Week 2: R for Data Science Ch 1, 2, and 3

Daniel Lee

2022-08-30

Welcome

Ch1 Introduction

The data science project workflow

Prerequisites

- R
- RStudio
- r packages

Install the tidyverse package

```
install.packages("tidyverse")
library(tidyverse)
```

Running R code

```
1+2
```

```
## [1] 3
```

Getting help

- Google
- Stackoverflow

Ch2 Introduction to Data Exploration

Ch3 Data Visualization

Set up

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

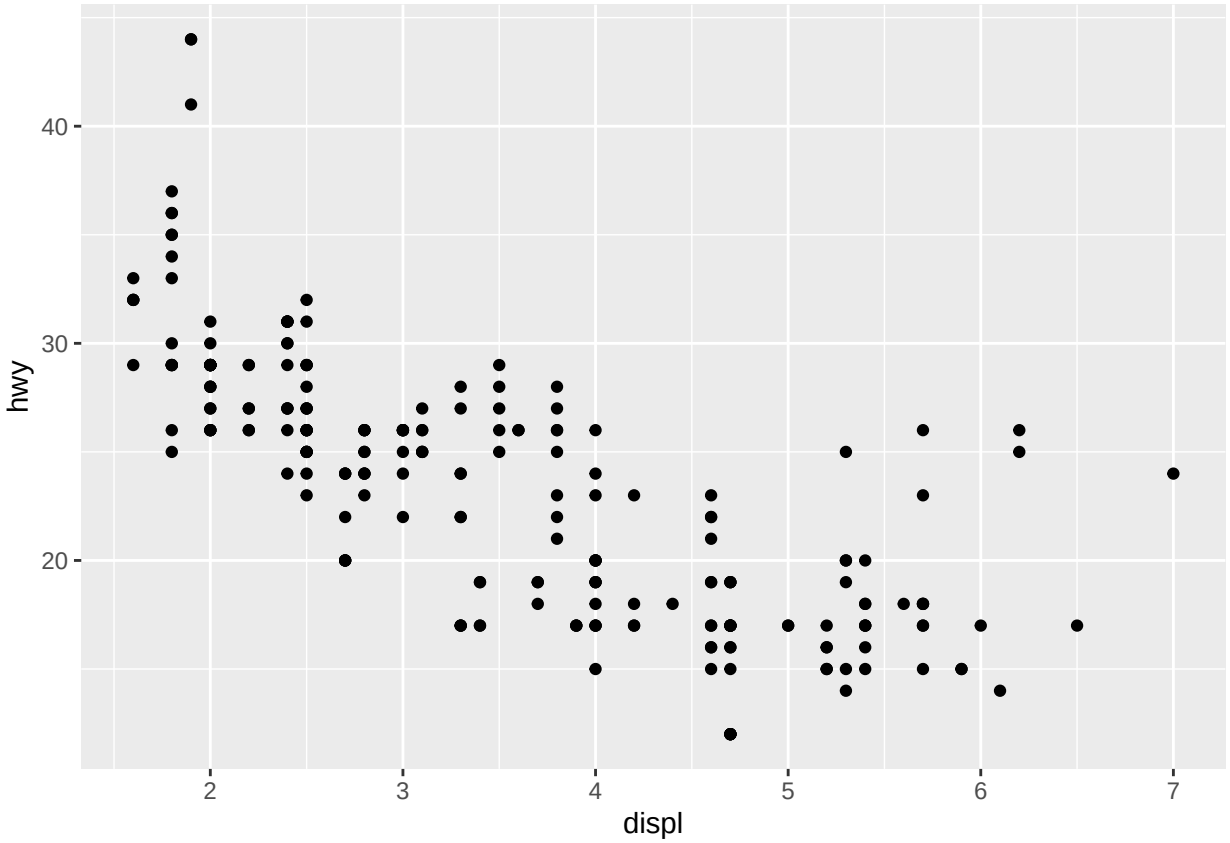
data

```
mpg
```

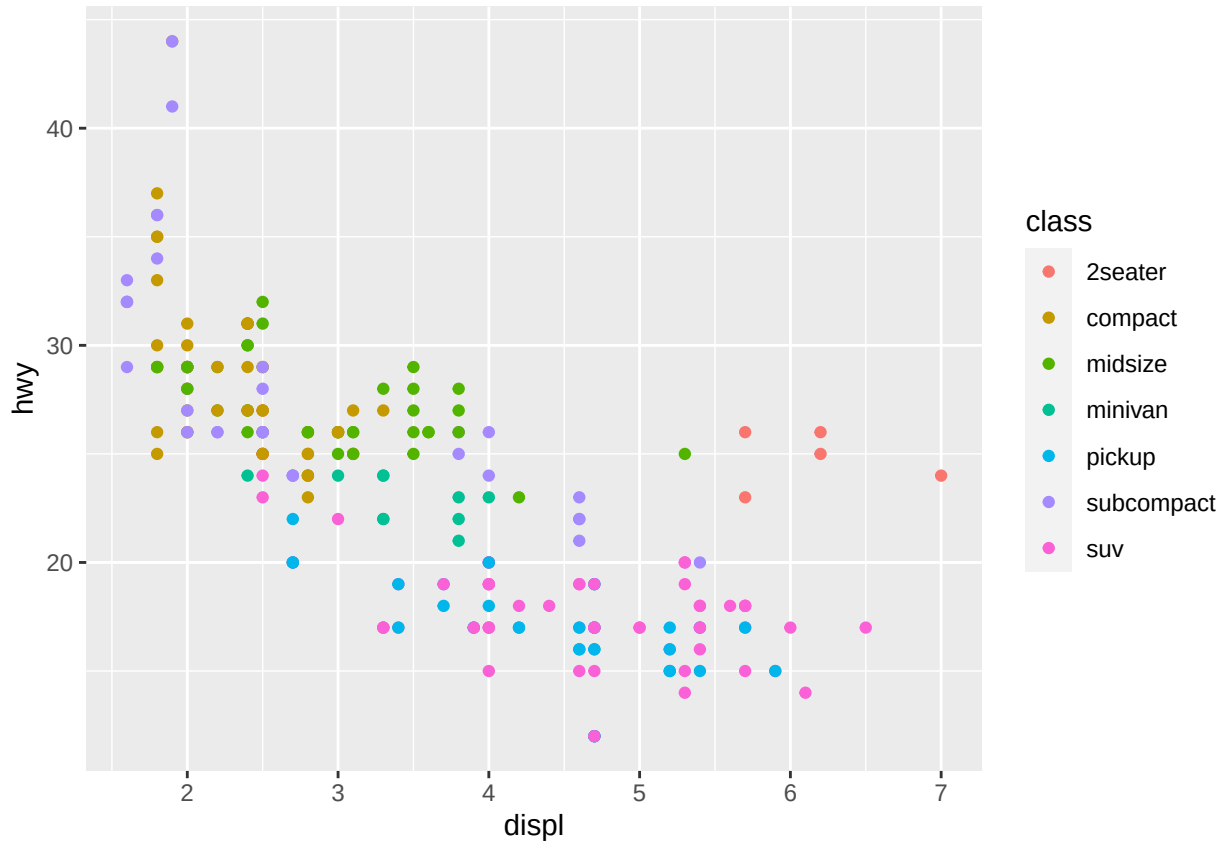
```
## # A tibble: 234 x 11
##    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
##  2 audi         a4           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 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
## # ... with 224 more rows
```

aesthetics

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

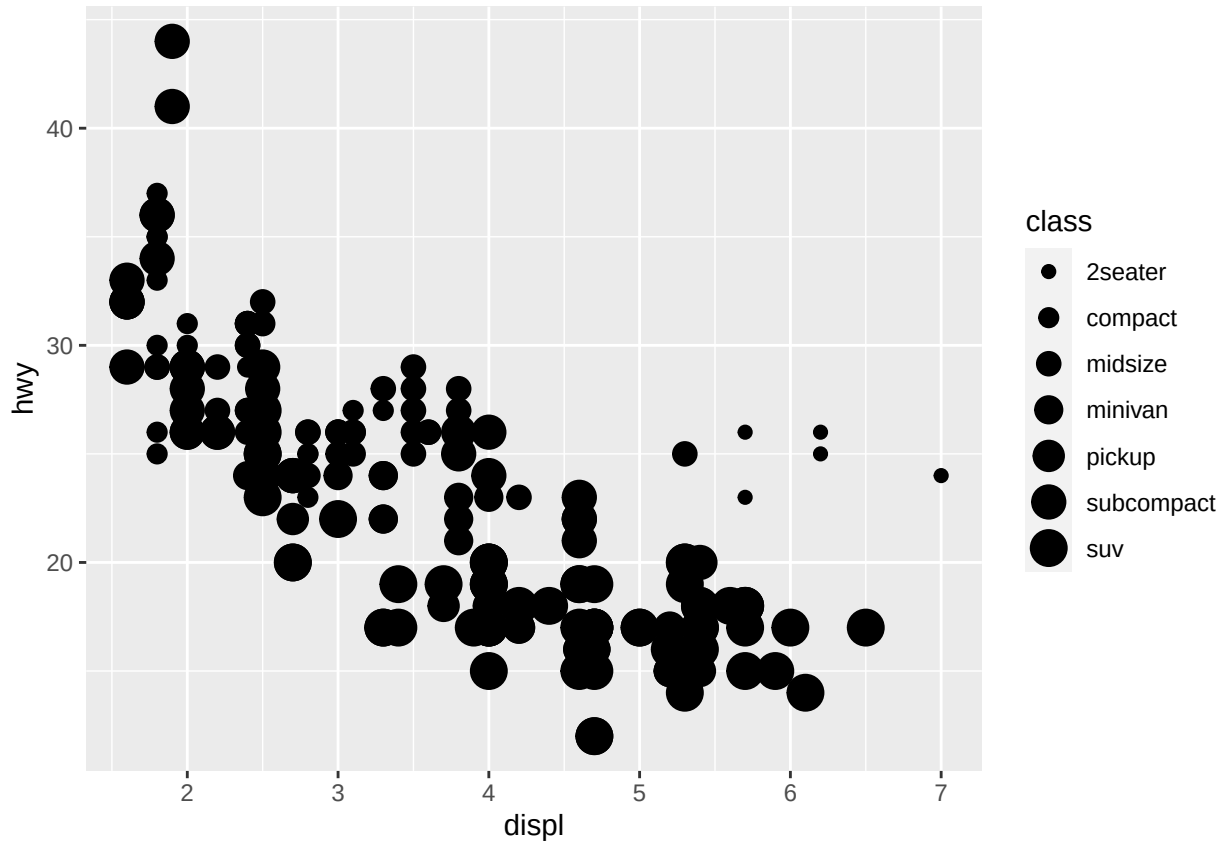



```
# Add a third variable: color
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



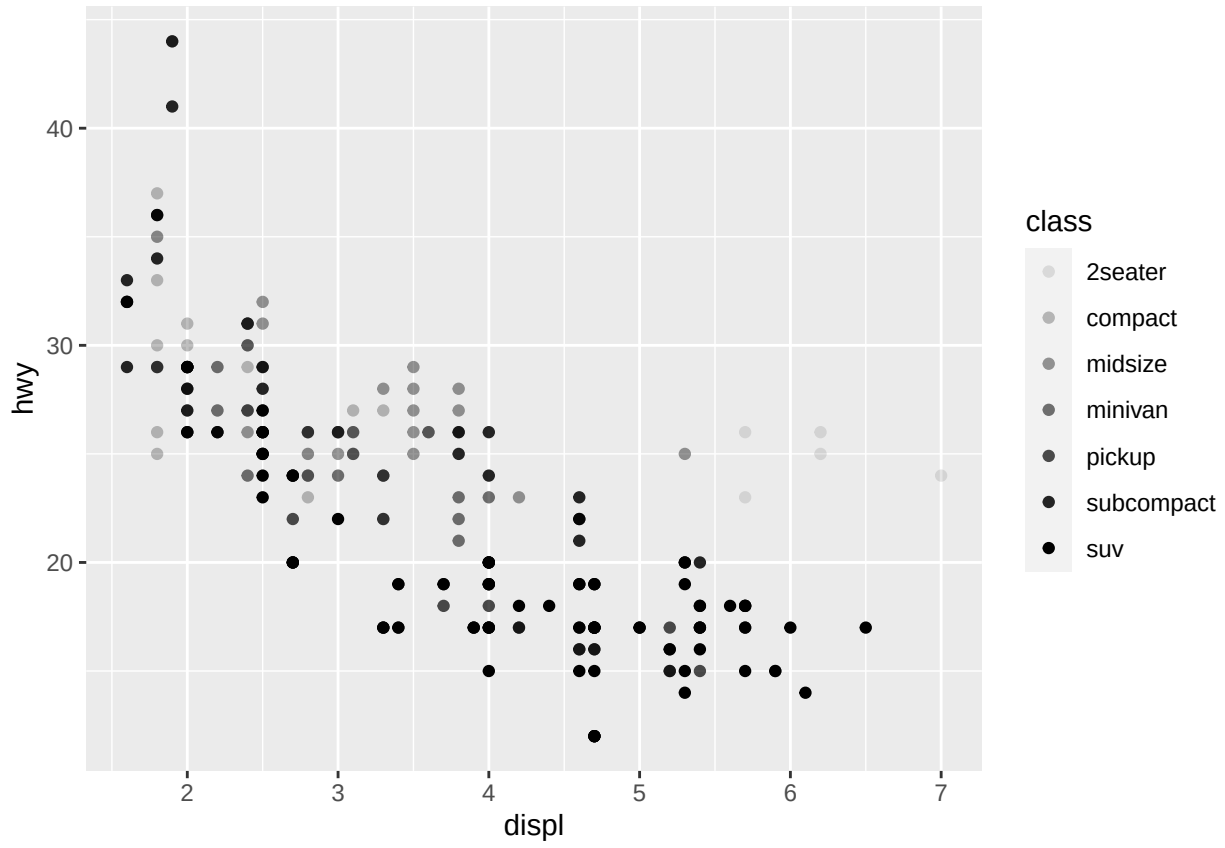

```
# Add a third variable: size
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))
```

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




```
# Add a third variable: alpha
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```

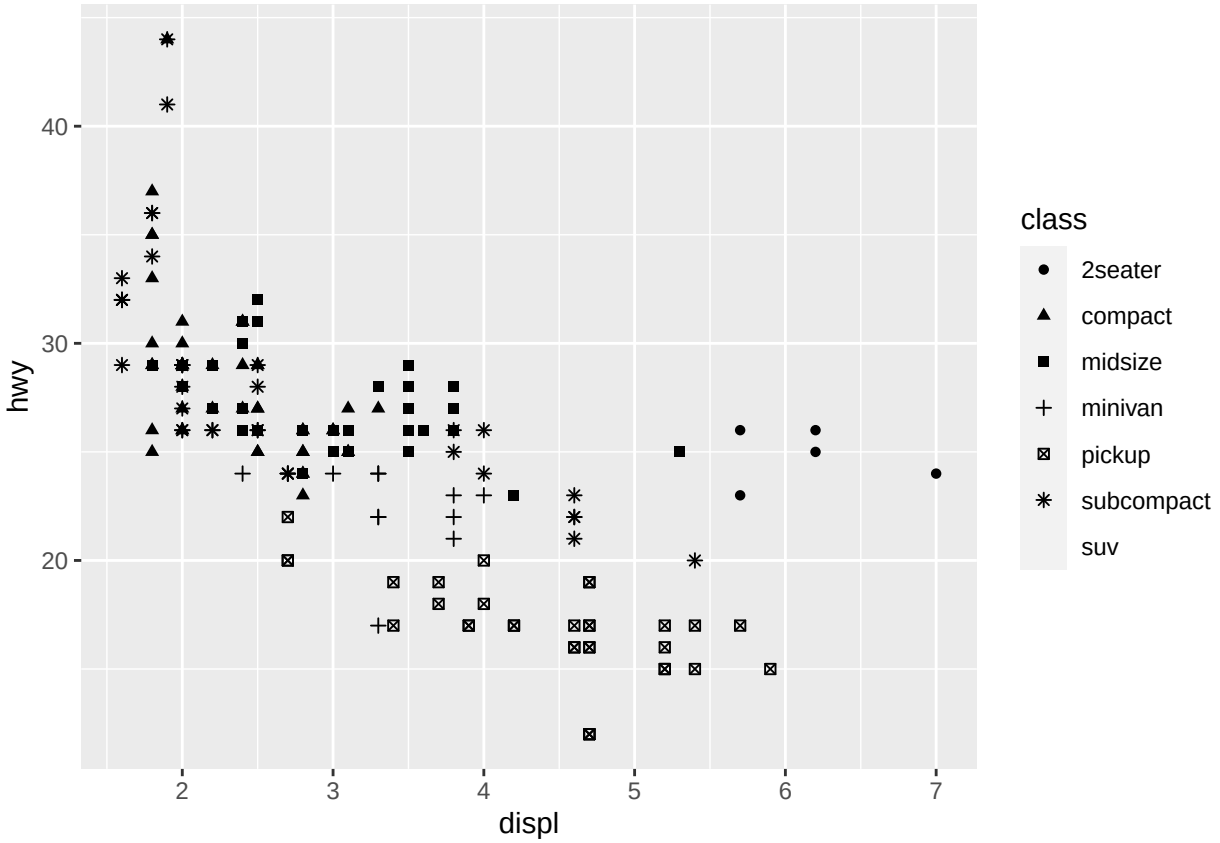
```
## Warning: Using alpha for a discrete variable is not advised.
```



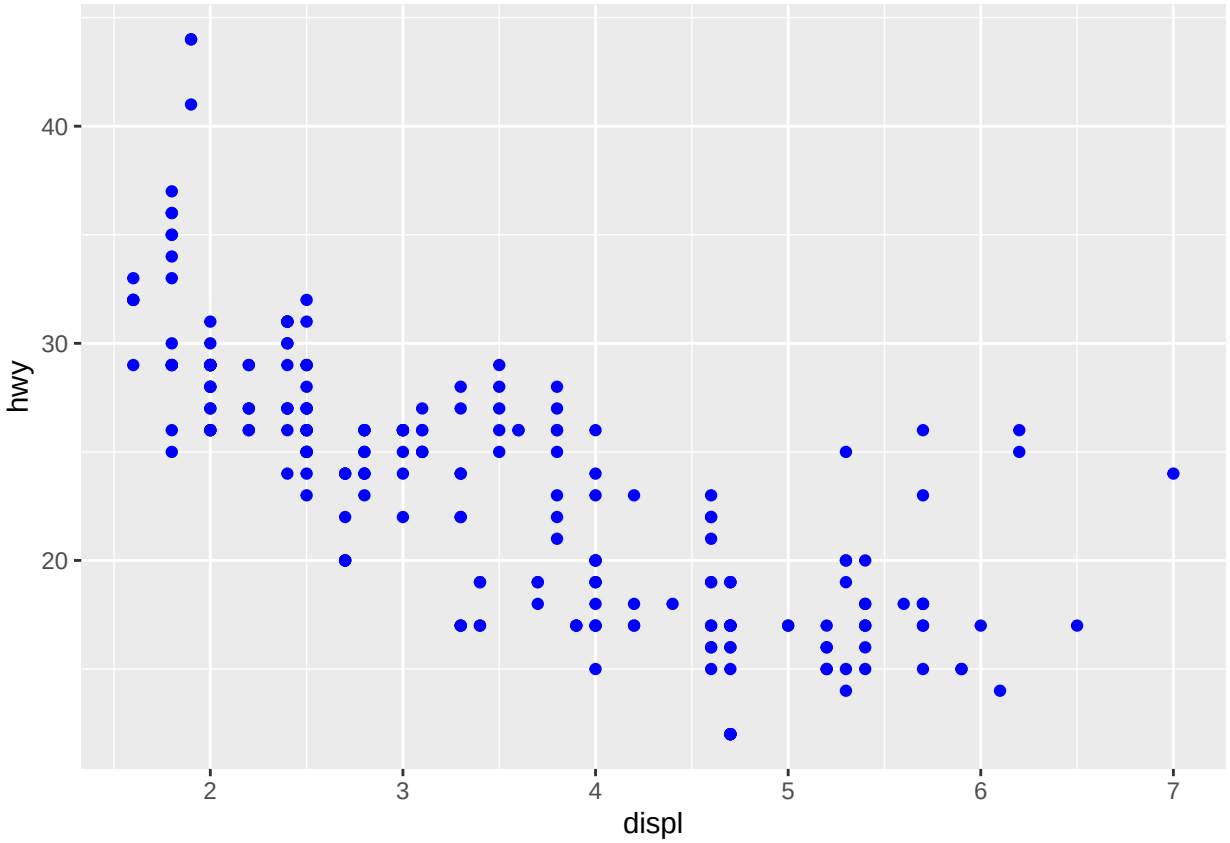

```
# Add a third variable: shape
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```

```
## 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).
```




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



common problems

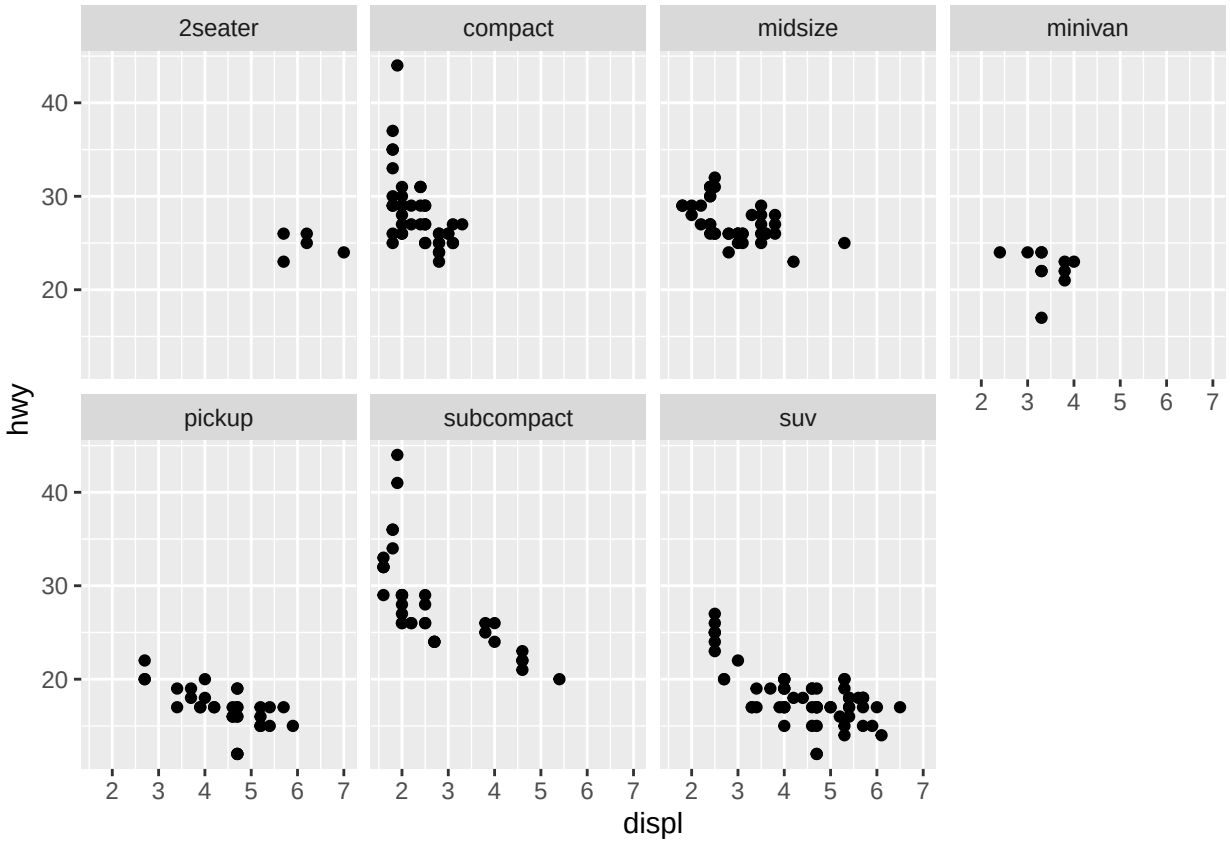
- Sometimes you'll run the code and nothing happens.
- Putting the + in the wrong place.

How to get help

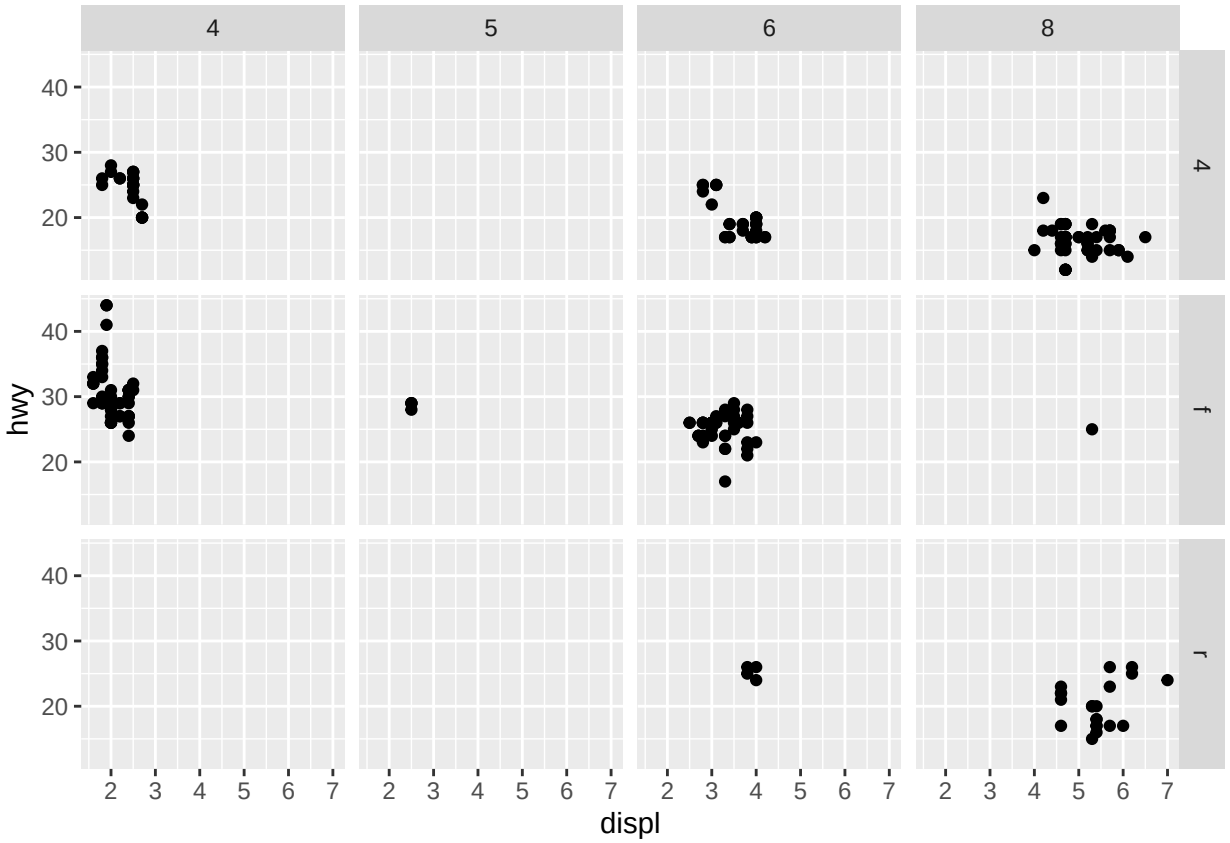
- ? function name
- Select the function name and press F1
- Read the error message
- Google the error message

facets

```
# facet your plot by a single variable
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



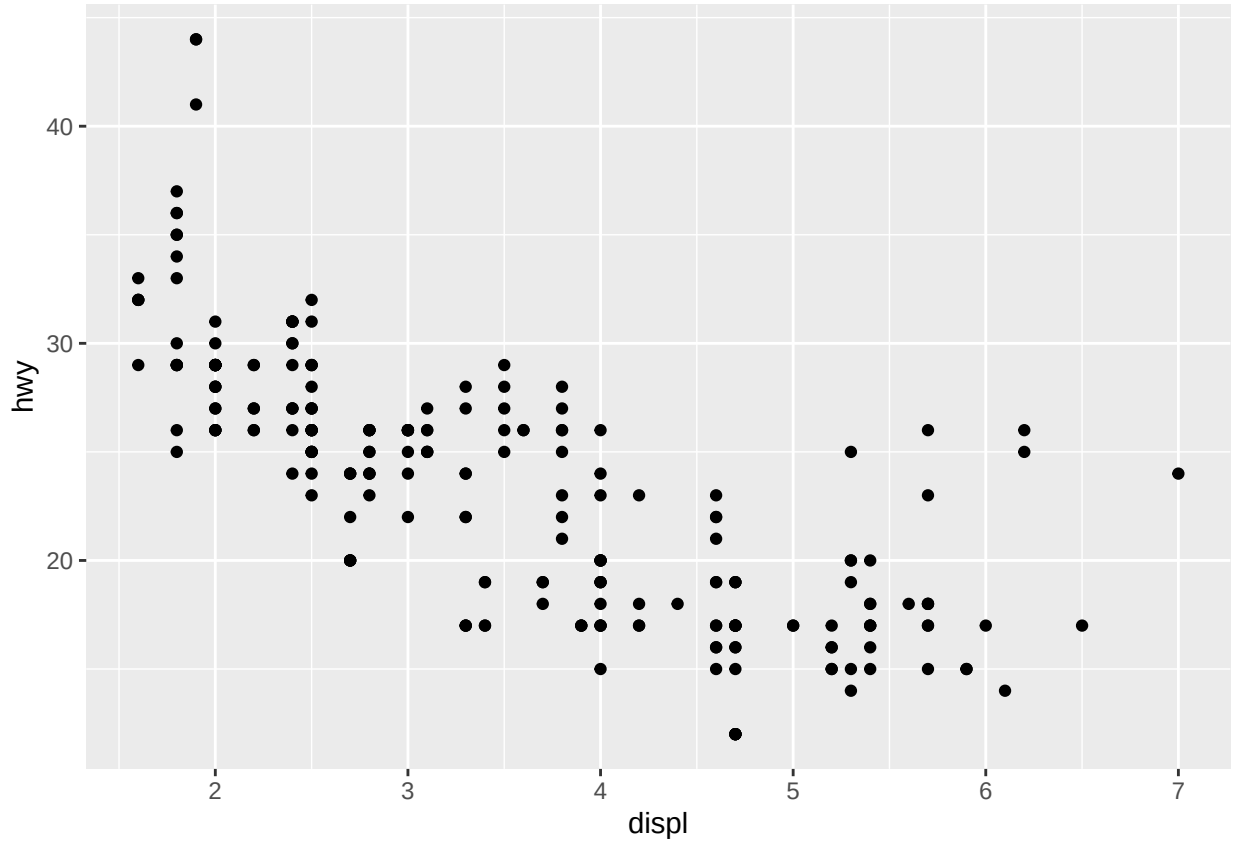

```
# facet your plot on the combination of two variables
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



geometric objects

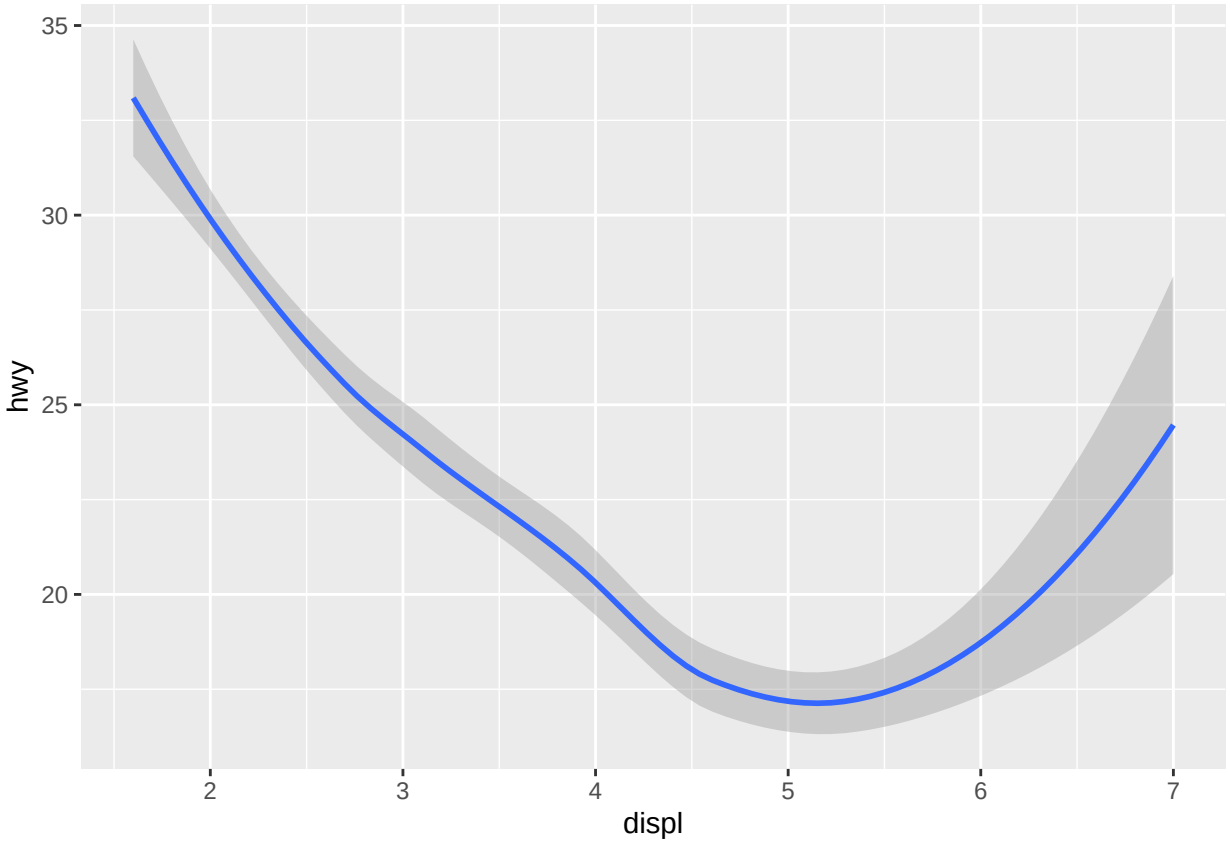
different visual object to represent data

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




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

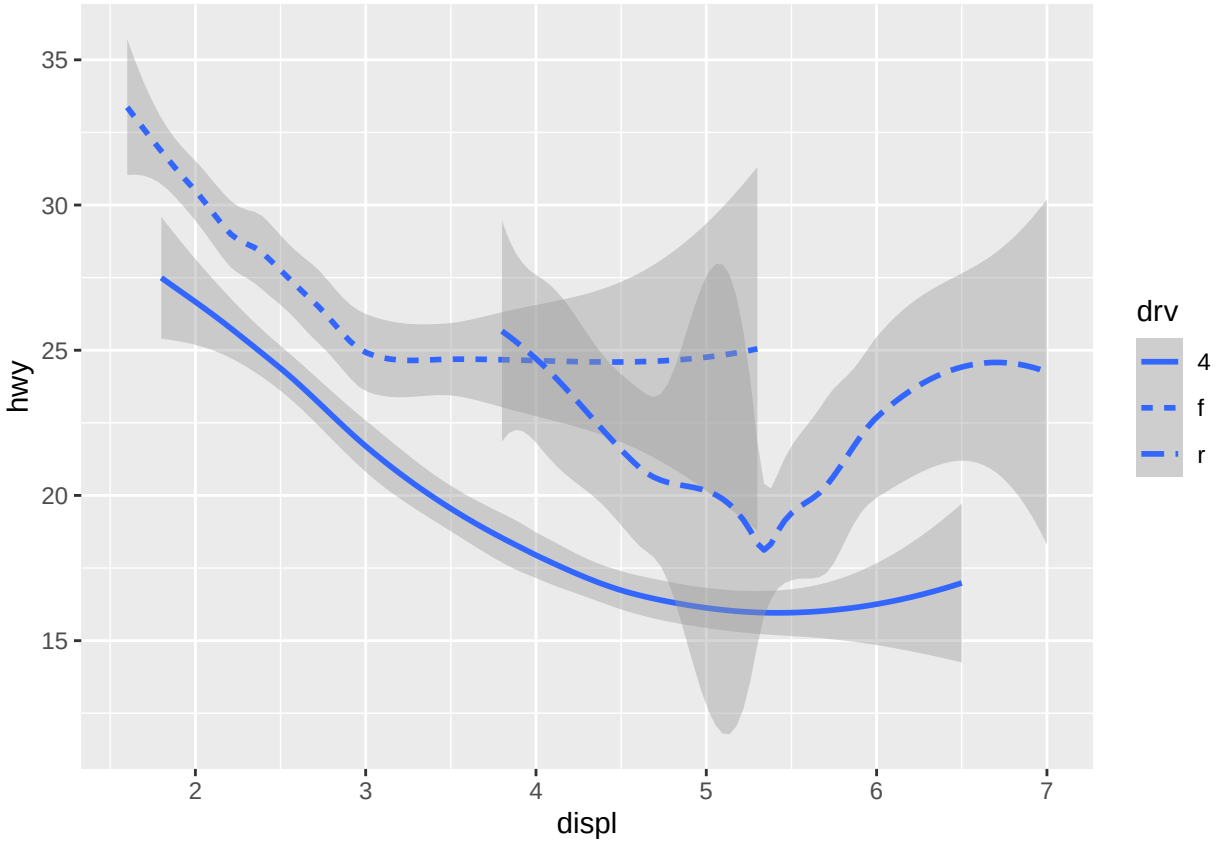
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



not every aesthetic works with every geom

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

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

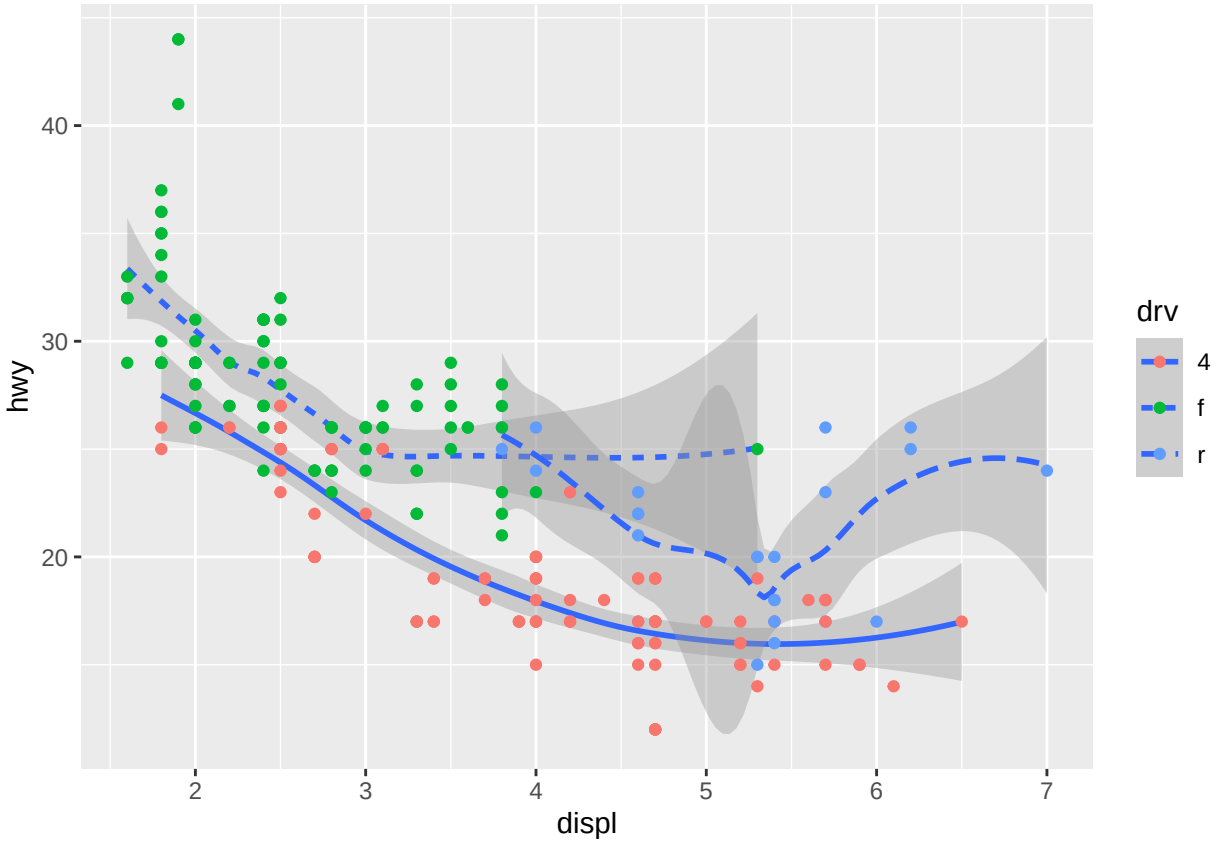


two geoms in the same graph!

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

  # Add another geom
  geom_point(mapping = aes(x = displ, y = hwy, color = drv))
```

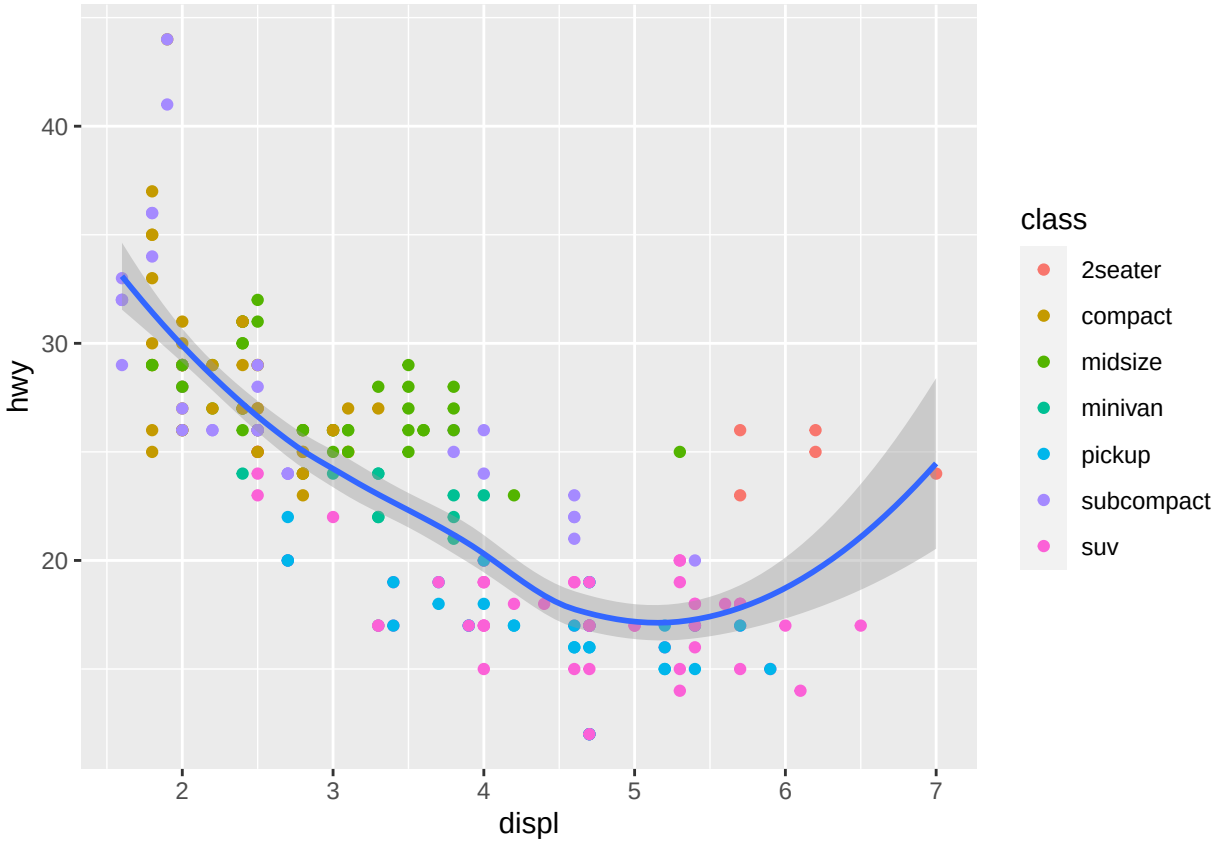
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



local vs. global mappings This makes it possible to display different aesthetics in different layers.

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

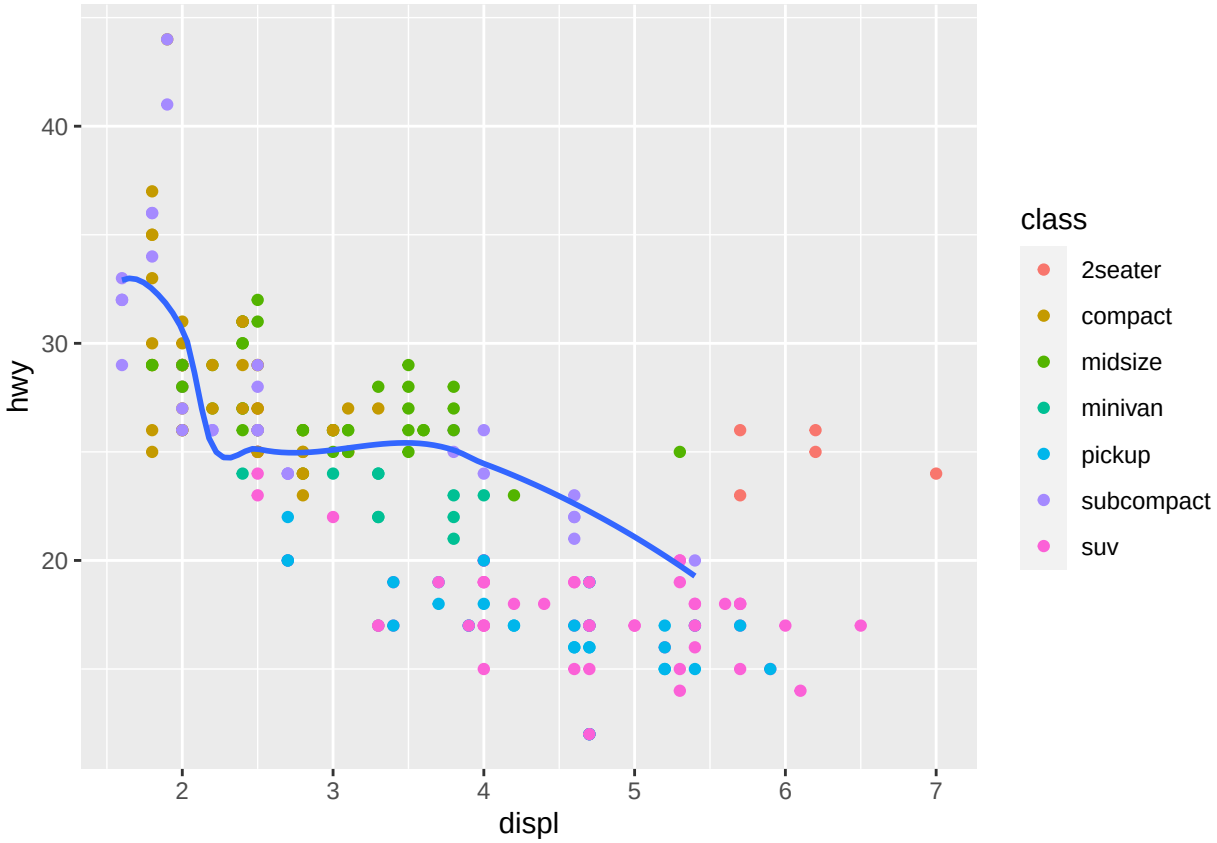
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



specify different data for each layer

```
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'
```

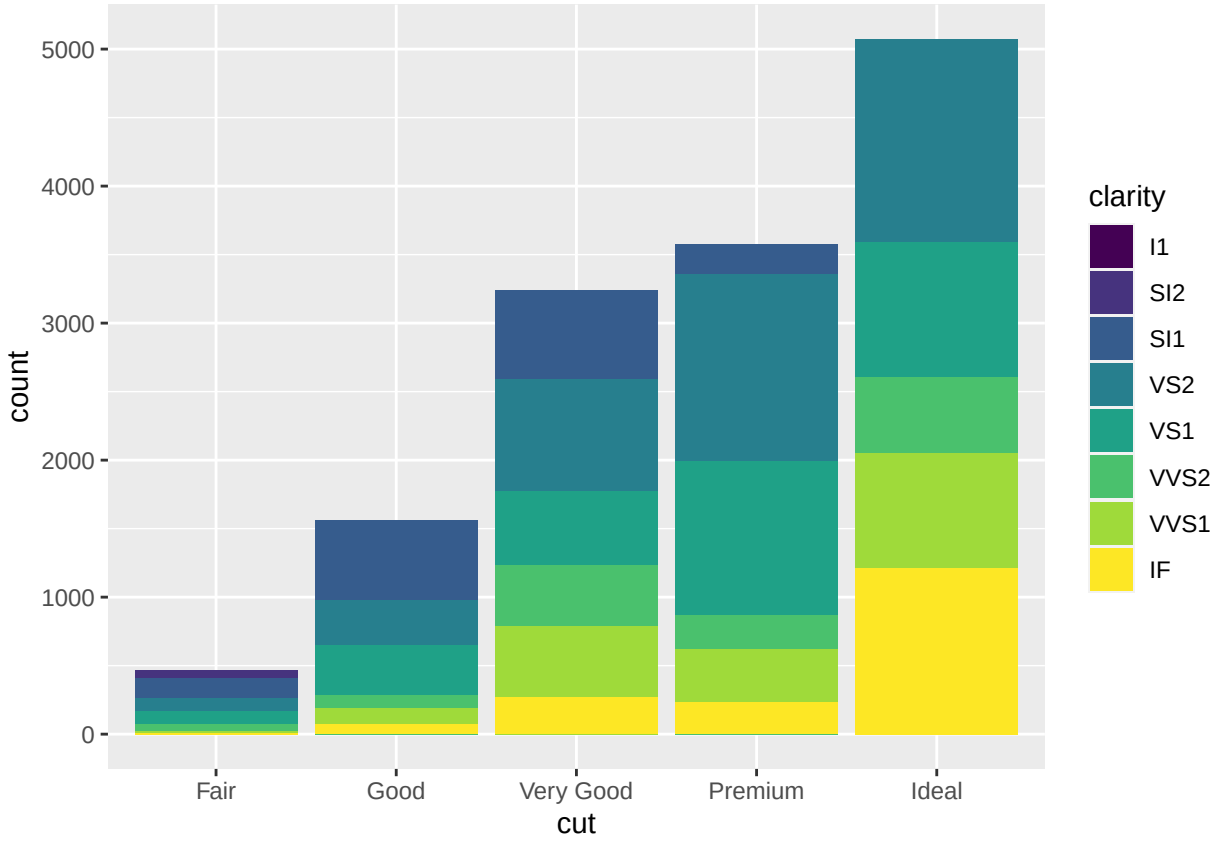


statistical transformation

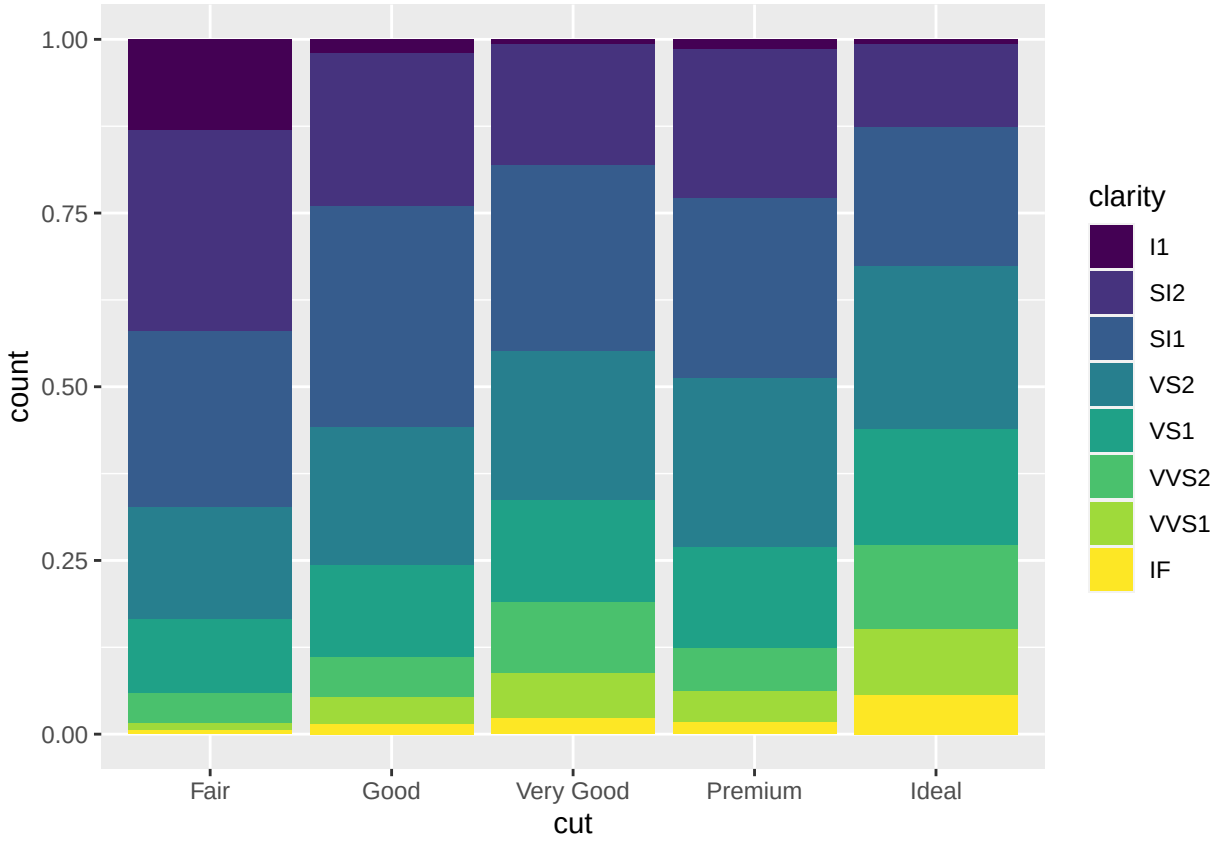
position adjustments

adjustments for bar charts

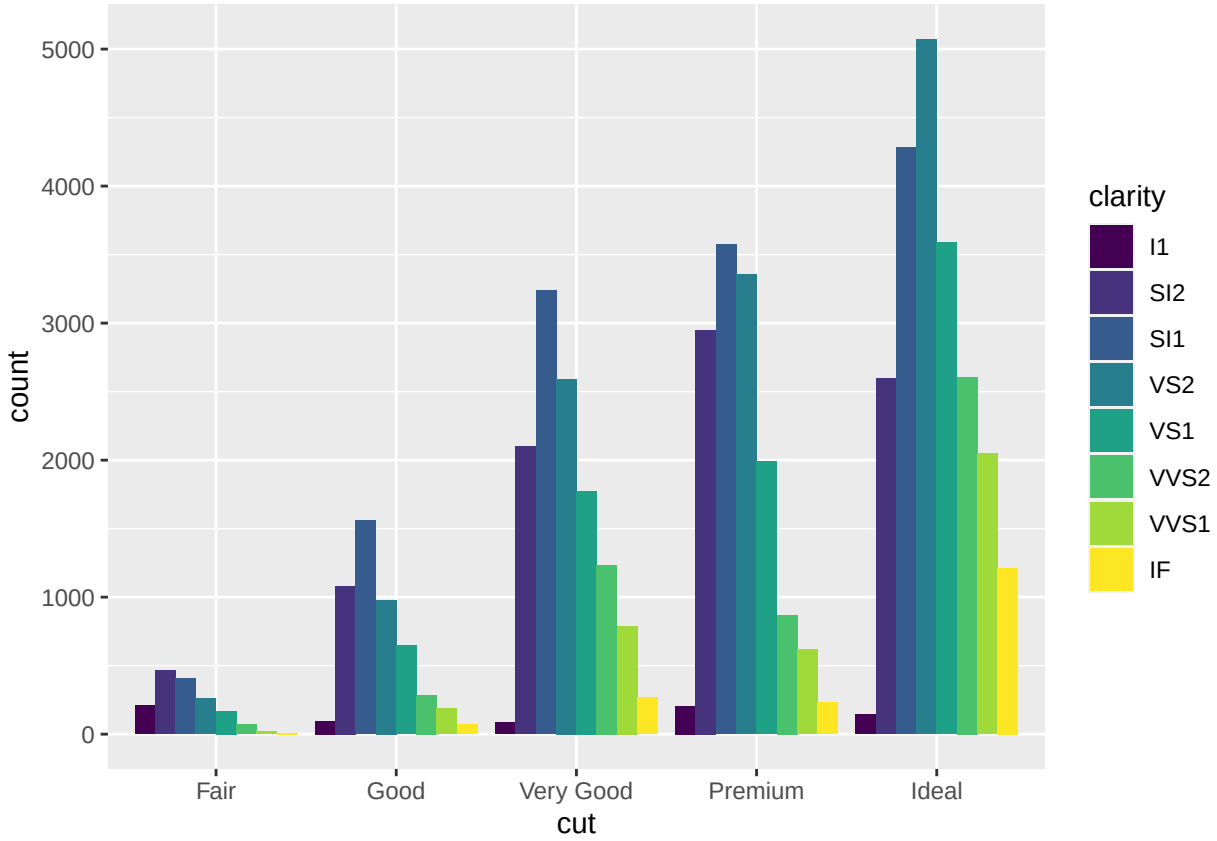
```
# to place each object exactly where it falls
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(position = "identity")
```




```
# to compare proportions across groups
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(position = "fill")
```

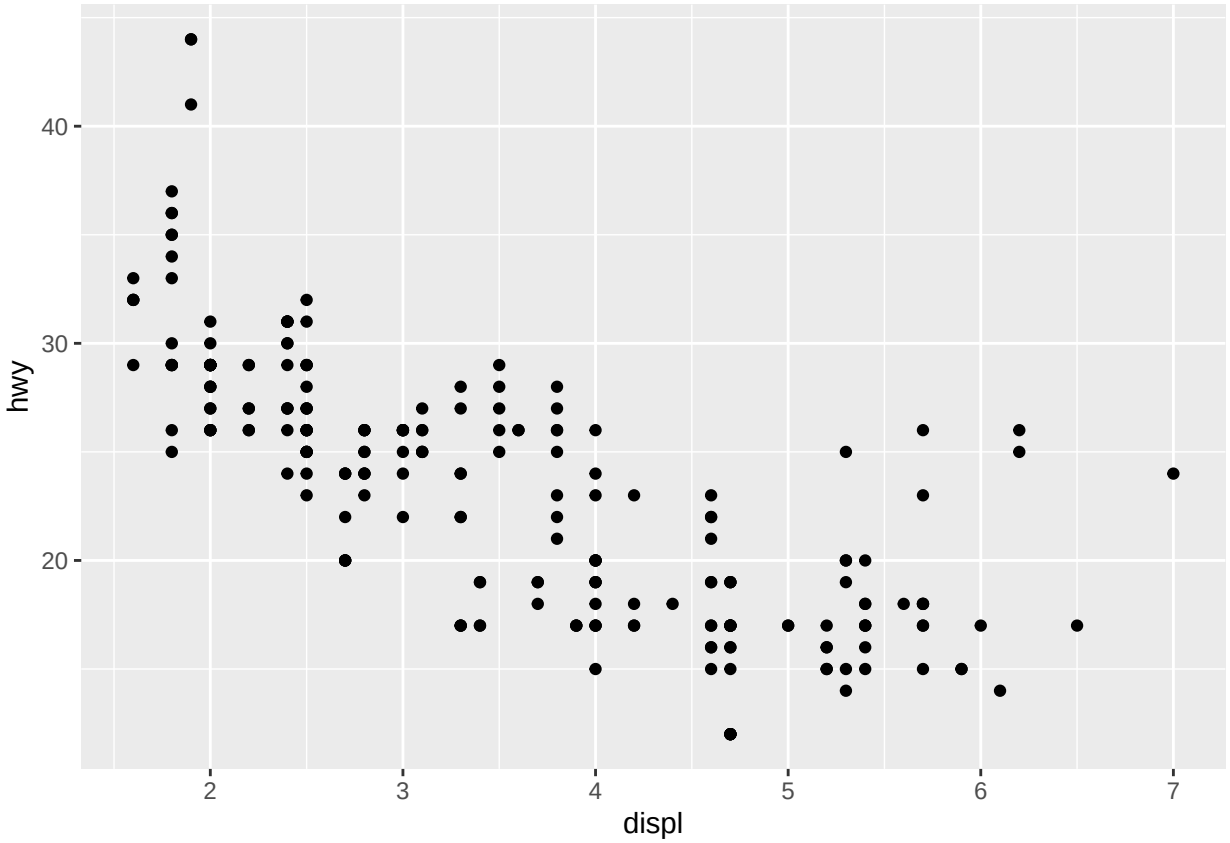



```
# to compare individual values
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(position = "dodge")
```

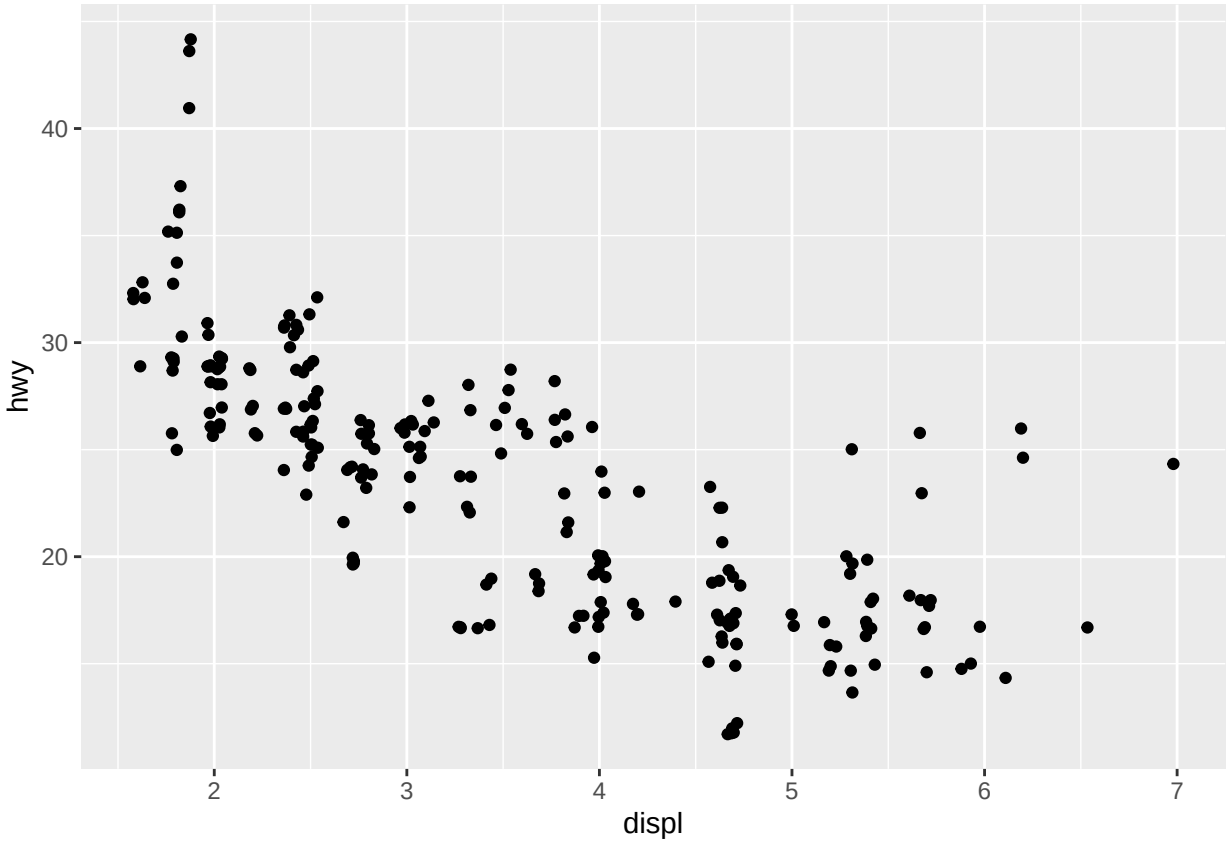


adjustments for scatterplots

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



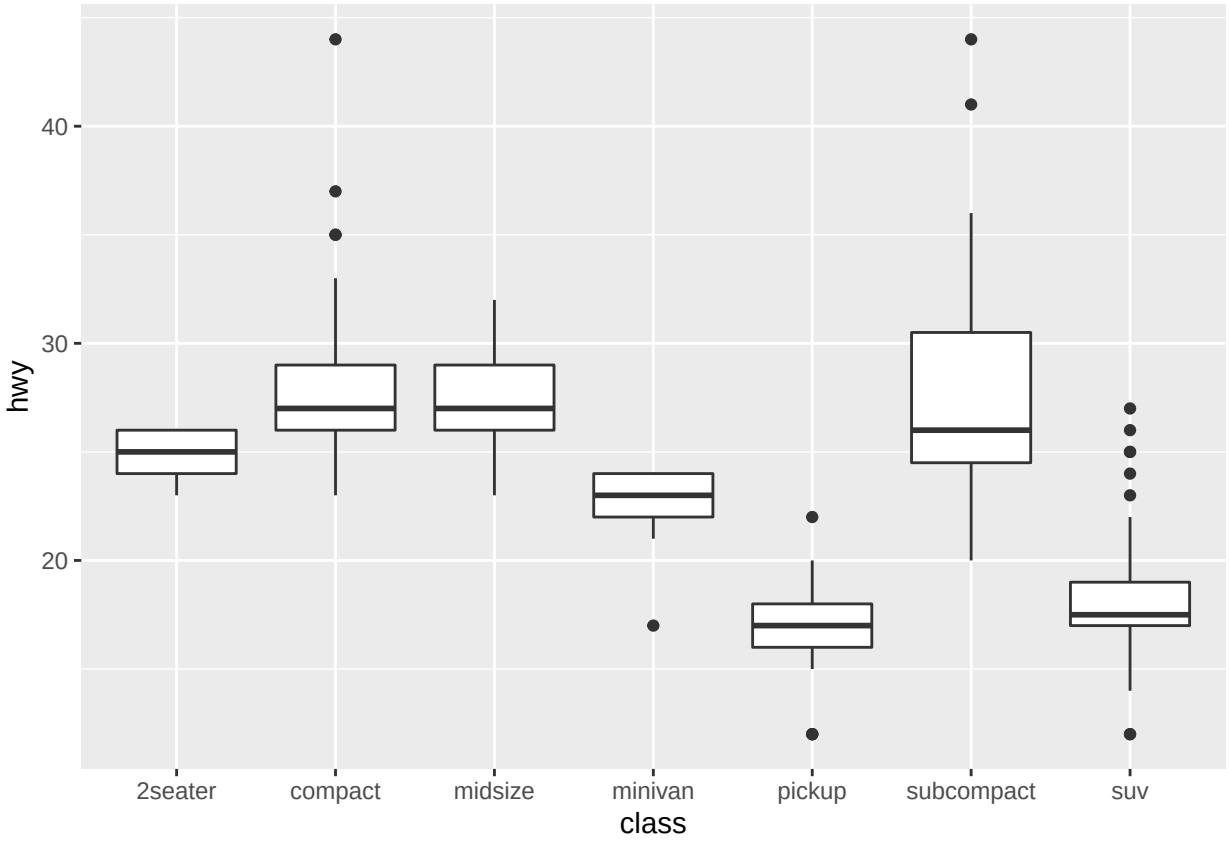

```
# jitter
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_jitter()
```



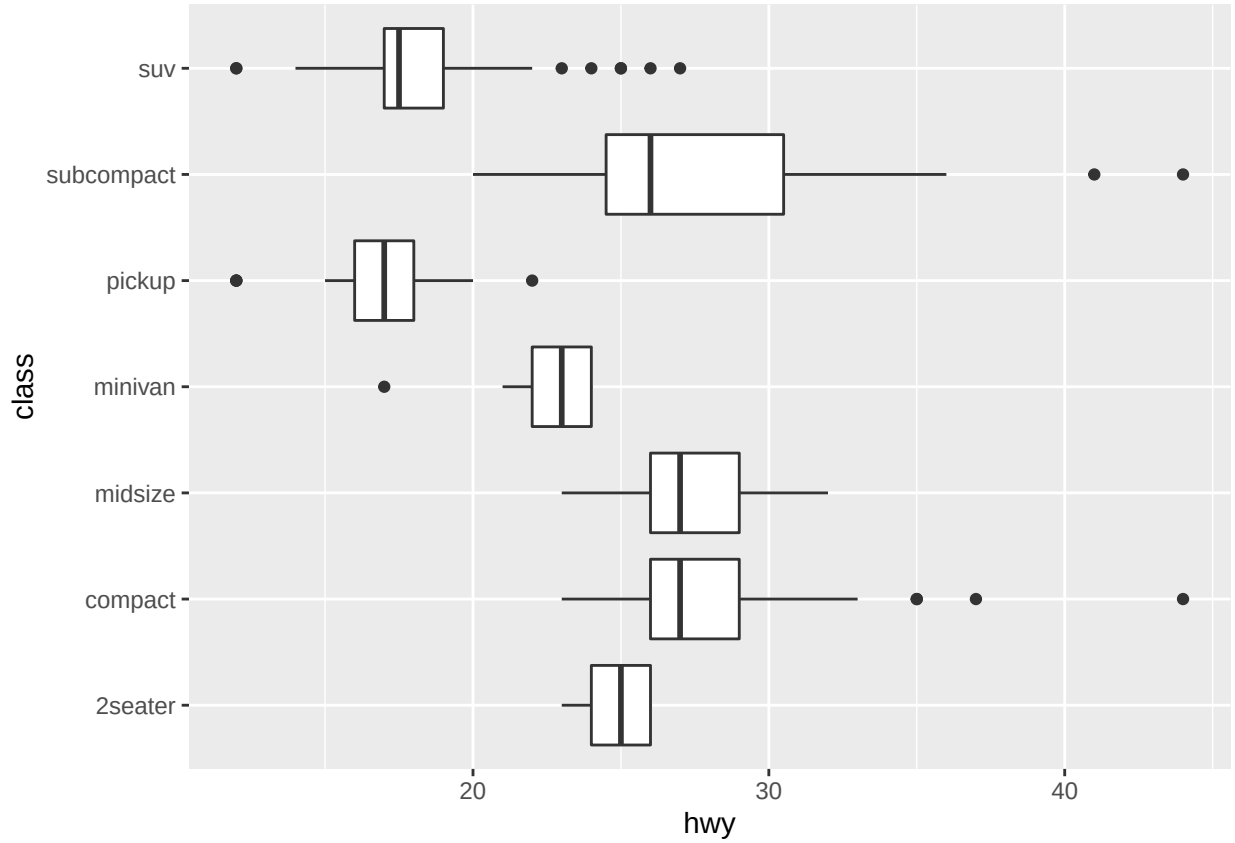
coordinate systems

switch x and y

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



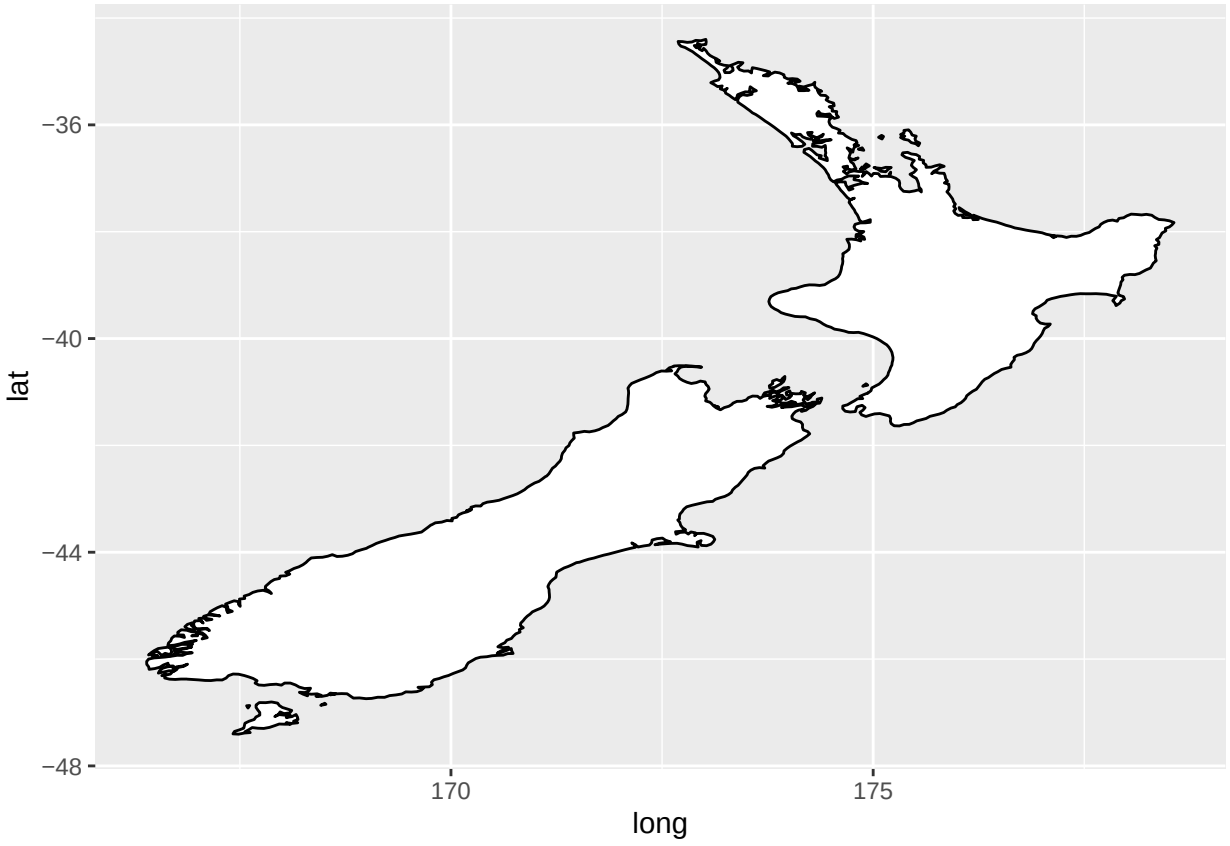

```
# switch x and y
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```



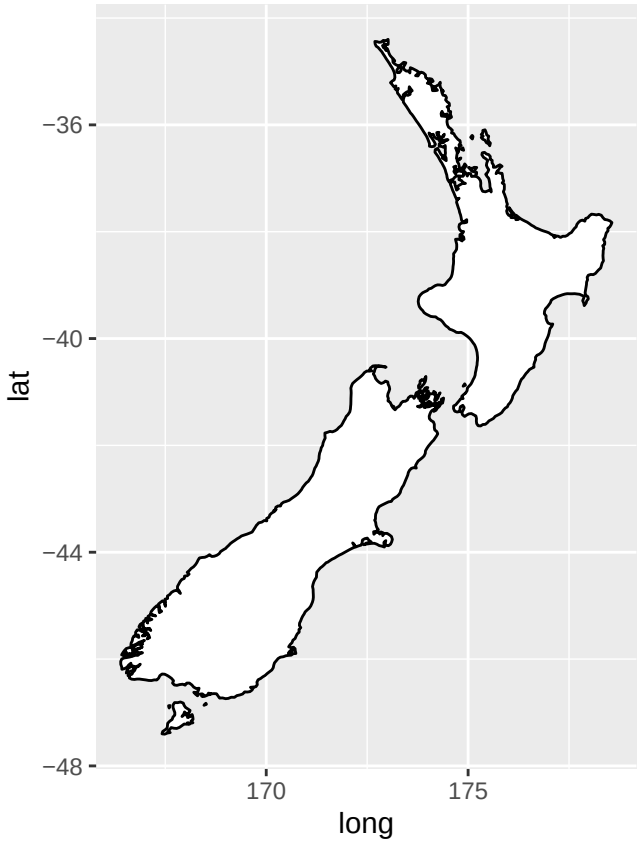
set the aspect ratio correctly for maps

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

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



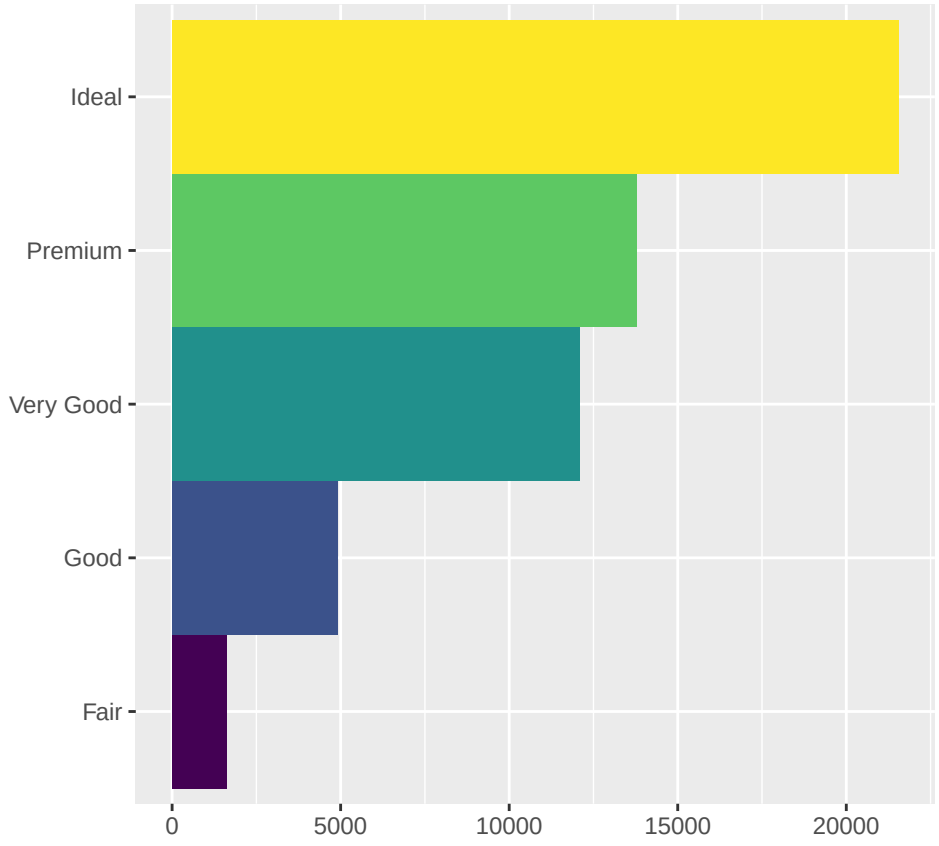

```
ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap()
```



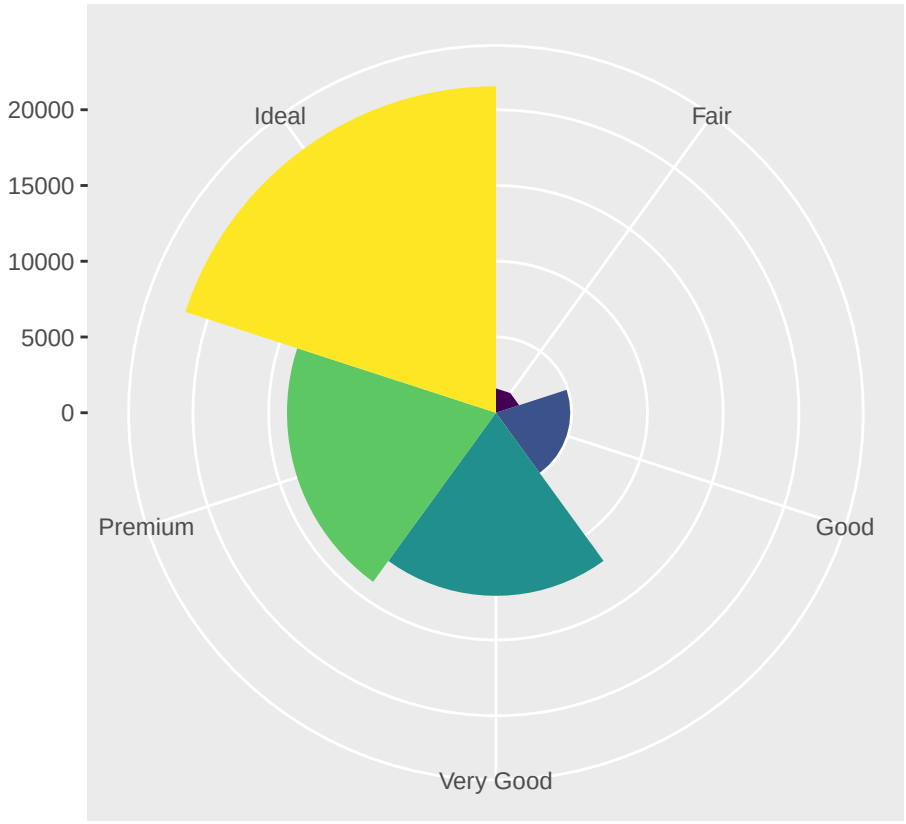
Polar coordinates reveal an interesting connection between a bar chart and a Coxcomb chart.

```
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()
```



the layered grammar of graphics

The grammar of graphics is based on the insight that you can uniquely describe any plot as a combination of:

- a dataset,
- a geom,
- a set of mappings,
- a stat,
- a position adjustment,
- a coordinate system, and
- a faceting scheme.