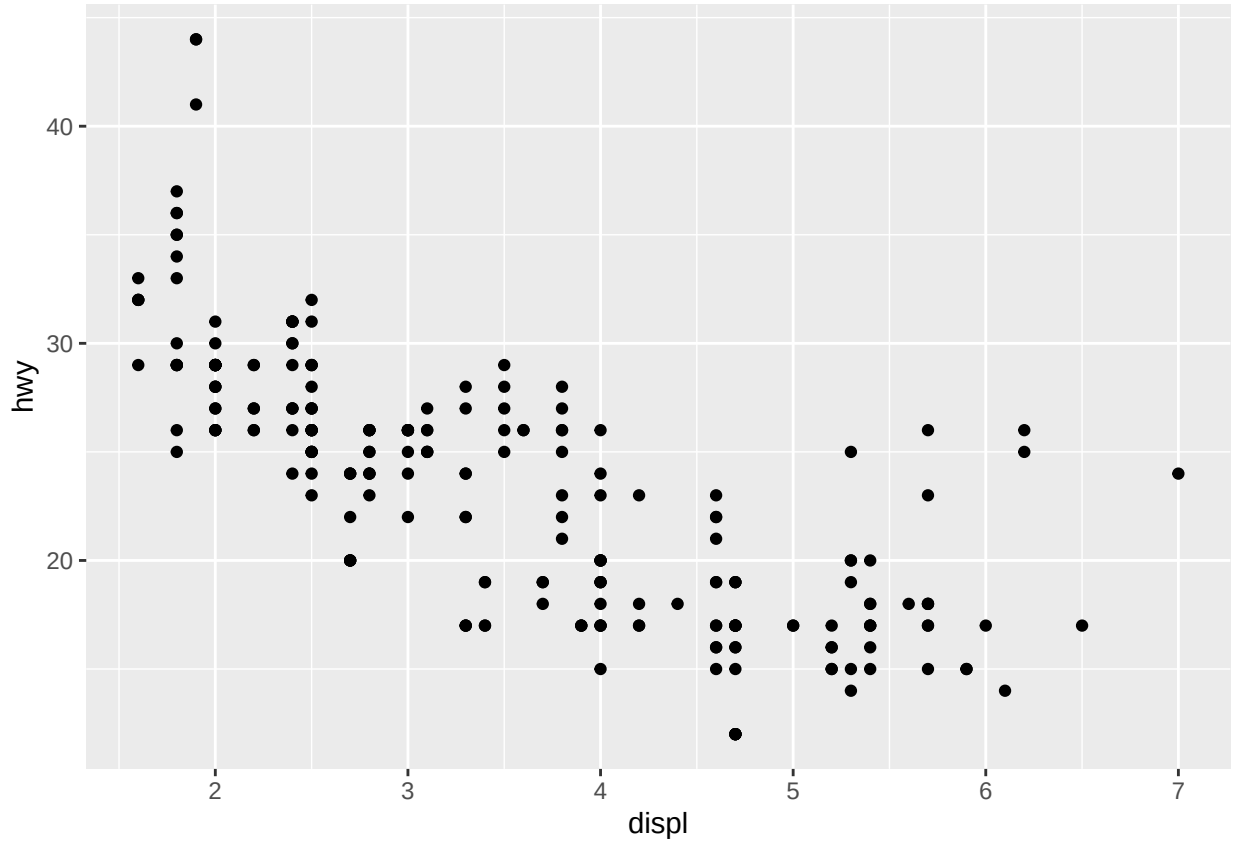
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



Geometric objects.

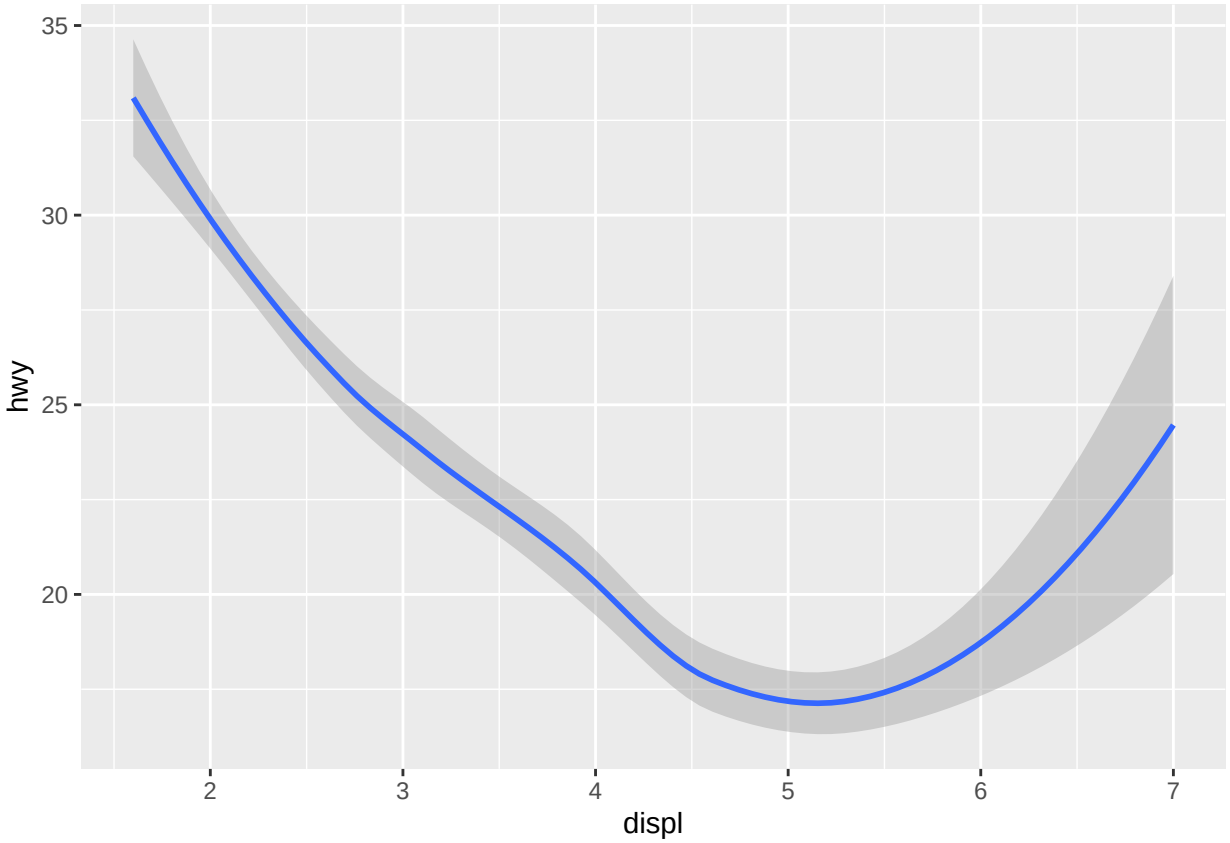
When each plot uses same data but different visual object, it means they are using diffrent geoms.

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




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

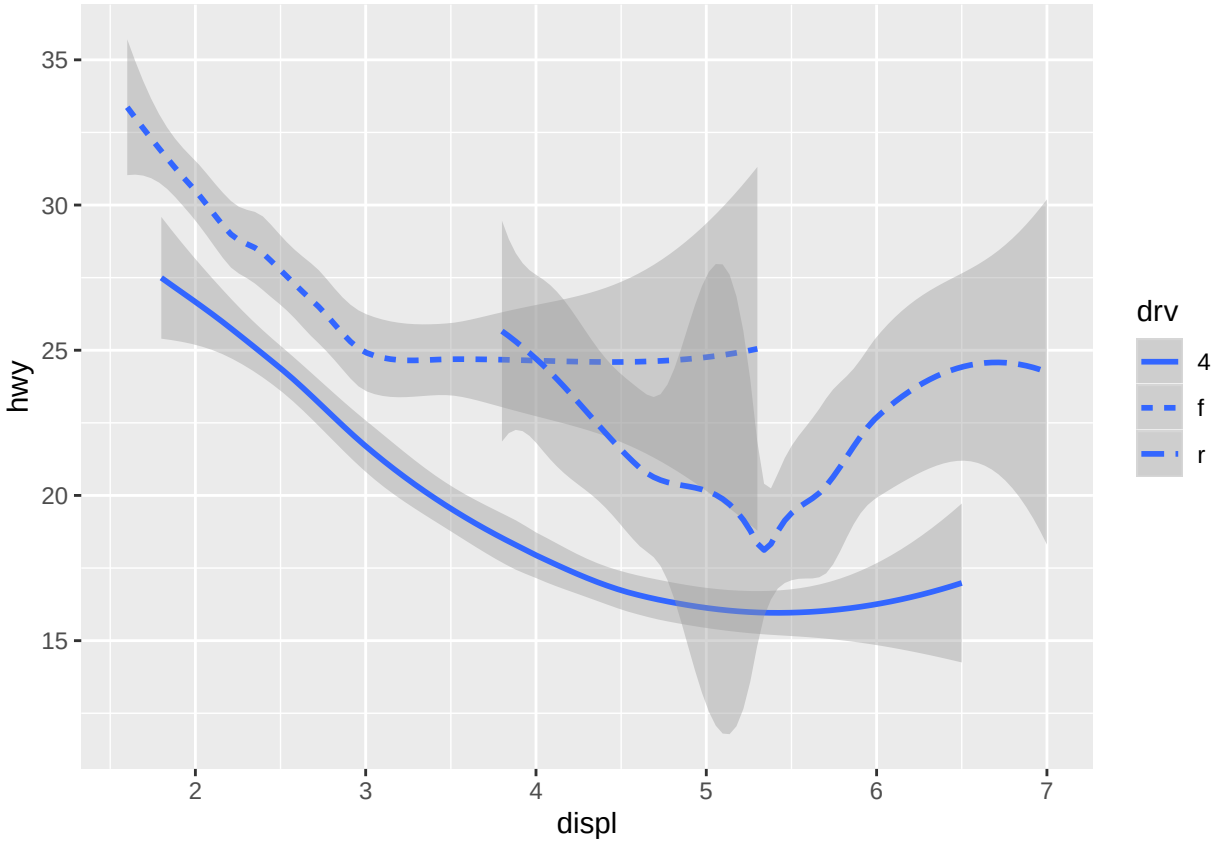
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Function geom_smooth() draws a different line, 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))
```

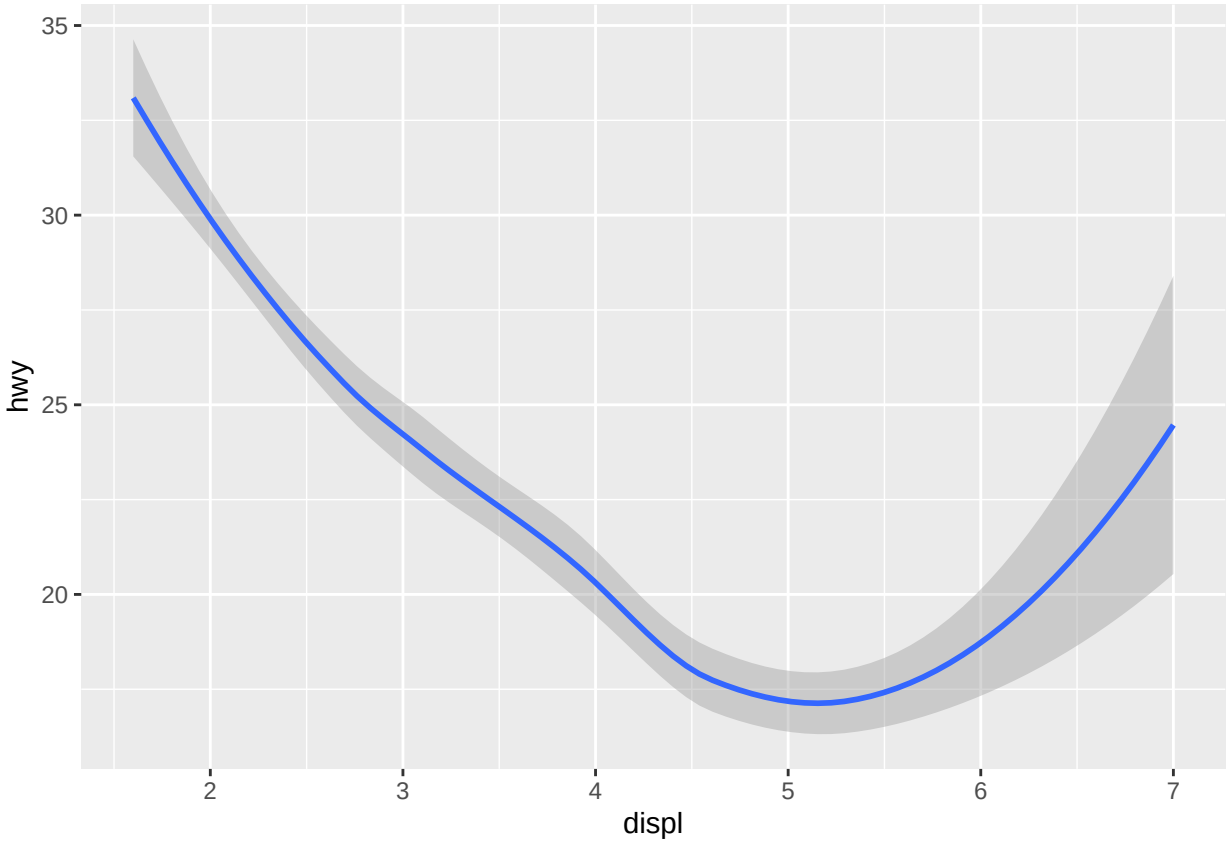
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Coloring according to drv. Draw a separate object for each unique value of the grouping variable

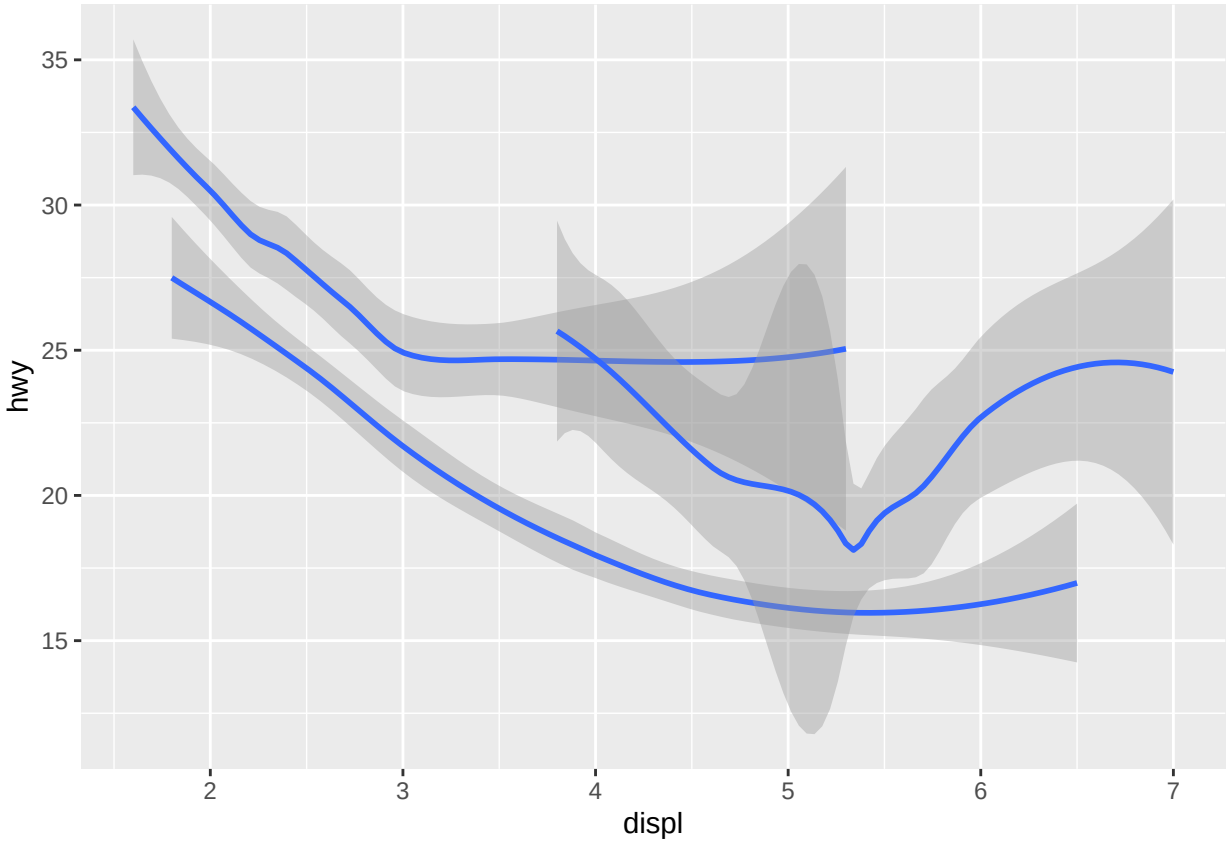
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

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



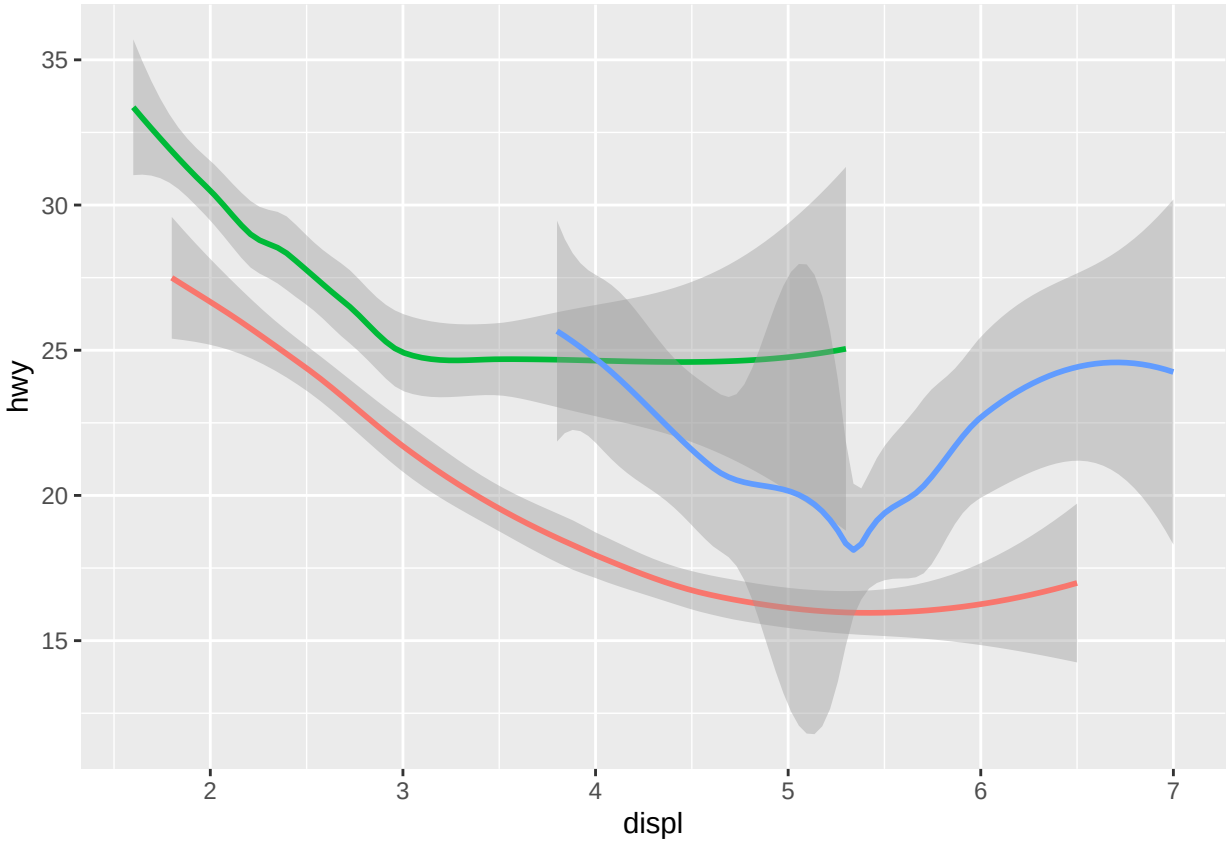

```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```

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



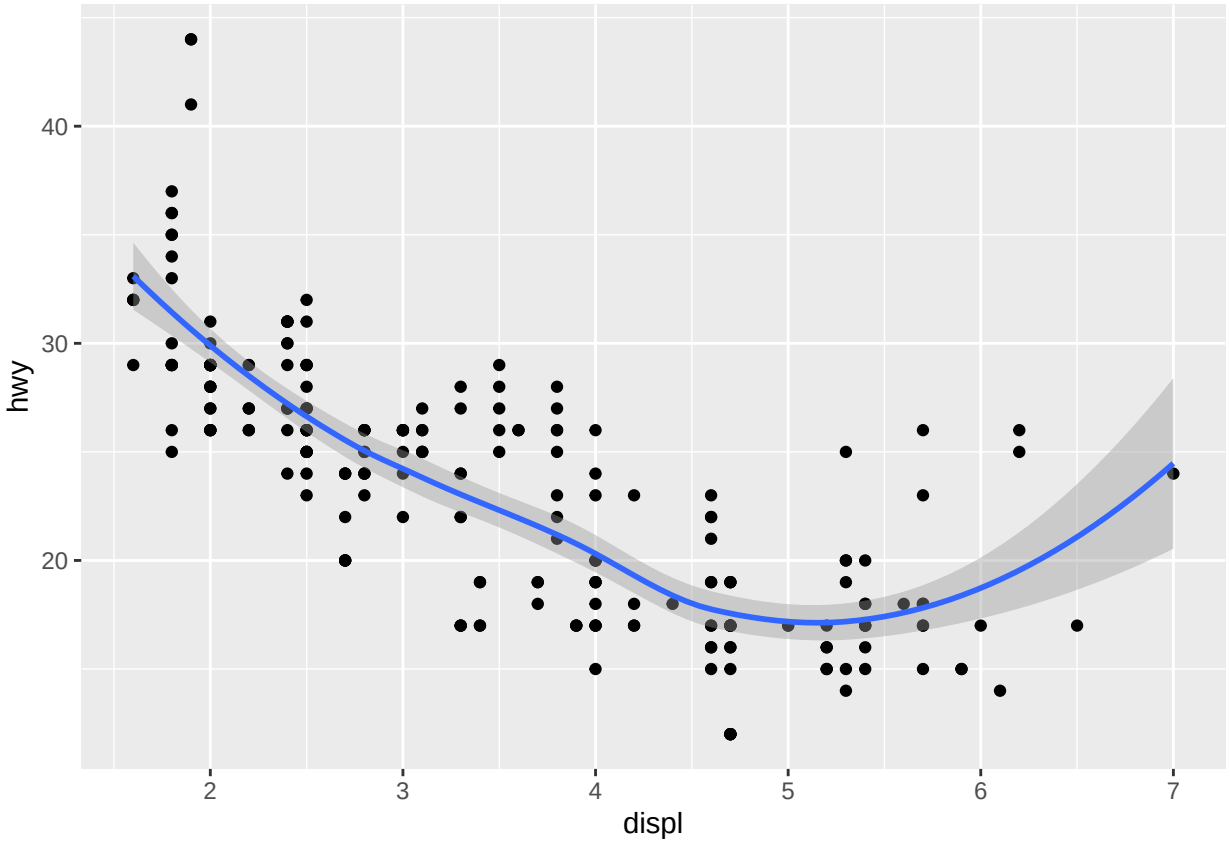

```
ggplot(data = mpg) +
  geom_smooth(
    mapping = aes(x = displ, y = hwy, color = drv),
    show.legend = FALSE
  )
```

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



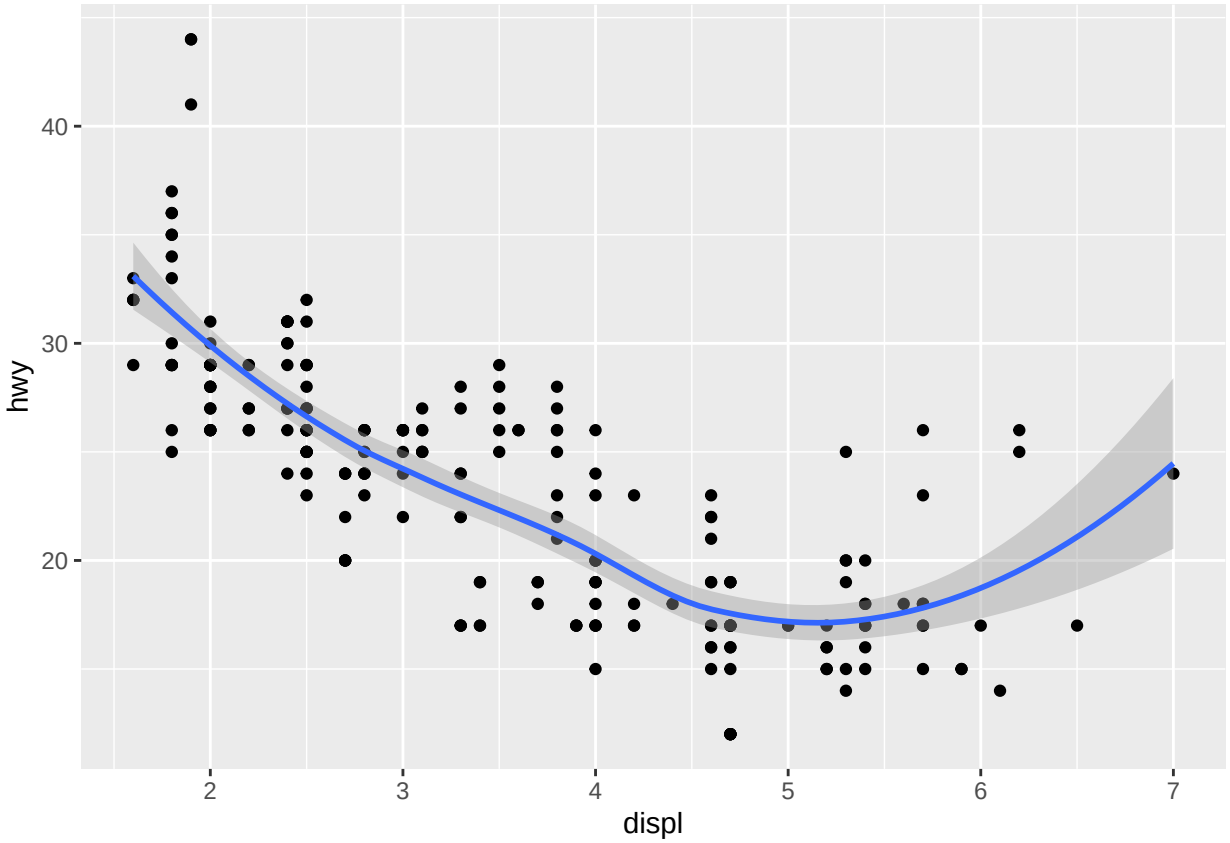

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

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



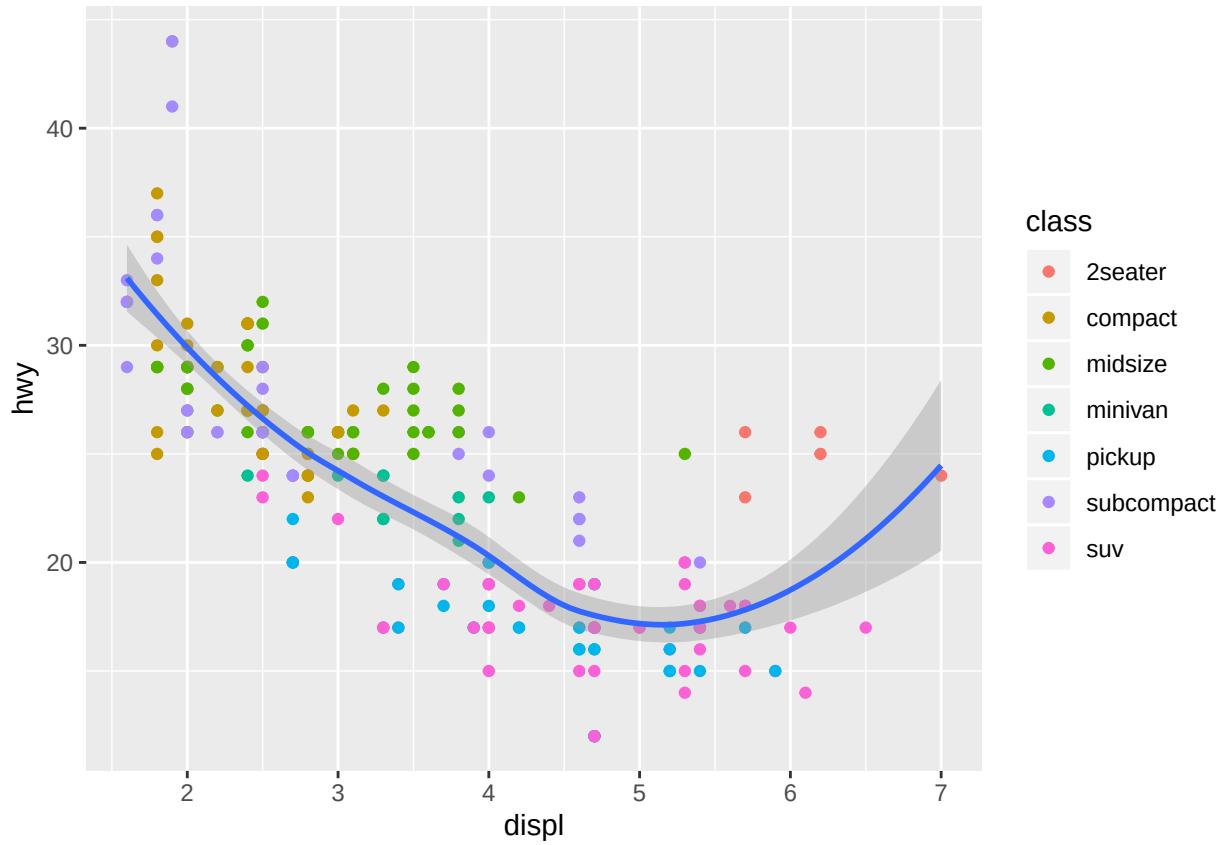

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

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



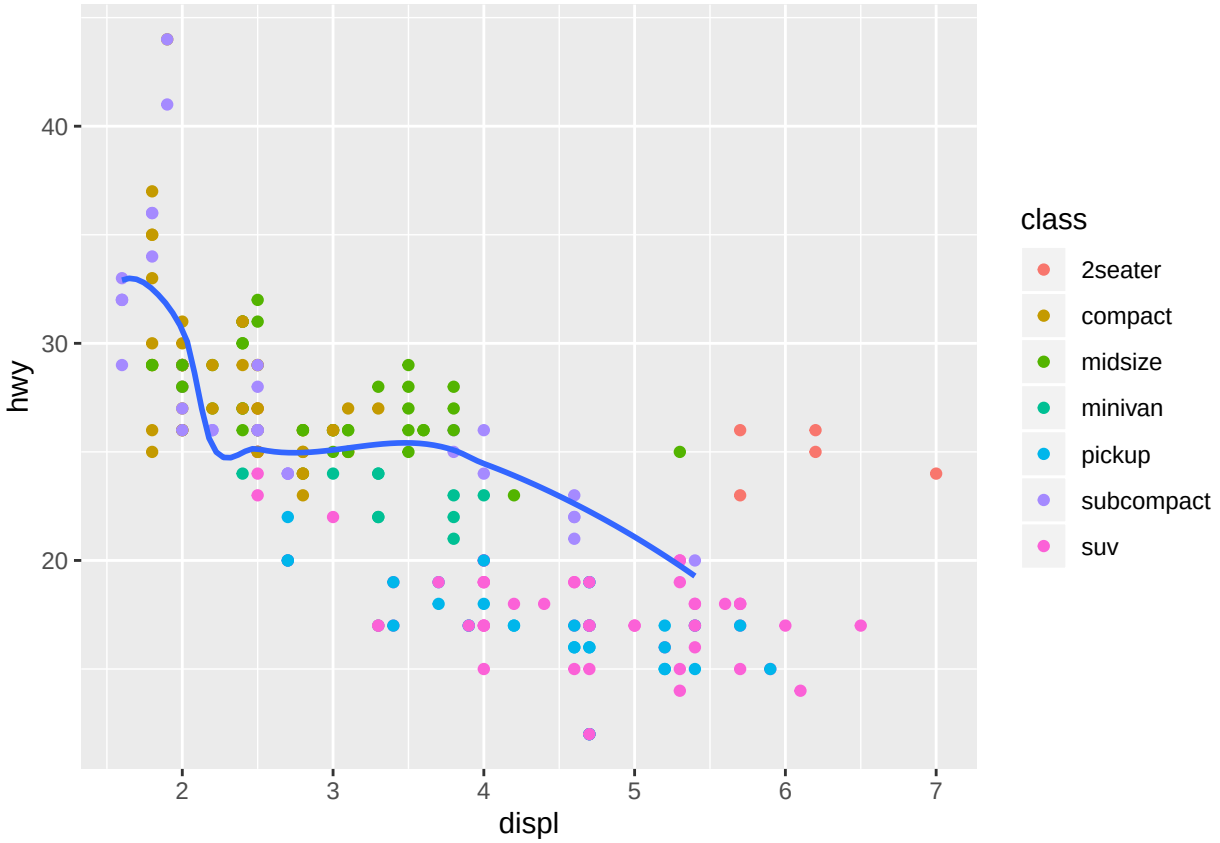

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth()
```

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




```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)
```

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

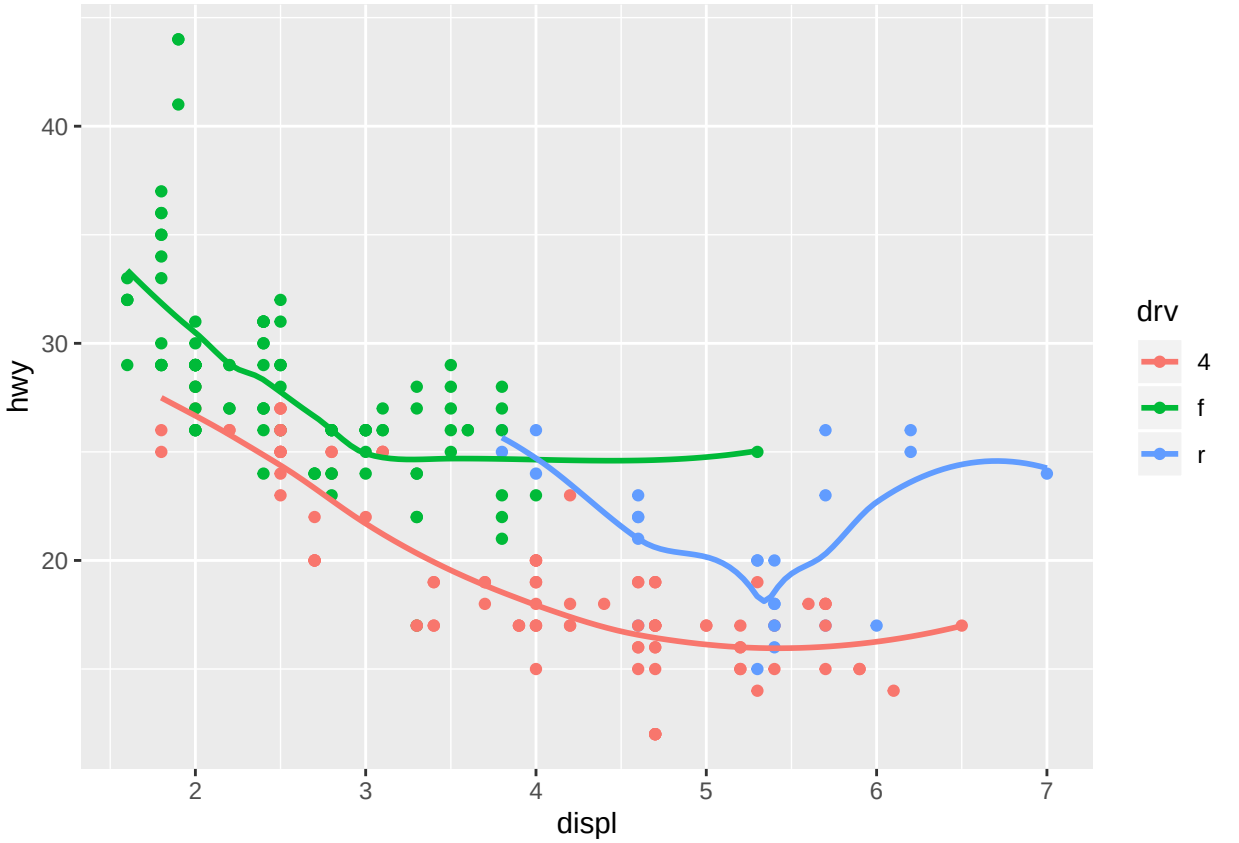


Exercises 1. What geom would you use to draw a line chart? A boxplot? A histogram? An area chart?

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.

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

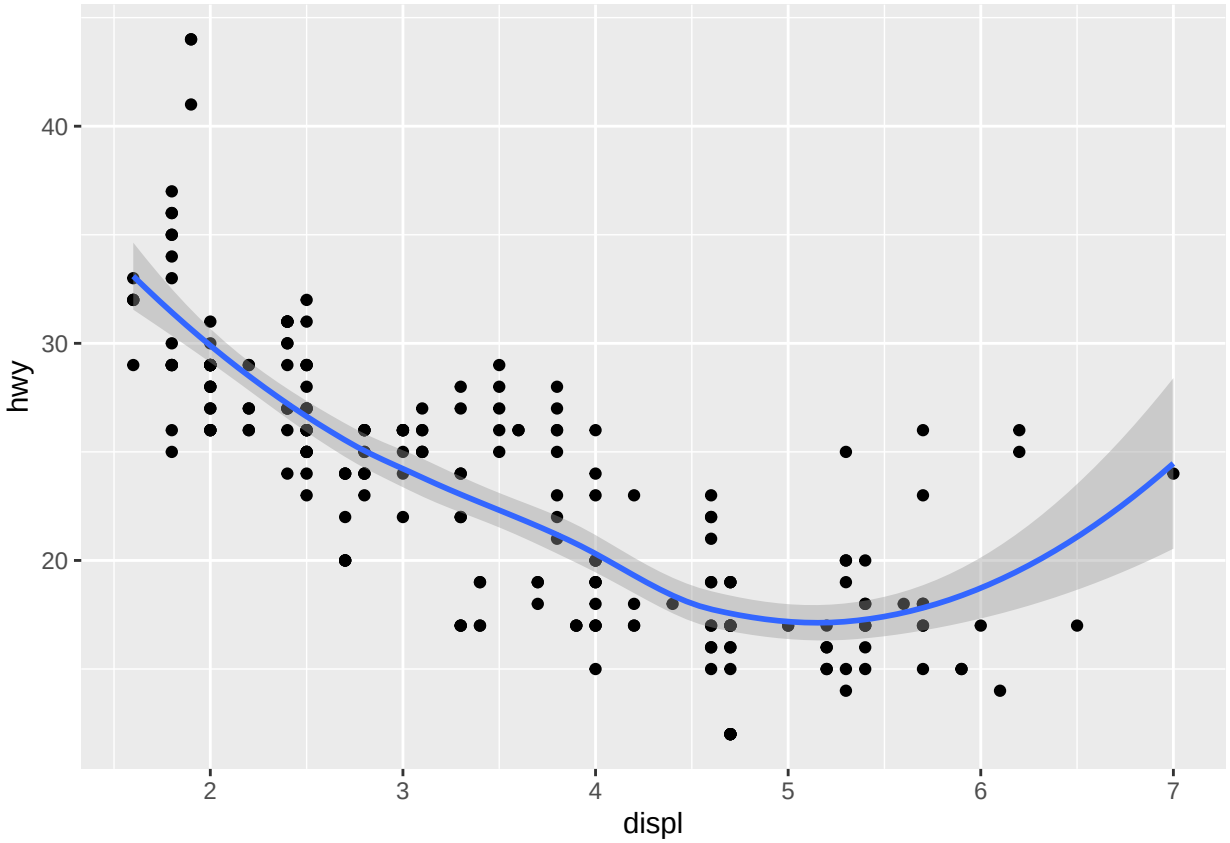
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



3. What does show.legend = FALSE do? What happens if you remove it? Why do you think I used it earlier in the chapter?
4. What does the se argument to geom_smooth() do?
5. Will these two graphs look different? Why/why not?
6. Recreate the R code necessary to generate the following graphs.

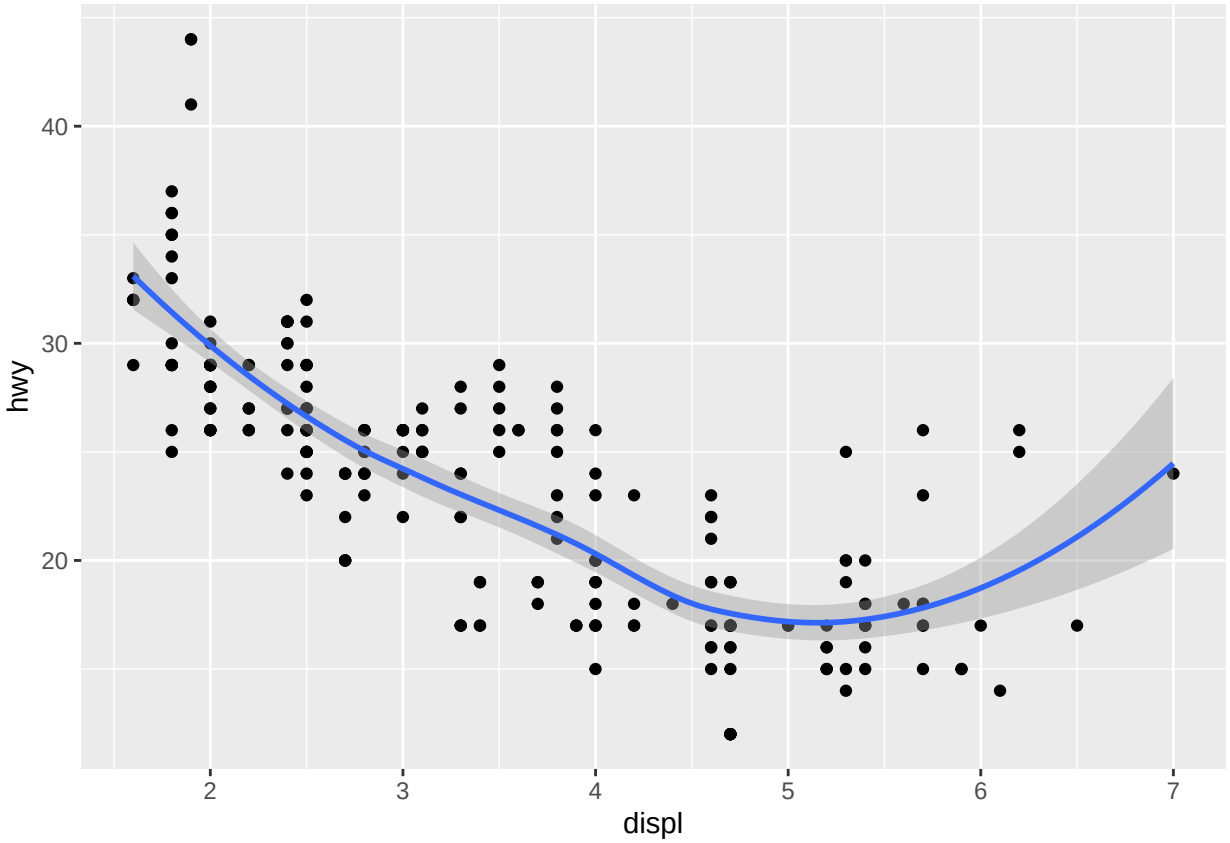
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

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



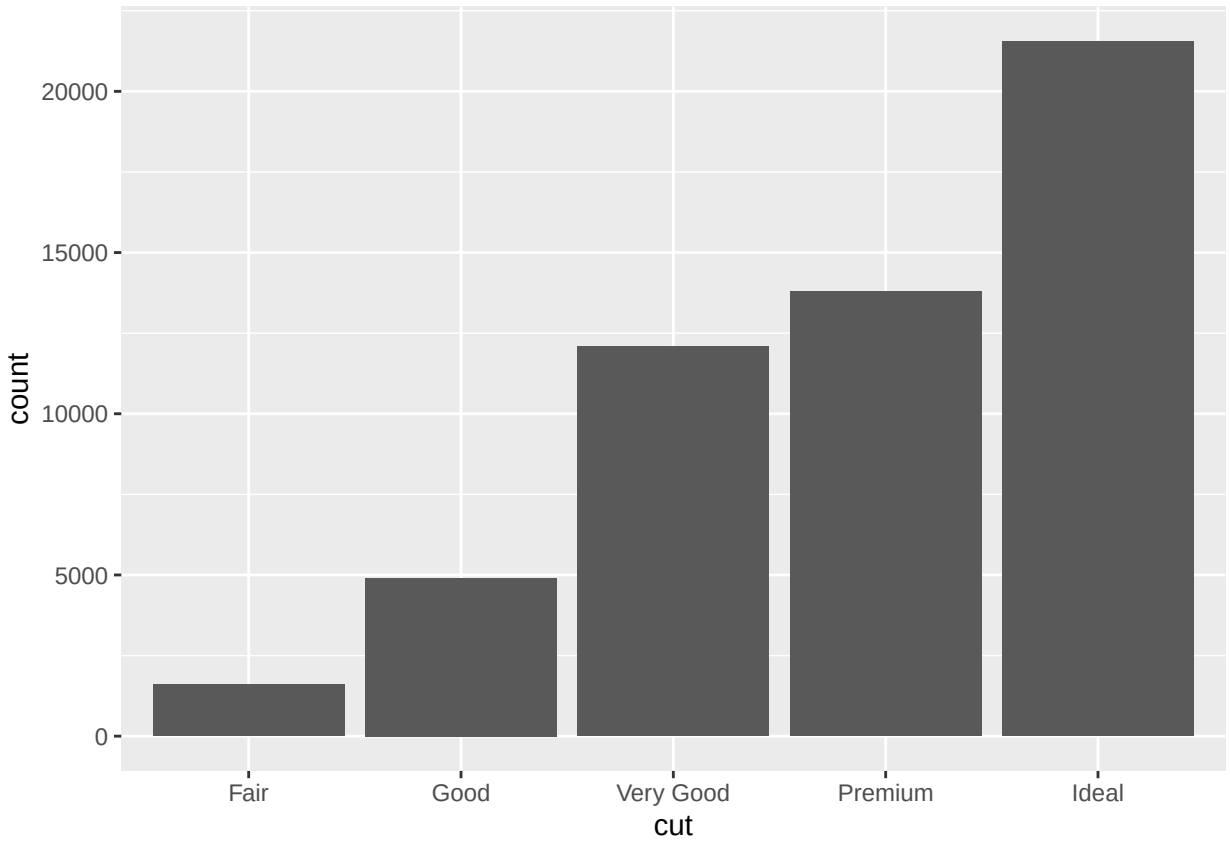

```
ggplot() +
  geom_point(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(data = mpg, mapping = aes(x = displ, y = hwy))
```

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

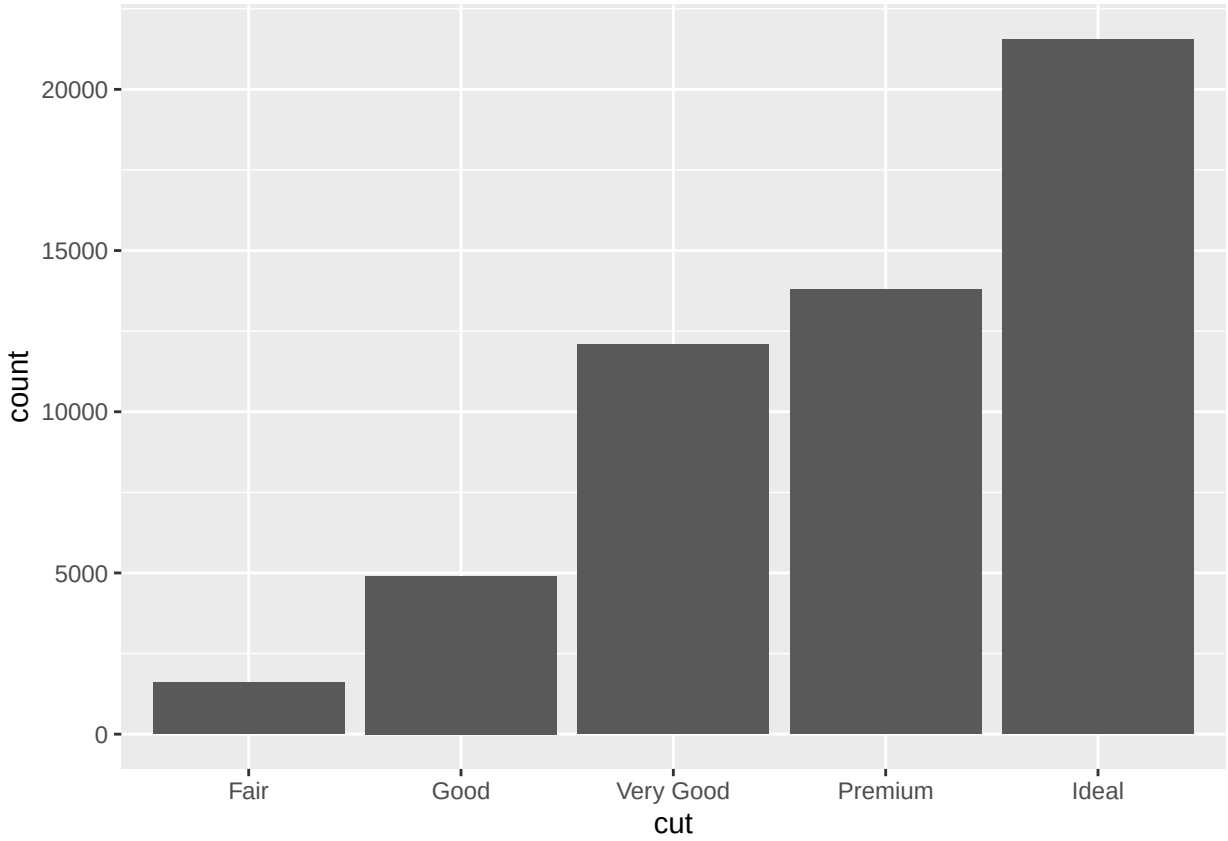


Statistical transformations

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



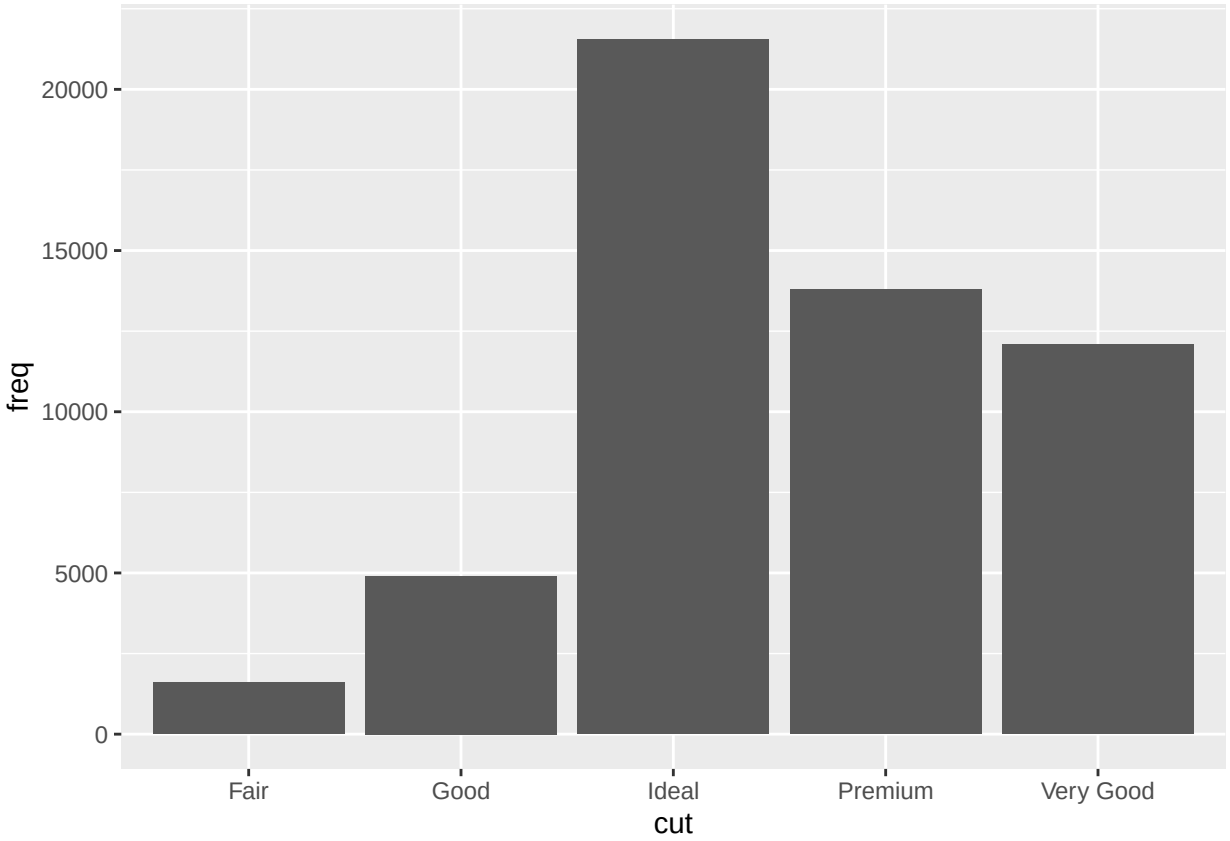

```
ggplot(data = diamonds) +
  stat_count(mapping = aes(x = cut))
```



1. You might want to override the default stat. In the code below, I change the stat of geom_bar() from count (the default) to identity. This lets me map the height of the bars to the raw values of a y variable. Unfortunately when people talk about bar charts casually, they might be referring to this type of bar chart, where the height of the bar is already present in the data, or the previous bar chart where the height of the bar is generated by counting rows.

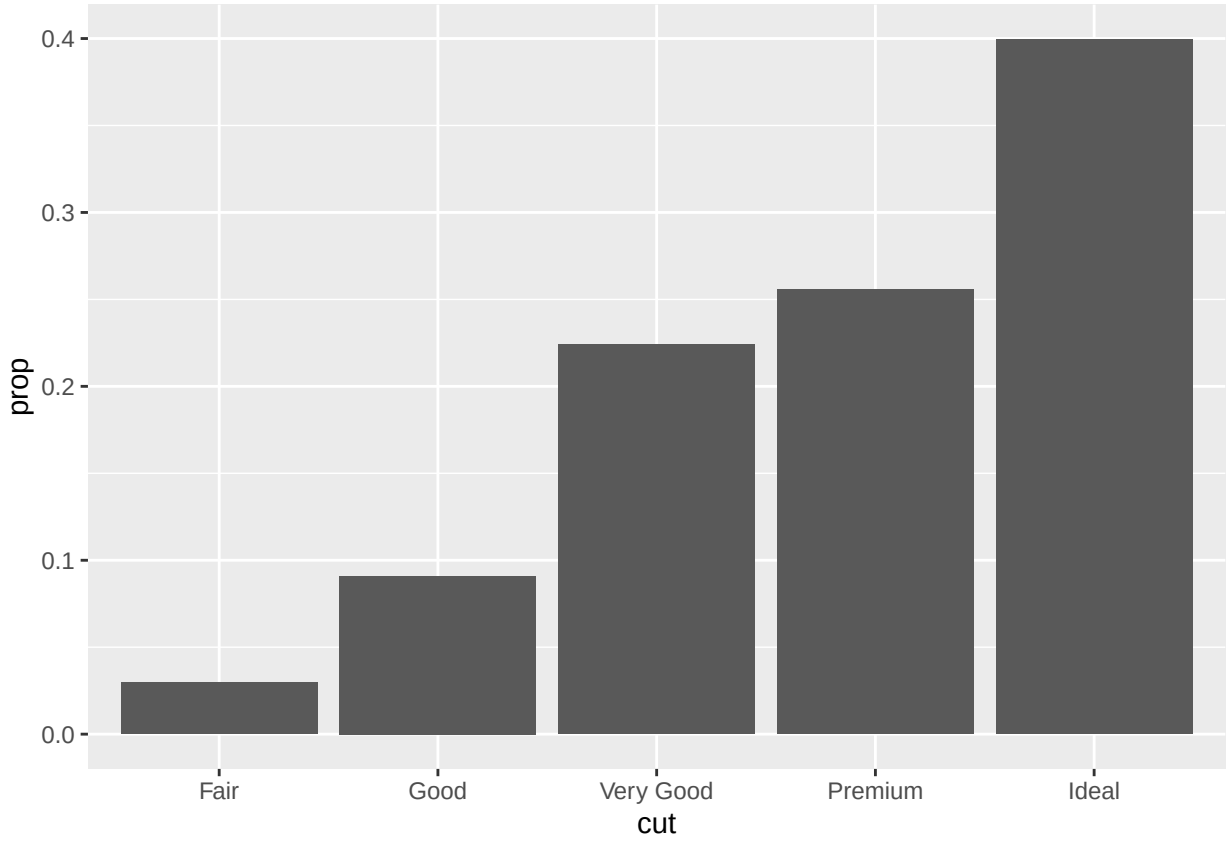
```
demo <- tribble(
  ~cut,          ~freq,
  "Fair",        1610,
  "Good",        4906,
  "Very Good",   12082,
  "Premium",     13791,
  "Ideal",       21551
)

ggplot(data = demo) +
  geom_bar(mapping = aes(x = cut, y = freq), stat = "identity")
```



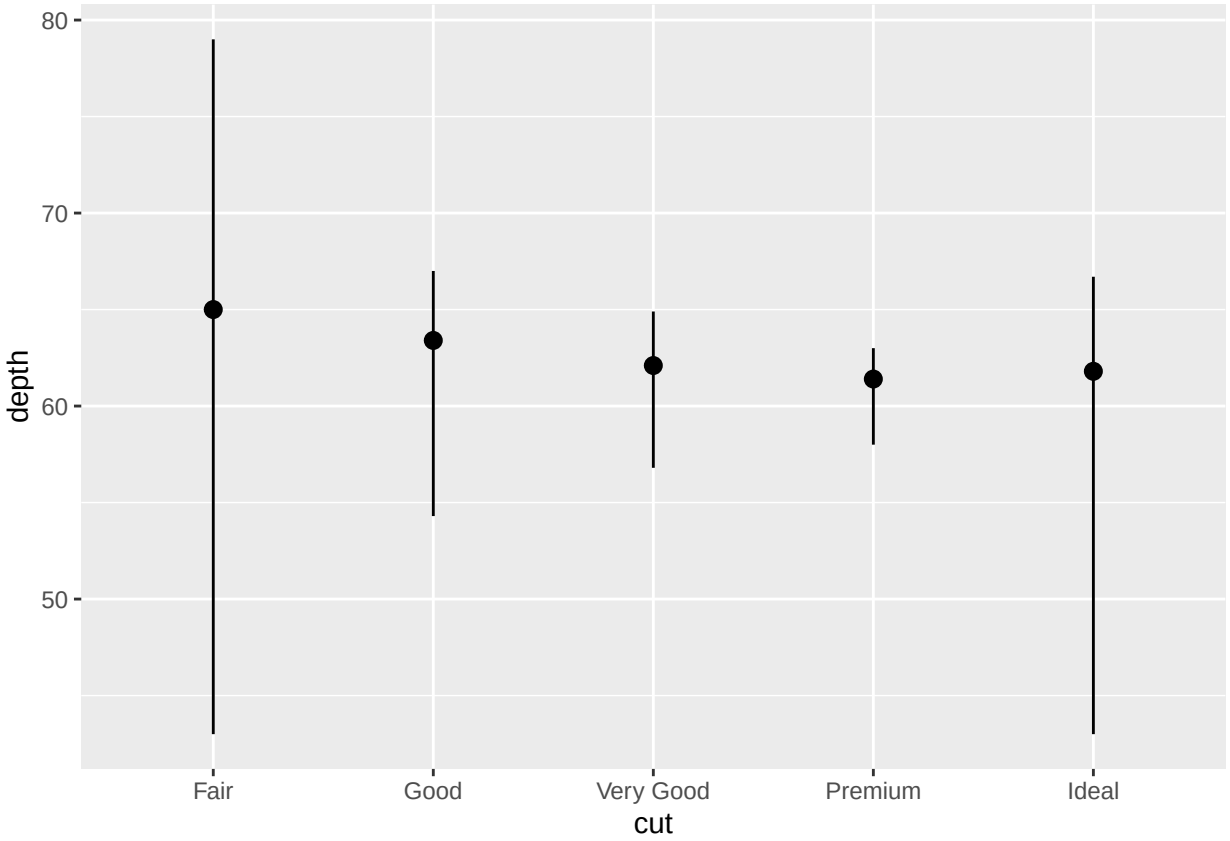
2. You might want to override the default mapping from transformed variables to aesthetics. For example, you might want to display a bar chart of proportion, rather than count:

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



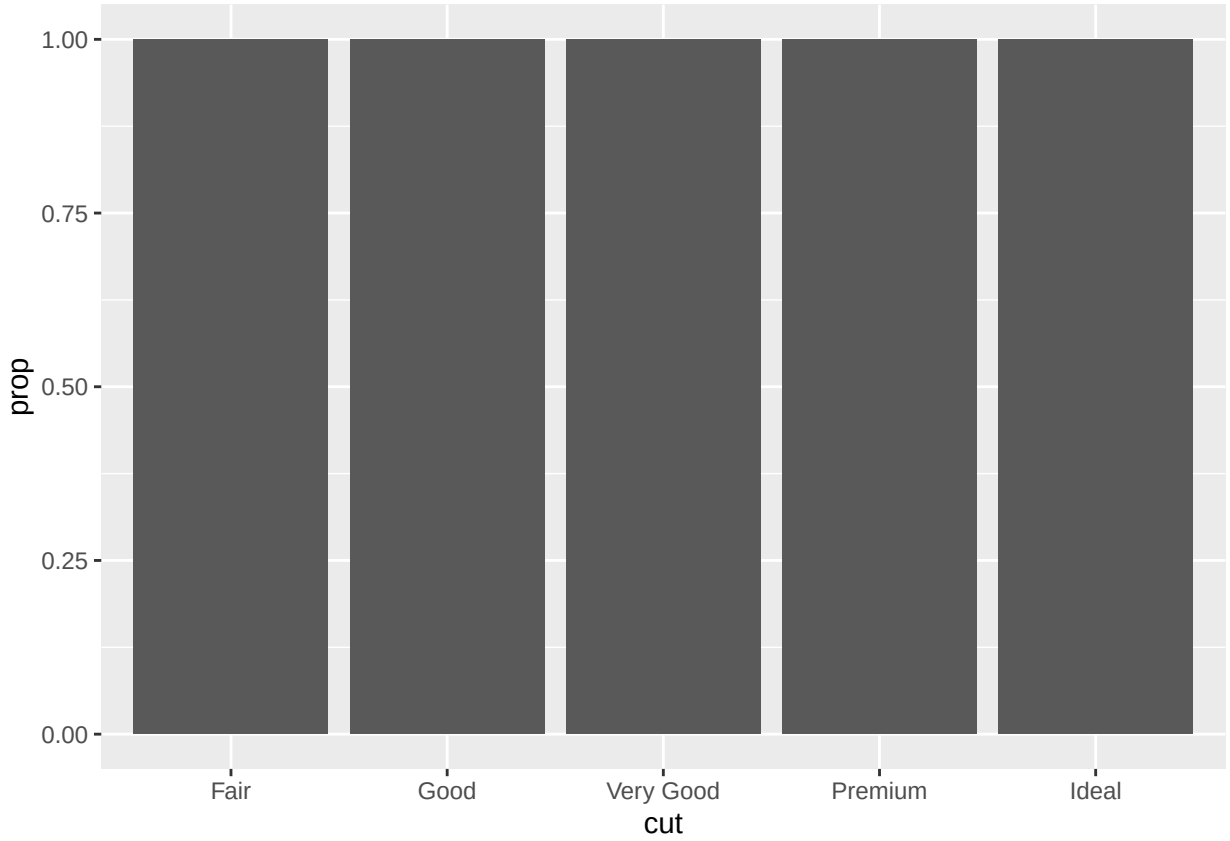
3. You might want to draw greater attention to the statistical transformation in your code. For example, you might use stat_summary(), which summarises the y values for each unique x value, to draw attention to the summary that you're computing:

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

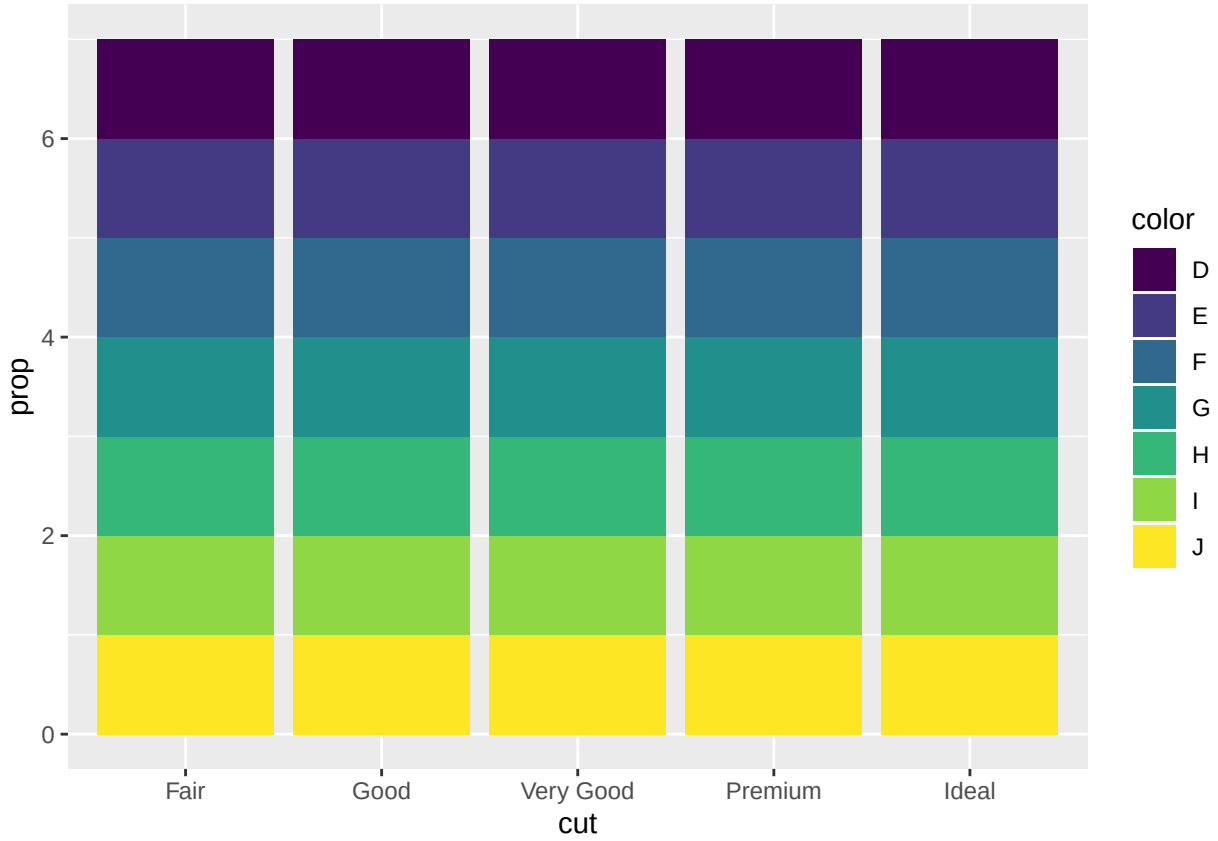


Exercises

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



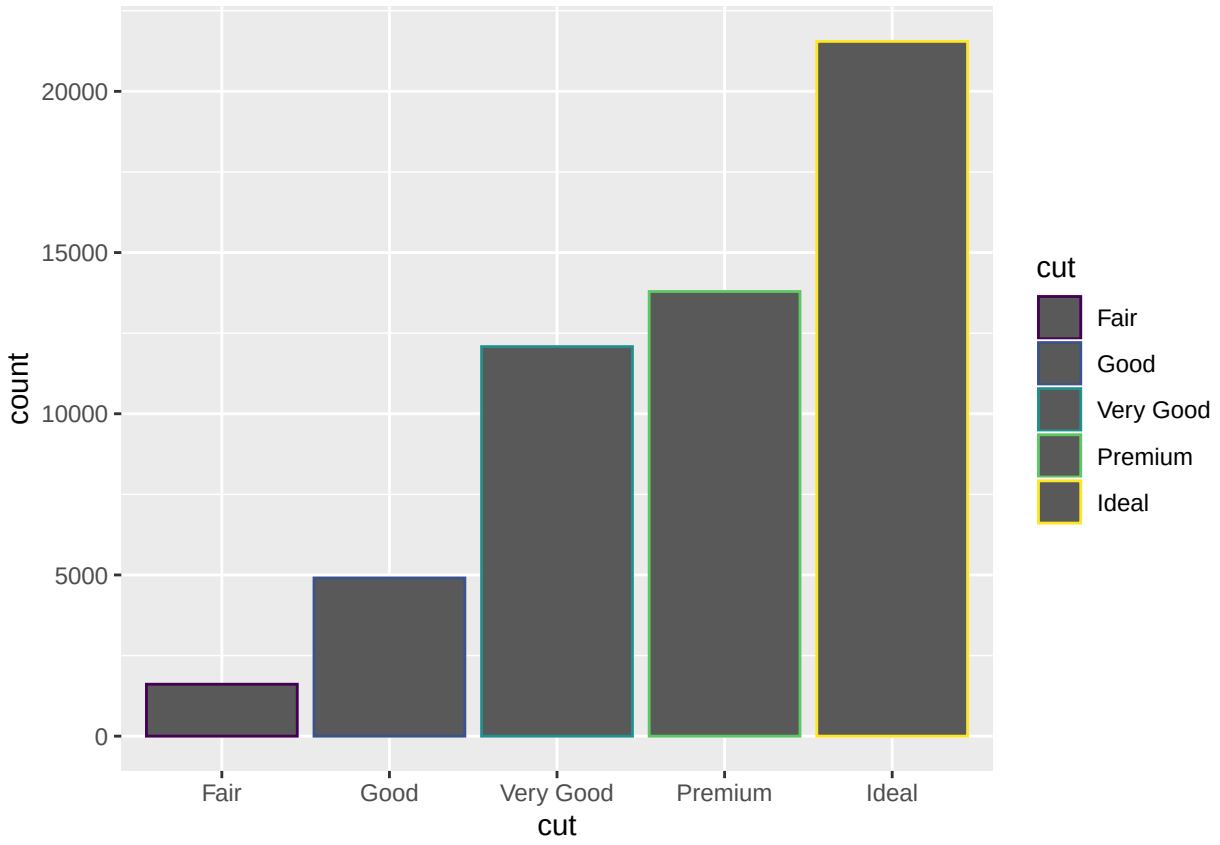

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop..))
```



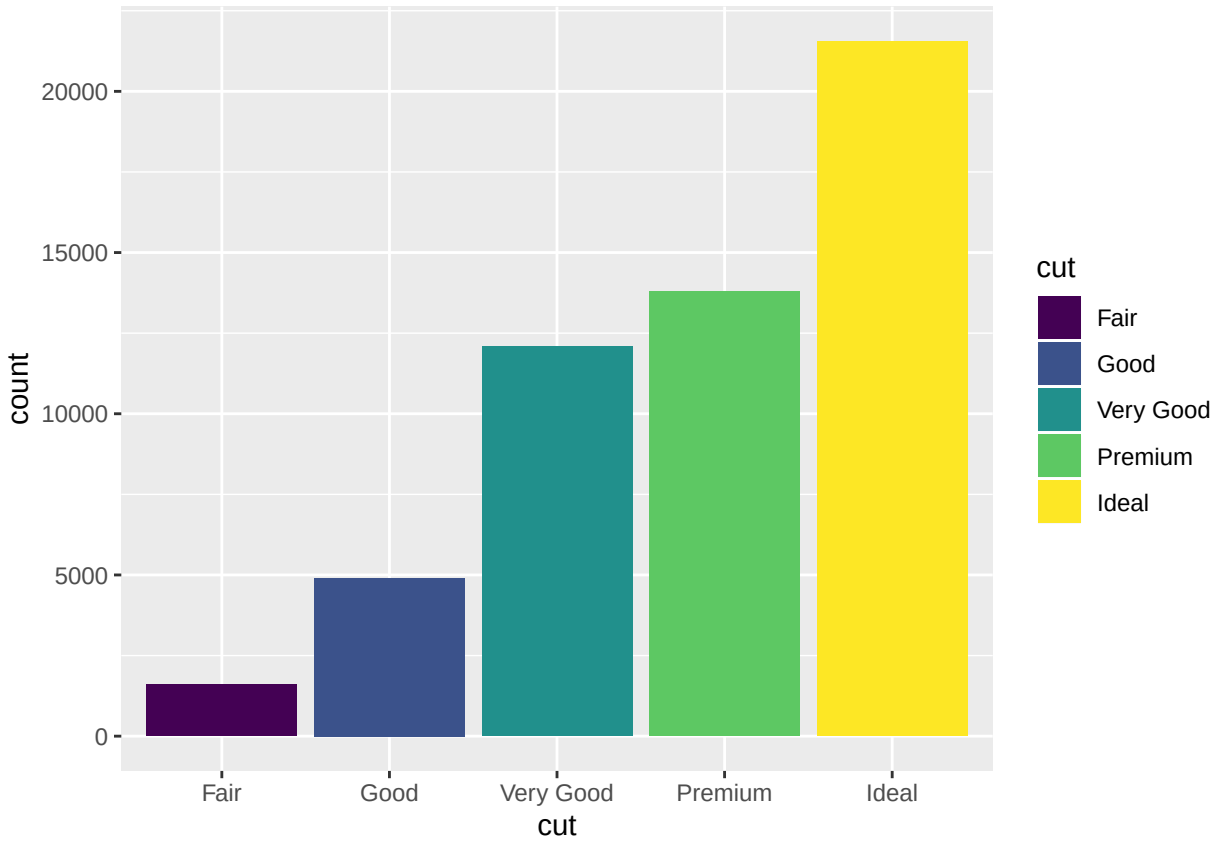
Position adjustments

Diffrent possible postion are identity, fill, dodge, jitter.

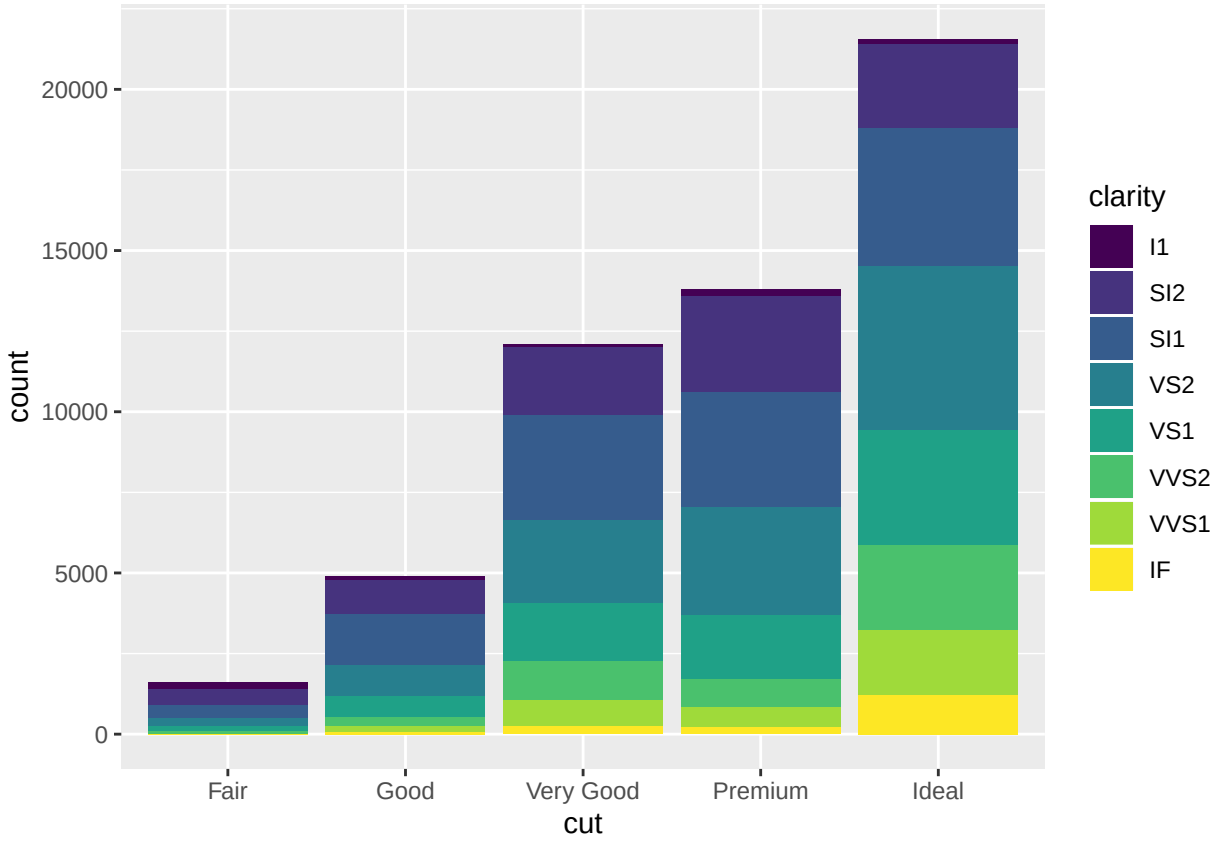
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour = cut))
```




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

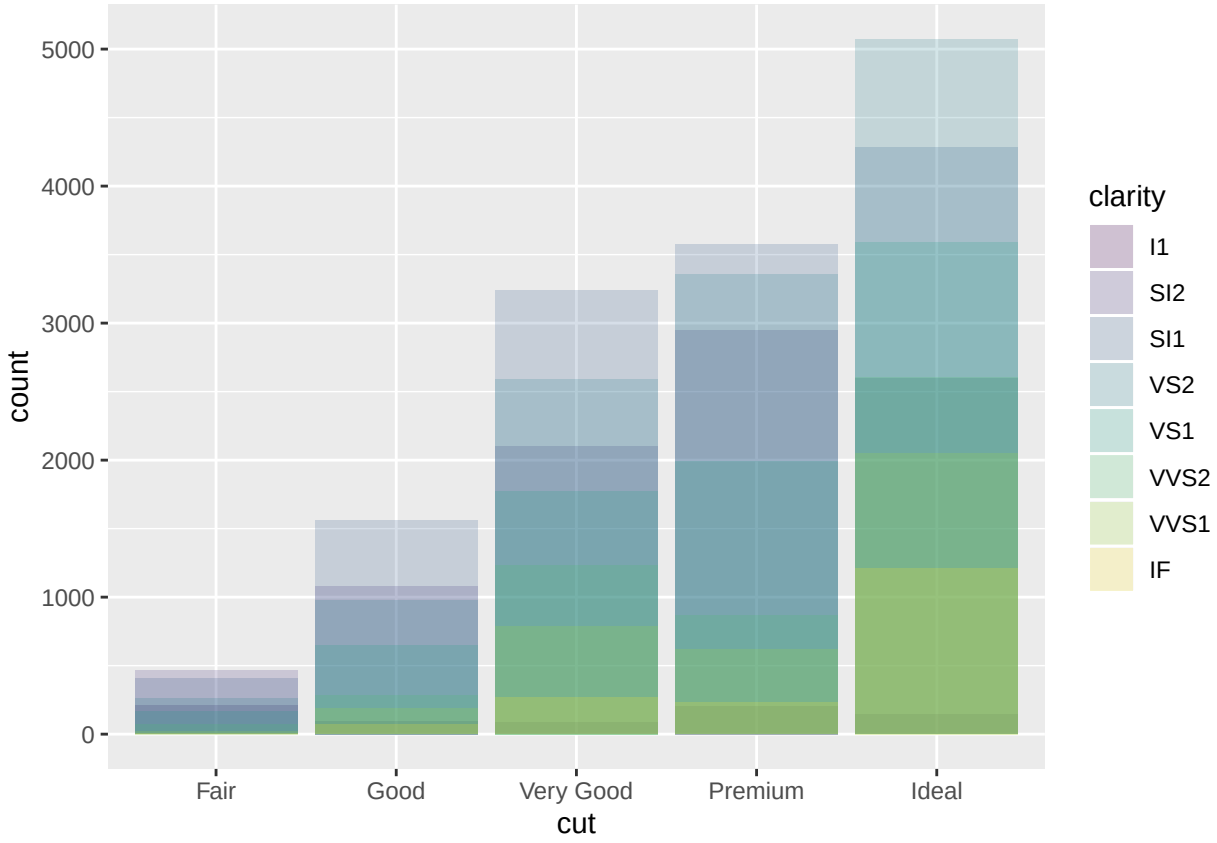



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```

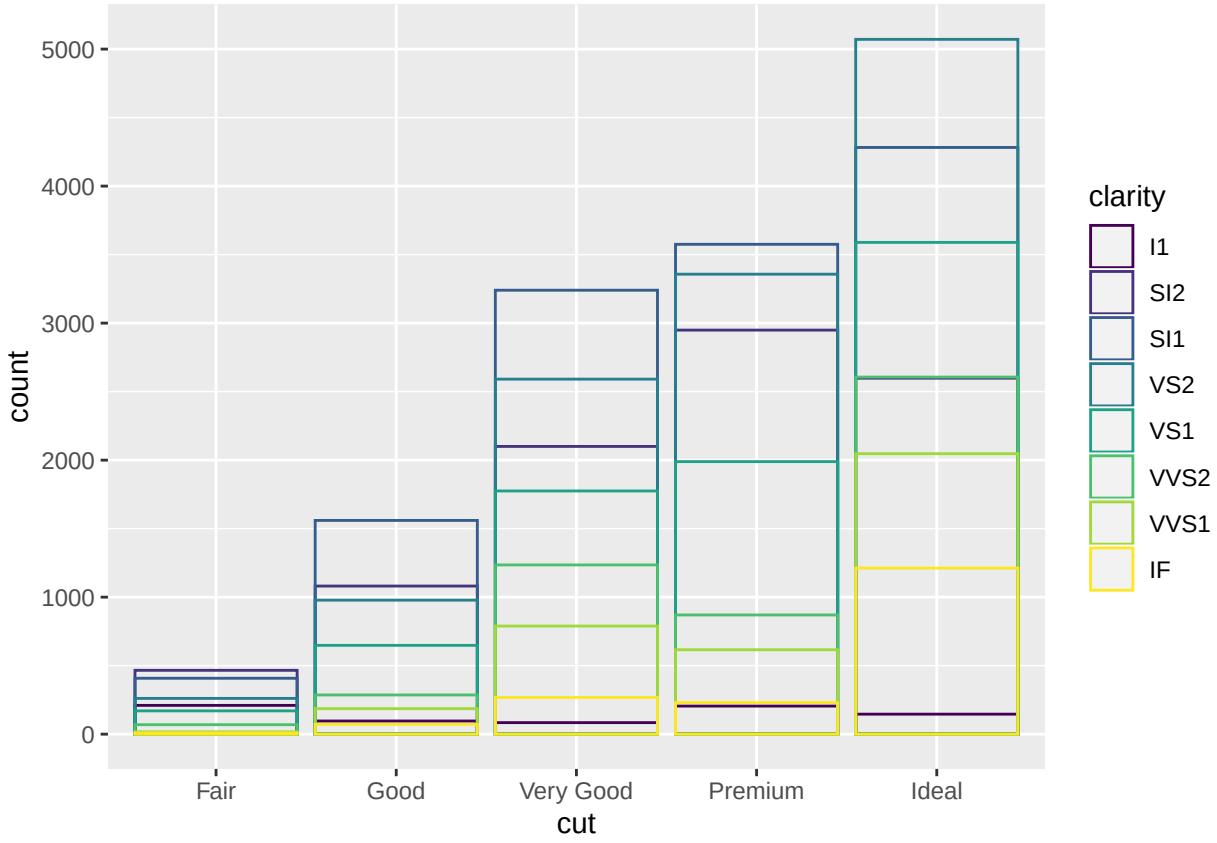


Example with position as identity.

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

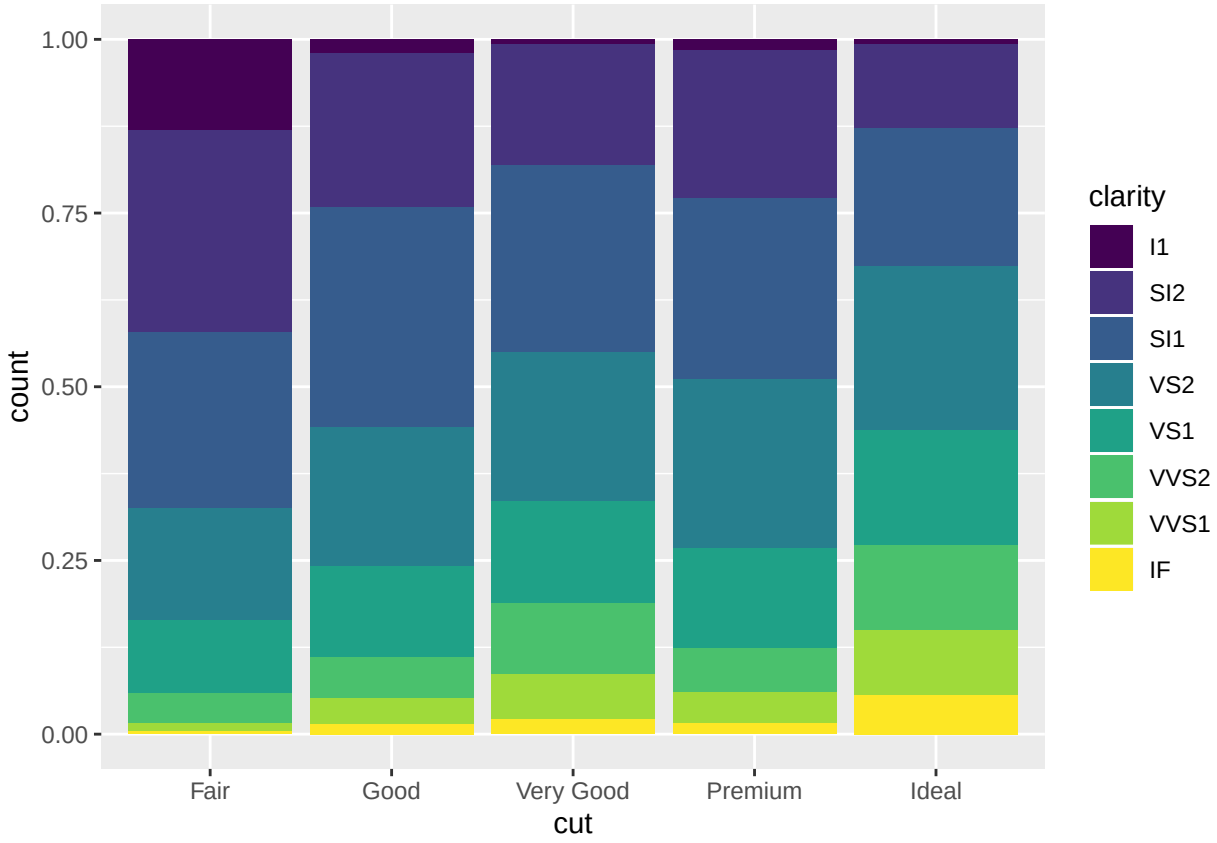



```
ggplot(data = diamonds, mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity")
```



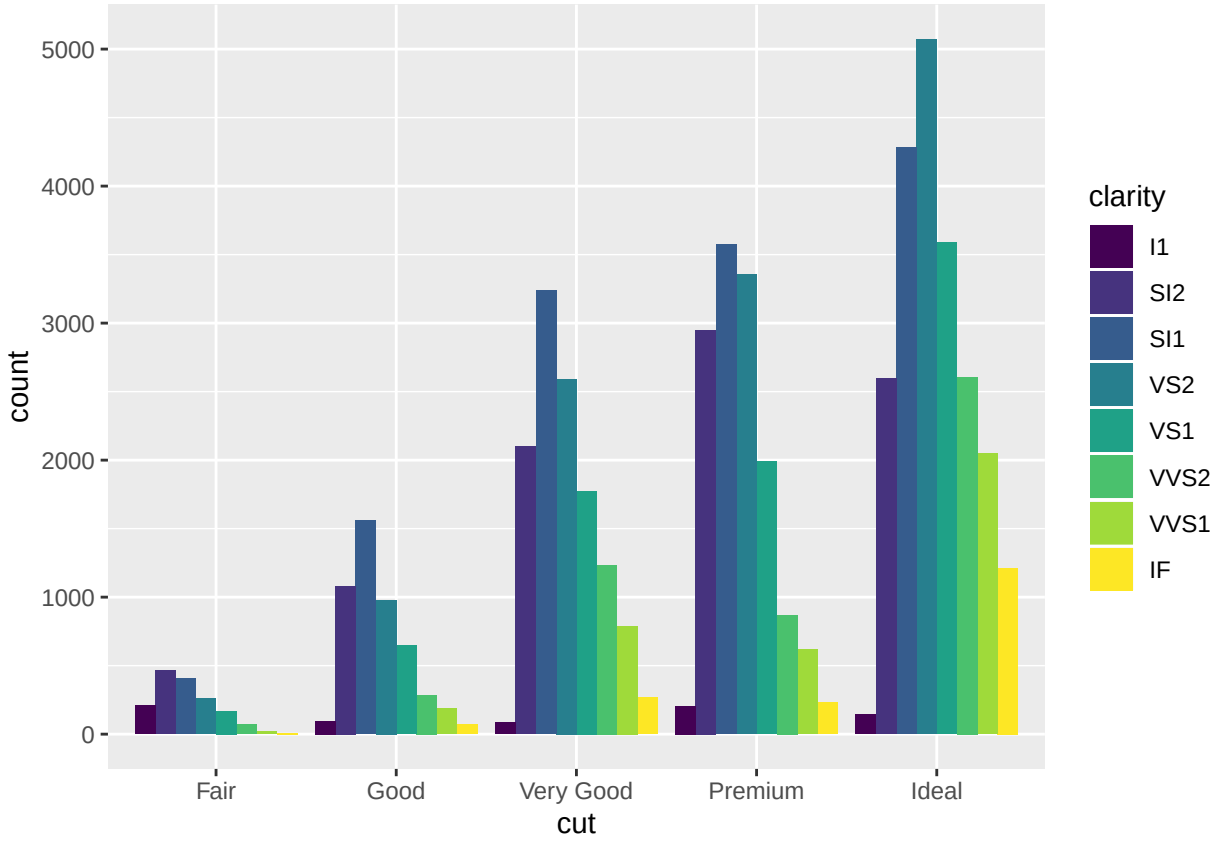
Example with position as fill.

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



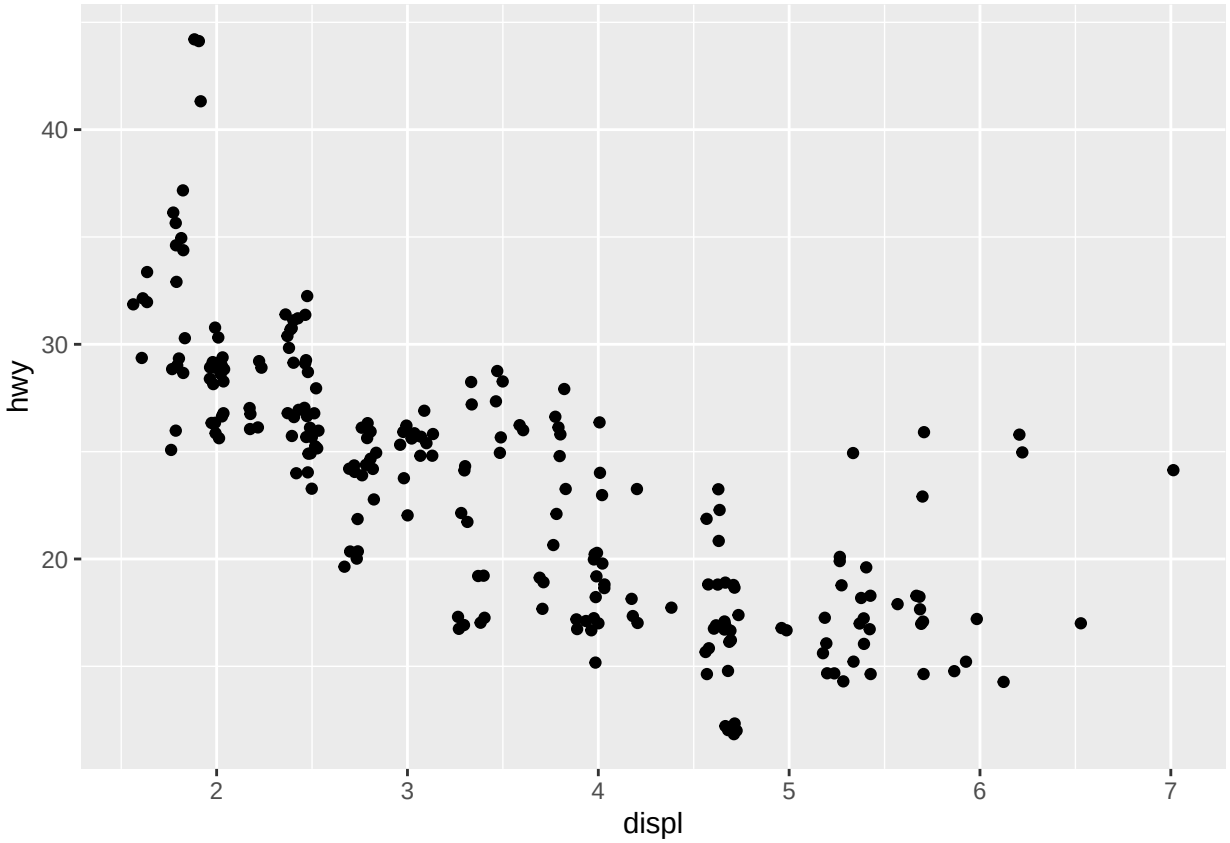
Example with position as dodge.

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



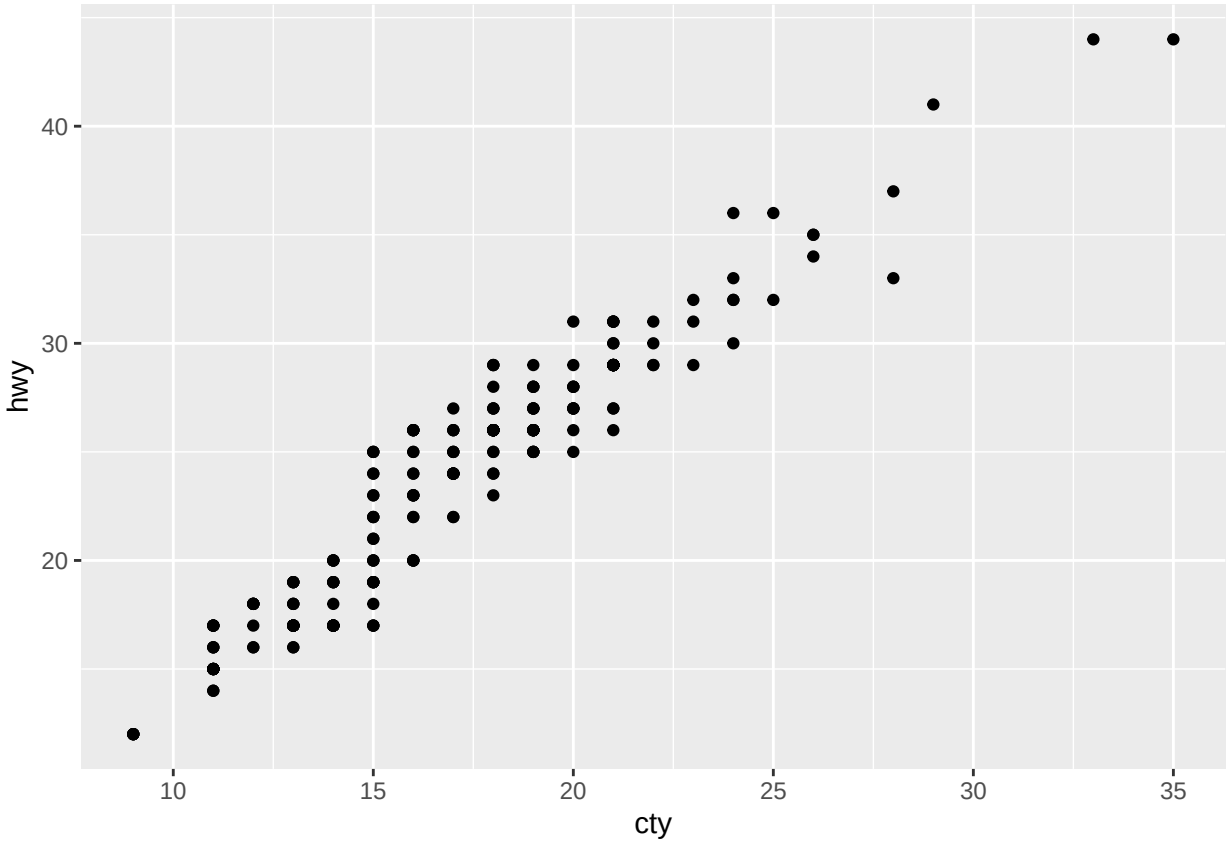
Example with position as jittery.

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



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

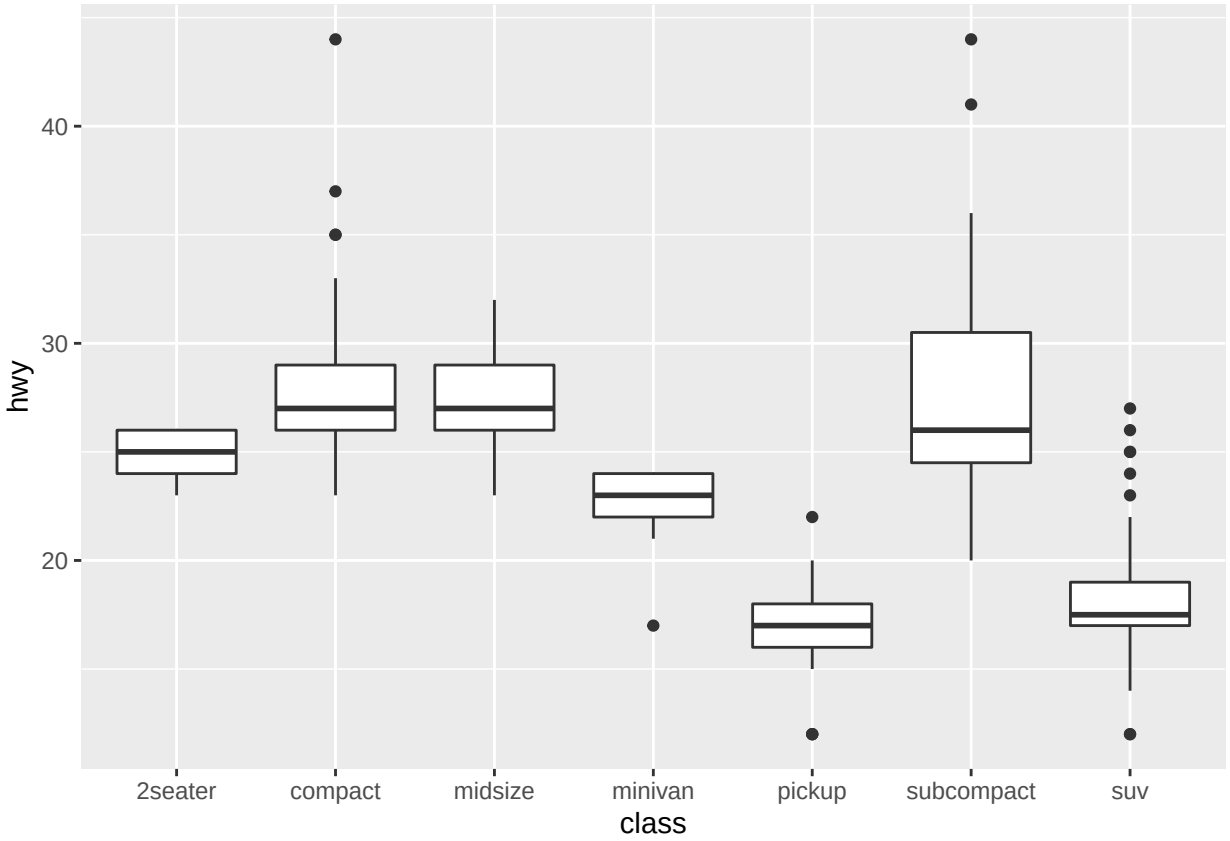
```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point()
```



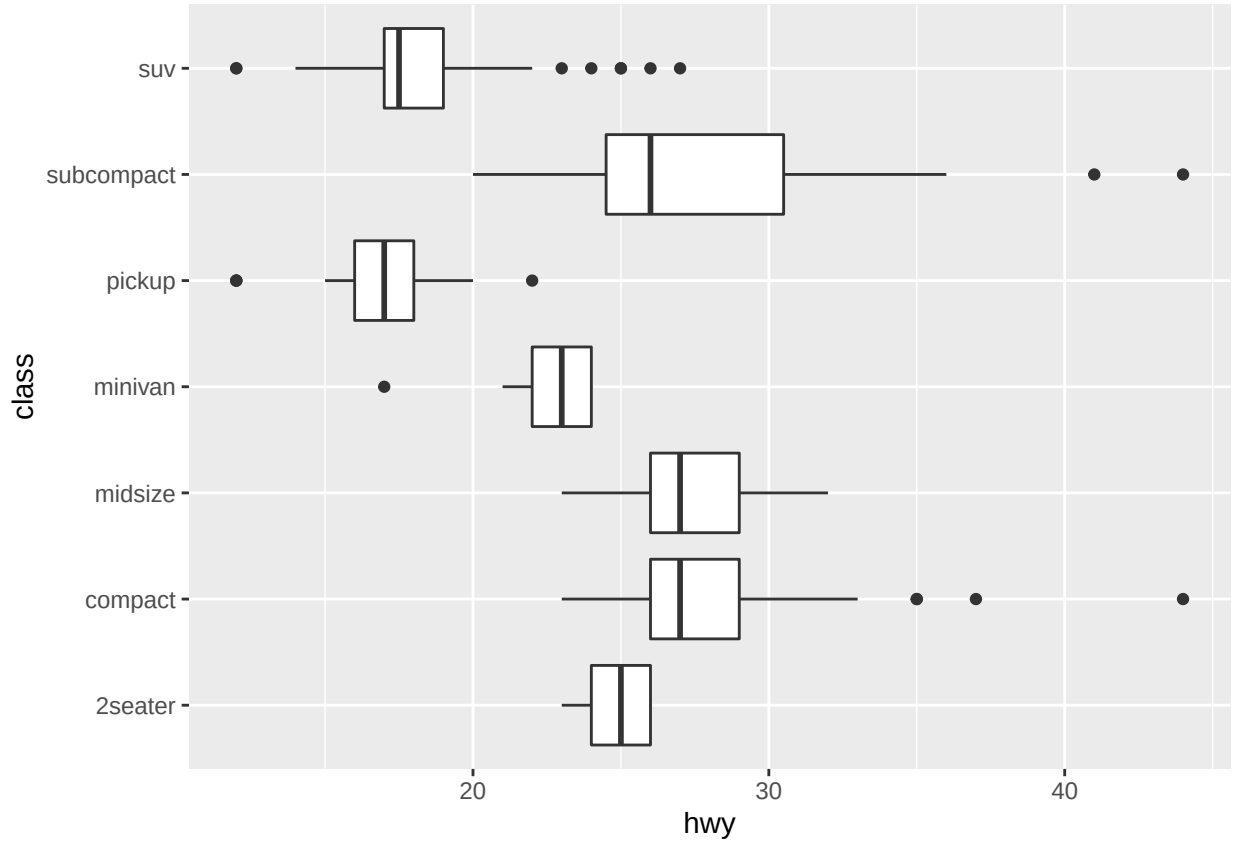
Coordinate systems

Default coordinate system used is the Cartesian coordinate system. Cartesian coordinate system x and y positions act independently to determine the location of each point. Other coordinate system which are helpful are coord_flip(), coord_quickmap(), coord_polar(). Example using coord_flip() coordinate system.

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




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



Example using coord_quickmap() coordinate system.

```
library(maps)
```

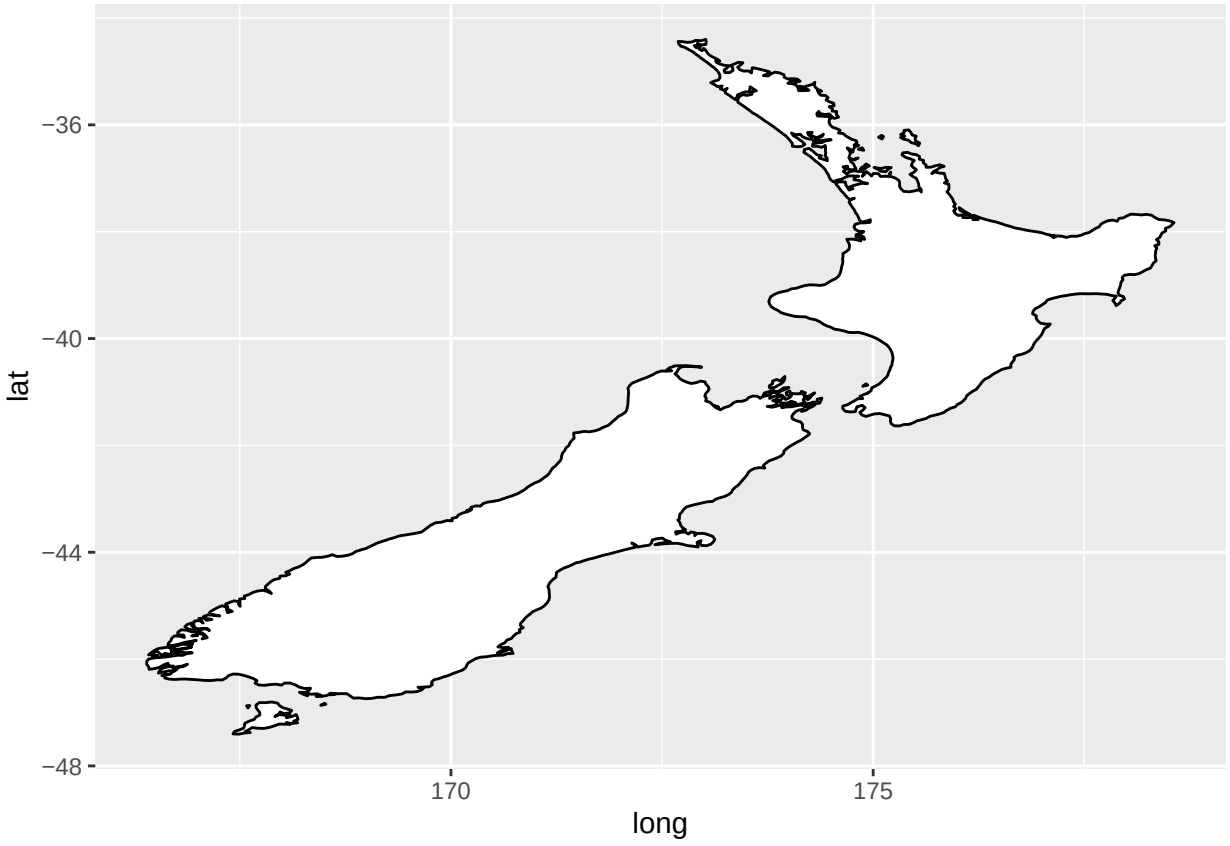
```
## Warning: package 'maps' was built under R version 3.5.3
```

```
##
## Attaching package: 'maps'
```

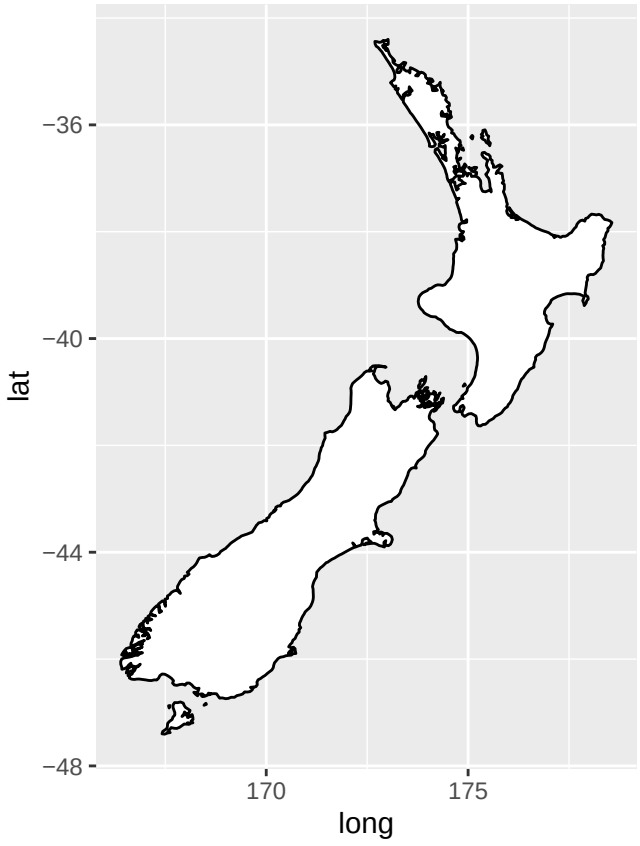
```
## The following object is masked from 'package:purrr':
##
##     map
```

```
nz <- map_data("nz")

ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black")
```



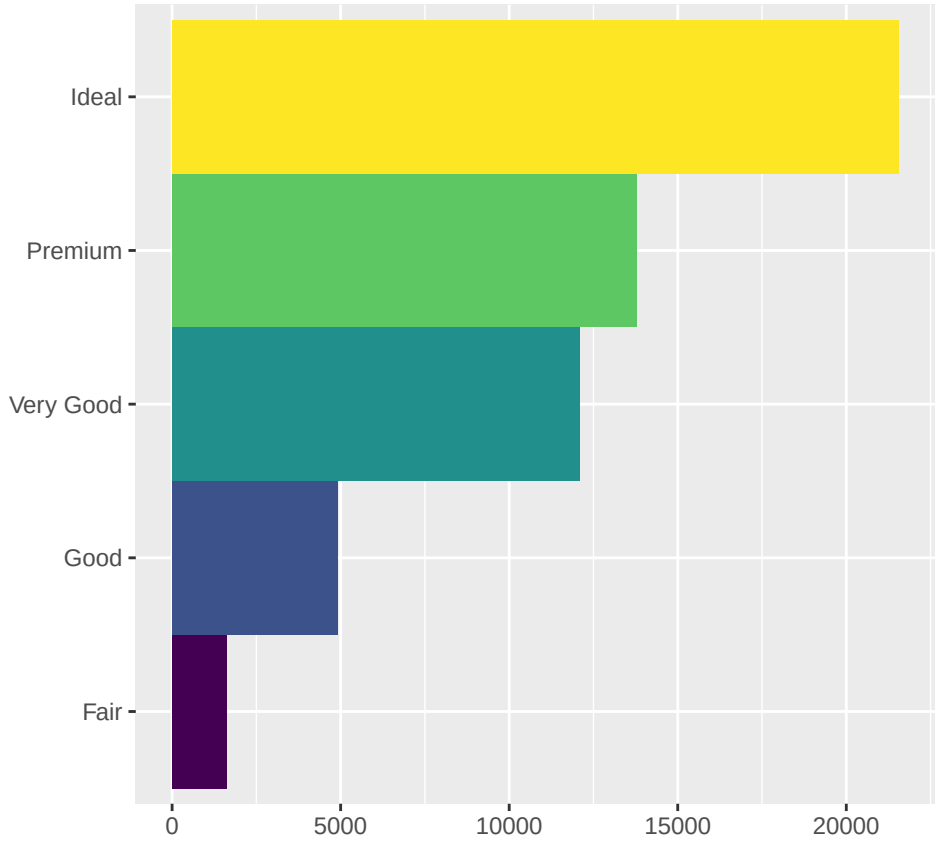

```
ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap()
```



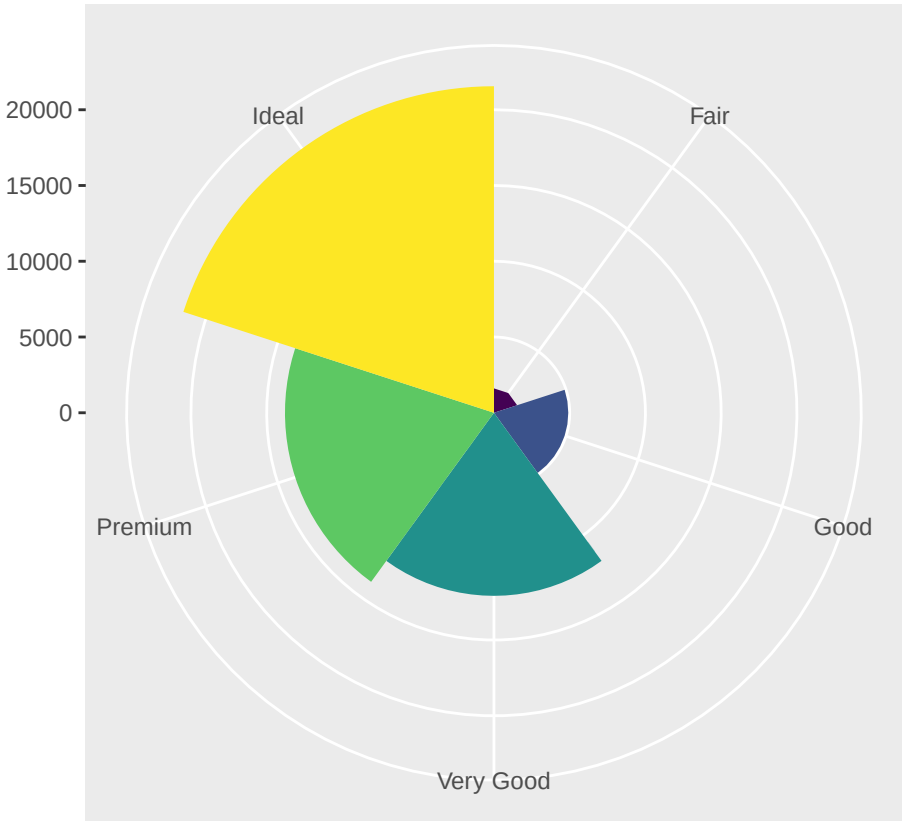
Example using coord_polar() coordinate system.

```
bar <- ggplot(data = diamonds) +
  geom_bar(
    mapping = aes(x = cut, fill = cut),
    show.legend = FALSE,
    width = 1
  ) +
  theme(aspect.ratio = 1) +
  labs(x = NULL, y = NULL)

bar + coord_flip()
```

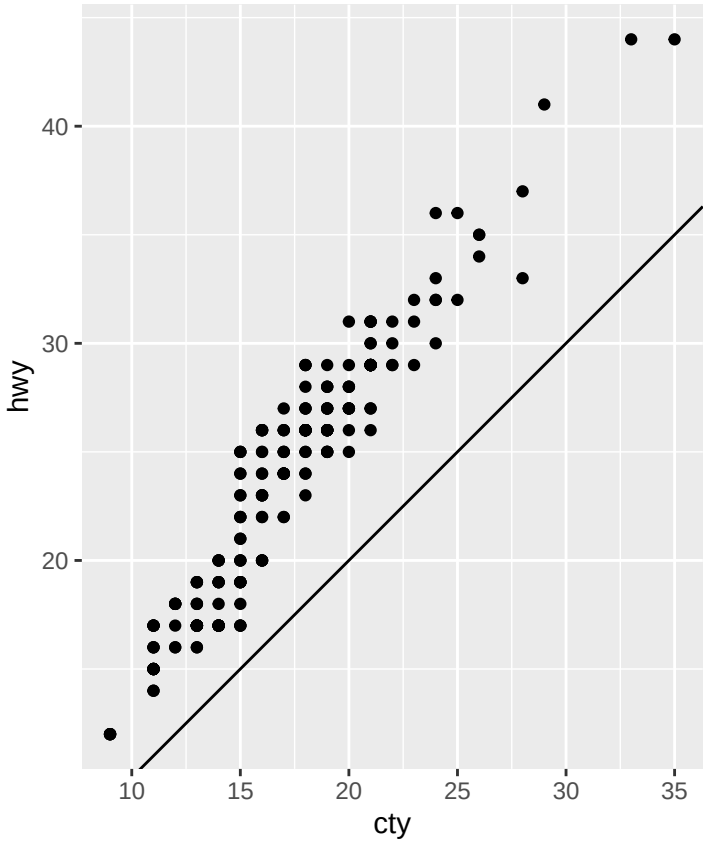



```
bar + coord_polar()
```



Exercises

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



The layered grammar of graphics

New template takes seven parameters, the bracketed words that appear in the template

ggplot(data =) + (mapping = aes(), stat = , position =) + +