01-ggplot

조근수

2020 12 22

ggplot을 활용한 데이터 시각화

가장 먼저 필요한 패키지를 불러온다

```
library(tidyverse)
```

- tidyverse는 다양한 패키지를 포함한다.(**dplyr**, **ggplot2**, **haven** 등)

데이터 불러오기(diamonds)

```
data(diamonds)
diamonds %>% head()
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

- 총 10개의 변수로 구성되어있다.

데이터 불러오기(mpg)

```
data(mpg)
mpg %>% head()
```

```
## # A tibble: 6 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(l5)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

- 총 11개의 변수로 구성되어있다.
- **displ** 변수는 자동차 엔진의 크기
- **hwy** 변수는 연료 효율성 (miles per gallon)

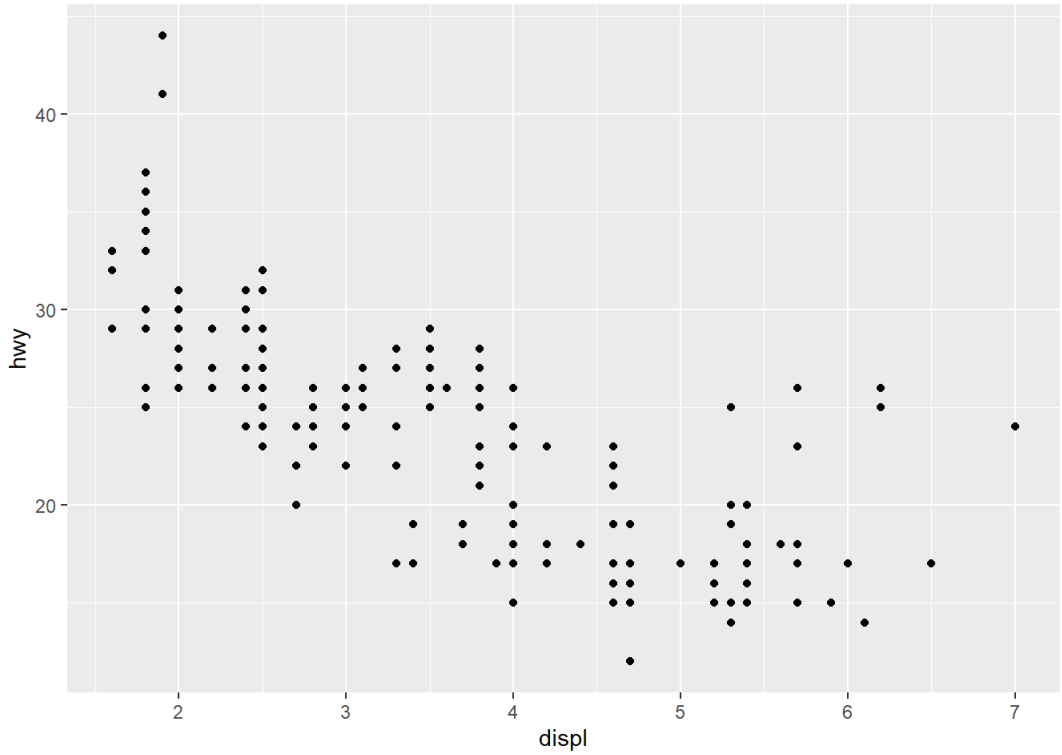
geom_point 활용

그래프 상에 점들을 출력하는 함수

산점도 그리기

displ 를 x축,**hwy** 를 y축

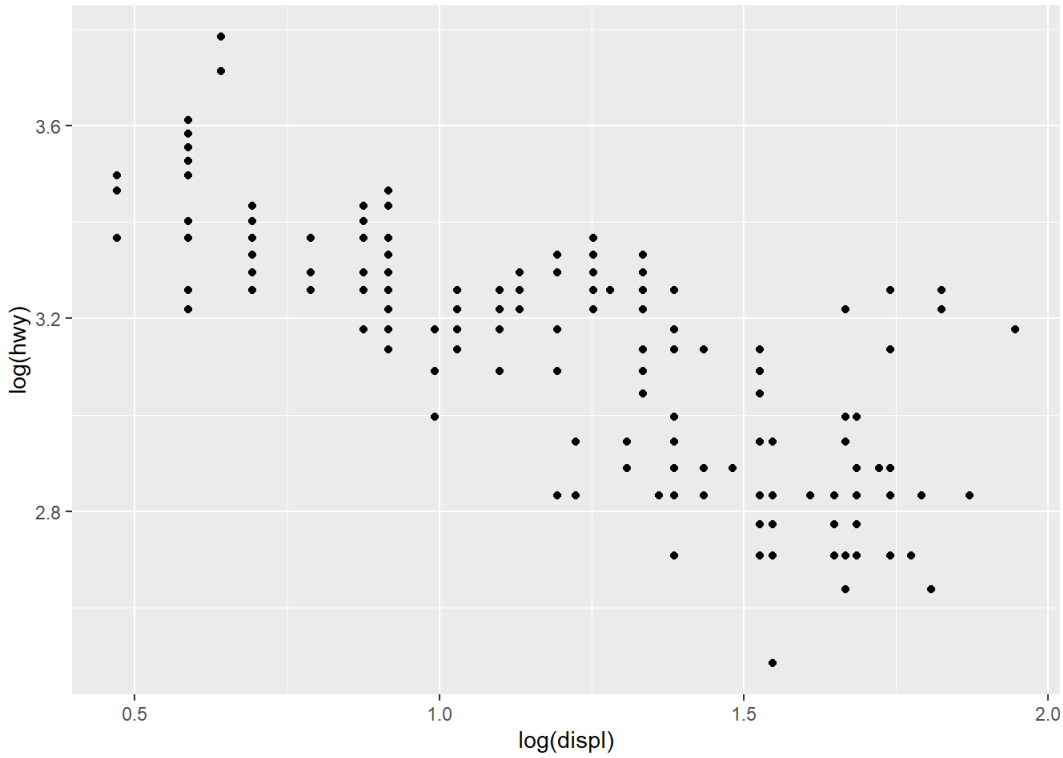
```
ggplot(data=mpg)+
geom_point(mapping = aes(x = displ, y = hwy))
```



- 두 변수간의 **음의 상관관계** 가 보인다

x, y 축을 log scaling

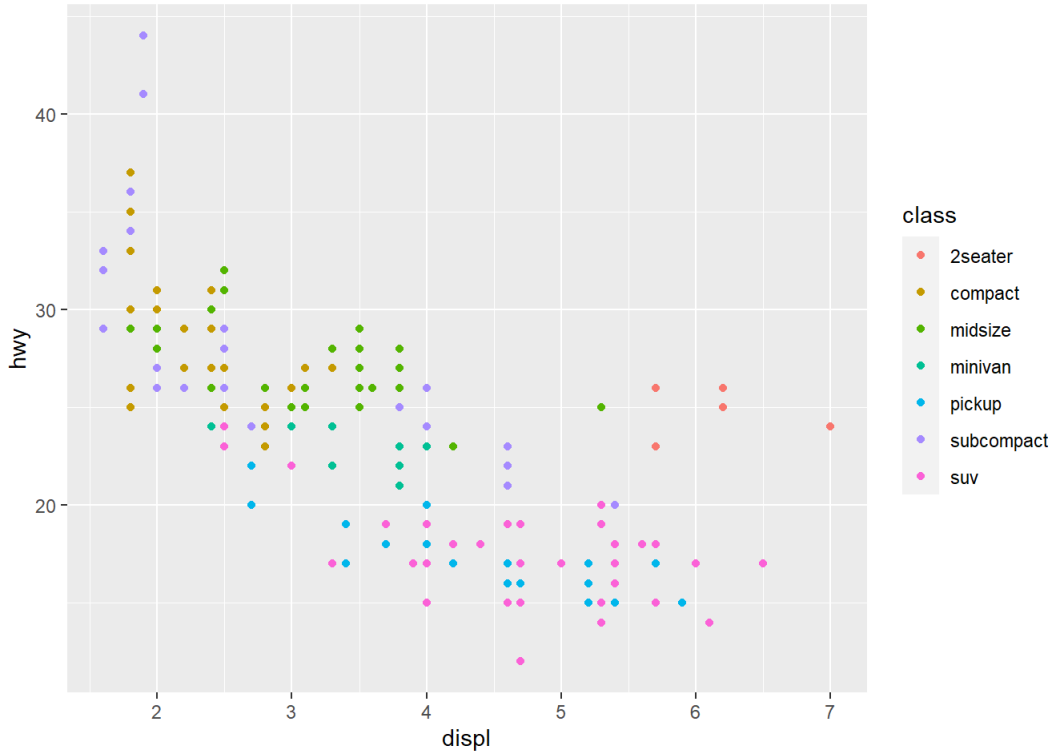
```
ggplot(data=mpg)+
geom_point(mapping = aes(x = log(displ), y = log(hwy)))
```



- mapping의 x,y 값에 log를 취한다
- 역시 음의 상관관계

그룹별 색상(color) 지정

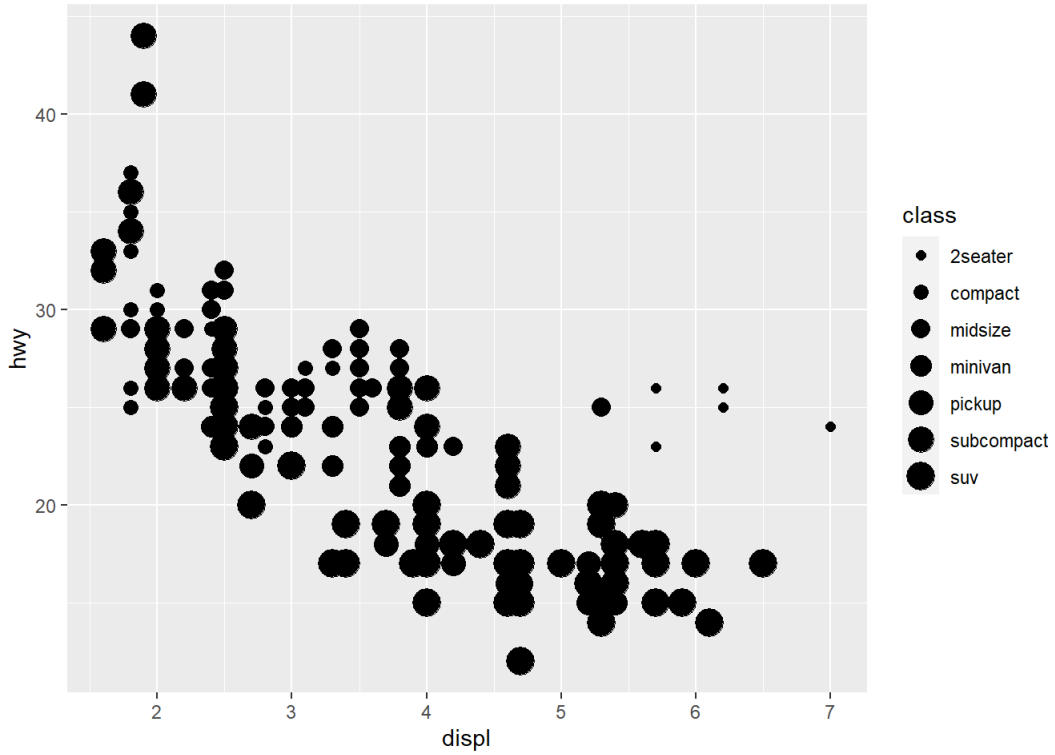
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



그룹별 크기(size) 지정

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

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

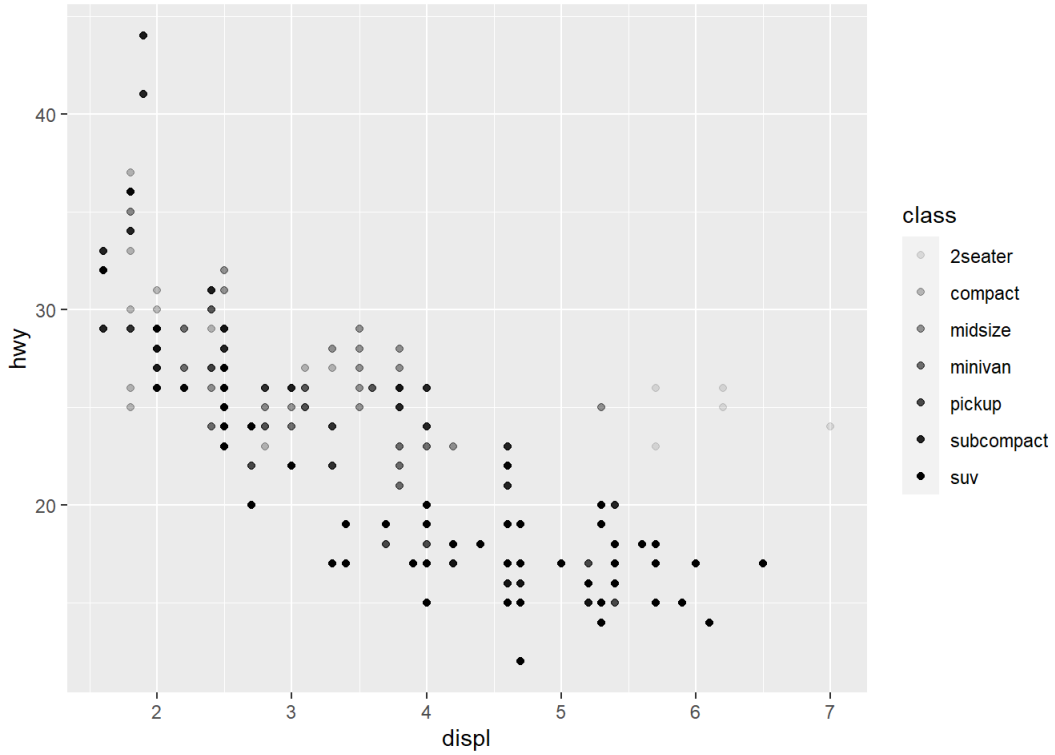


- **class** 가 order인 경우 의미가 있을 듯

그룹별 투명도(alpha) 지정

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

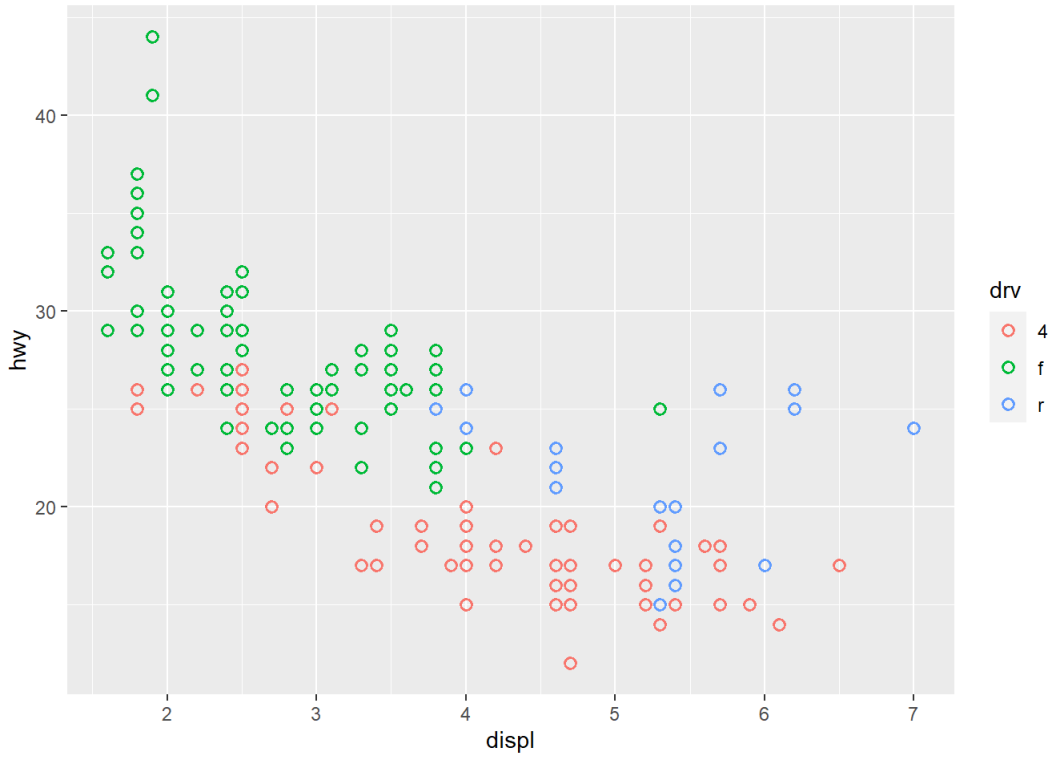
```
## Warning: Using alpha for a discrete variable is not advised.
```



- **class** 가 order인 경우 의미가 있을 듯

다양한 point 형태

```
mpg %>% ggplot(aes(displ, hwy, group=class)) +
  geom_point(aes(color=drv), fill=20, stroke=1, size=2, shape=1)
```



geom_tile 활용

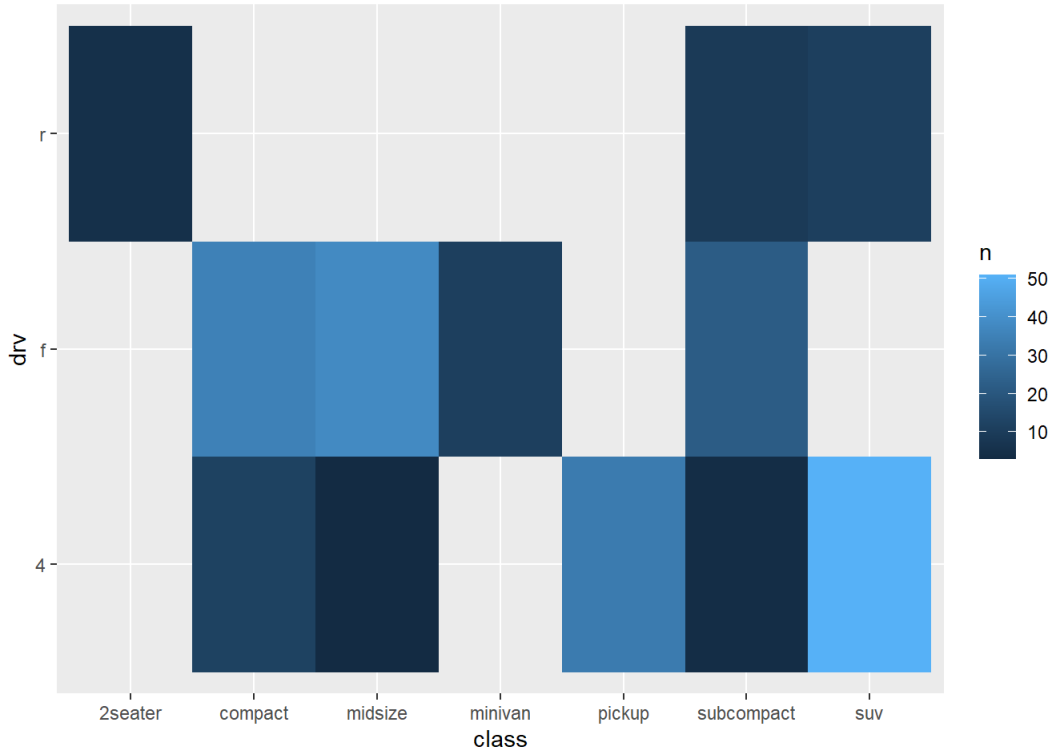
타일(tile) 모양의 그래프 출력 **count()** 함수로 그룹별 count

```
mpg %>% count(class, drv)
```

```
## # A tibble: 12 x 3
##    class      drv       n
##    <chr>      <chr> <int>
##  1 2seater    r         5
##  2 compact    4        12
##  3 compact    f        35
##  4 midsize    4         3
##  5 midsize    f        38
##  6 minivan    f        11
##  7 pickup     4        33
##  8 subcompact 4         4
##  9 subcompact f        22
## 10 subcompact r         9
## 11 suv        4        51
## 12 suv        r        11
```

- class, drv 변수는 모두 categorical variable

```
mpg %>% count(class, drv) %>%
  ggplot(aes(x=class, y=drv)) + geom_tile(mapping = aes(fill = n))
```



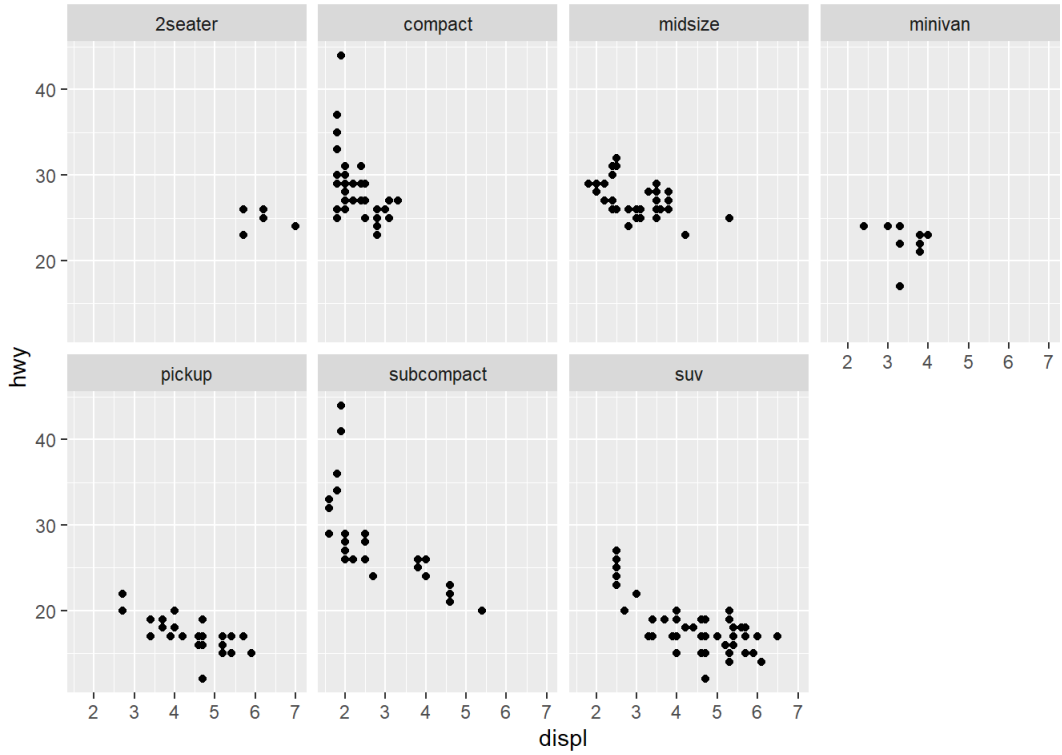
facet 활용

앞서 geom_point의 color나 alpha를 조정해주며 하나의 그래프에 여러 그룹을 구분해주었다. **facet** 을 활용하면 그룹별 그래프를 따로 출력할 수 있다.

facet_wrap

한개 의 그룹일 경우 사용

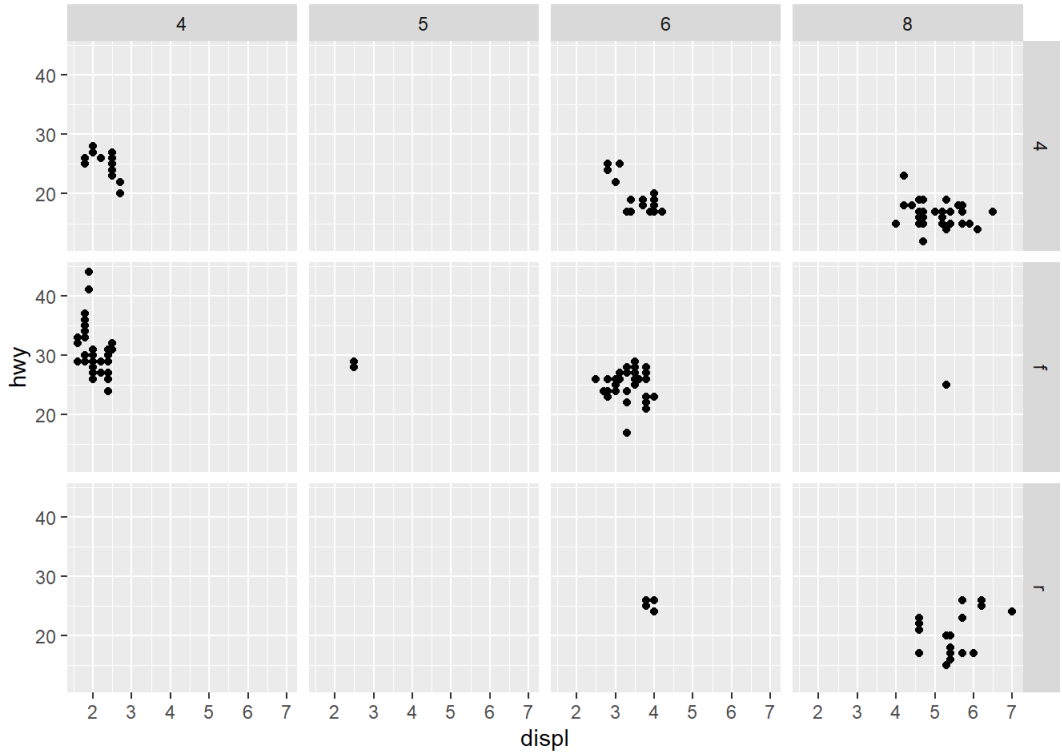
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
facet_wrap(~ class, nrow = 2)
```



- facet_wrap(~class) 대신 facet_grid(.~class) ### facet_grid

두개 의 그룹일 경우 사용

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

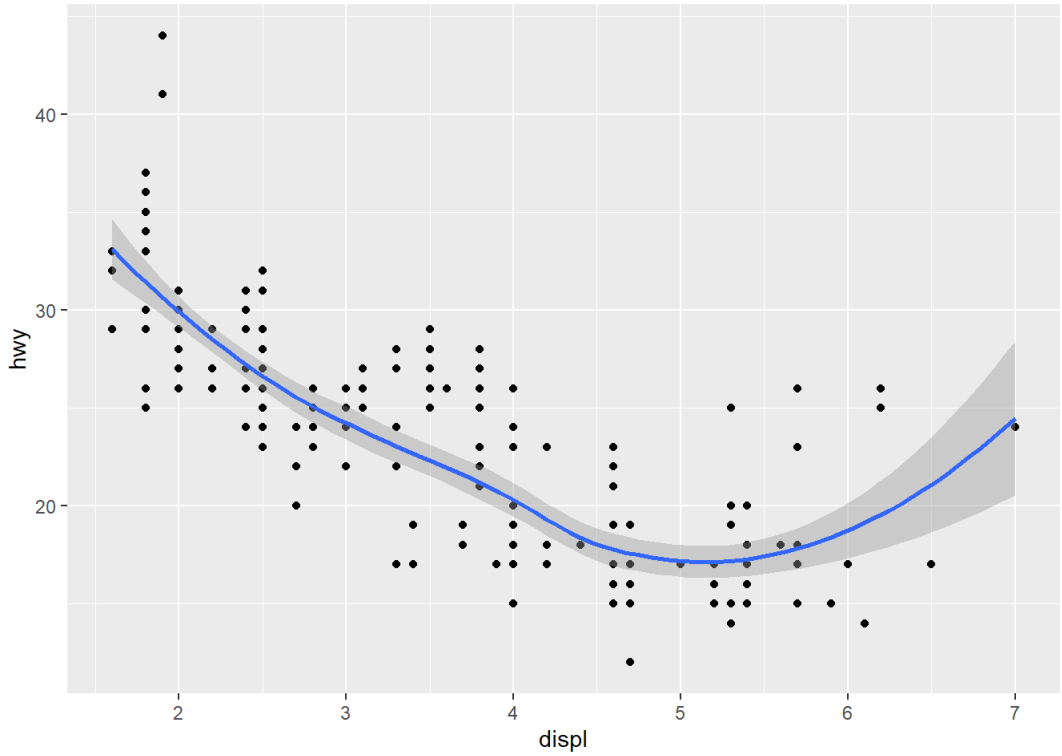


geom_smooth 활용

추세선을 구할때 사용된다.
geom_point() 와 함께 자주 사용된다.

```
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'
```

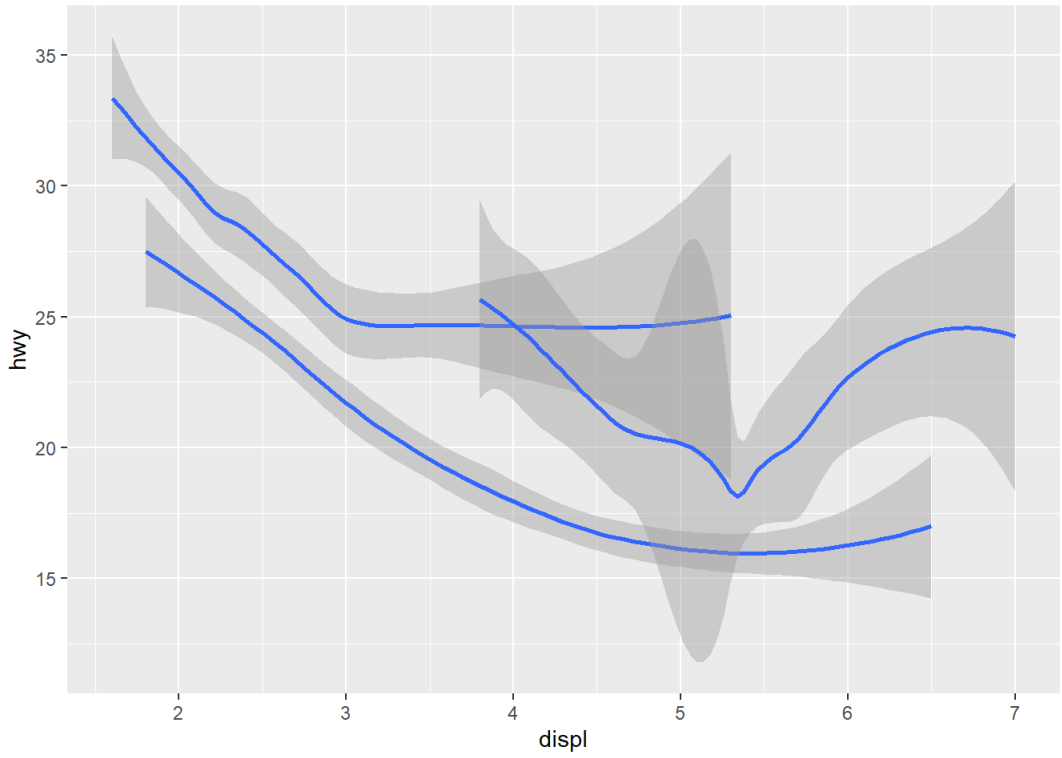


- **geom_point()** 와 함께 사용한 기본적인 형태
- point와 smooth에 mapping이 모두 들어갔다.
- 위의 경우는 mapping이 동일하므로 ggplot() 내에 mapping 해주어도 실행된다.

그룹별 smooth

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

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

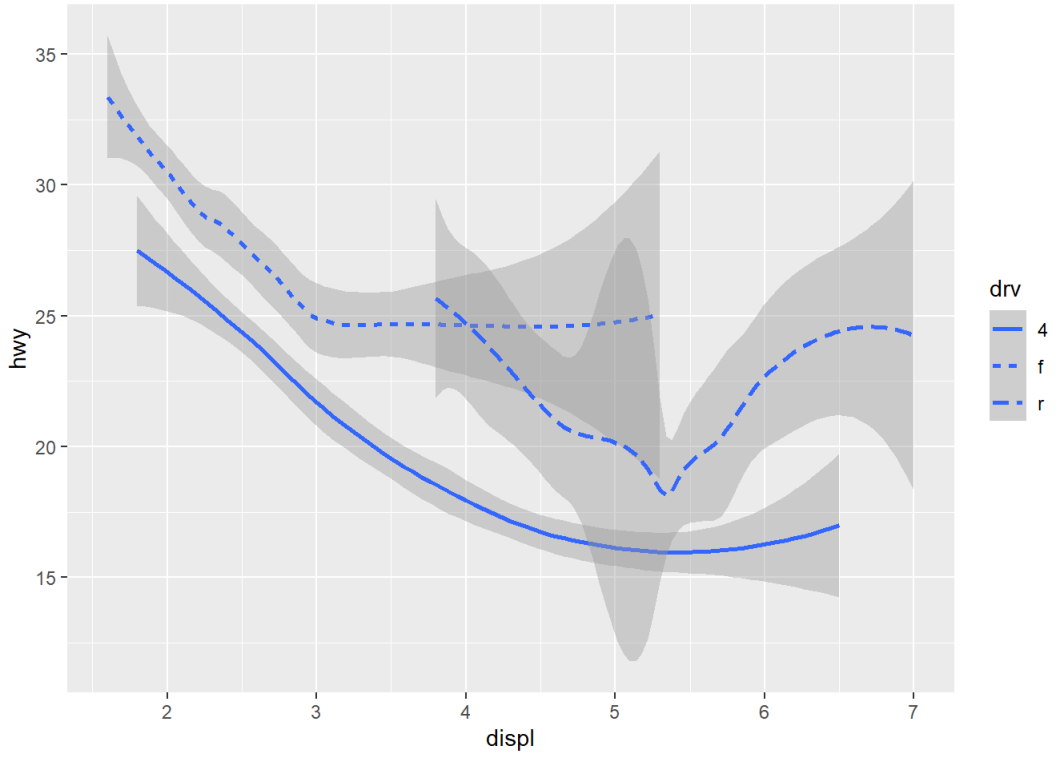


- 위의 결과는 어떤 group의 smooth인지 알 수 없다.

linetype으로 그룹 구별

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

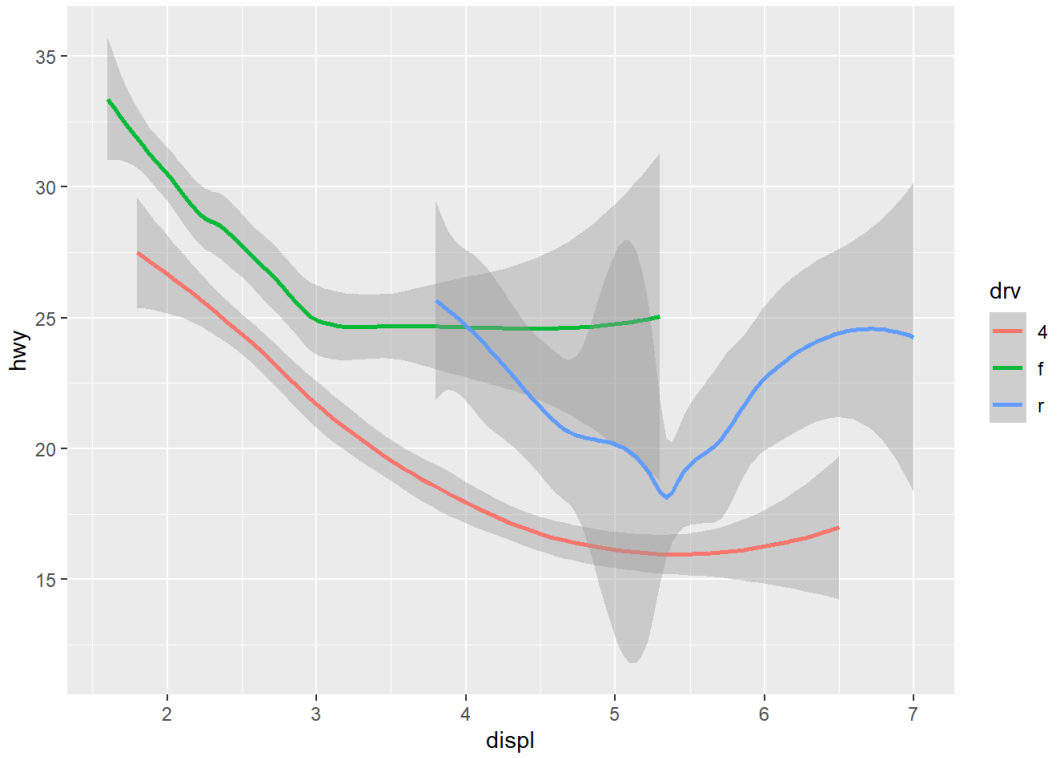
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



color로 그룹 구별

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

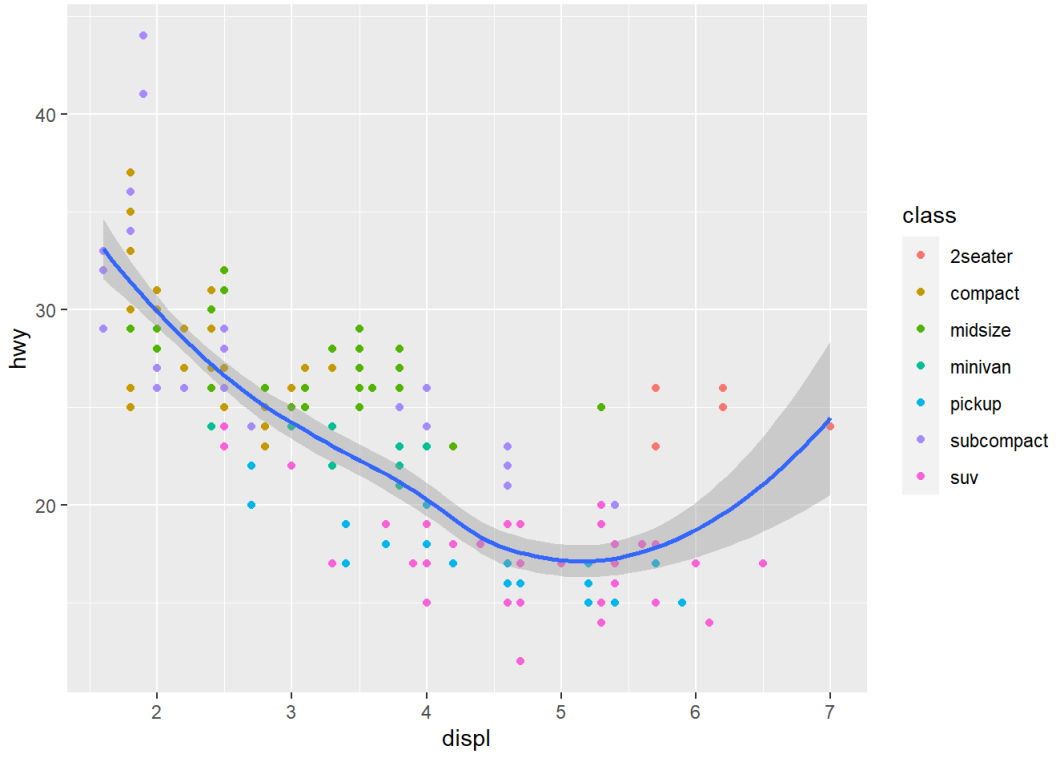
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



point를 group 별로 구분

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

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

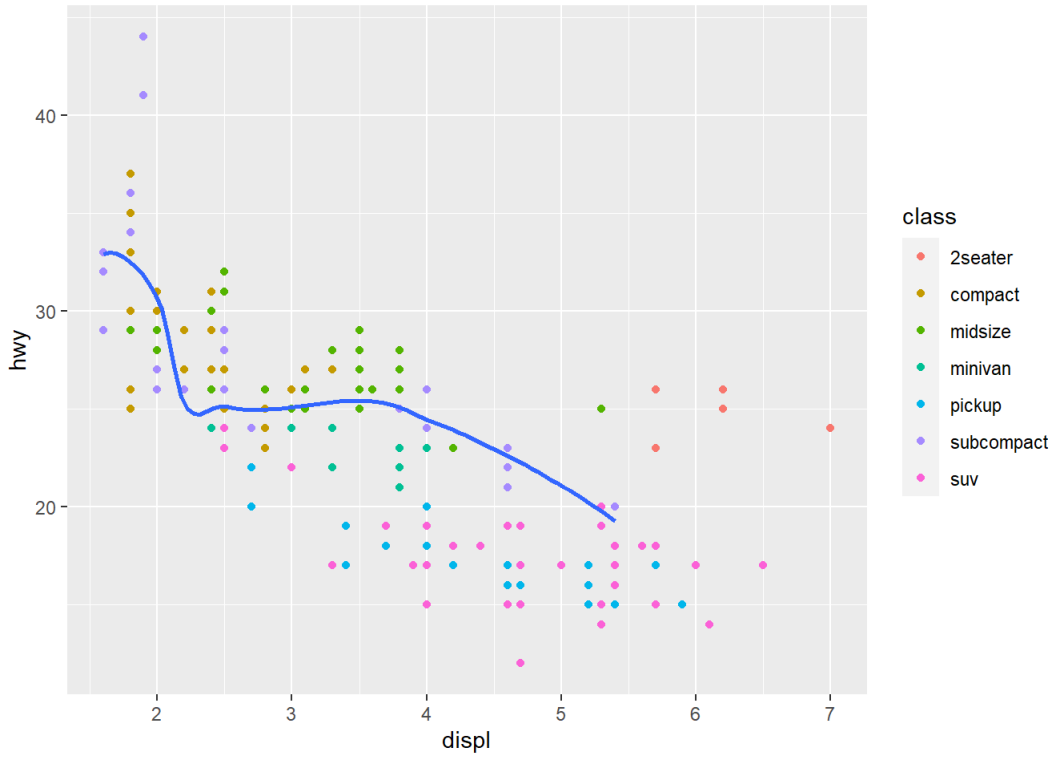


- show.legend = TRUE 는 범례 출력

특정 group에 대해서만 추세선 & SE(standard error) 제거

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

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

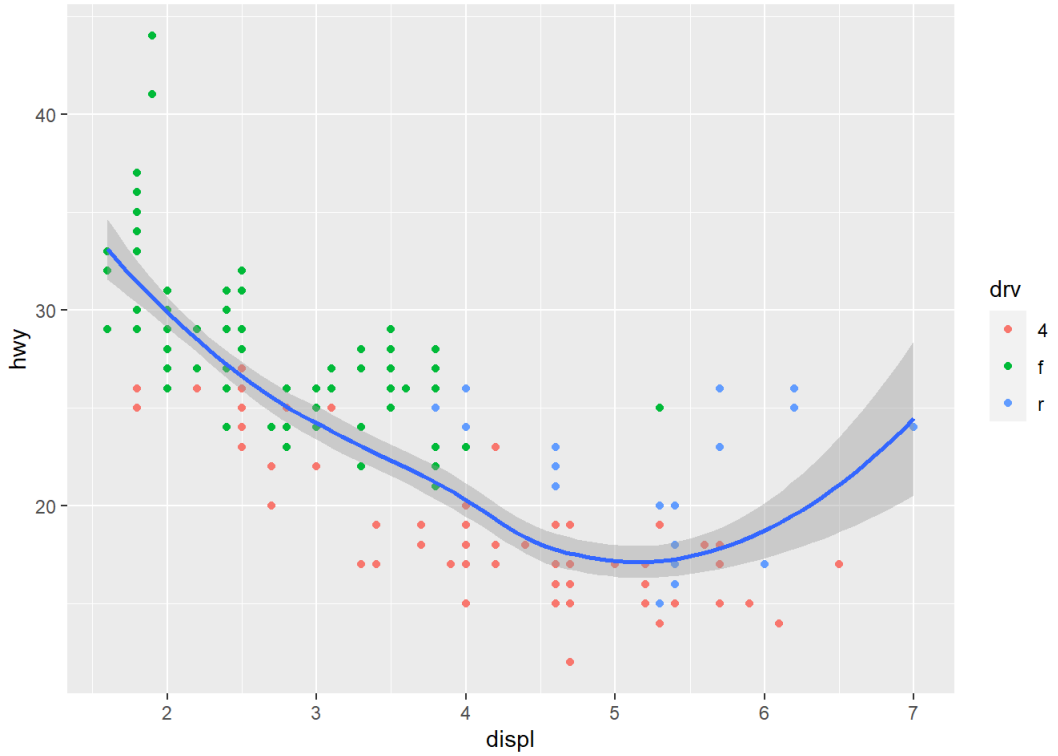


- **filter** 를 이용해 원하는 class 추출
- **se = F** 를 이용해 표준오차 제거

다양한 종류의 그래프

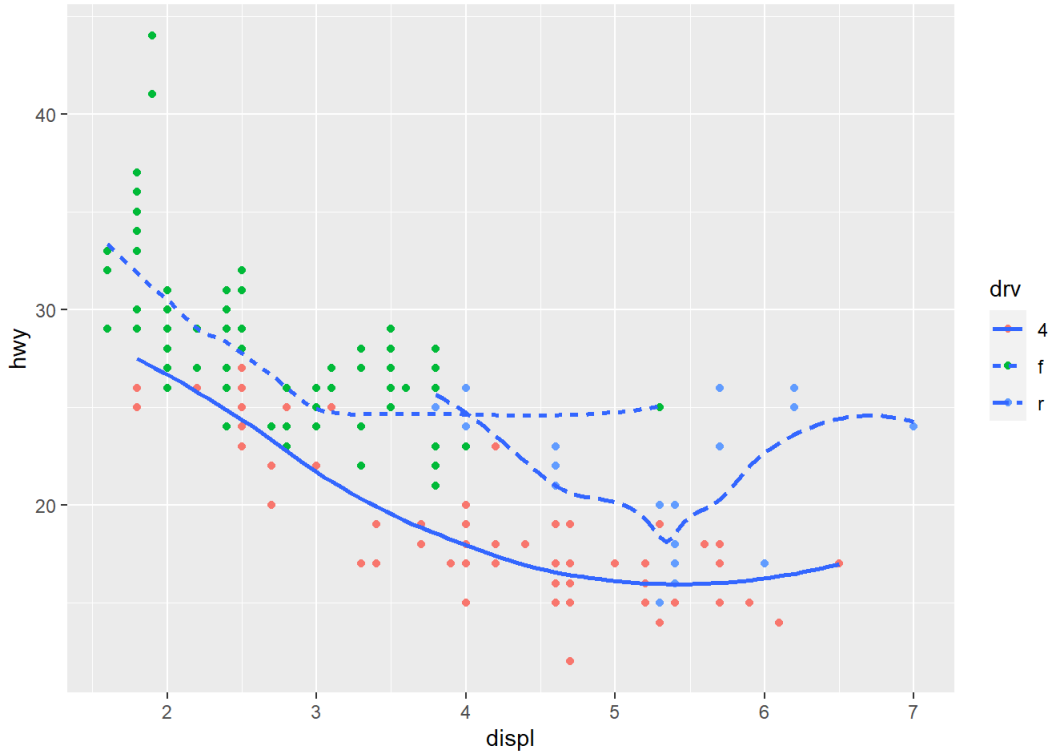
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv), show.legend = TRUE) +
  geom_smooth()
```

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

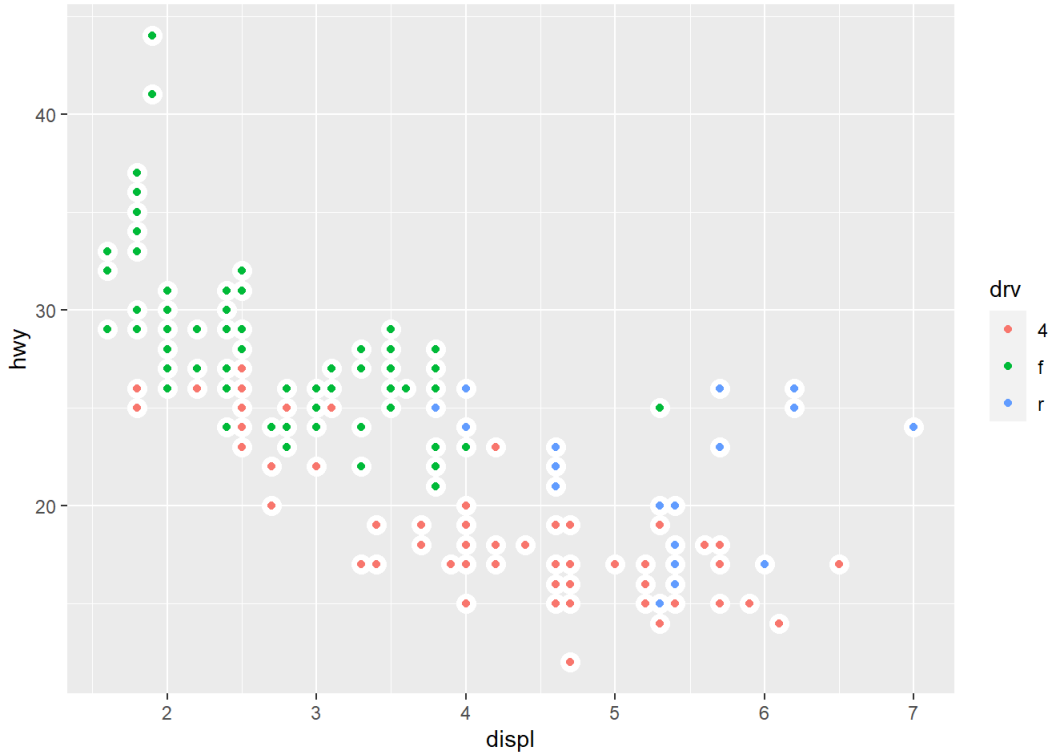



```
mpg %>% ggplot(aes(displ, hwy, group=drv)) +
  geom_point(aes(color=drv)) +
  geom_smooth(se=F, aes(linetype=drv))
```

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




```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(size=4, color='white') +
  geom_point(aes(colour=drv))
```

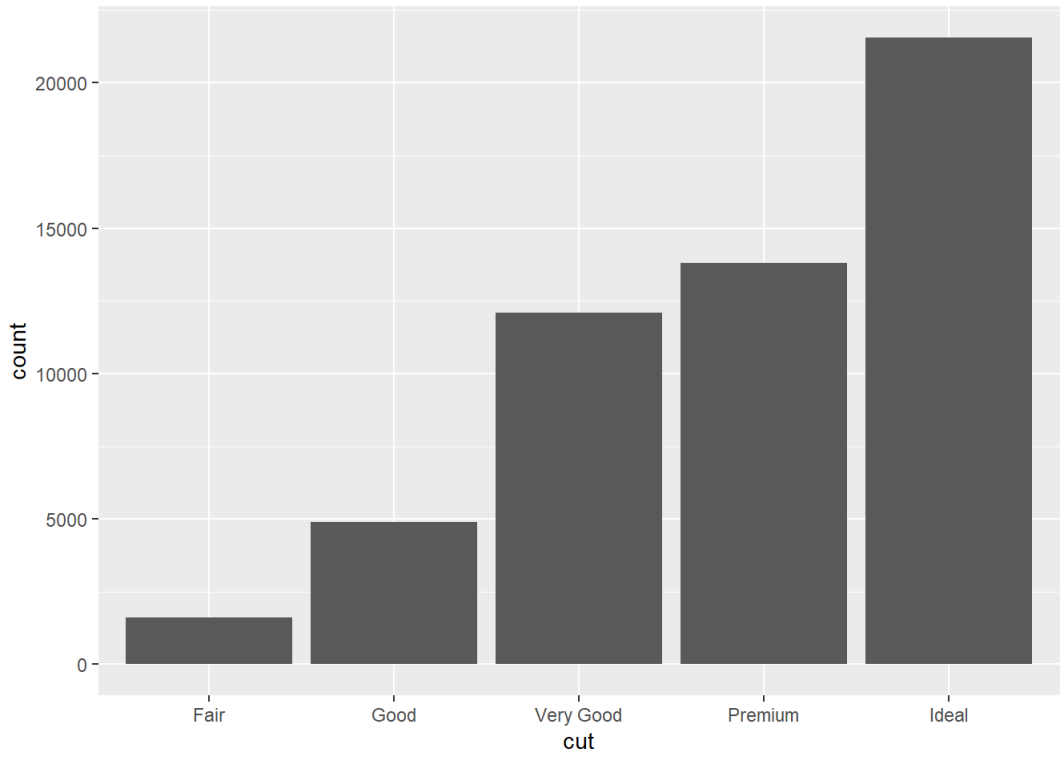



```
mpg %>% ggplot(aes(displ, hwy, group=drv)) +
  geom_point(aes(color=drv), fill=20, stroke=1, size=2, shape=1)
```

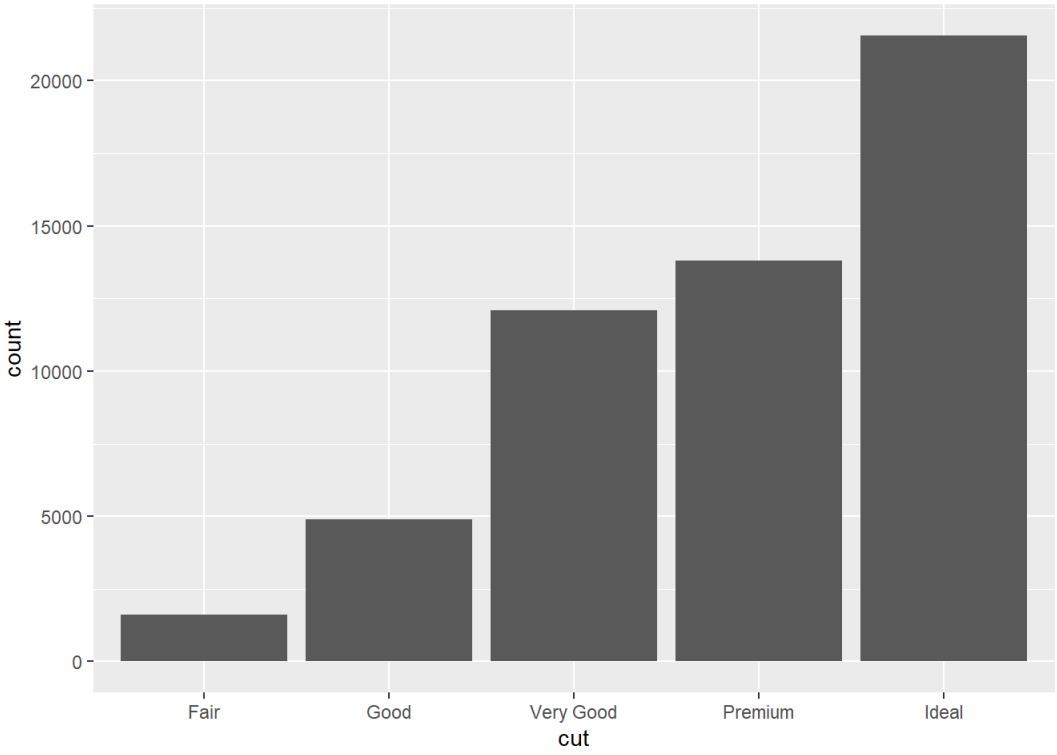
statistical transformation

geom_bar와 geom_count 비교

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



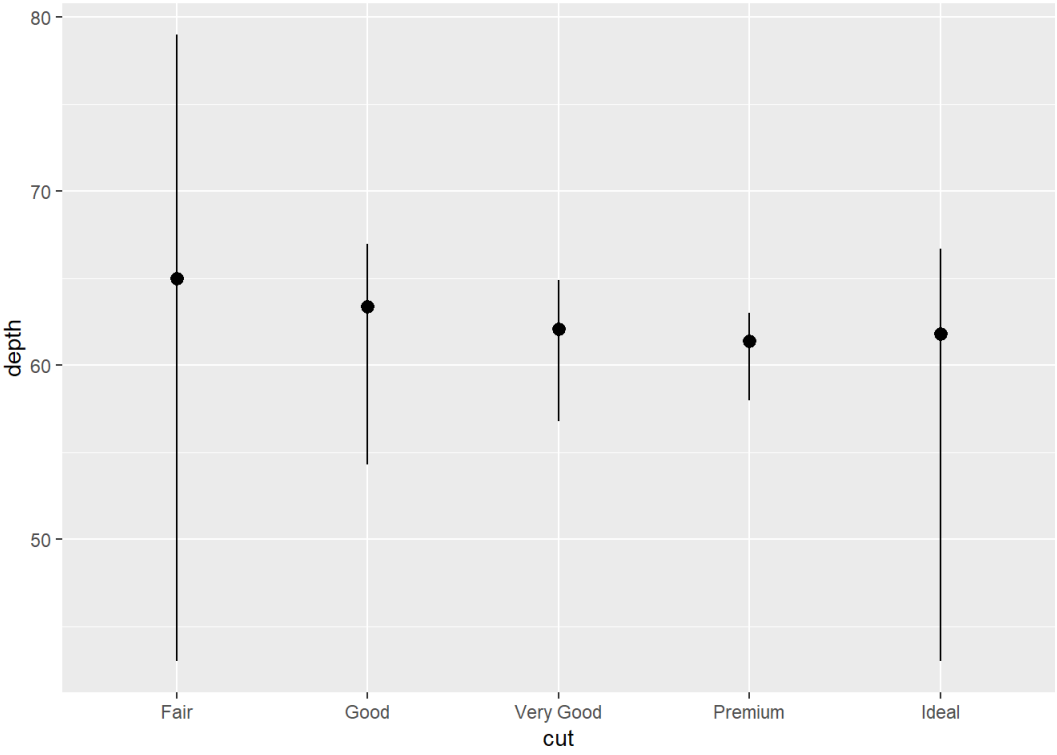

```
ggplot(data = diamonds) +
  stat_count(mapping = aes(x = cut))
```



- 위의 두 결과가 완전히 동일
- **geom_bar()** 에 자연스럽게 count 성질을 포함

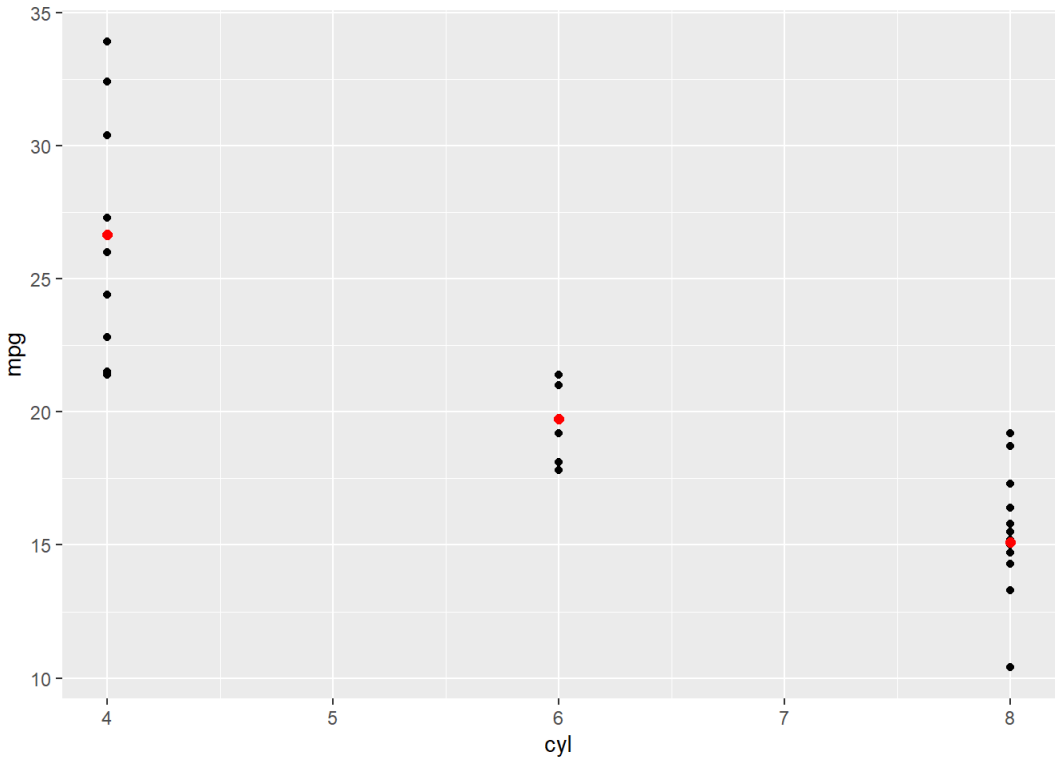
stat_summary()

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



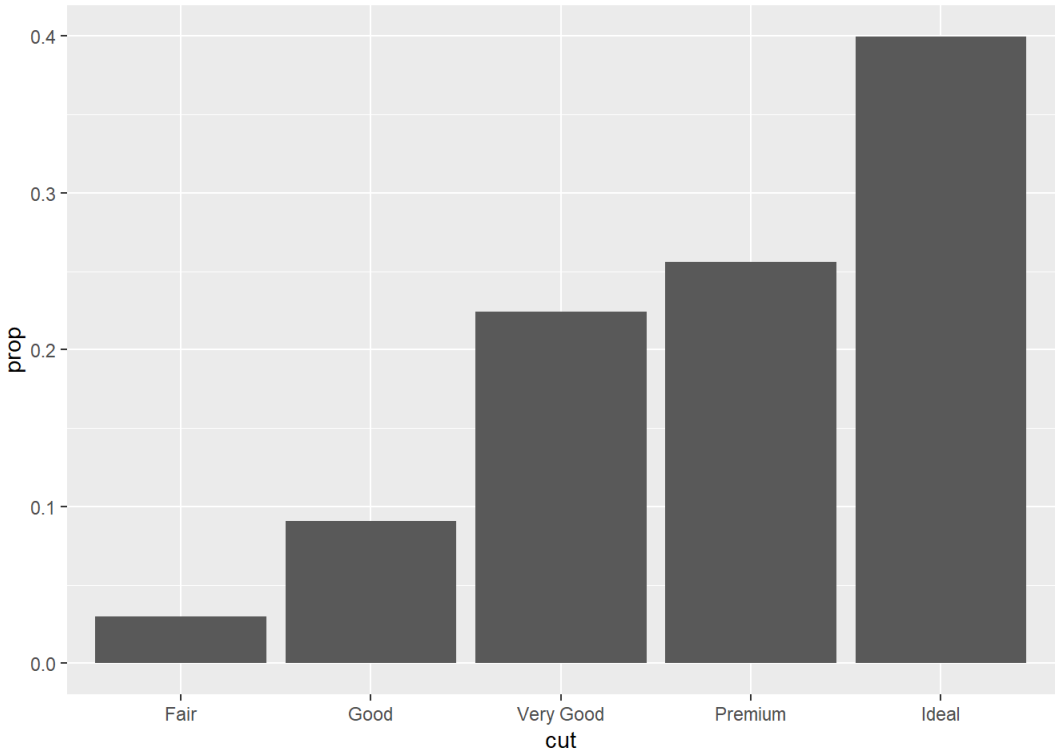
- point는 **중앙값** 을 나타냄
- 직선의 양 끝값은 **최댓값** 과 **최솟값**

```
d <- ggplot(mtcars, aes(cyl, mpg)) + geom_point()
d + stat_summary(fun.y = "mean", colour = "red", size = 2,
                 geom = "point")
```



지금까지 geom_bar은 y축에 count 결과를 출력 propotion은 어떻게 계산하는가?

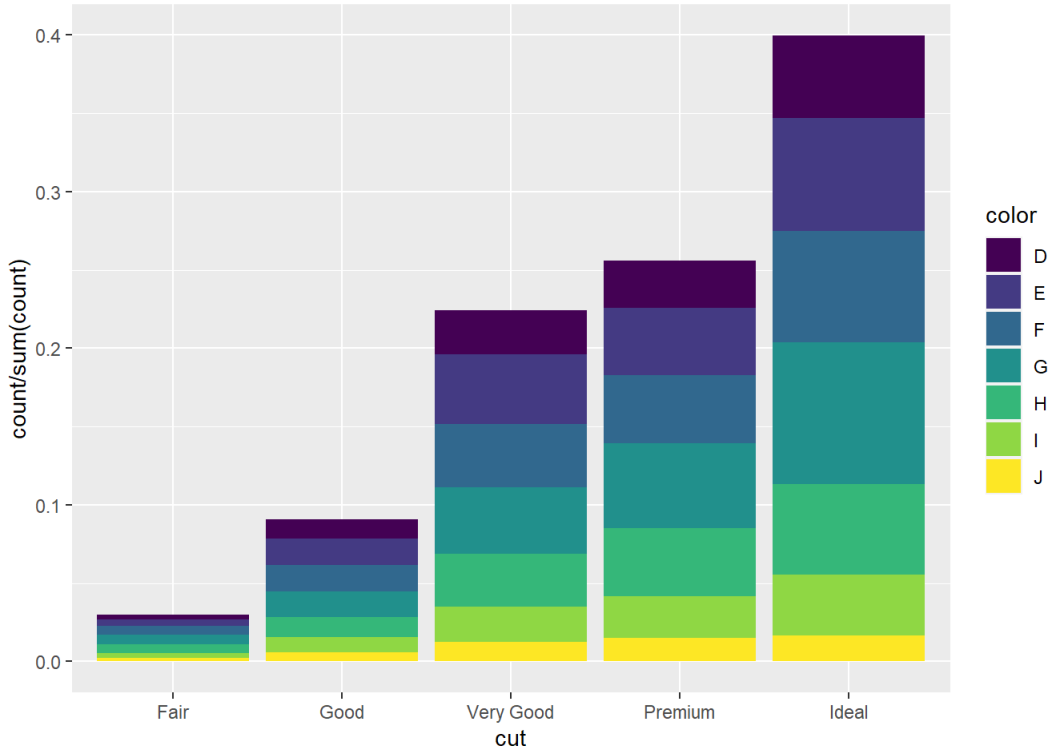
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop.., group = 1))
```



- **group=1** 을 추가해주어야 propotion을 구할 수 있다.
- 다소 직관적이진 못하다.

혹은 다음과 같이 propotion을 구할 수 있다.

```
ggplot(data = diamonds) +
  geom_bar(aes(x = cut, y = ..count.. / sum(..count..),
               fill = color))
```

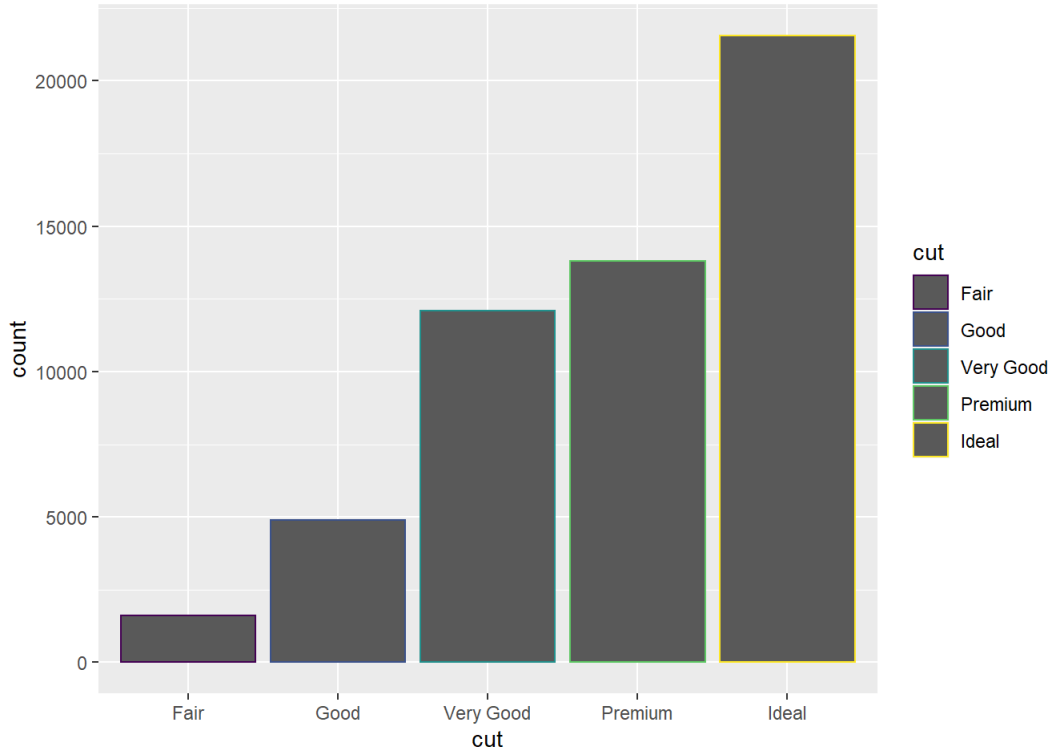


- ..count.. 를 활용하여 propotion을 구했다.
- fill= 옵션으로 그룹별 색상을 달리했다.

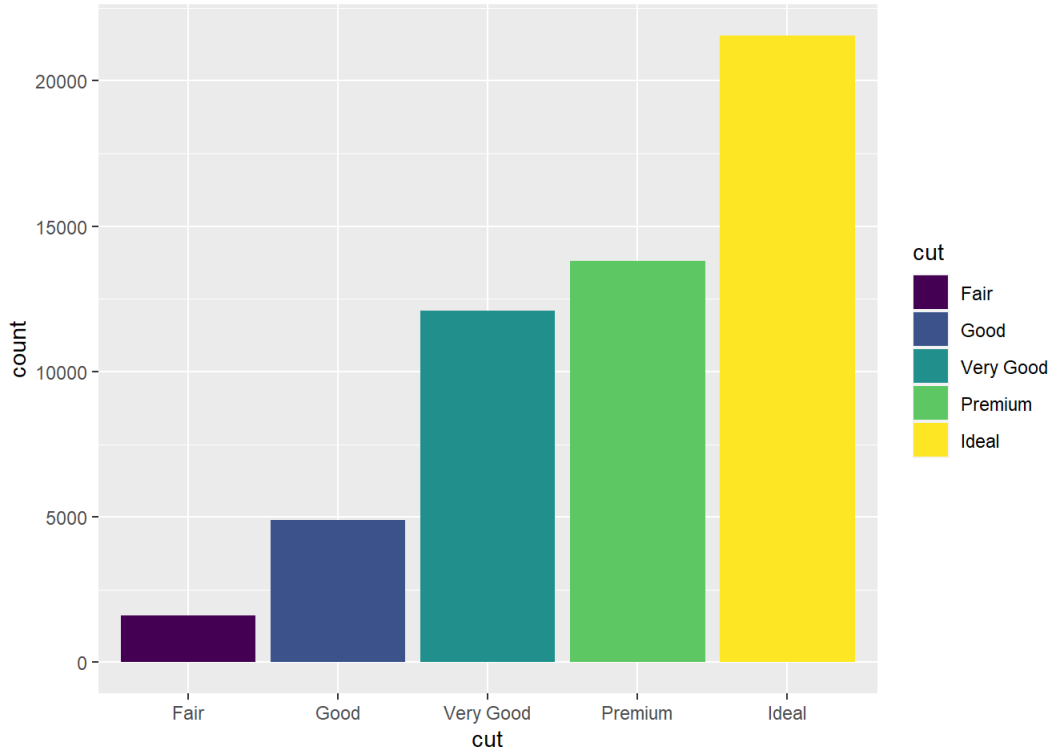
Positon adjustment

bar chart에서는 colour 옵션으로 색 지정하는 것 보다 fill이 더 유용하다.

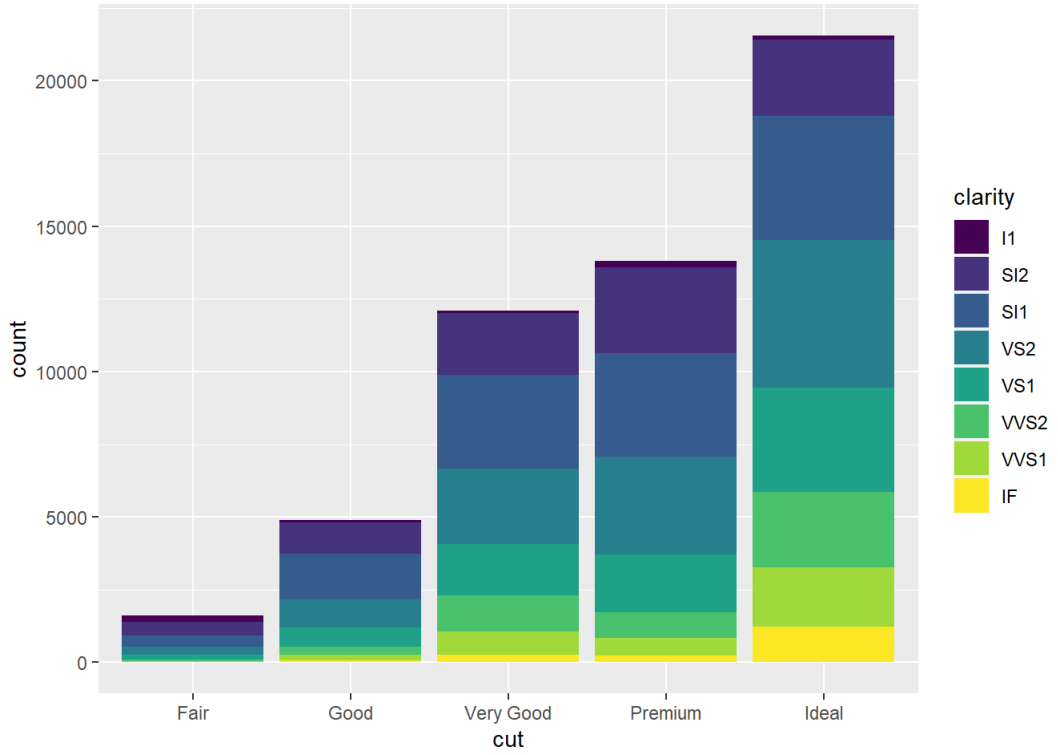
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour = cut))
```



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

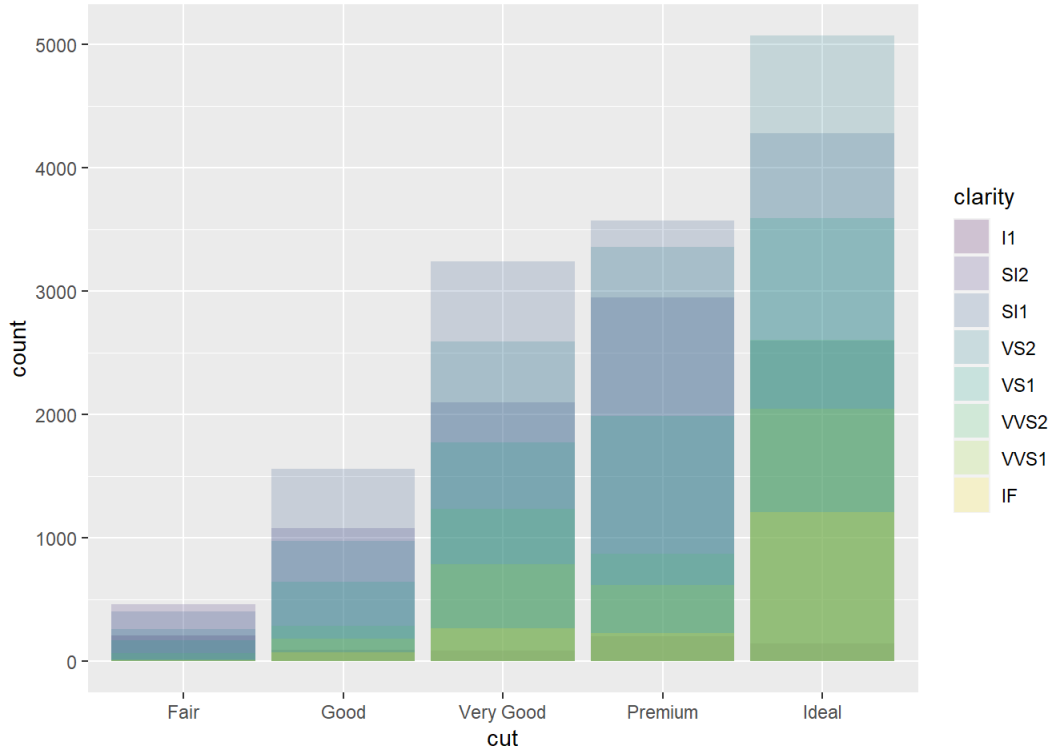



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```

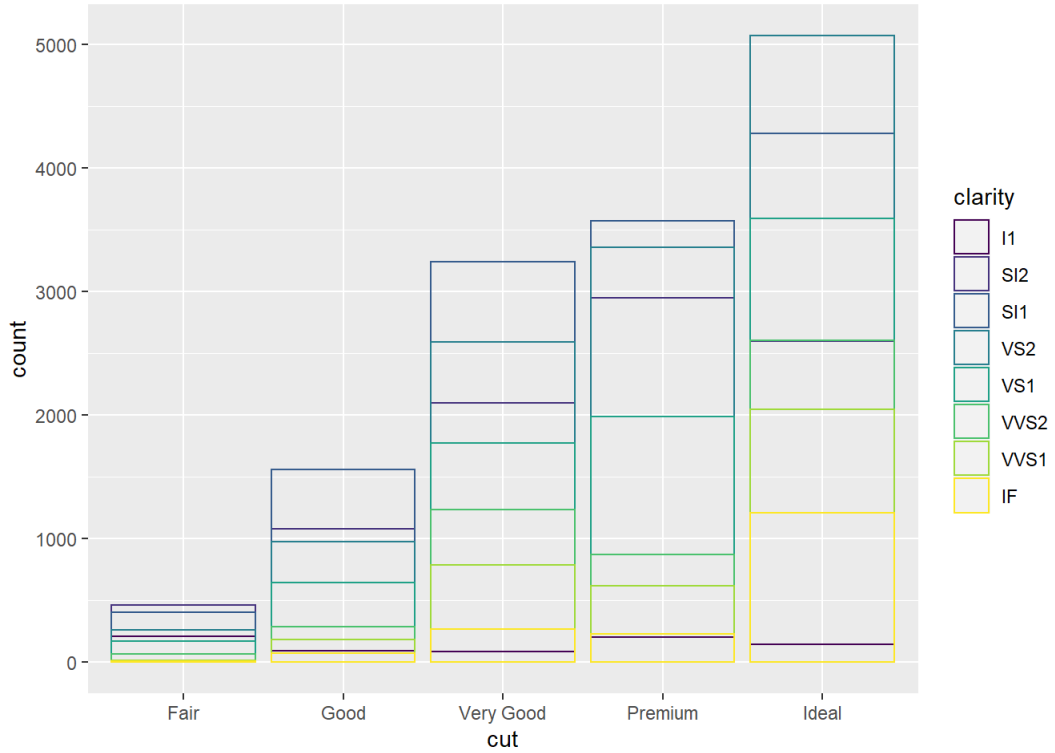


position = “identity”

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



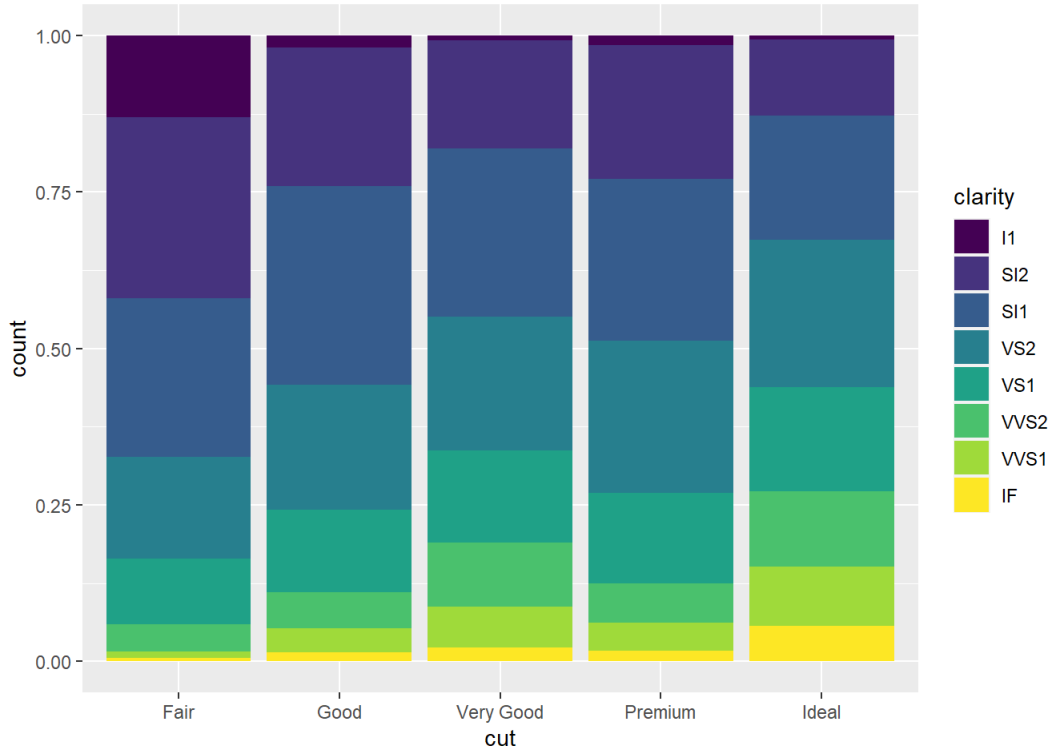

```
ggplot(data = diamonds,
       mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity")
```



- position = identity는 default 값인듯
- alpha 값으로 투명도 조절

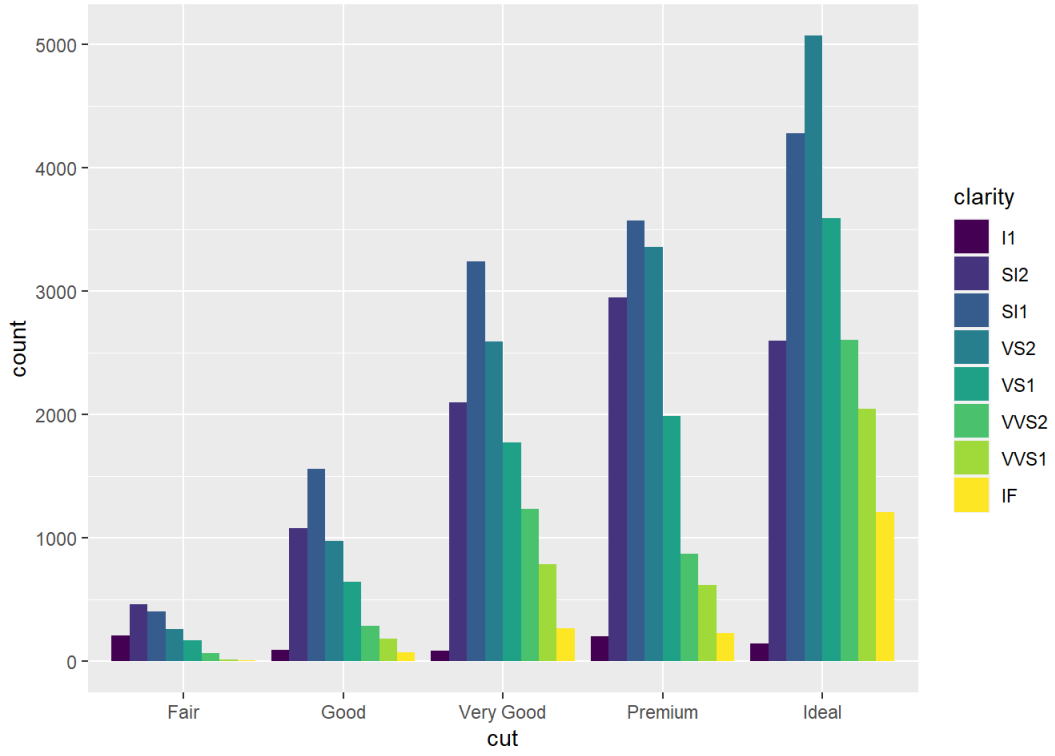
position = “fill”

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



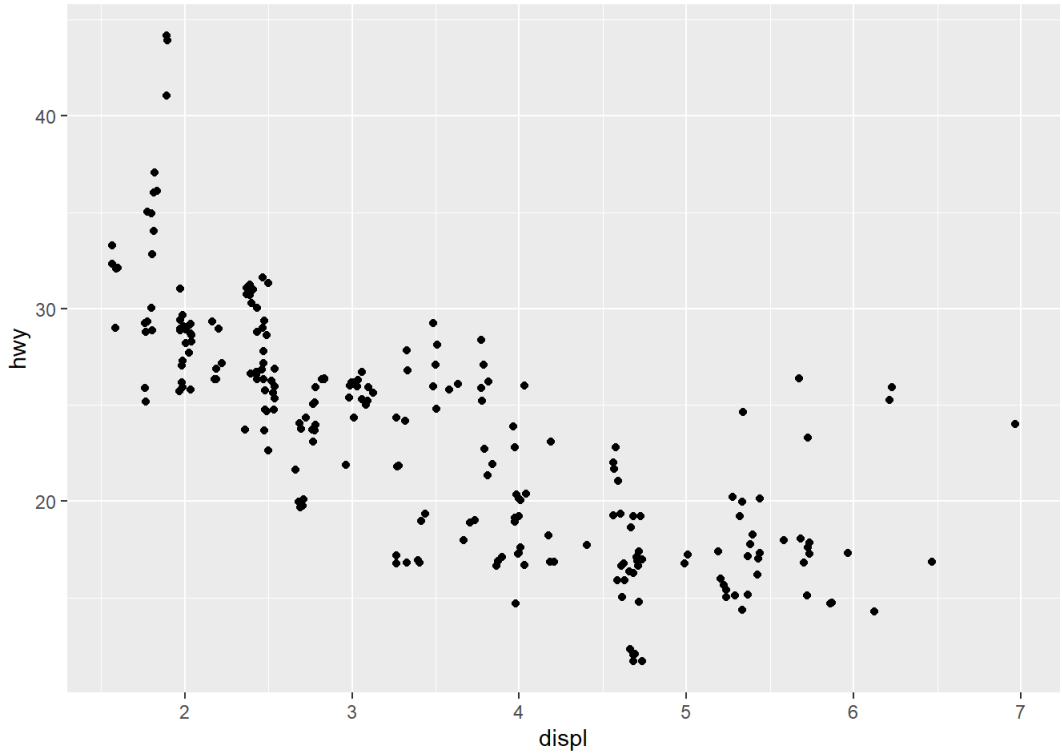
position = “dodge”

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



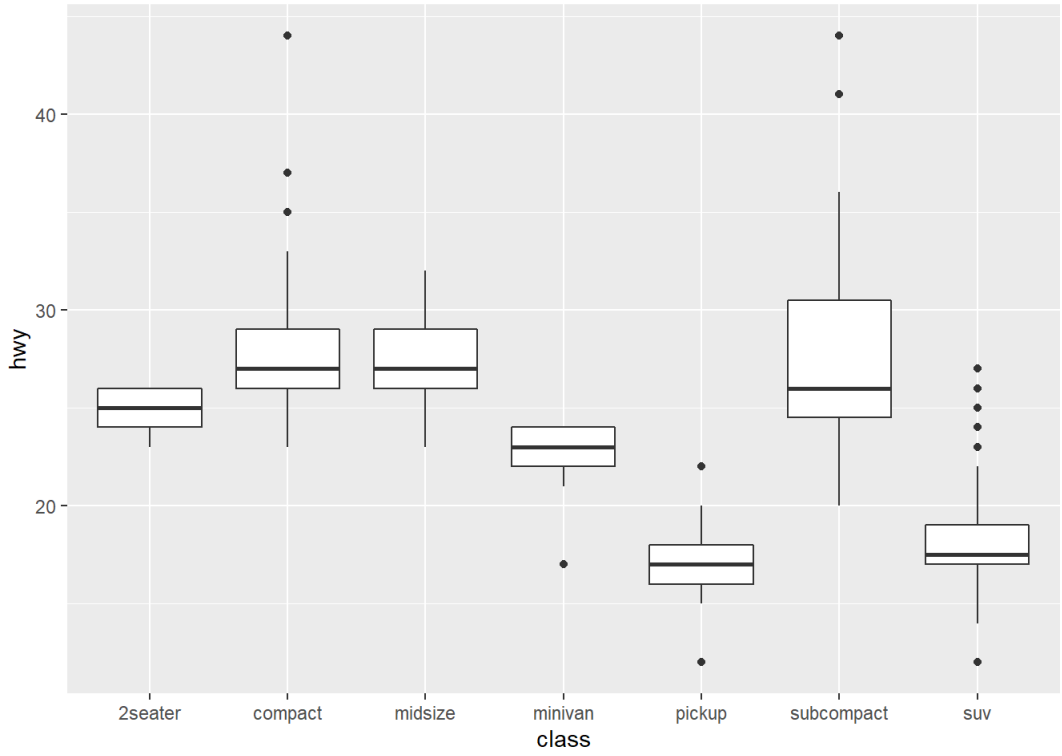
position = “jitter”

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

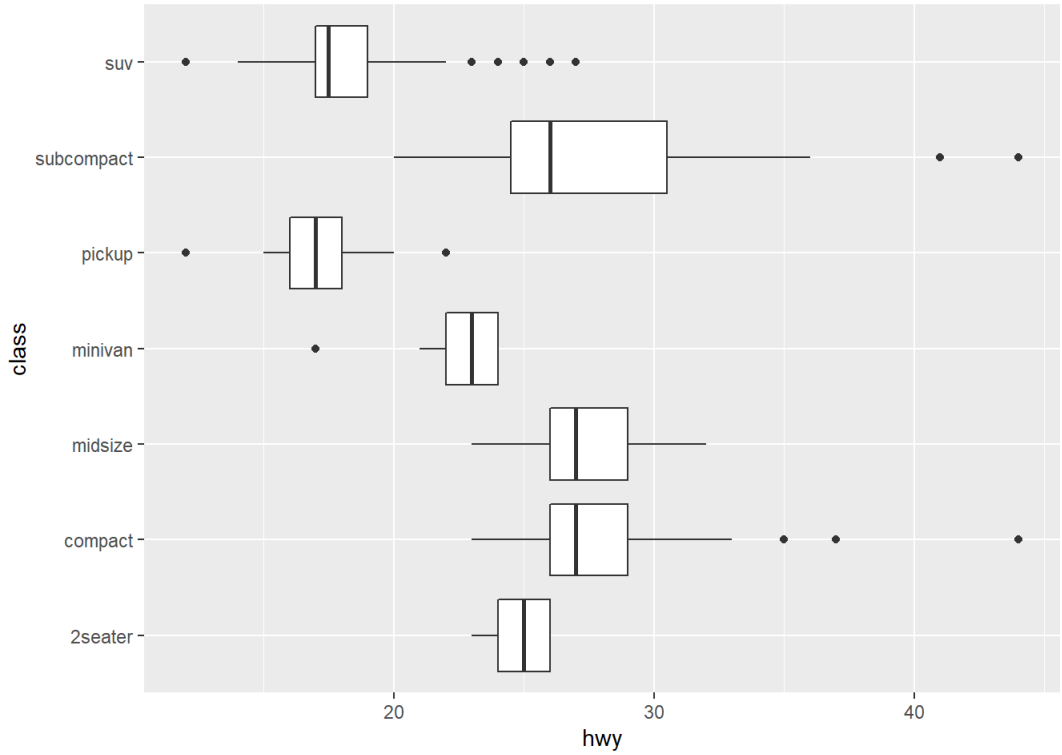


Coordinate system

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

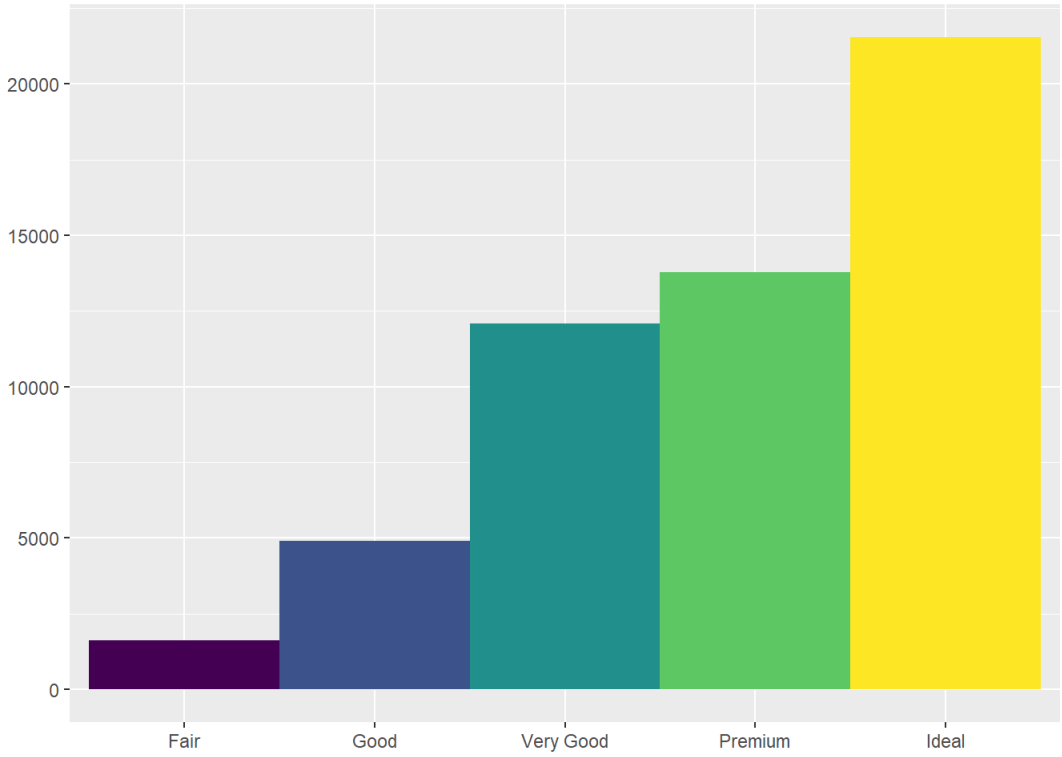



```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() + coord_flip()
```

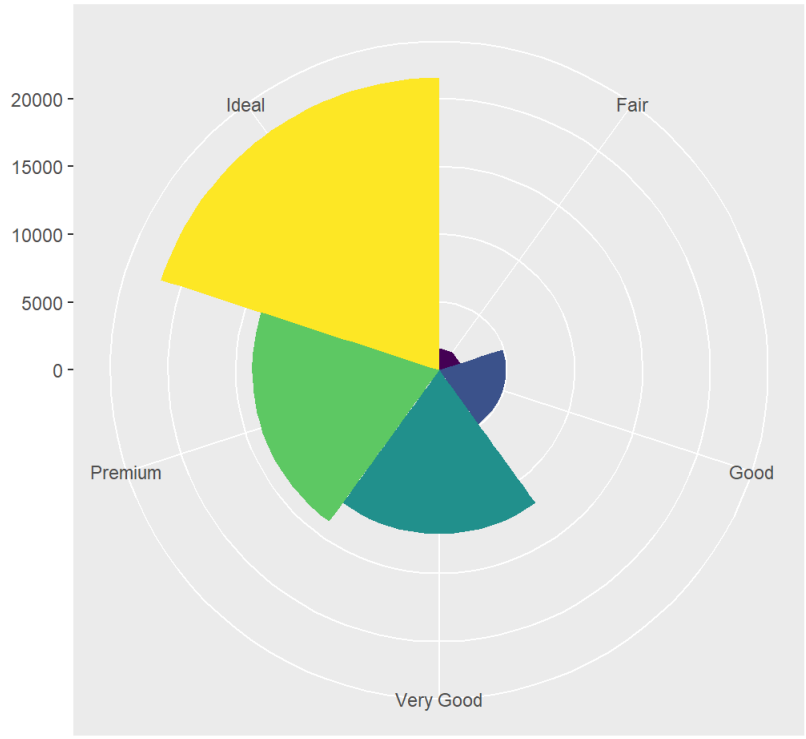


- **coord_flip()** 를 이용해 가로방향 출력

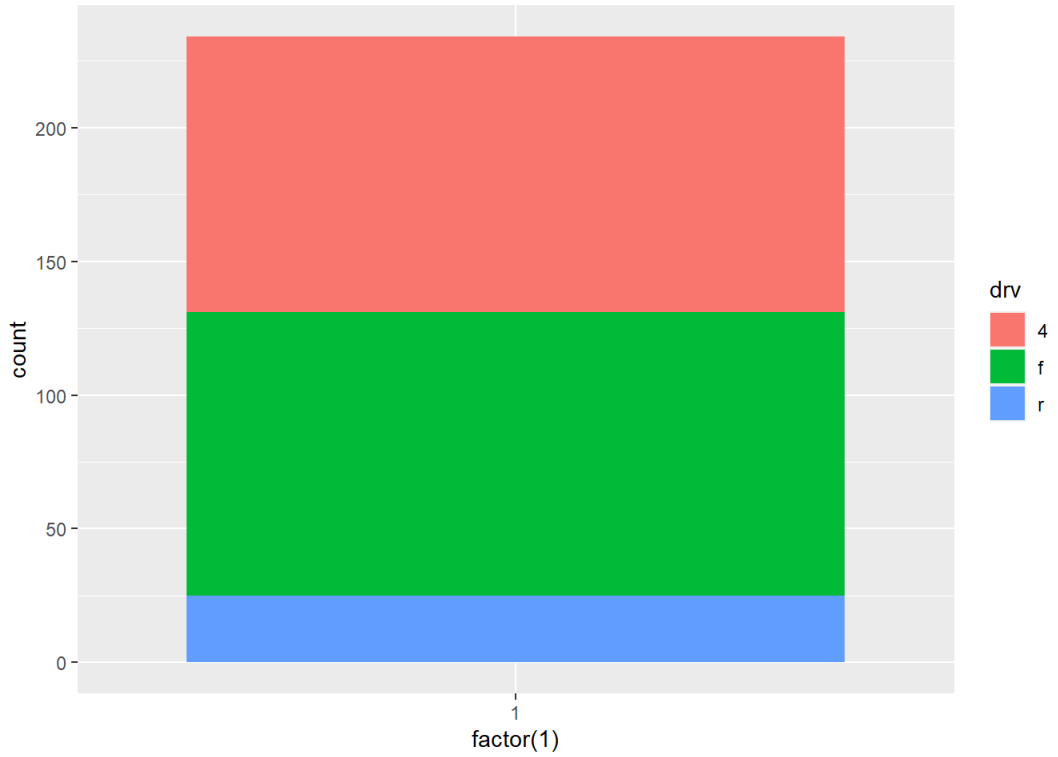
```
bar <- ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = cut), show.legend = FALSE, width = 1) +
  labs(x = NULL, y = NULL)
bar
```



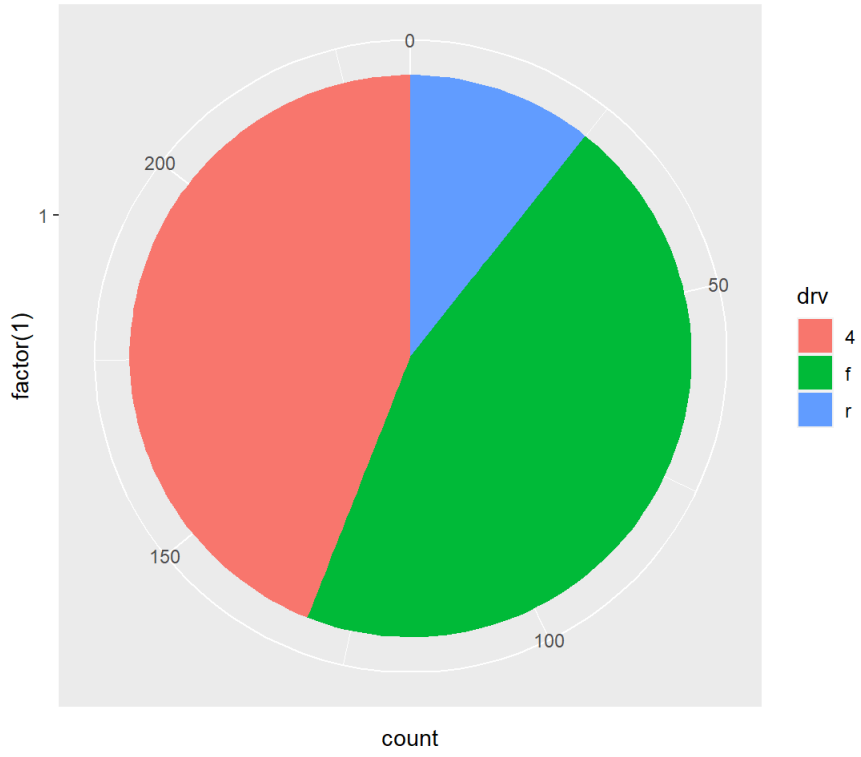

```
bar + coord_polar()
```



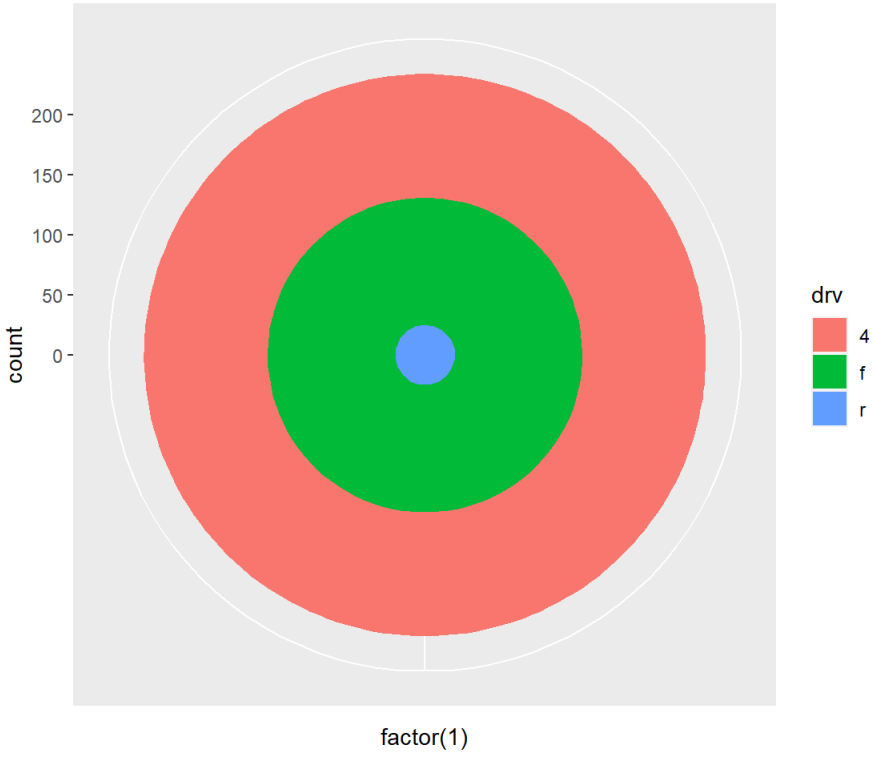

```
ggplot(mpg, aes(x = factor(1), fill = drv)) + geom_bar()
```




```
ggplot(mpg, aes(x = factor(1), fill = drv)) + geom_bar() +
  coord_polar(theta = "y")
```

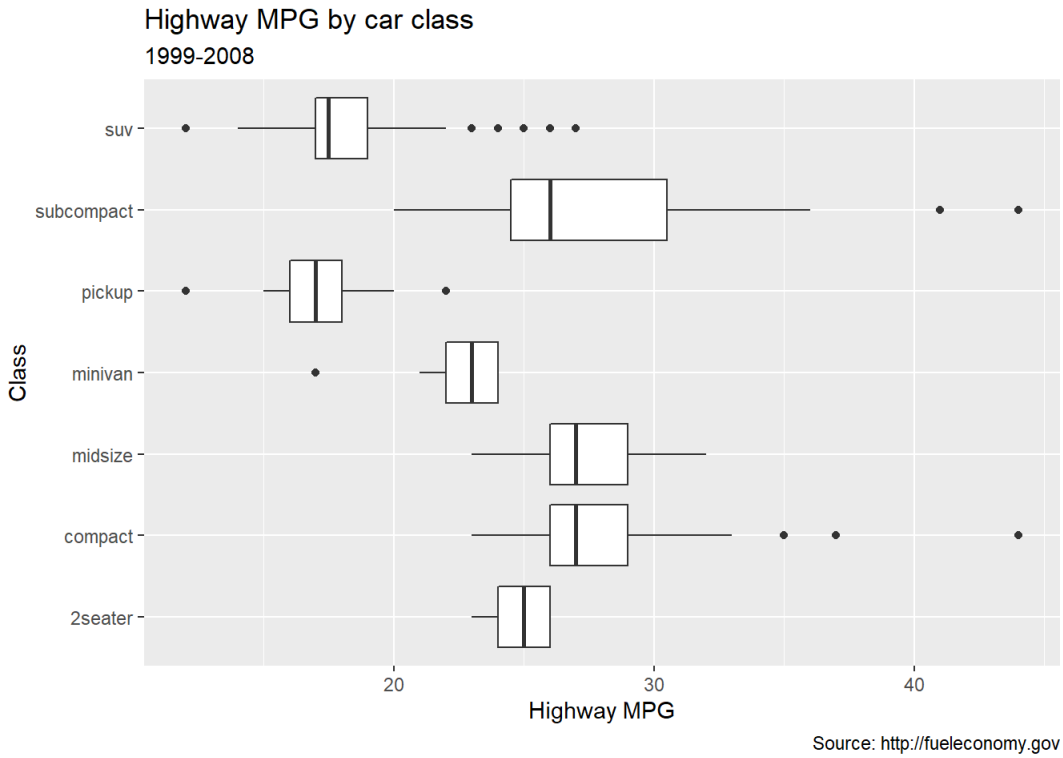



```
ggplot(mpg, aes(x = factor(1), fill = drv)) + geom_bar(width = 1) +
  coord_polar()
```



labs 활용

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
geom_boxplot() +
coord_flip() +
labs(y = "Highway MPG",
x = "Class",
title = "Highway MPG by car class",
subtitle = "1999-2008",
caption = "Source: http://fueleconomy.gov")
```


Highway MPG by car class
1999-2008
suv
subcompact
pickup
minivan
midsize
compact
2seater
Class
20
30
40
Highway MPG
Source: http://fueleconomy.gov