

## Assignment 4

```
library(tidyverse)

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

## v tibble 3.1.4      v dplyr 1.0.7
## v tidyr 1.1.4       v stringr 1.4.0
## v purrr 0.3.4       v forcats 0.5.1

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

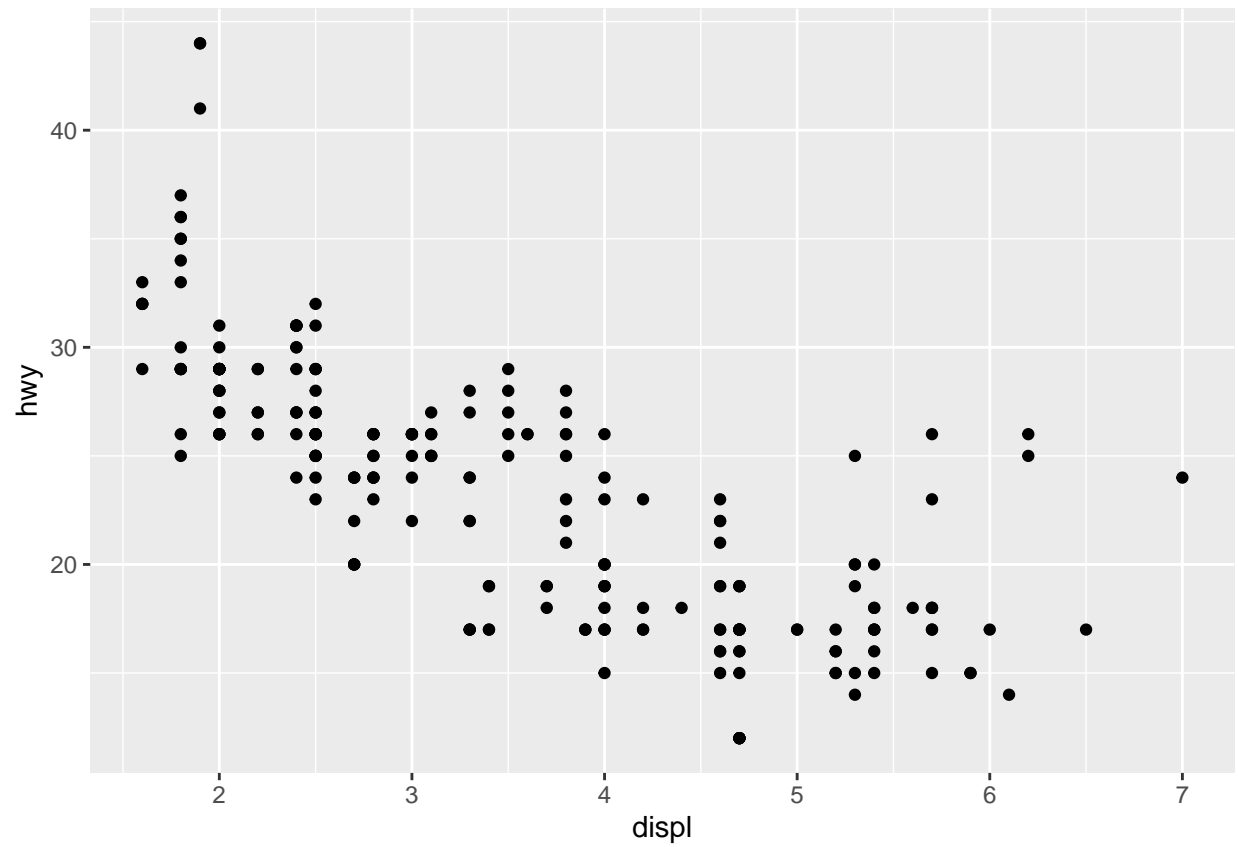
### Data Visualization Article Notes:

```
ggplot2::mpg

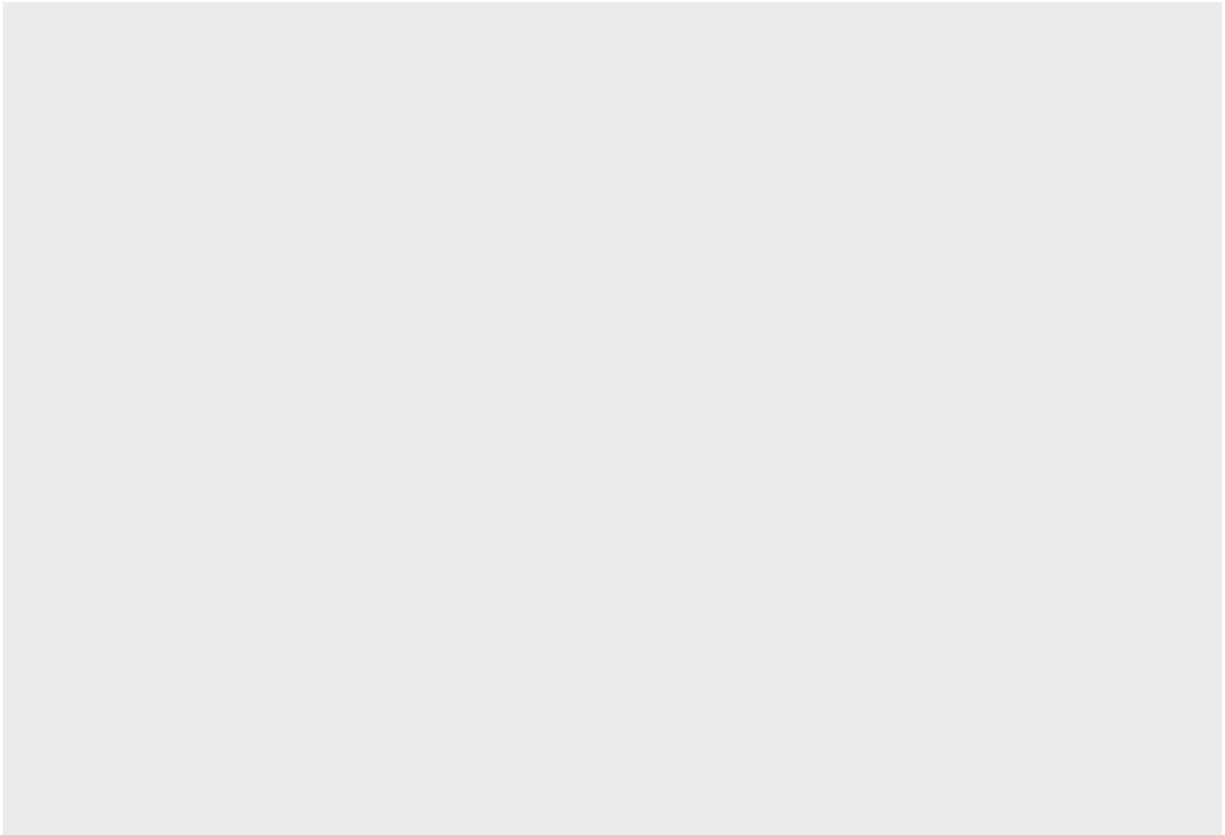
## # A tibble: 234 x 11
##   manufacturer model      displ  year  cyl trans drv      cty   hwy fl      class
##   <chr>          <chr>    <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
## 1 audi          a4          1.8  1999    4 auto~ f      18    29 p      comp~
## 2 audi          a4          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
```

-data frame found in ggplot2 (a rectangular collection of variables & observations) -mpg contains observations collected on diff. models of cars

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



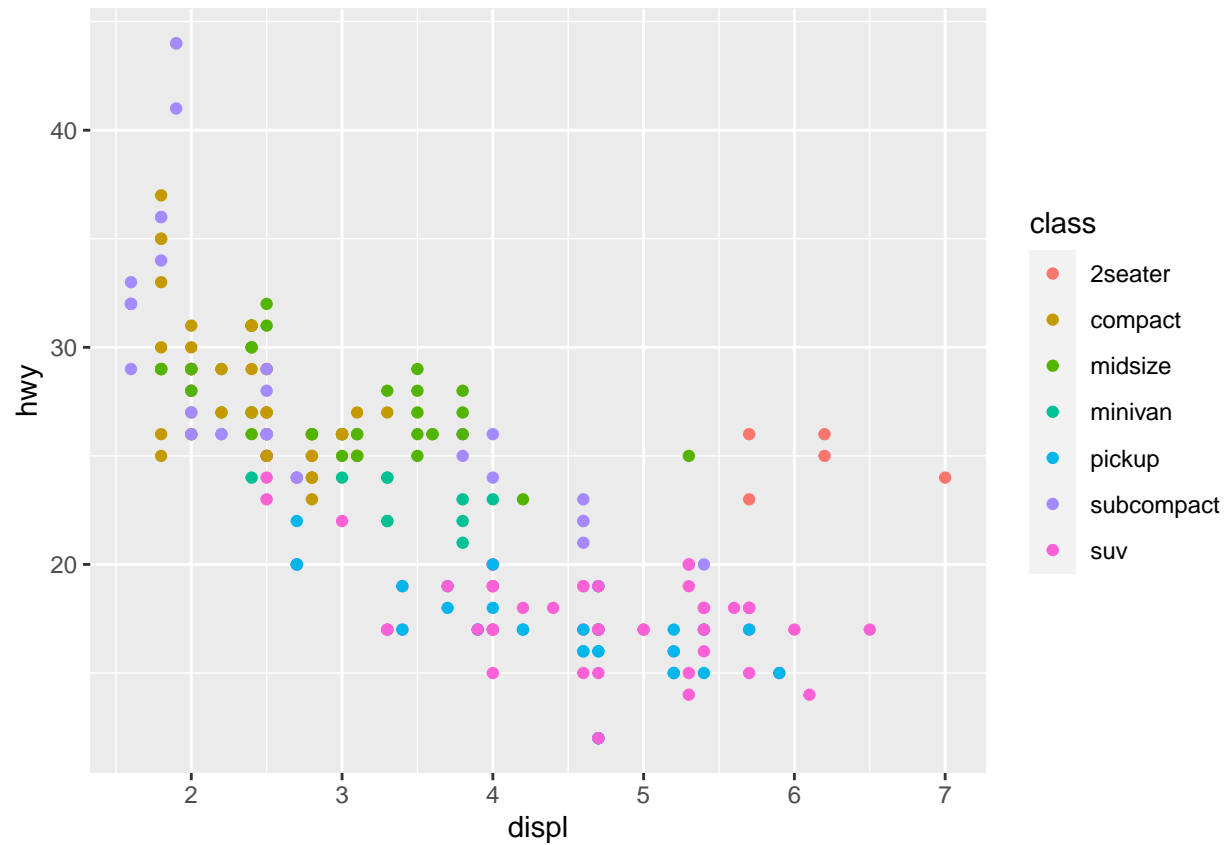
```
ggplot(data = mpg)
```



-plot mpg & specify what's on each axis -ggplot(data=mpg) creates an empty graph but u have to add layers -geom\_point() adds layer of points to plot (creates scatterplot)

CODE PRESET: ggplot(data = ) + (mapping = aes()) -replace bracketed sections in code below w/ dataset  
-aesthetic: visual property of objects in plot (size, shape, color, etc.) -> referred to as level

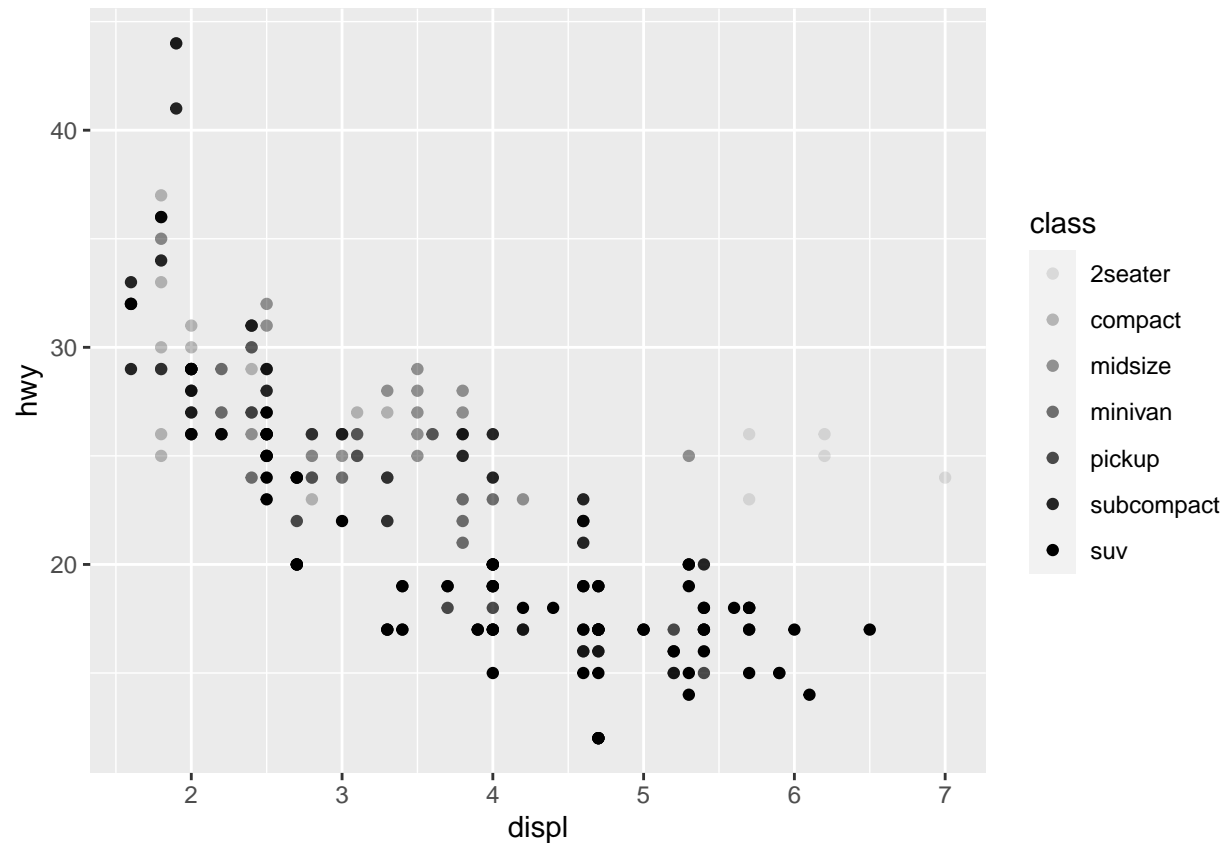
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



-maps the colors of each point to the class variable (reveals class of each car)

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

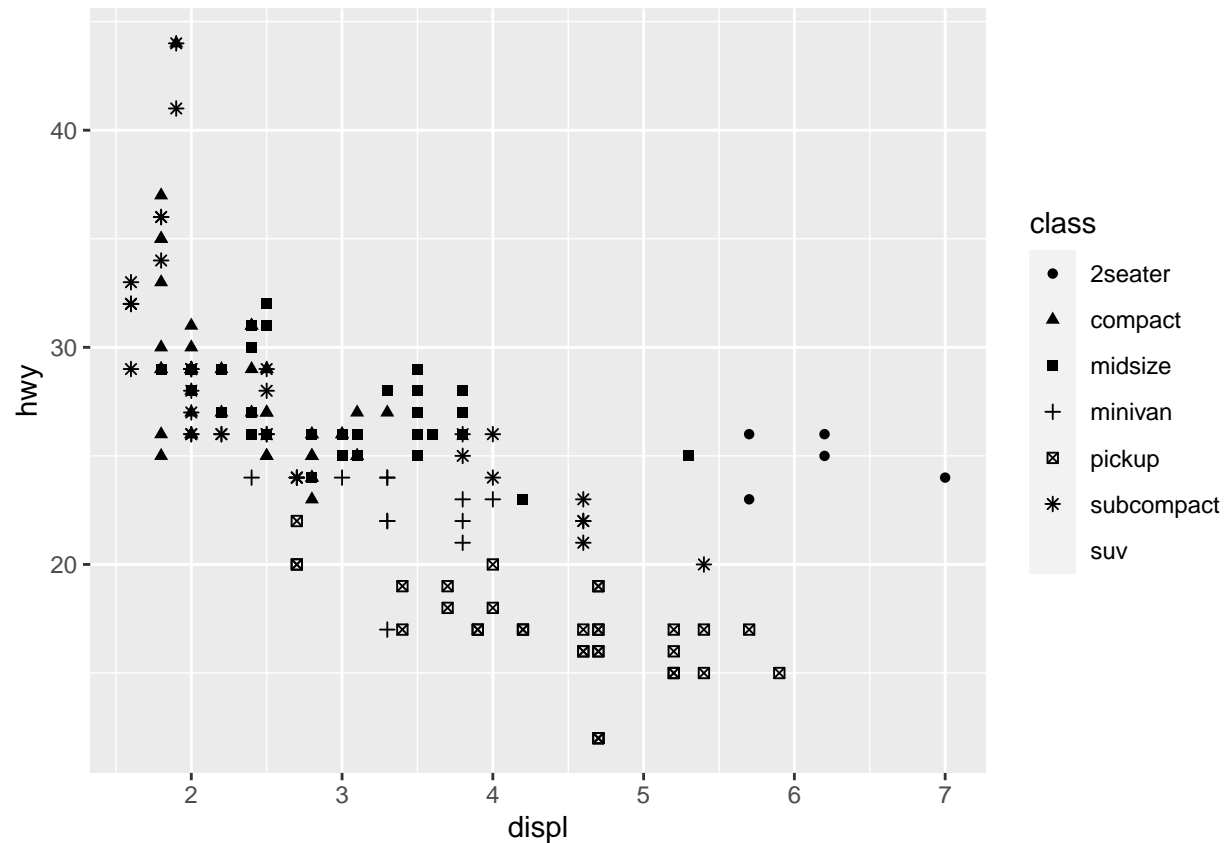
```
## Warning: Using alpha for a discrete variable is not advised.
```



```
# 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.
```

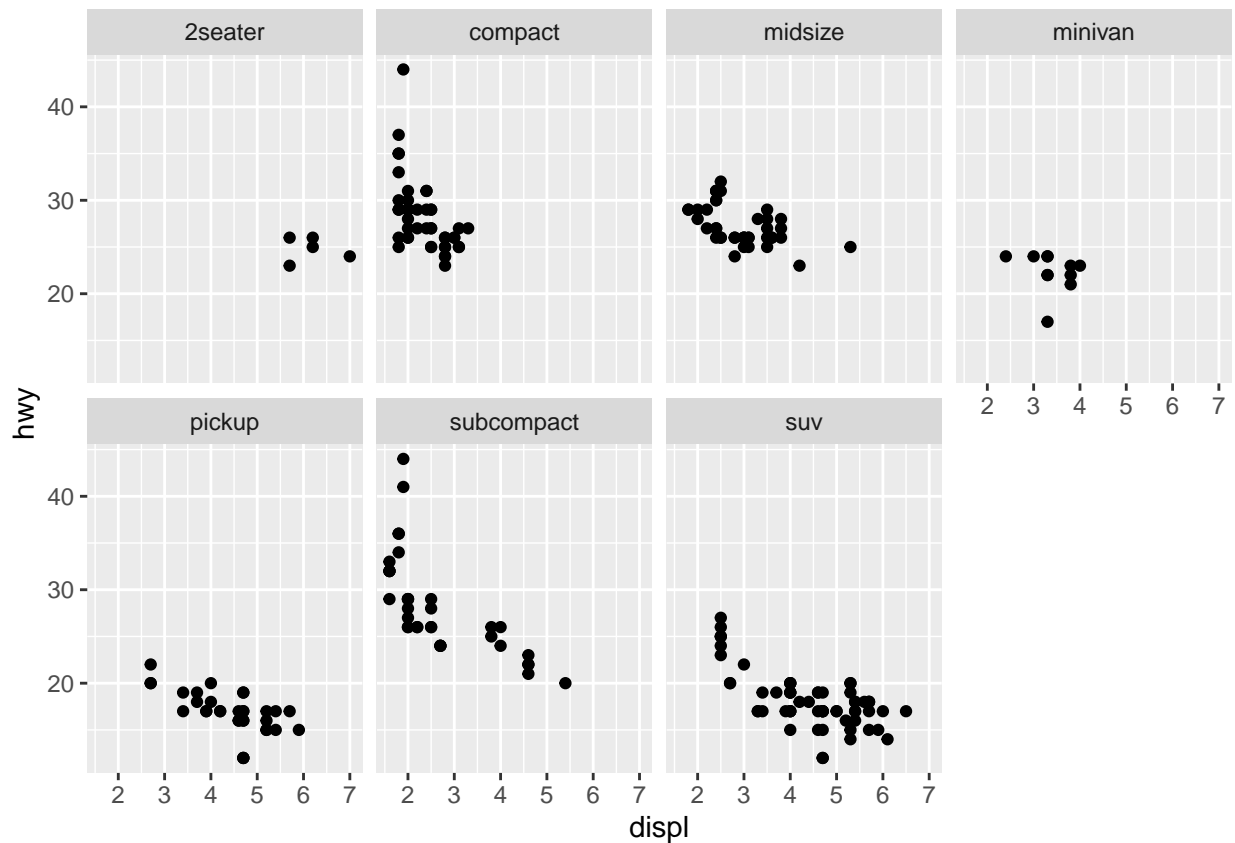
```
## Warning: Removed 62 rows containing missing values (geom_point).
```



-mapping class to alpha controls transparency of points -mapping class to shape controls shape of points  
 -you use aes() to associate the name of the aesthetic with a variable to display -+ signs have to be at end of line not the beginning (check above)

-facets: sunplots that each display one subset of data

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

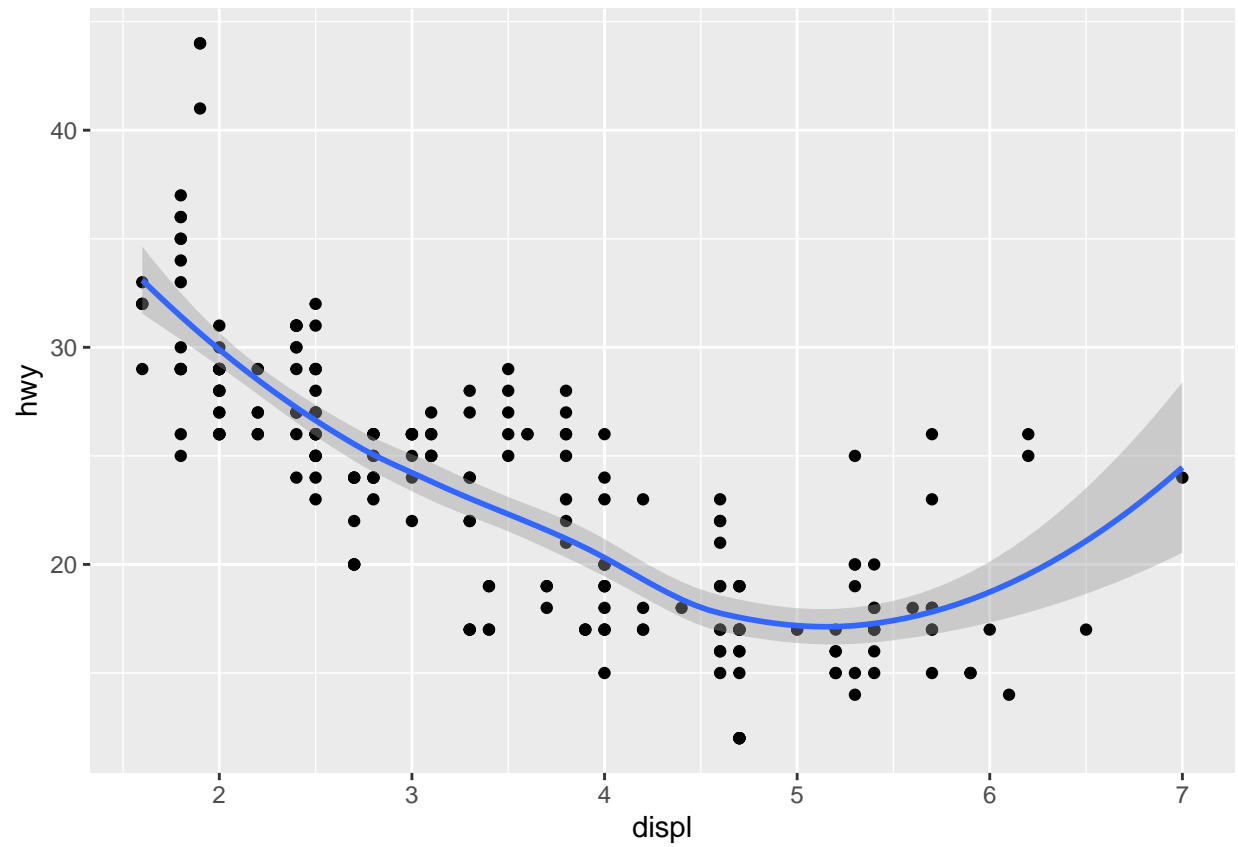


-add `facet_grid()` to plot on combination of 2 variables -use a `.` instead of a variable name if u don't wanna facet in row/column dimension

-geom: geometrical object that plot uses to represent data

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

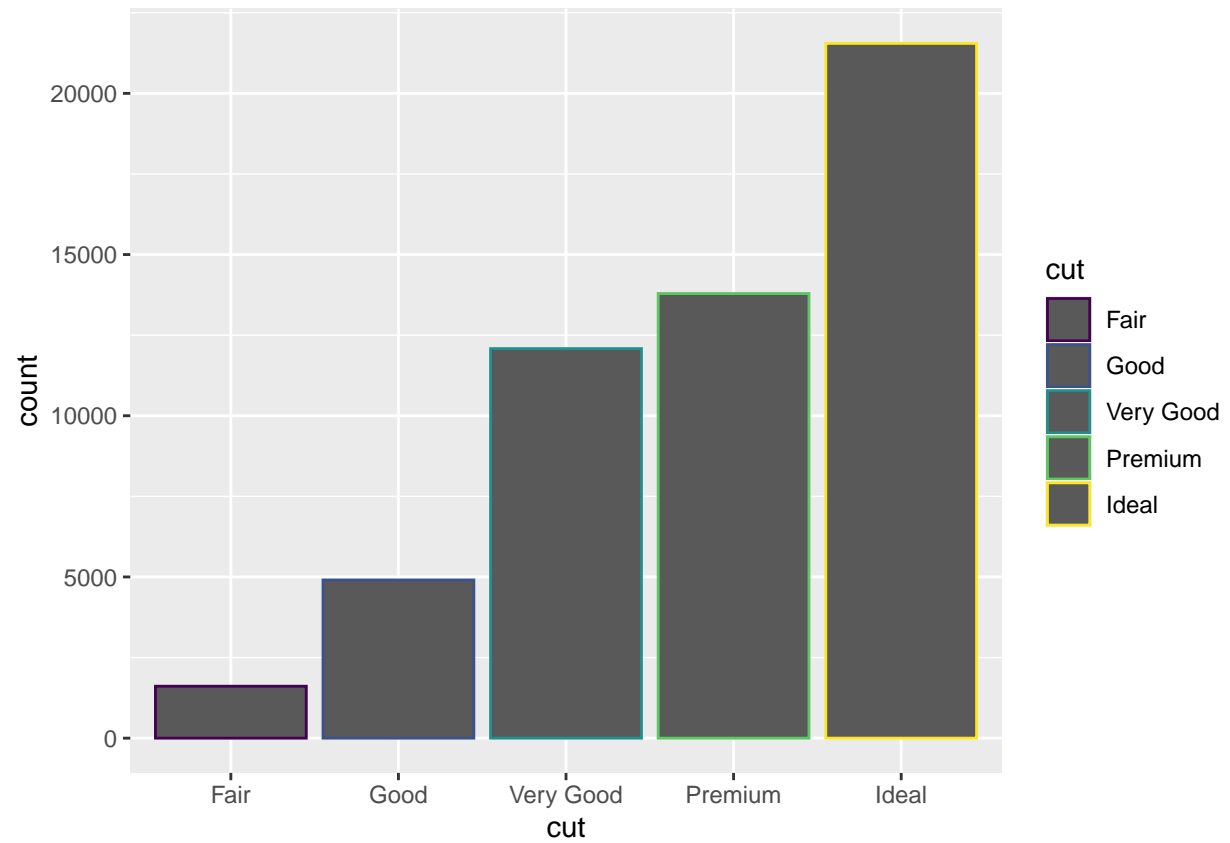
## 'geom\_smooth()' using method = 'loess' and formula 'y ~ x'



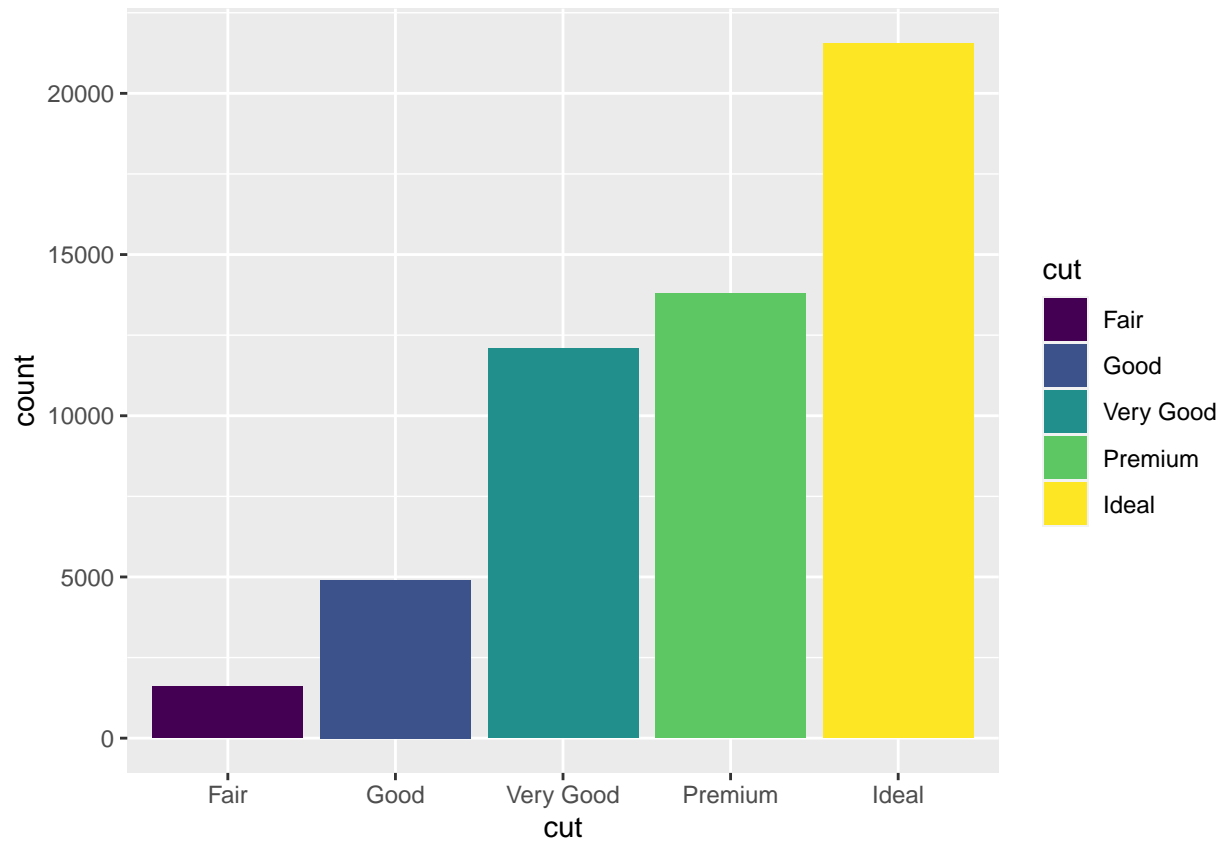
-lets you display multiple geoms in same plot

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





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



-allows you to change colors on bar graphs

-coord\_flip() switches the x and y axes -> useful for horizontal box plots

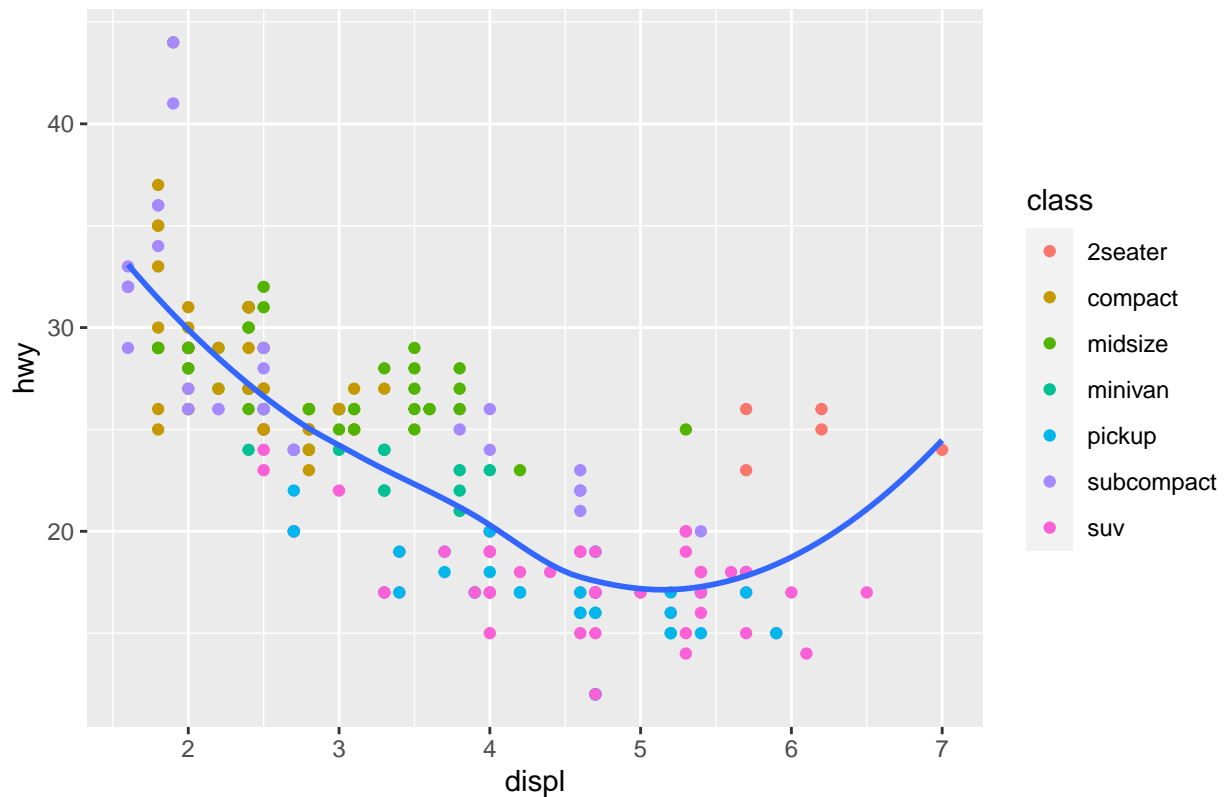
## Graphics for Communication Article Notes:

-Label, Annotations, Scales

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(se = FALSE) +
  labs(title = "Fuel efficiency generally decreases with engine size")
```

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

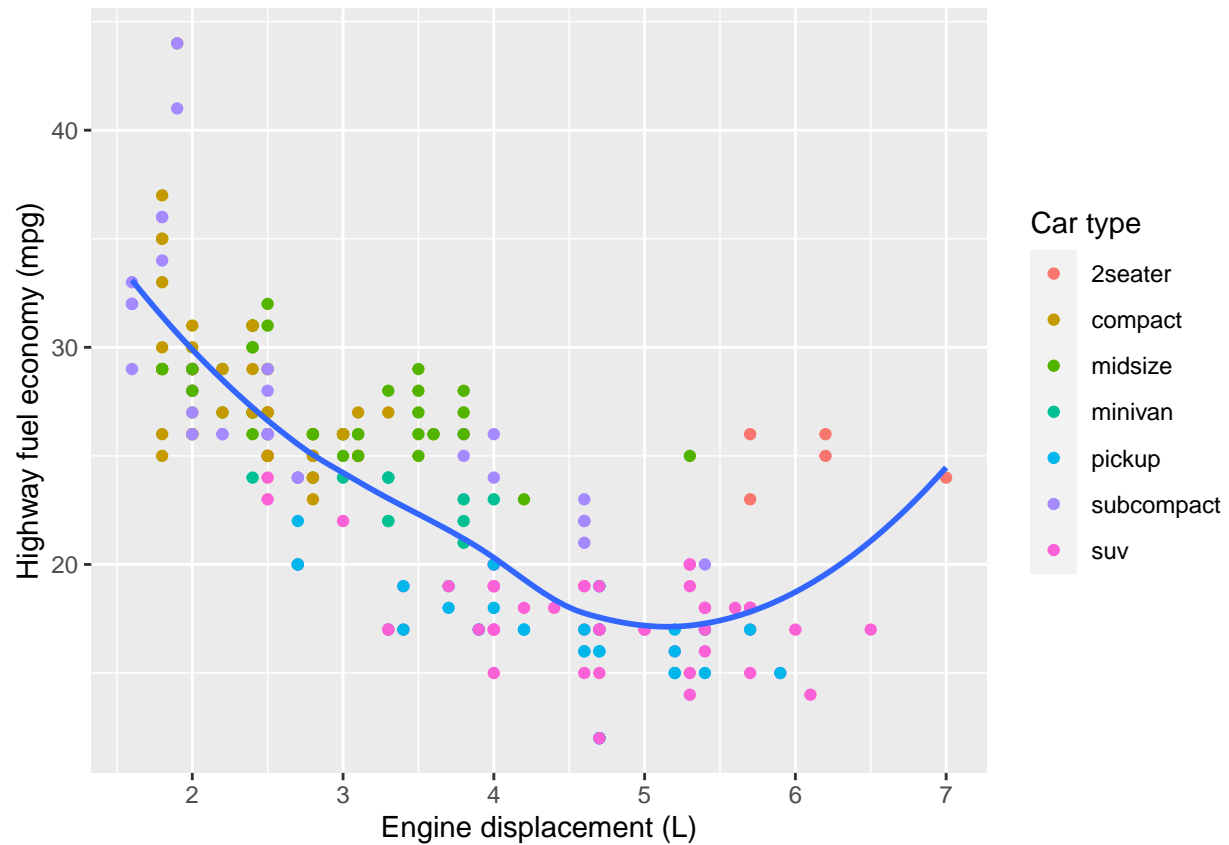
Fuel efficiency generally decreases with engine size



-labs() function lets you add labels (used to add title here)^ -plot title should summarize main finding instead of saying vs

```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(colour = class)) +  
  geom_smooth(se = FALSE) +  
  labs(  
    x = "Engine displacement (L)",  
    y = "Highway fuel economy (mpg)",  
    colour = "Car type"  
  )
```

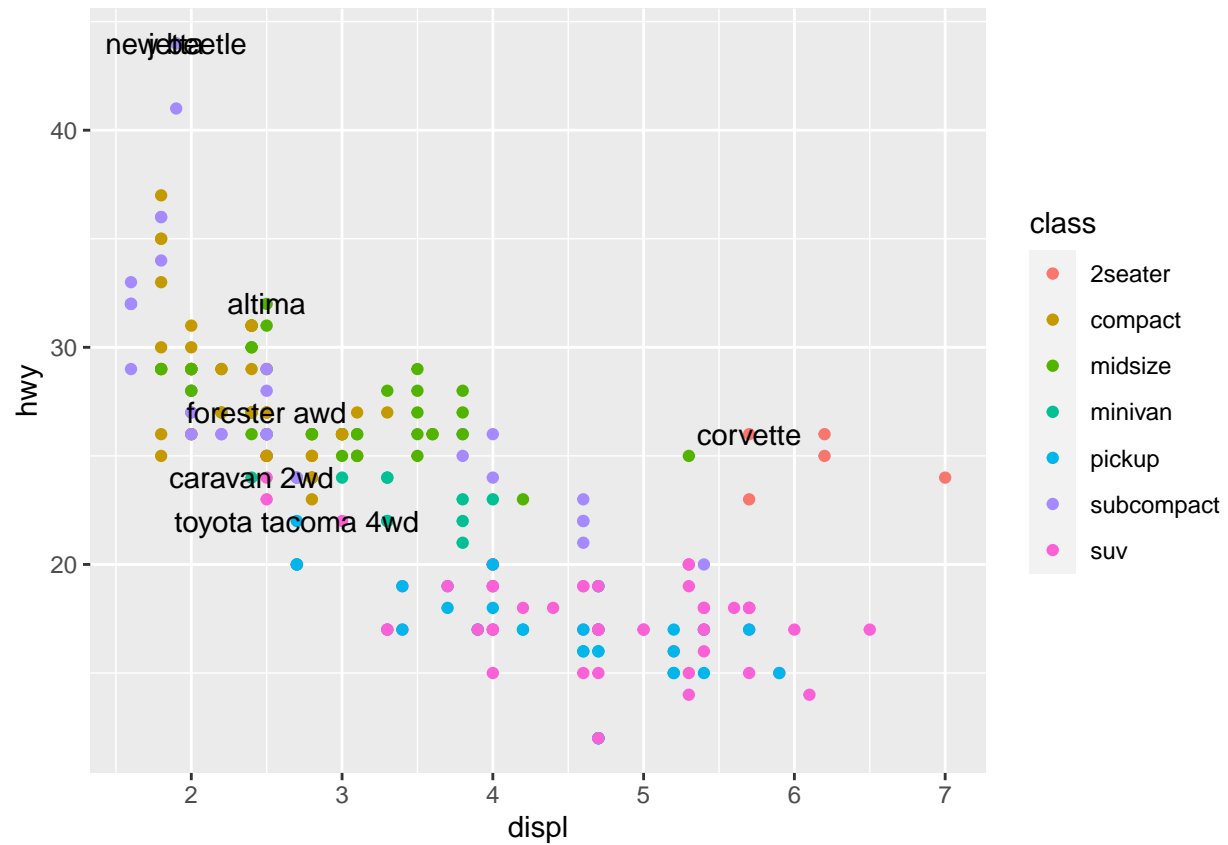
## 'geom\_smooth()' using method = 'loess' and formula 'y ~ x'



-labs() also let's you replace axis & legend titles

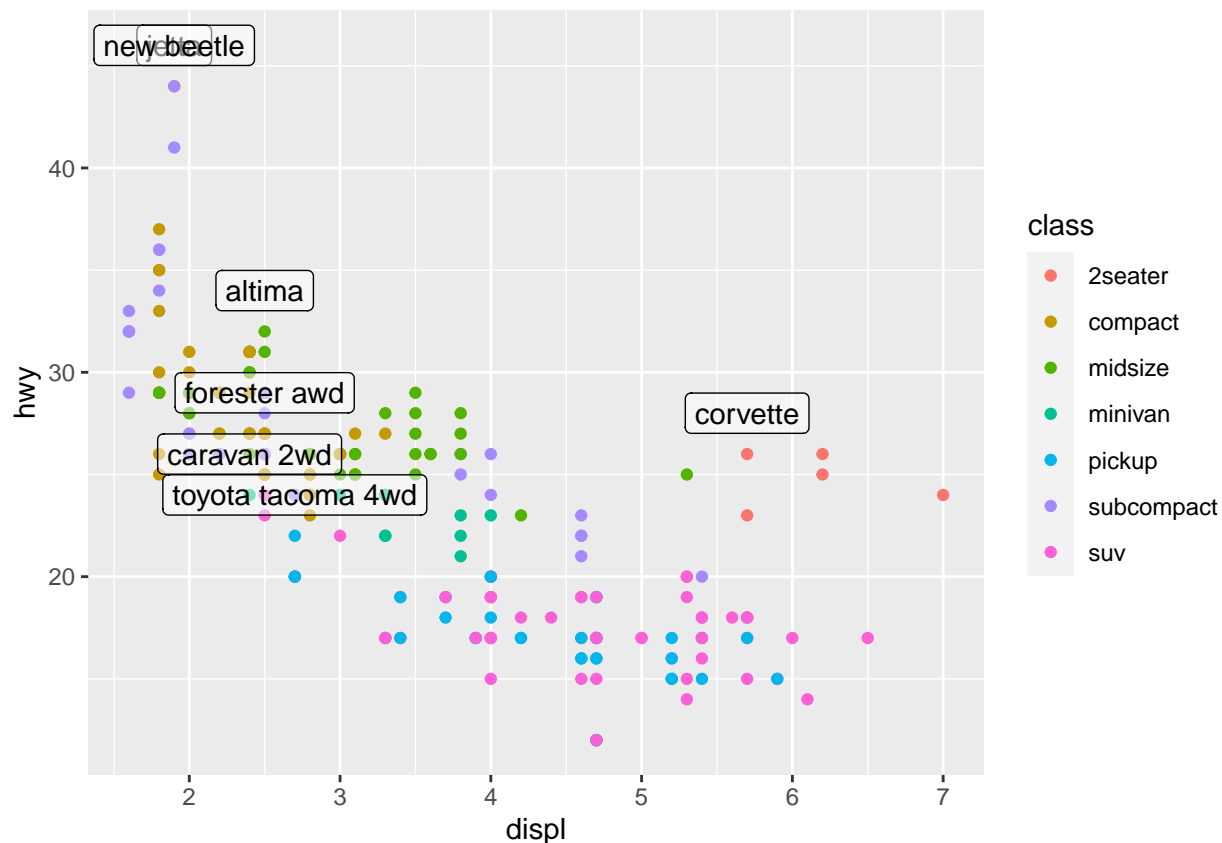
```
best_in_class <- mpg %>%
  group_by(class) %>%
  filter(row_number(desc(hwy)) == 1)

ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(colour = class)) +
  geom_text(aes(label = model), data = best_in_class)
```



-pulls out most efficient car in each class w/ dplyr then labels it -text makes it a little messy with the overlap

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(colour = class)) +
  geom_label(aes(label = model), data = best_in_class, nudge_y = 2, alpha = 0.5)
```



-using label puts rectangles behind text “{r} ggplot(mpg, aes(displ, hwy)) + geom\_point(aes(colour = class)) + geom\_point(size = 3, shape = 1, data = best\_in\_class) + ggrepel::geom\_label\_repel(aes(label = model), data = best\_in\_class)

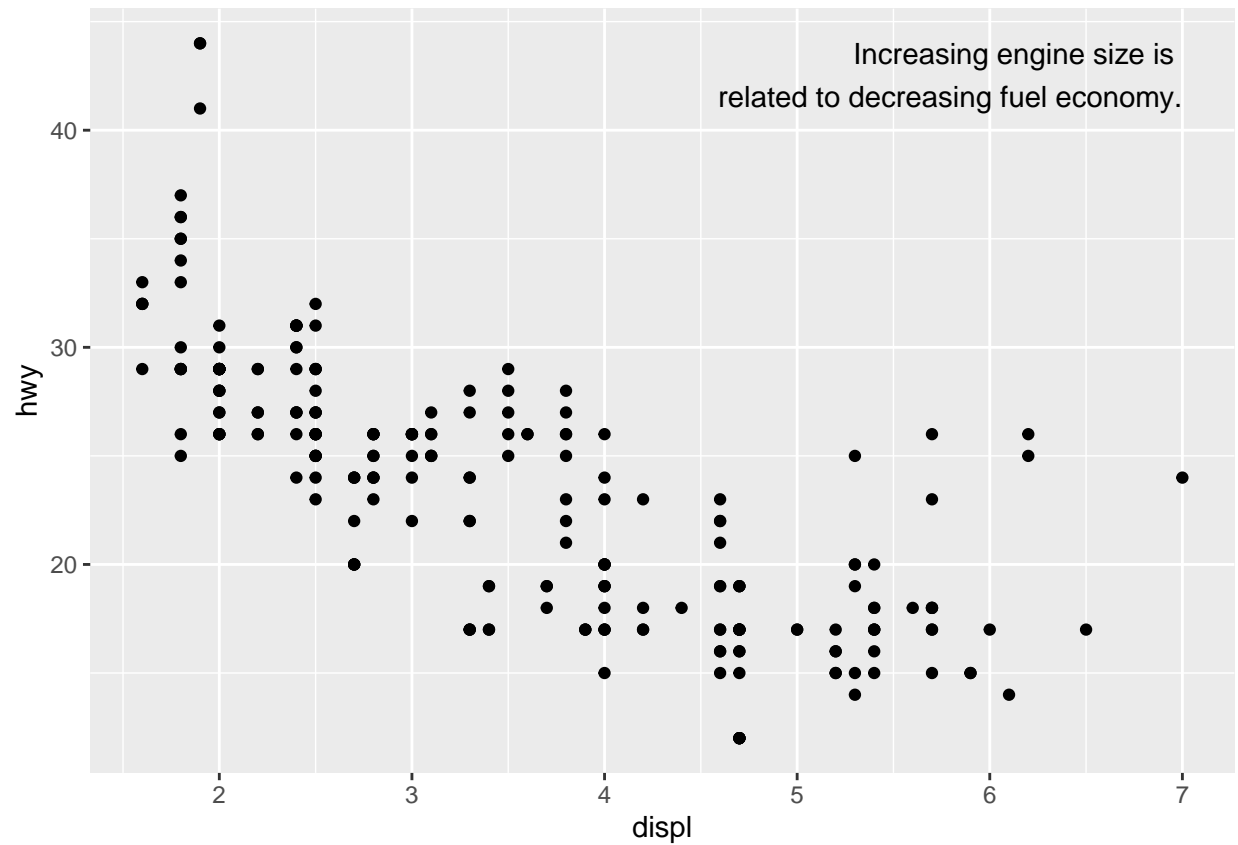
-not working but ggrepel package automatically adjusts labels to remove overlap

```

“{r}
label <- mpg %>%
  summarise(
    displ = max(displ),
    hwy = max(hwy),
    label = "Increasing engine size is \nrelated to decreasing fuel economy."
  )

ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_text(aes(label = label), data = label, vjust = "top", hjust = "right")

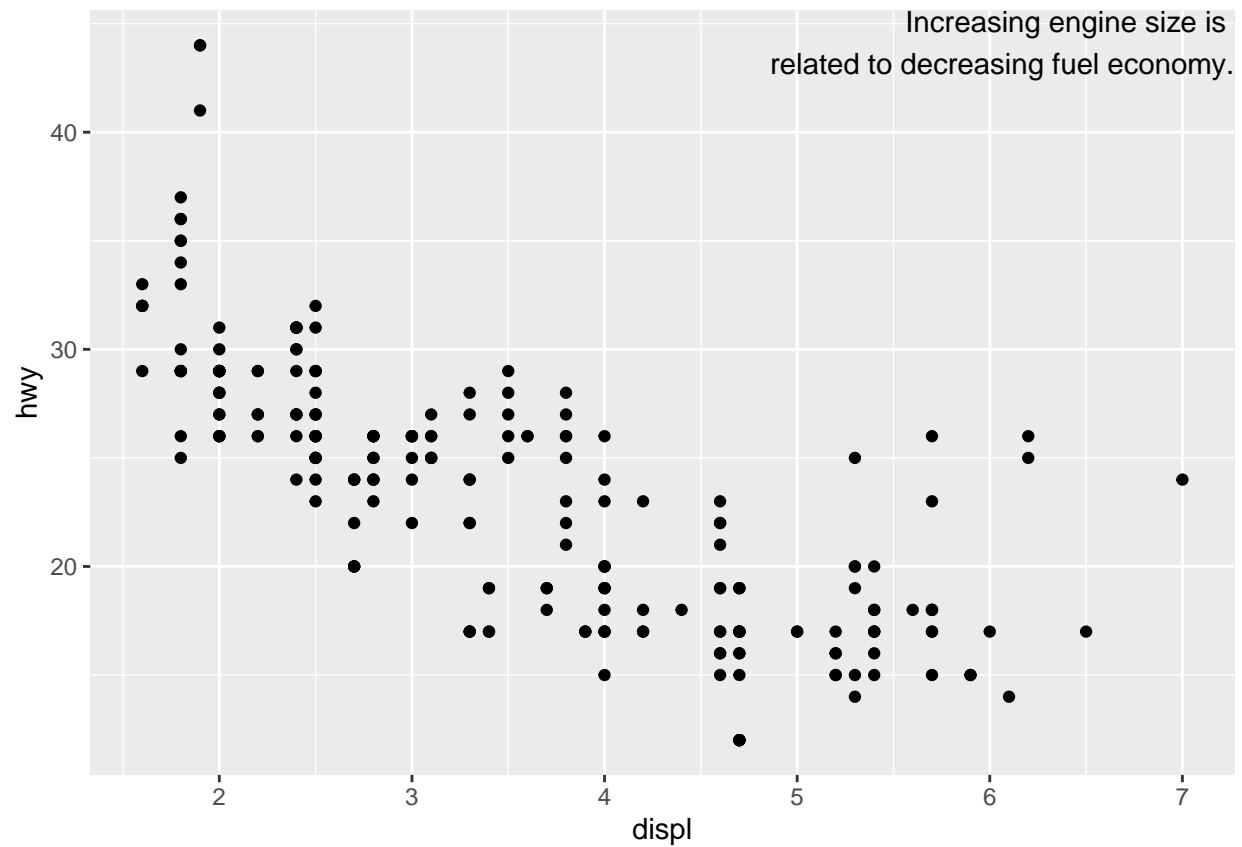
```



-computes maximum value of x and y

```
label <- tibble(
  displ = Inf,
  hwy = Inf,
  label = "Increasing engine size is \nrelated to decreasing fuel economy."
)

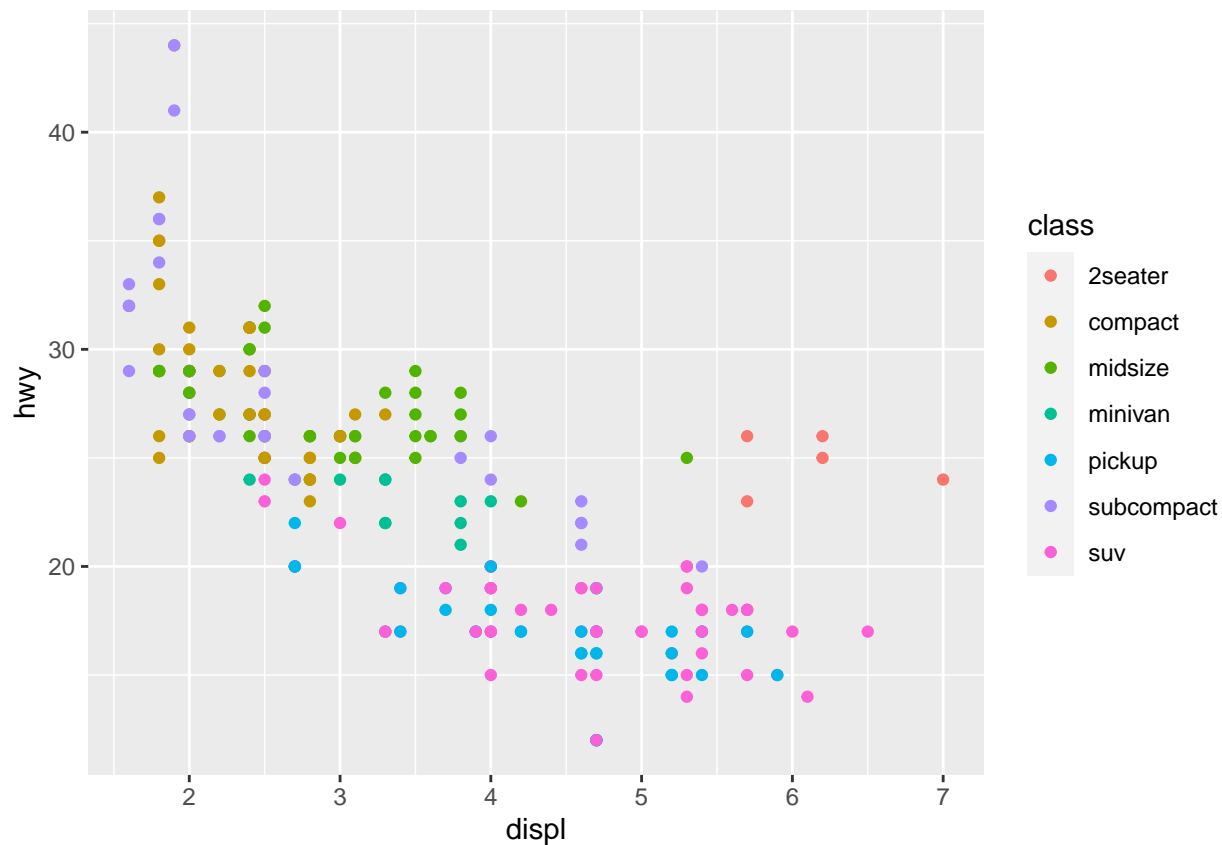
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_text(aes(label = label), data = label, vjust = "top", hjust = "right")
```



-moves text exactly on borders of plot using +Inf and -Inf

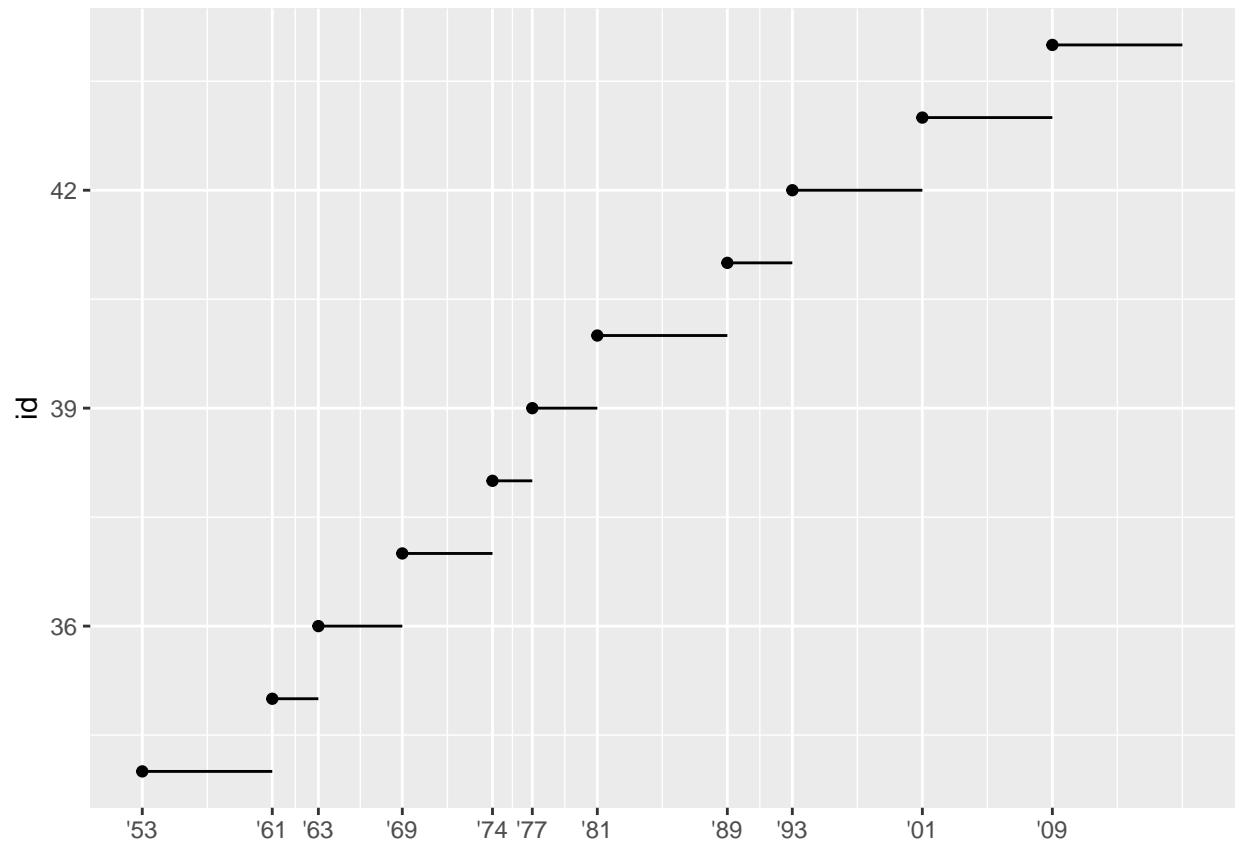
```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(colour = class))
```





-ggplot2 automatically adds scales for you -breaks controls position of ticks or values associated w/ keys  
 -labels control text associated w/ each tick/key -breaks & labels can also control appearance of legends  
 -breaks can be used when there's relatively little data & want to highlight exactly where observations occur

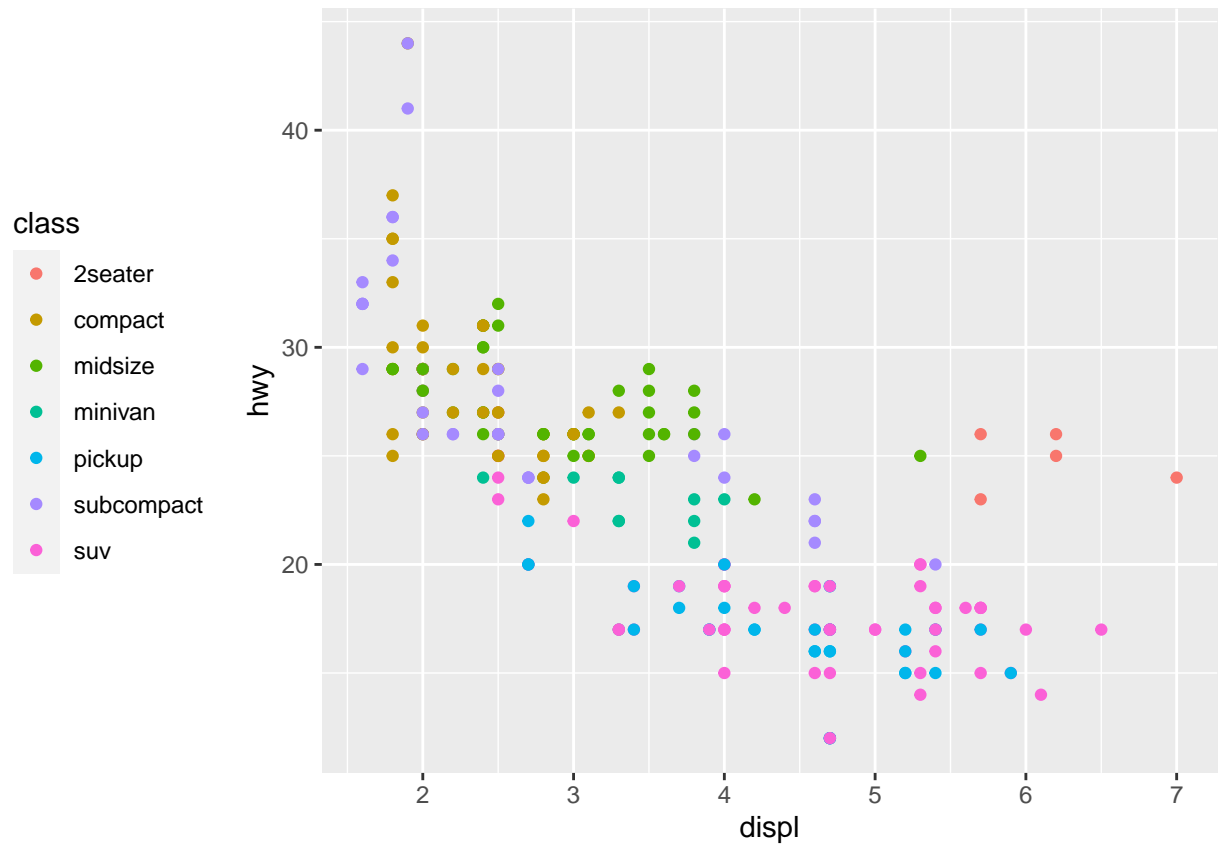
```
presidential %>%
  mutate(id = 33 + row_number()) %>%
  ggplot(aes(start, id)) +
    geom_point() +
    geom_segment(aes(xend = end, yend = id)) +
    scale_x_date(NULL, breaks = presidential$start, date_labels = "'%y")
```



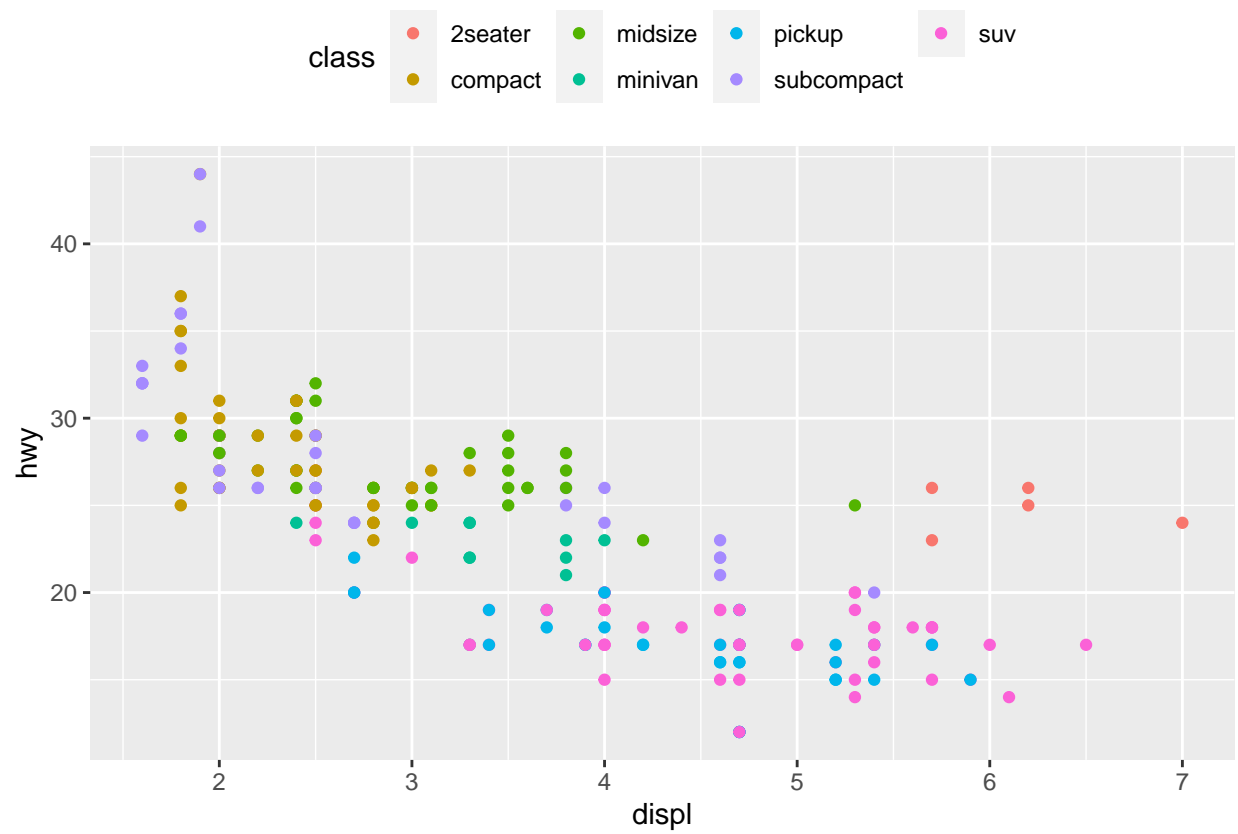
-theme() setting can be used to control overall position of legend

```
base <- ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(colour = class))

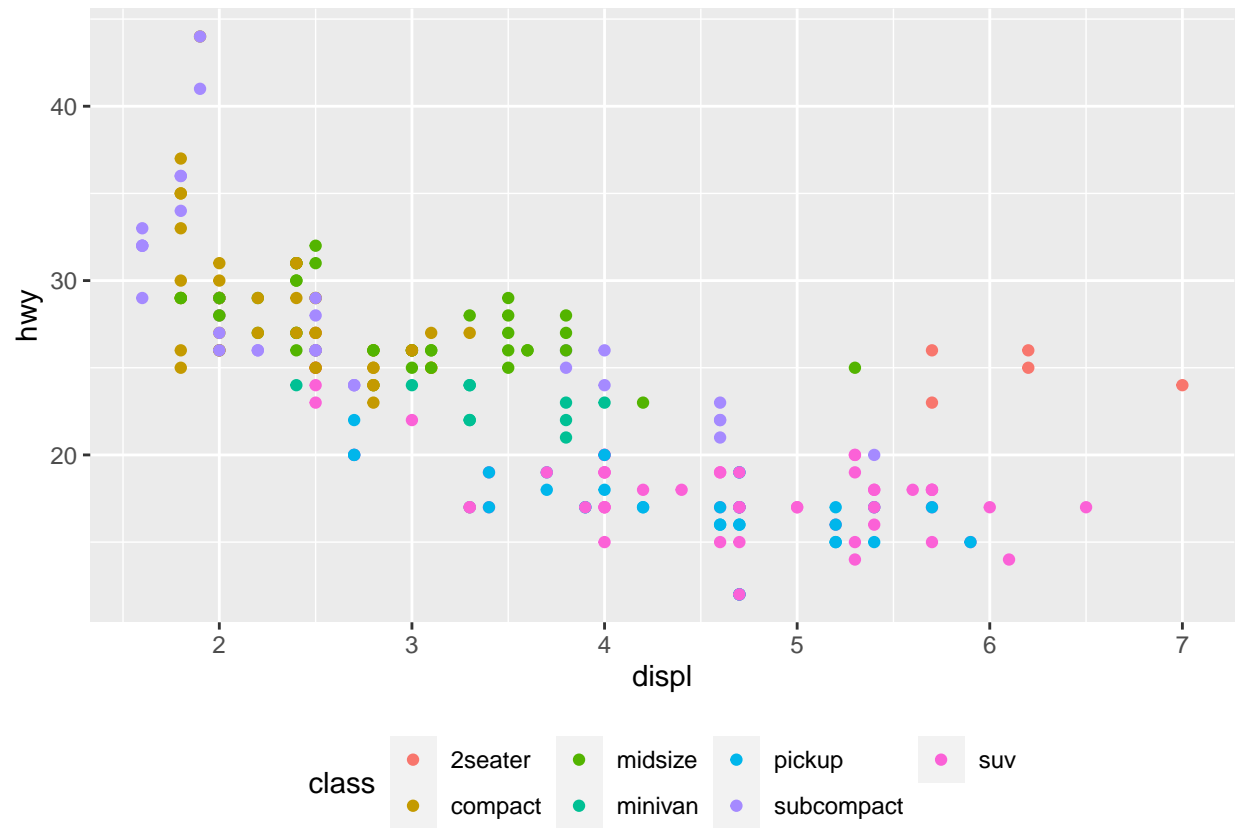
base + theme(legend.position = "left")
```



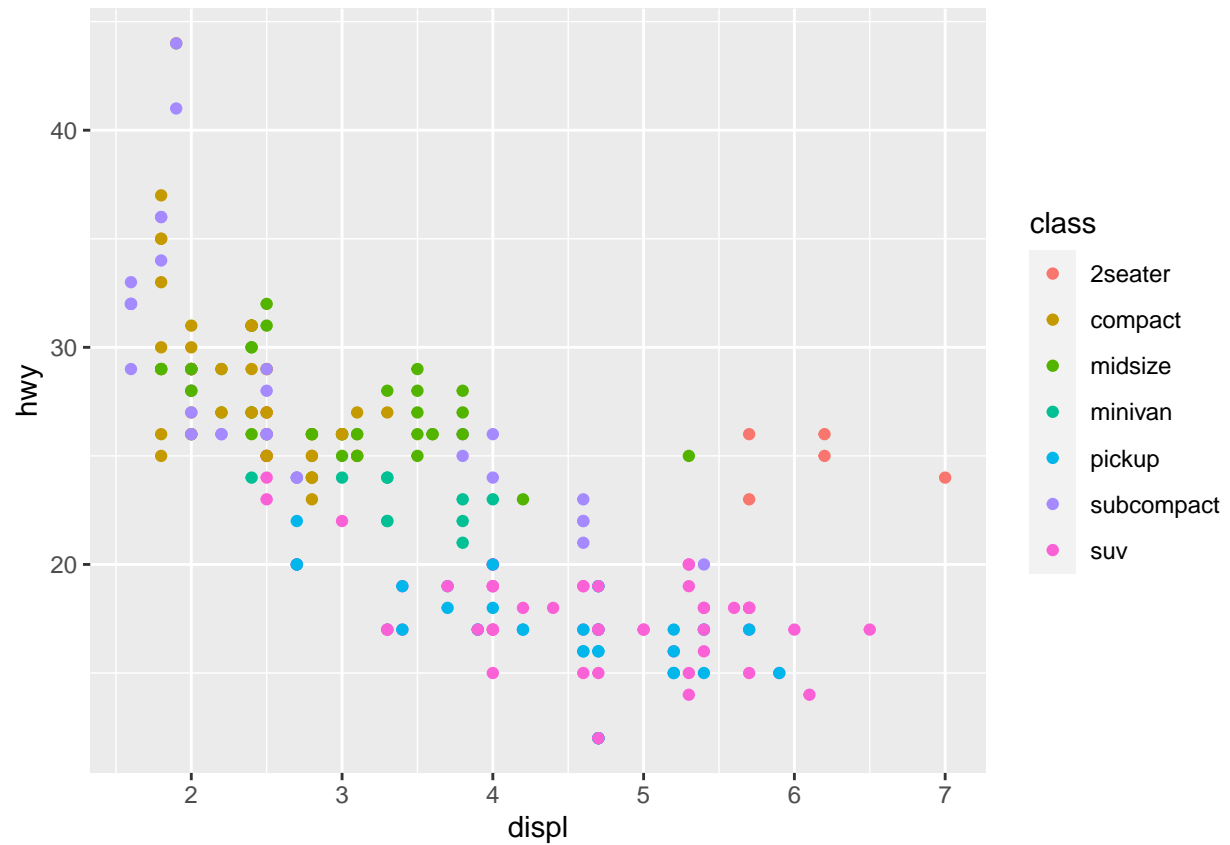
```
base + theme(legend.position = "top")
```



```
base + theme(legend.position = "bottom")
```

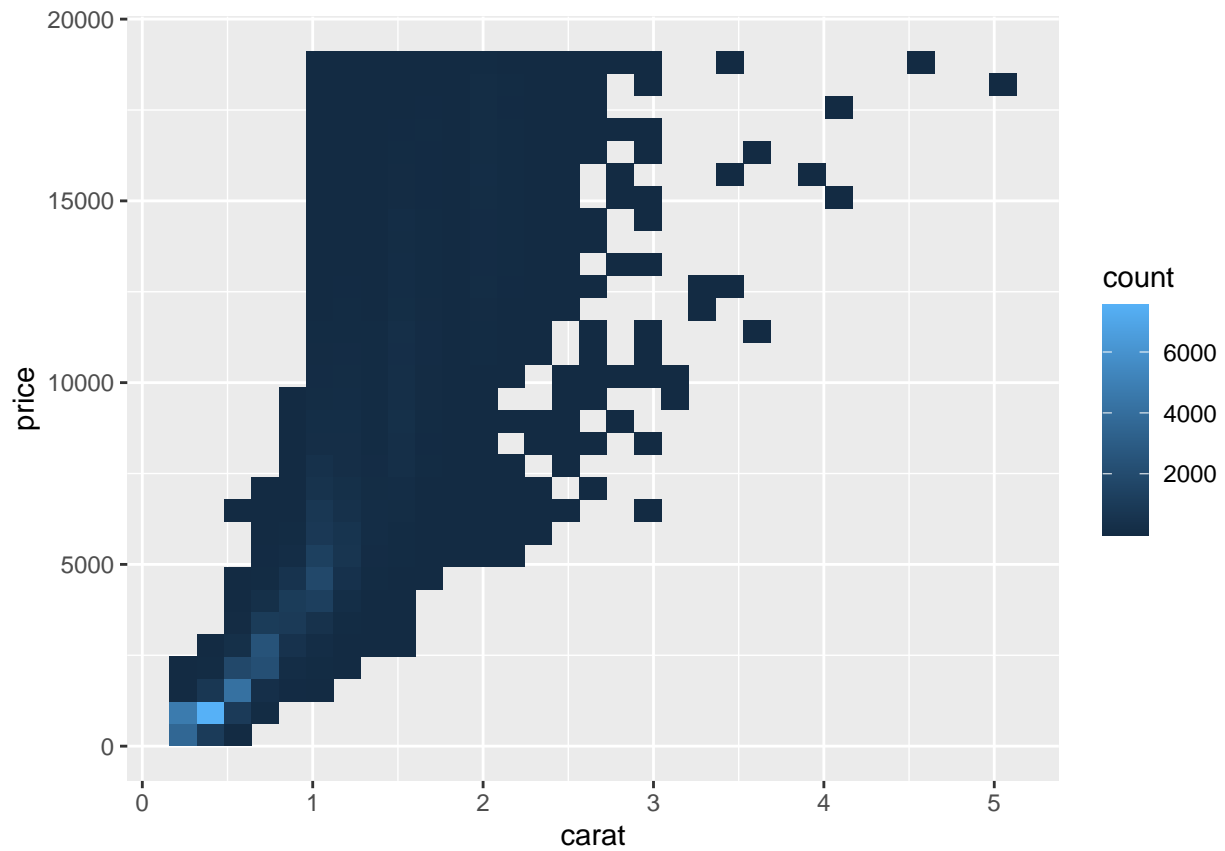


```
base + theme(legend.position = "right") # the default
```

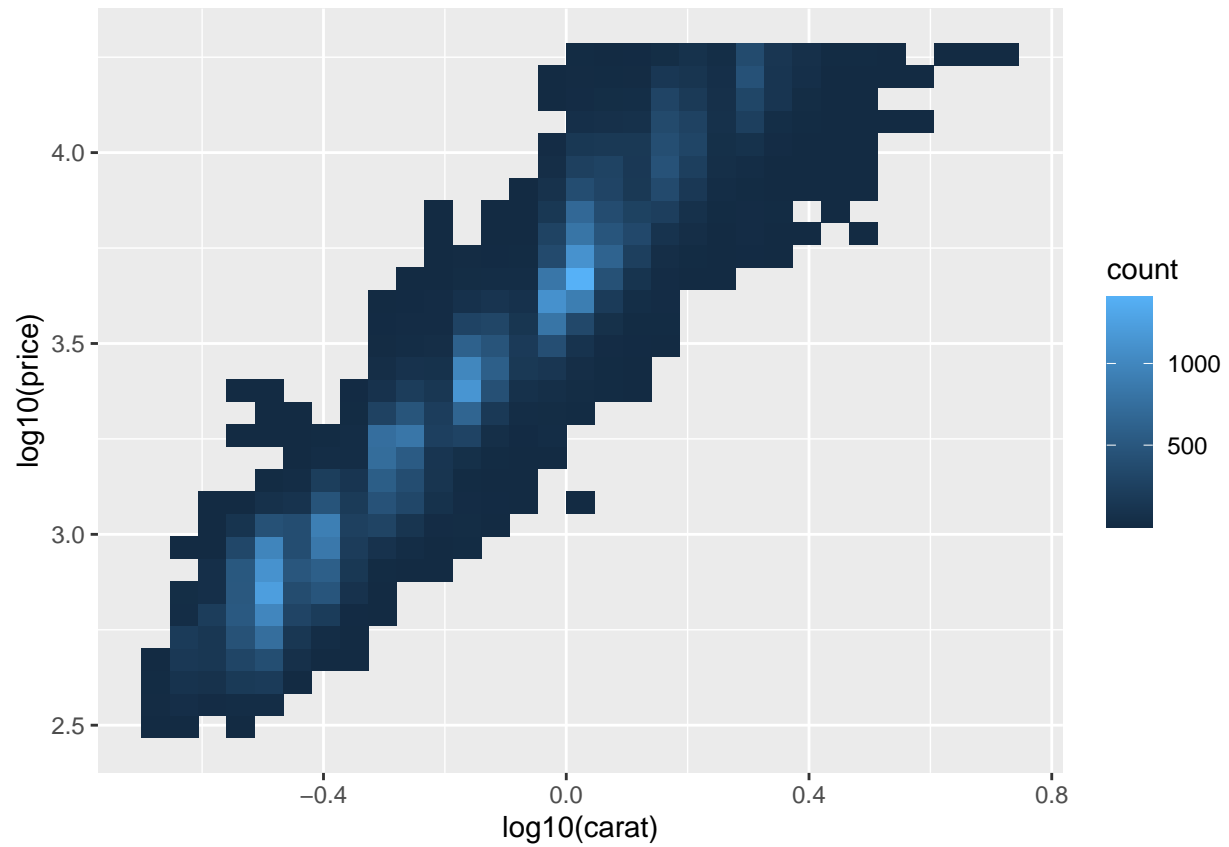


-legend.position = "none" would remove the legend  
 replacing a scale:

```
ggplot(diamonds, aes(carat, price)) +  
  geom_bin2d()
```



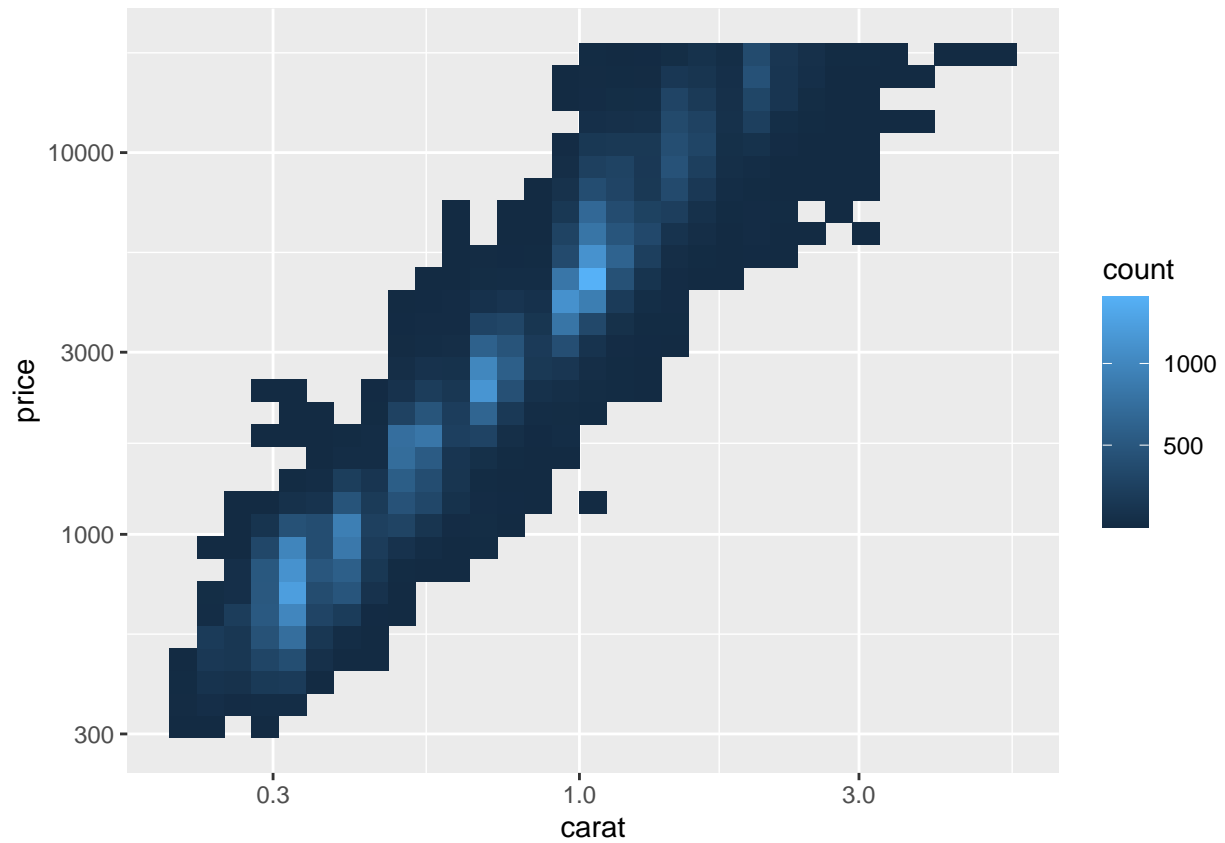
```
ggplot(diamonds, aes(log10(carat), log10(price))) +  
  geom_bin2d()
```



-here a log transformation performed -slight problem: axis labels changed

```
ggplot(diamonds, aes(carat, price)) +  
  geom_bin2d() +  
  scale_x_log10() +  
  scale_y_log10()
```

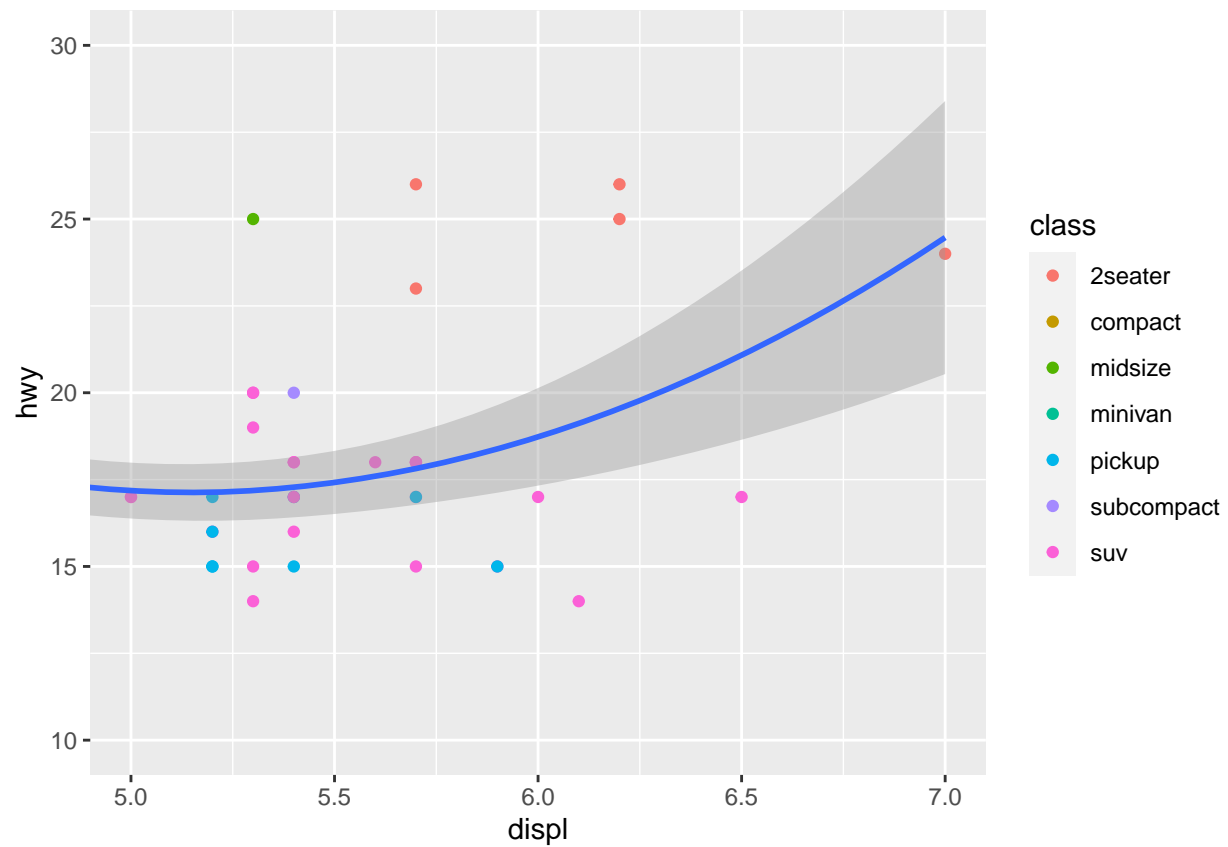




-axis now have original labels

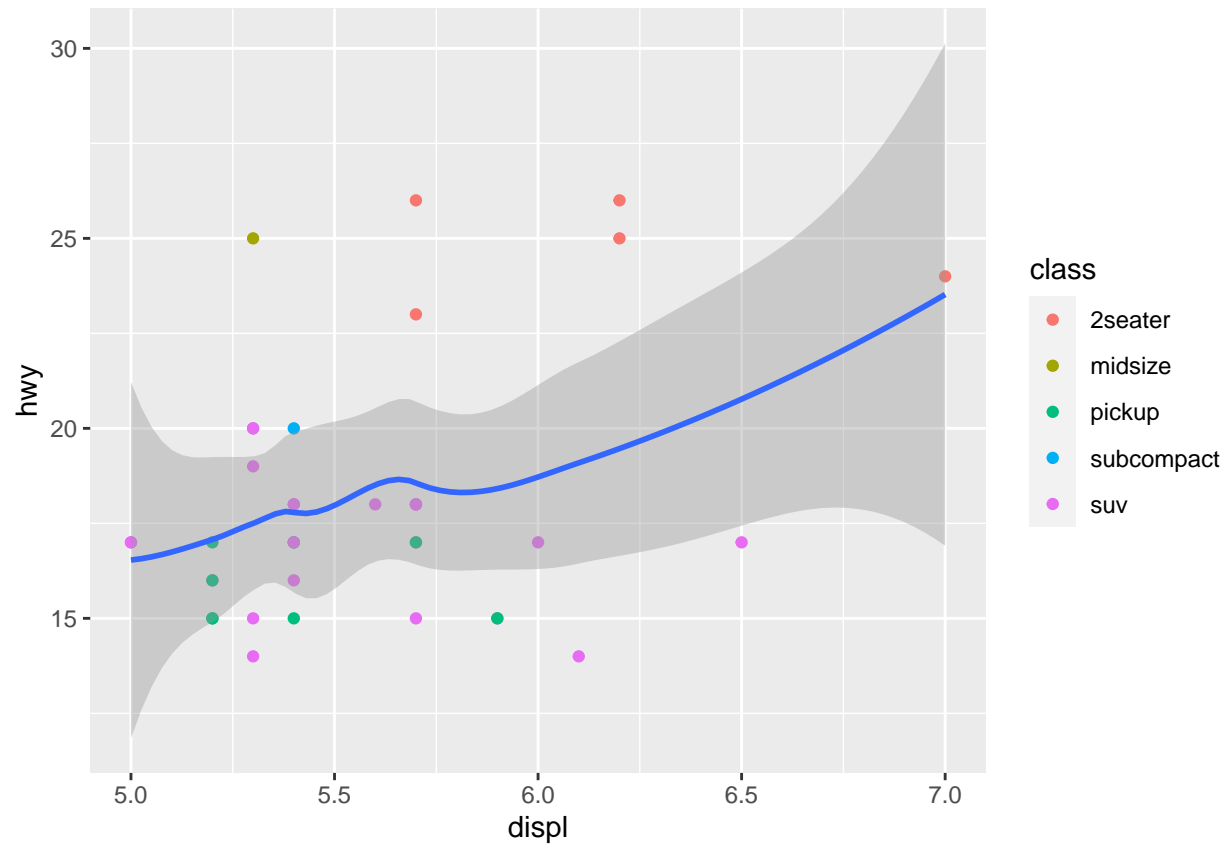
```
ggplot(mpg, mapping = aes(displ, hwy)) +  
  geom_point(aes(color = class)) +  
  geom_smooth() +  
  coord_cartesian(xlim = c(5, 7), ylim = c(10, 30))
```

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



```
mpg %>%  
  filter(displ >= 5, displ <= 7, hwy >= 10, hwy <= 30) %>%  
  ggplot(aes(displ, hwy)) +  
  geom_point(aes(color = class)) +  
  geom_smooth()
```

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

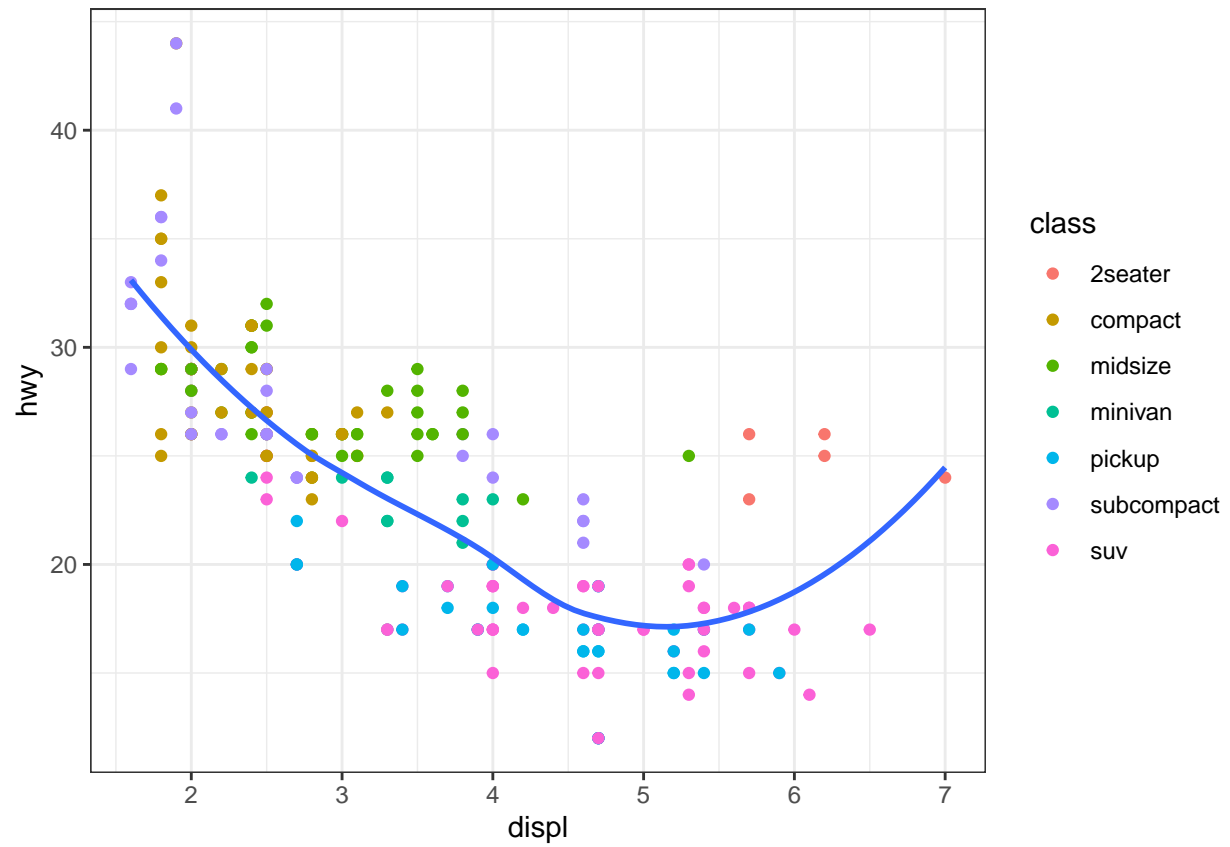


-3 ways to control plot limits -adjusting what data plotted -setting limits in each scale -setting xlim & ylim in coord\_cartesian()

themes:

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(se = FALSE) +
  theme_bw()
```

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



-allows you to customize non-data elements in plot w/ a theme