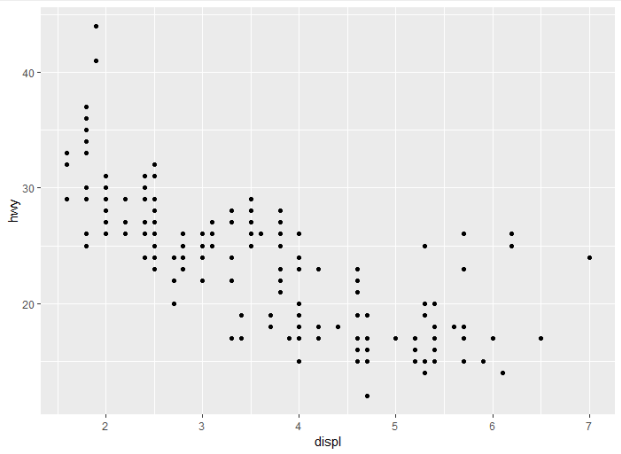
3. ggplot2

자료의 시각화

1. ggplot2 시작하기

- 패키지 ggplot2에 있는 데이터 프레임 mpg의 변수 displ과 hwy의 산점도 작성

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



- 함수 ggplot( ): 데이터 프레임 data 지정. 그래프가 작성될 비어있는 좌표계 작성.
- 함수 geom_point( ): 실질적인 그래프, 레이어(layer)를 작성하는 geom 함수 중 하나
- mapping: geom 함수 내에서 함수 aes( )와 함께 데이터와 시각적 요소를 서로 연결

- ggplot2에서 그래프 작성의 최소 요소

  - 그래프 작성을 위한 법칙이 있음: 그래프의 문법
  - 모든 그래프 작성에 일정하게 적용
  - 익숙해지면 복잡한 형태의 그래프도 어렵지 않게 작성 가능

  - 그래프 작성을 위한 3가지 최소 요소: <Data>, <Geom_function>, <Mappings>

```
ggplot(data=<Data>) +
              <Geom_function>(mapping=aes(<Mappings>)
```

<Data>: 그래프 작성에 사용될 데이터 프레임 지정

<Geom_function>: geom 함수 중 하나. 레이어(layer) 작성. 여러 개의 레이어를 겹치기 위해서는 여러 개의 geom 함수를 덧셈 기호로 연결

<Mappings>: 시각적 요소(점의 크기, 모양, 색깔, ...)와 데이터 연결

2. 시각적 요소와 연결: Mapping

- 시각적 요소
  - 그래프를 시각적으로 인식할 때 필요한 요소
  - 산점도의 경우, 점의 위치, 크기, 모양 및 색깔 등이 시각적 요소

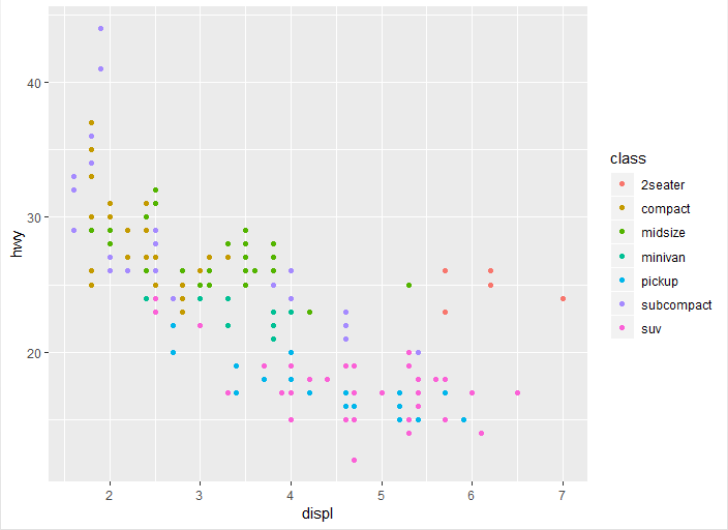
- 시각적 요소의 mapping과 setting
  - mapping: 데이터의 값과 연결되어 결정. 함수 aes( ) 안에서 연결
  - setting: 사용자가 일정한 값을 지정

- 시각적 요소의 mapping
  - 기존의 그래프에 다른 변수의 정보 추가 가능

- 예제: 데이터 프레임 mpg의 변수 displ과 hwy의 산점도에 시각적 요소와의 mapping으로 다른 변수 정보 추가

  - 변수 class를 시각적 요소 color와 mapping

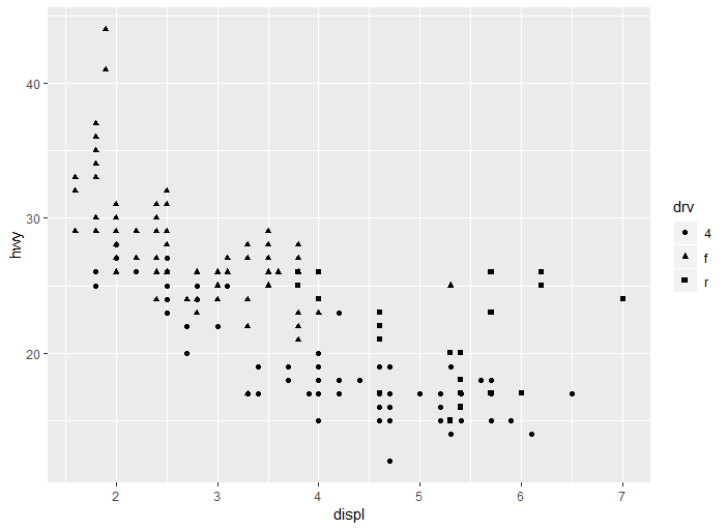
```
> ggplot(data=mpg) +
    geom_point(mapping=aes(x=displ, y=hwy, color=class))
```



- 변수 class: 문자형 벡터
- 변수 class의 값에 따라 다른 색 사용
- 사용된 색깔에 대한 범례는 자동으로 추가

- 변수 drv를 시각적 요소 shape와 mapping

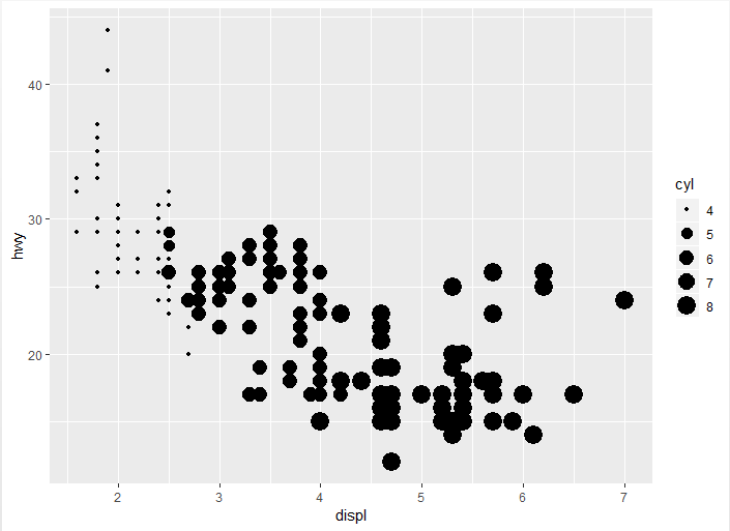
```
> ggplot(data=mpg) +
    geom_point(mapping=aes(x=displ, y=hwy, shape=drv))
```



- shape에 mapping되는 변수는 이산형
- 변수 drv: 문자형 벡터
- 변수 drv의 값에 따라 다른 모양의 점 사용
- 범례 자동 추가

- 변수 cyl을 시각적 요소 size와 mapping

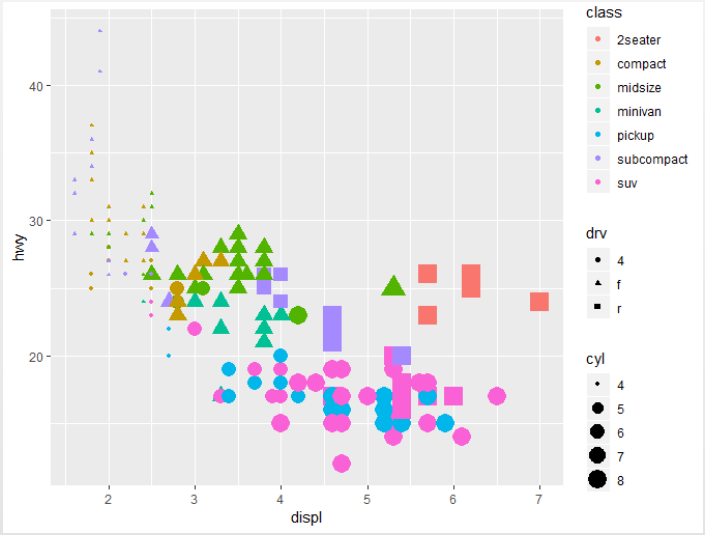
```
> ggplot(data=mpg) +
    geom_point(mapping=aes(x=displ, y=hwy, size=cyl))
```



- size에 mapping되는 변수는 연속형이 좋음
- 변수 cyl: 정수형 변수
- cyl의 값에 따라 점의 크기 조절
- 범례 자동 추가

- 여러 시각적 요소를 동시에 mapping
  - 변수 class는 color와, drv는 shape와, cyl은 size와 mapping

```
> ggplot(data=mpg) +
    geom_point(mapping=aes(x=displ, y=hwy,
                           color=class, shape=drv, size=cyl))
```



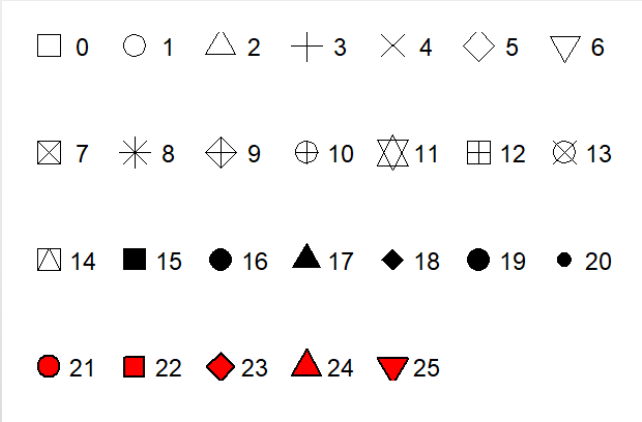
- 너무 많은 정보
- 그래프의 의미가 모호

- 시각적 요소의 setting

  - 함수 aes( ) 밖에서 사용자가 원하는 값으로 지정
  - geom 함수의 입력 요소가 됨

- 시각적 요소 color, size, shape에 값 지정 법칙

  1) color: 색깔을 나타내는 문자열 지정
  2) size: 점 크기를 mm 단위로 지정
  3) shape: 점의 형태를 나타내는 0~26 사이의 숫자

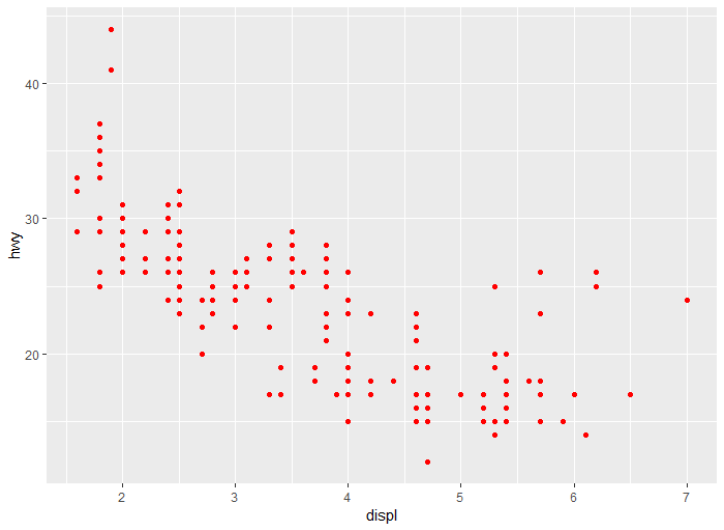


도형에 색깔 지정 방법

1) 0~14의 외곽선 및 15~20의 도형 색: color 사용
2) 21~25의 외곽선: color 사용
3) 21~25의 내부 색: fill 사용

- 시각적 요소 color의 setting: 모든 점을 빨간 색으로

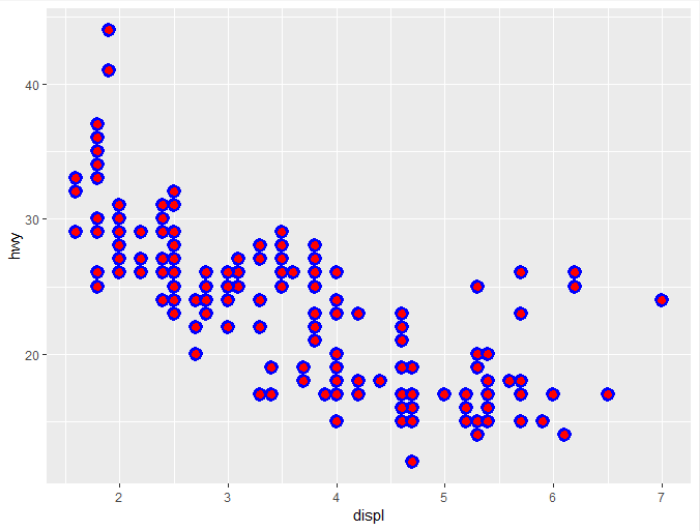
```
> ggplot(data=mpg) +
    geom_point(mapping=aes(x=displ, y=hwy), color="red")
```



- color를 함수 aes( ) 밖에서 지정
- 함수 geom_point( )의 입력 요소

- 여러 시각적 요소를 동시에 setting

```
> ggplot(data=mpg) +
    geom_point(mapping=aes(x=displ, y=hwy), color="blue", size=3,
               shape=21, fill="red", stroke=2)
```



- 점의 모양: shape=21
- 점의 내부 색: 빨간색
- 점의 외곽선 색: 파란색
- 점의 크기 확대: size=3
- 점의 외곽선 두께 조절: stroke=2

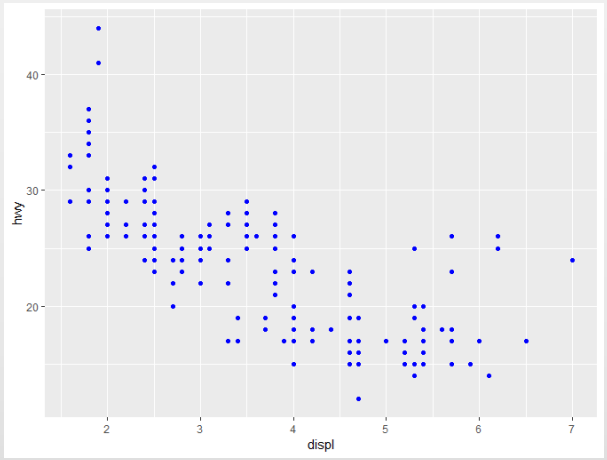
- 함수 aes( ) 안에서 시각적 요소에 특정 값을 setting한 경우의 결과

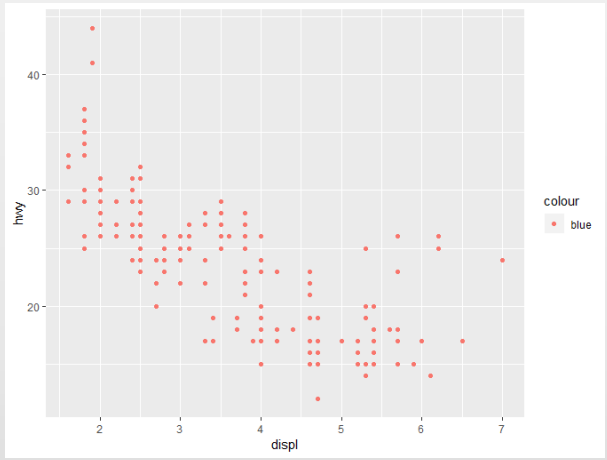
Setting

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

Mapping

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





- mapping은 변수와의 연결을 의미
- "blue"라는 값을 갖는 변수 생성

3. 그룹별 그래프 작성: Facet

- 범주형 변수가 다른 변수에 미치는 영향력을 그래프로 확인하는 방법

  1) 시각적 요소에 범주형 변수를 mapping
  2) 범주형 변수로 그룹 구성하고, 각 그룹별 그래프 작성: faceting

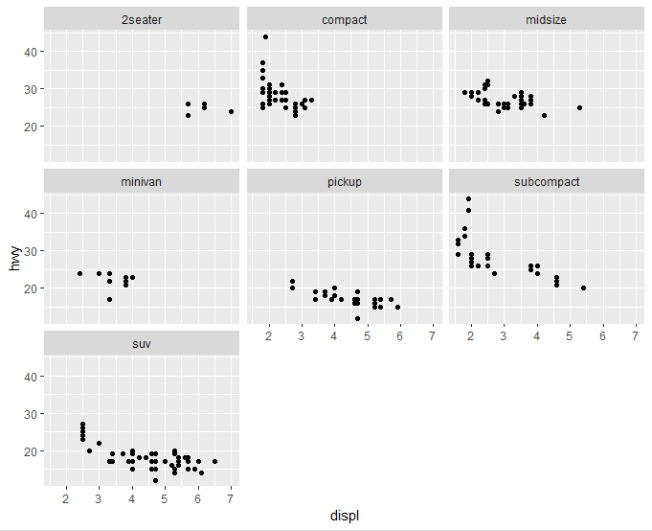
- facet을 적용하기 위한 함수
  ① facet_wrap( ): 한 변수에 의한 facet
  ② facet_grid( ): 한 변수 또는 두 변수에 의한 facet

- 함수 facet_wrap( )에 의한 faceting

  - 데이터를 구분하는 변수가 하나인 경우: facet_wrap(~ x)

• 데이터 프레임 mpg의 변수 displ과 hwy의 산점도를 class의 범주별로 작성

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

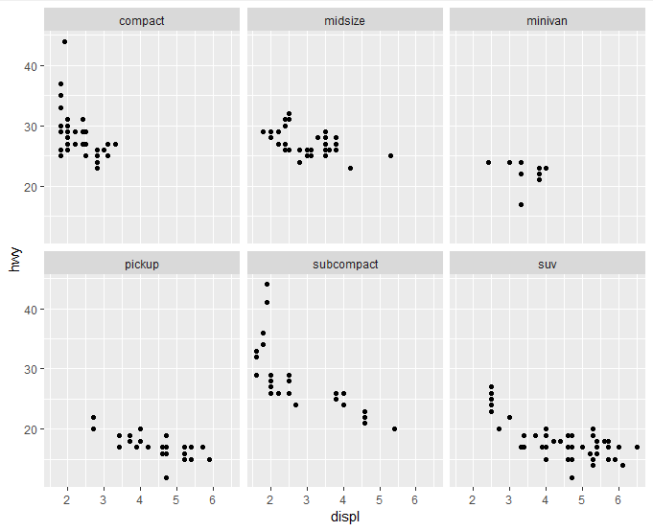


- 패널 '2seater'에는 적은 수의 데이터 존재
- class가 '2seater'인 케이스 제거 후 다시 작성

- 데이터 프레임 mpg의 변수 displ과 hwy의 산점도를 class의 범주별로 작성 (2seater 케이스 제외)

```
> mpg_1 <- mpg %>% filter(class != "2seater")

> ggplot(data=mpg_1) +
    geom_point(mapping=aes(x=displ,y=hwy)) +
    facet_wrap(~class)
```



패널 배치 조절

- 열의 수 지정: ncol=
- 행의 수 지정: nrow=

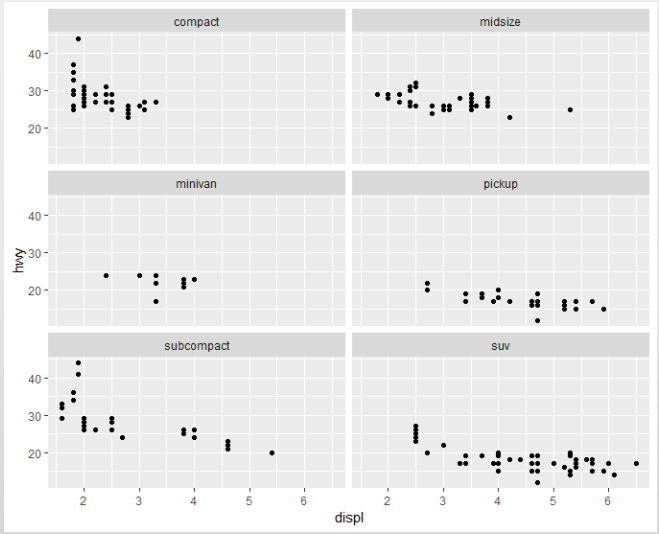
- 배치 순서 지정:
  행 단위: 디폴트
  열 단위: dir="v"

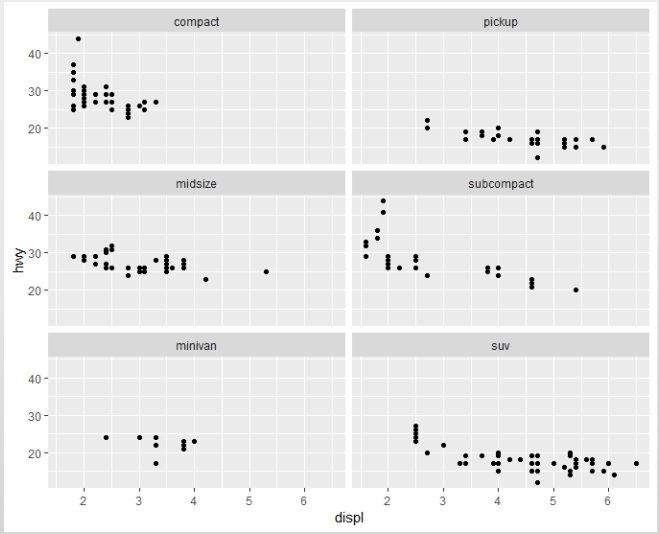
- 2×3 패널 패치를 3×2 배치로 수정: ncol=2
- 패널에 그래프 배치 순서를 열 단위로 수정: dir="v"

```
> pp <- ggplot(data=mpg_1) +
    geom_point(mapping=aes(x=displ,y=hwy))
```

```
> pp + facet_wrap(~class, ncol=2)
```

```
> pp + facet_wrap(~class, ncol=2, dir="v")
```





- 함수 facet_grid( )에 의한 faceting

  - 한 변수에 의한 faceting:
    하나의 행으로 패널 배치: `facet_grid(.~x)`
    하나의 열로 패널 배치: `facet_grid(x~.)`

  - 두 변수에 의한 faceting: `facet_grid(y~x)`
    행 범주: 변수 y의 범주
    열 범주: 변수 x의 범주

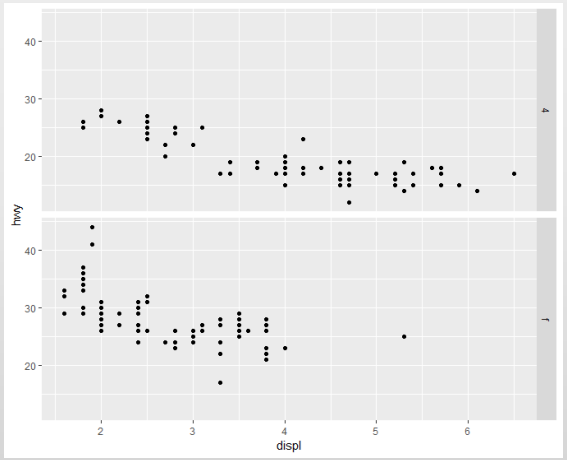
- 데이터 프레임 mpg에서 데이터를 변수 drv와 cyl로 구분하여 displ과 hwy의 산점도 작성. 단, drv가 "r"인 자료와 cyl이 5인 자료 제외

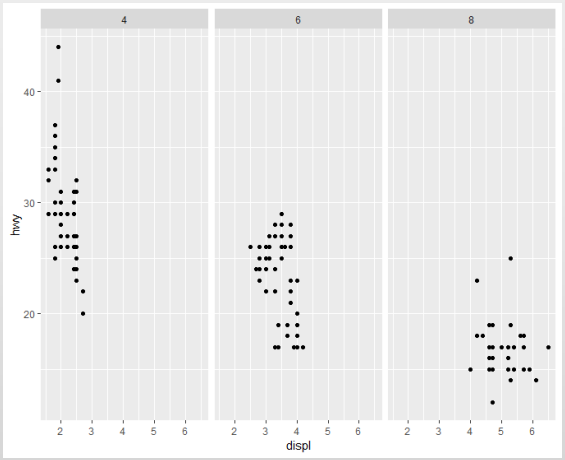
```
> mpg_2 <- mpg %>%
    filter(cyl!=5, drv!="r")
> my_plot <- ggplot(data=mpg_2) +
    geom_point(mapping=aes(x=displ, y=hwy))
```

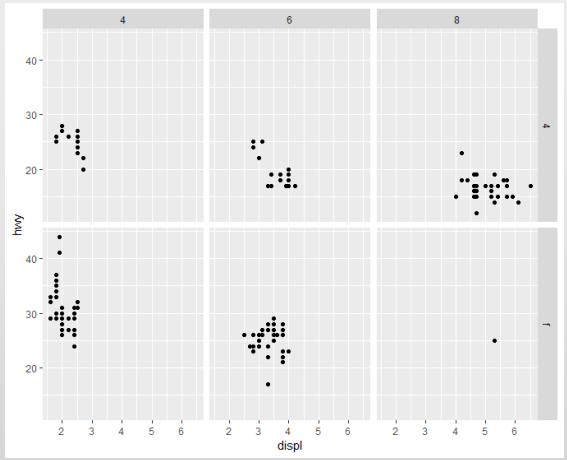
```
> my_plot + facet_grid(drv~.)
```

```
> my_plot + facet_grid(.~cyl)
```

```
> my_plot + facet_grid(drv~cyl)
```







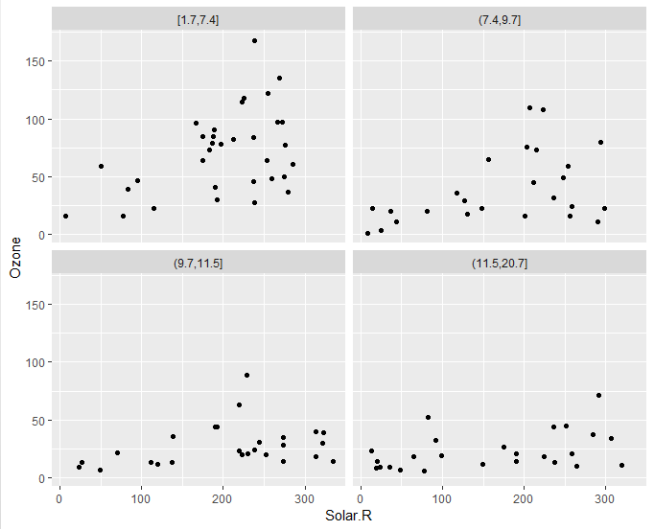
- 연속형 변수에 의한 faceting

  - 연속형 변수를 범주형 변수로 변환 후 faceting

  - 유용한 함수
    `cut_interval(x, n)`: 벡터 x를 n개의 같은 길이의 구간으로 구분
    `cut_width(x, width)`: 벡터 x를 길이가 width인 구간으로 구분
    `cut_number(x, n)`: 벡터 x를 n개의 구간으로 구분하되 각 구간에 속한 데이터의 개수가 비슷하게 구분

- 데이터 프레임 airquality에서 변수 Ozone, Solar.R, Wind의 관계 탐색
  1) 변수 Wind를 4개의 구간으로 구분하되 속한 자료의 개수가 비슷하도록
  2) 4개의 구간에서 Ozone과 Solor.R의 산점도 작성

```
> airquality %>%
    mutate(Wind_d=cut_number(Wind, n=4)) %>%
    ggplot() +
    geom_point(mapping=aes(x=Solar.R, y=Ozone)) +
    facet_wrap(~Wind_d)
```



- 변수 Wind가 큰 값을 가질수록 두 변수의 관계는 점점 미약해지고 있음
- 세 연속형 변수의 관계 탐색 방법 중 하나

4. 기하 객체: Geometric object

- Base graphics에서 그래프 작성 방식: pen on paper
  - 높은-수준의 그래프 함수: 좌표축과 주요 그래프 작성
  - 낮은-수준의 그래프 함수: 점, 선, 문자 등을 추가하여 원하는 그래프 작성

- ggplot2에서 그래프 작성 방식
  - 작성하고자 하는 그래프: 몇몇 유형의 그래프(점 그래프, 선 그래프 등등)를 겹쳐 놓은 것
  - 몇몇 유형의 그래프를 각기 따로 작성
  - 작성된 그래프를 겹쳐 놓음으로써 원하는 그래프 작성

- ggplot2 시스템

  원하는 유형의 그래프(점 그래프, 선 그래프 등등) 작성

  ↔ 해당되는 기하 객체(geom)를 사용하여 그래프 작성

- 기하 객체의 사용

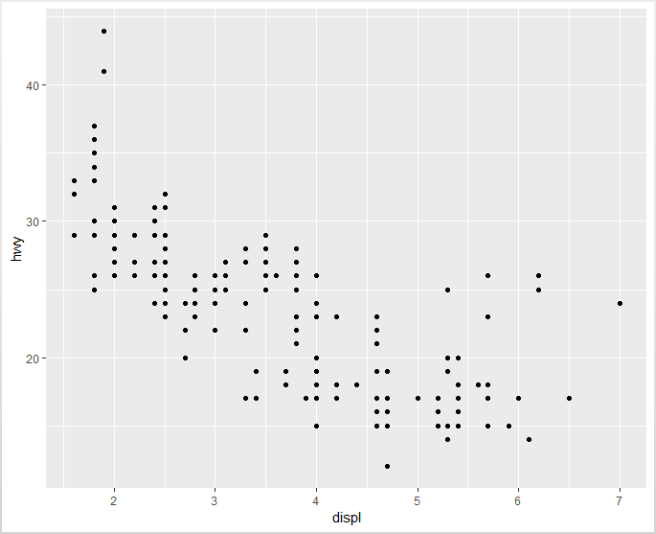
  - 해당되는 geom 함수의 실행
  - geom 함수 실행 → 해당 유형의 그래프가 작성된 layer 생성
  - 여러 개의 geom 함수 실행: 여러 layer 생성되고 이것들이 겹쳐져서 원하는 그래프 완성

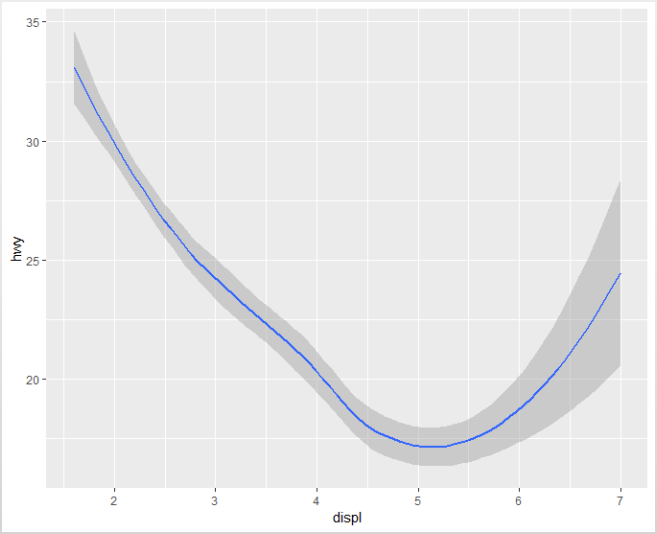
- 동일 자료에 다른 geom 적용

  - mpg의 변수 displ과 hwy를 대상으로 point geom과 smooth geom 적용
    - point geom: 점 그래프 작성
    - smooth geom: 비모수 회귀곡선 작성

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

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





- geom 함수 리스트

  - 현재 대략 30개 이상의 geom 함수가 있음

  - 한 변수에 대한 함수: geom_bar( ), geom_histogram( ), geom_density( ), geom_dotplot( ) 등등

  - 두 변수에 대한 함수: geom_point( ), geom_smooth( ), geom_text( ), geom_line( ), geom_boxplot( ) 등등

  - 세 변수에 대한 함수: geom_contour( ), geom_tile( ) 등등

  - geom 함수의 리스트: R studio의 메뉴에서 'Help > Cheatsheets > Data Visualization with ggplot2' 에서 확인 가능

- 글로벌 매핑과 로컬 매핑

  - 글로벌 매핑: 함수 ggplot( )에서의 매핑. 해당 그래프 작성에 참여한 모든 geom 함수에 적용

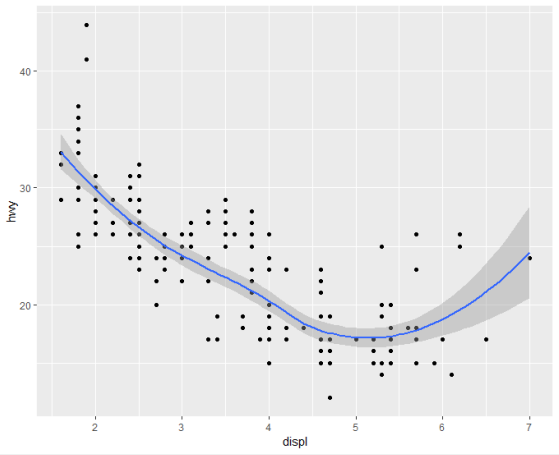
  - 로컬 매핑: geom 함수에서의 매핑. 해당 geom 함수로 작성되는 layer에만 적용. 해당 layer에서는 글로벌 매핑보다 우선해서 적용됨.

```
ggplot(data, mapping=aes( ) ) +
    geom_*(mapping=aes( ) ) +
    geom_*(mapping=aes( ) )
```

글로벌 매핑

로컬 매핑

- 예: mpg의 변수 displ과 hwy의 산점도에 비모수 회귀곡선 추가



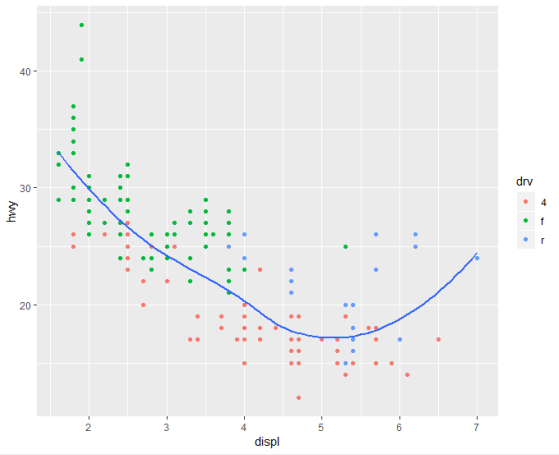
- 두 geom 함수에 동일한 내용의 매핑이 중복되어 입력

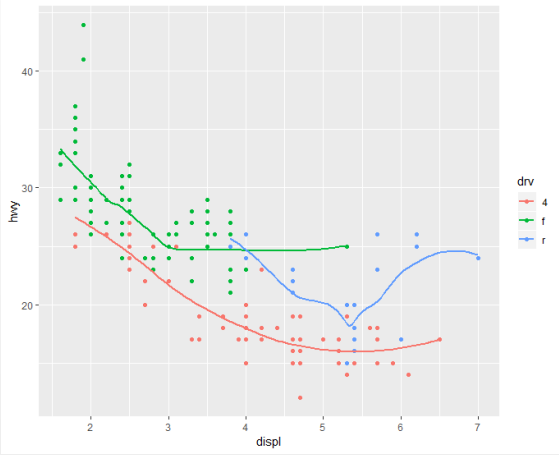
```
> ggplot(data=mpg) +
    geom_point(mapping=aes(x=displ,y=hwy)) +
    geom_smooth(mapping=aes(x=displ,y=hwy))
```

- 글로벌 매핑으로 중복 입력 문제 해결

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

- 예: mpg의 변수 displ과 hwy의 비모수 회귀곡선 작성. 그 위에 산점도 추가하되 drv의 값에 따라 점의 색을 구분.






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

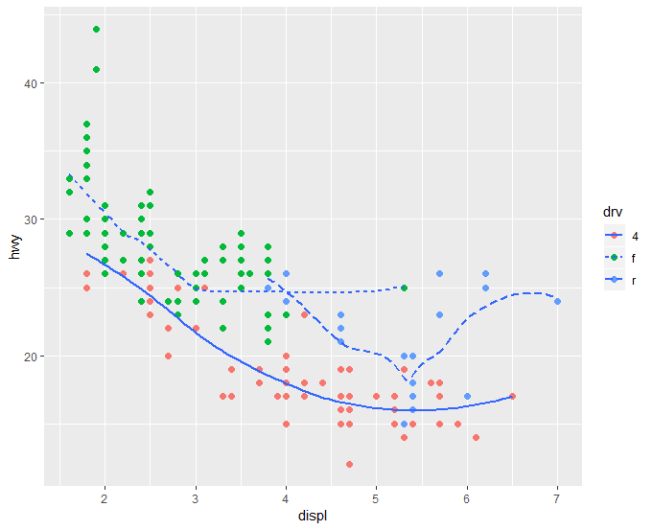
x, y: 글로벌 매핑
color: 로컬 매핑

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

x, y, color: 글로벌 매핑

- 예: mpg의 변수 displ과 hwy의 비모수 회귀곡선 작성하되 drv에 의해 구분되는 그룹별 각각 추정하여 선의 종류를 다르게 표시. 그 위에 산점도 추가하되 drv의 값에 따라 점의 색을 구분, 점의 크기 확대.

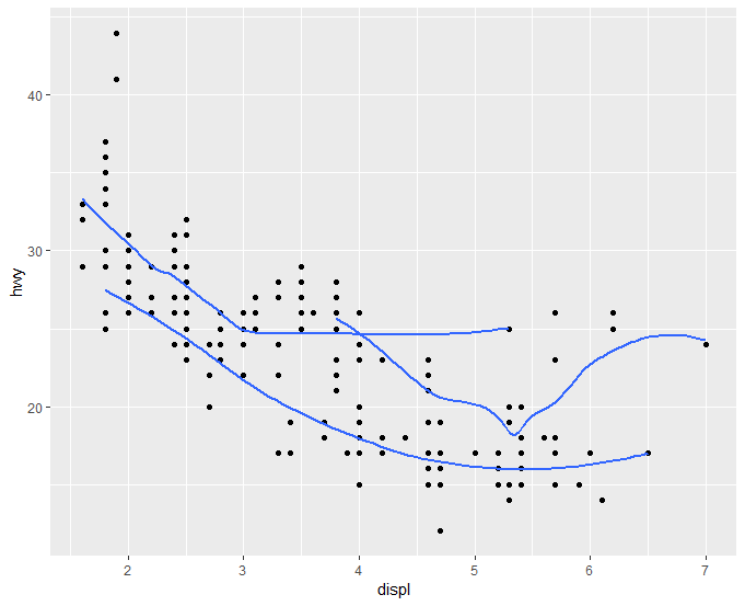
```
> ggplot(data=mpg, mapping=aes(x=displ, y=hwy)) +
    geom_point(mapping=aes(color=drv), size=2) +
    geom_smooth(mapping=aes(linetype=drv), se=FALSE)
```



linetype: 선의 종류를 나타내는 시각적 요소

- 예: 다음의 그래프 작성
  - 변수 drv의 그룹별로 따로 비모수 회귀곡선 작성하되, 선의 색과 종류는 같은 것을 사용

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

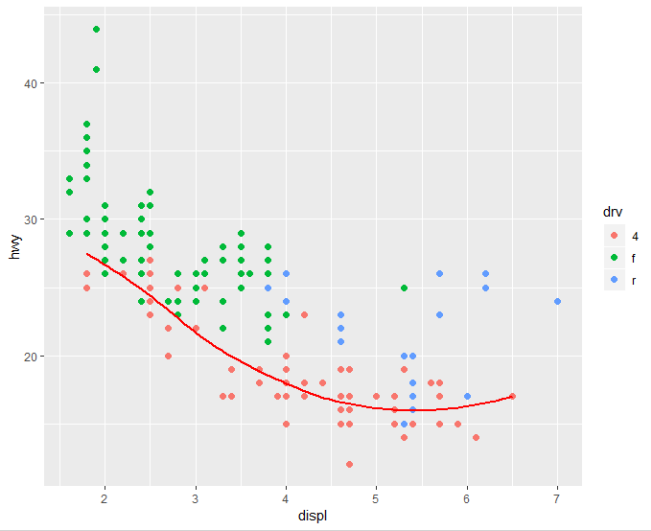


group: 그룹을 구성하는 시각적 요소

- 각 geom 함수에서 다른 데이터 사용
  - 각 geom 함수로 작성되는 layer마다 다른 데이터로 그래프 작성 가능

• 예: mpg의 변수 displ과 hwy의 산점도. drv에 따라 점의 색 구분.
비모수 회귀곡선 추가하되 drv가 4인 데이터만을 대상으로 추정.

```
> ggplot(data=mpg,mapping=aes(x=displ, y=hwy)) +
    geom_point(mapping=aes(color=drv), size=2) +
    geom_smooth(data=filter(mpg, drv=="4"),
                se=FALSE, color="red")
```



5. 통계적 변환: Statistical transformation

- 그래프 작성에 사용되는 자료
  1) 입력된 자료: 산점도
  2) 입력된 자료를 대상으로 통계적 변환 과정을 거쳐 생성된 자료: 비모수 회귀곡선 그래프

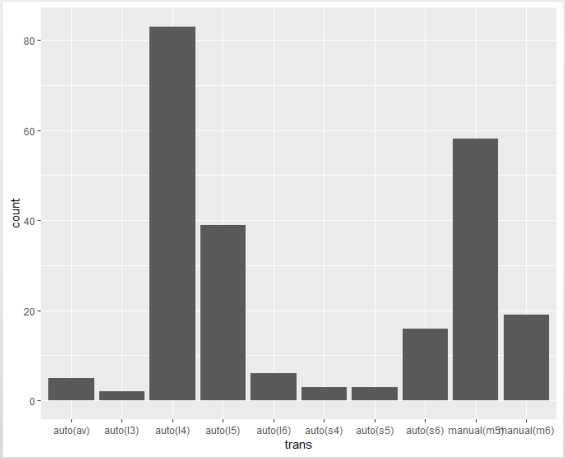
- 통계적 변환(stat)
  - 입력된 데이터 프레임 자료의 변환을 의미
  - 각 그래프 유형별 대응되는 stat 존재
    - 산점도: stat="identity"
    - 비모수 회귀곡선: stat="smooth"
    - 막대 그래프: stat="count"
  - 각 geom 함수마다 대응되는 디폴트 stat 존재
    - geom_point( ) → geom_point(stat="identity")
    - geom_smooth( ) → geom_smooth(stat="smooth")
    - geom_bar( ) → geom_bar(stat="count")

- stat 함수

  - geom 함수 대신 stat 함수로 그래프 작성 가능

  - 각 stat 함수마다 디폴트 geom 존재
    - stat_identity( ) → stat_identity(geom="point")
    - stat_smooth( ) → stat_smooth(geom="smooth")
    - stat_count( ) → stat_count(geom="bar")

- 예: mpg의 변수 trans의 막대 그래프 작성




```
> ggplot(data=mpg,mapping=aes(x=trans)) +
    geom_bar()
> ggplot(data=mpg,mapping=aes(x=trans)) +
    stat_count()
```

- 그래프 작성은 geom 함수 또는 stat 함수 모두 가능
- 사용자가 선택할 사항
- geom 함수의 사용이 비교적 더 명확함

- stat으로 계산된 변수의 이용

  - stat 함수: 입력된 데이터 프레임을 대상으로 변환을 실시하여 그래프 작성에 필요한 변수로 이루어진 데이터 프레임을 내부적으로 생성

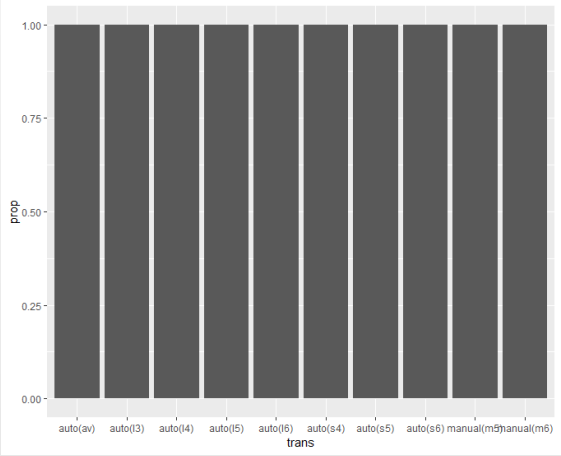
  - 생성된 변수를 사용자가 직접 지정해서 사용 가능

  - 예: 함수 geom_bar( ) 혹은 stat_count( )에서 계산된 변수
    - 변수 count: 각 범주의 빈도
    - 변수 prop: 그룹별 비율

  - 계산된 변수를 사용자가 지정할 때에는 변수를 ".."기호로 감싸야 함
    - 원래 데이터 프레임에 있는 변수와 혼동 방지
    - 예: `..count..` 또는 `..prop..`

- 예: mpg의 변수 trans의 막대 그래프를 상대 도수로 작성
  - 변수 ..prop..를 이용하여 막대 그래프 작성

```
> ggplot(data=mpg) +
    geom_bar(mapping=aes(x=trans, y=..prop..))
```



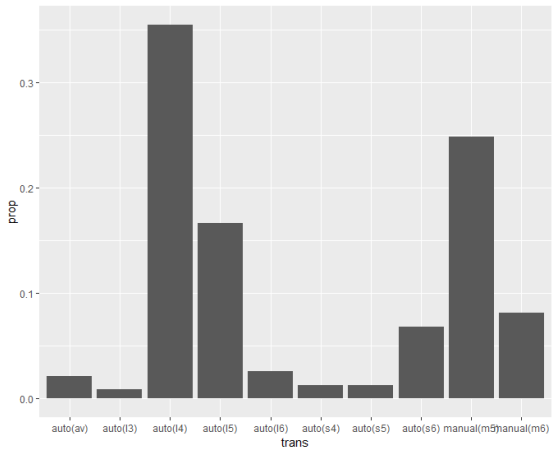
- stat_count( )에서 생성된 자료

	trans	count	prop	group
1	auto(av)	5	1	1
2	auto(l3)	2	1	2
3	auto(l4)	83	1	3
4	auto(l5)	39	1	4
5	auto(l6)	6	1	5
6	auto(s4)	3	1	6
7	auto(s5)	3	1	7
8	auto(s6)	16	1	8
9	manual(m5)	58	1	9
10	manual(m6)	19	1	10

- 변수 ..prop.. : 그룹별 비율
- 모든 범주를 하나의 그룹으로 구성

- 상대 도수 막대 그래프 작성

```
> ggplot(data=mpg) +
    geom_bar(mapping=aes(x=trans, y=..prop.., group=1))
```



- group에 하나의 값 지정
- 어떤 값도 가능

	trans	count	prop	group
1	auto(av)	5	0.021367521	1
2	auto(l3)	2	0.008547009	1
3	auto(l4)	83	0.354700855	1
4	auto(l5)	39	0.166666667	1
5	auto(l6)	6	0.025641026	1
6	auto(s4)	3	0.012820513	1
7	auto(s5)	3	0.012820513	1
8	auto(s6)	16	0.068376068	1
9	manual(m5)	58	0.247863248	1
10	manual(m6)	19	0.081196581	1

- geom 함수에서 stat을 따로 지정해야 하는 경우

  - geom 함수의 디폴트 stat이 아닌 다른 stat을 사용해야 하는 경우
  - 예: 도수분포표로 막대 그래프를 작성하는 경우

- mpg의 trans를 auto(av)에서 auto(s6)를 auto로, manual(m5)와 manual(m6)를 manual로 통합. 도수분포표 작성. 막대 그래프 작성.

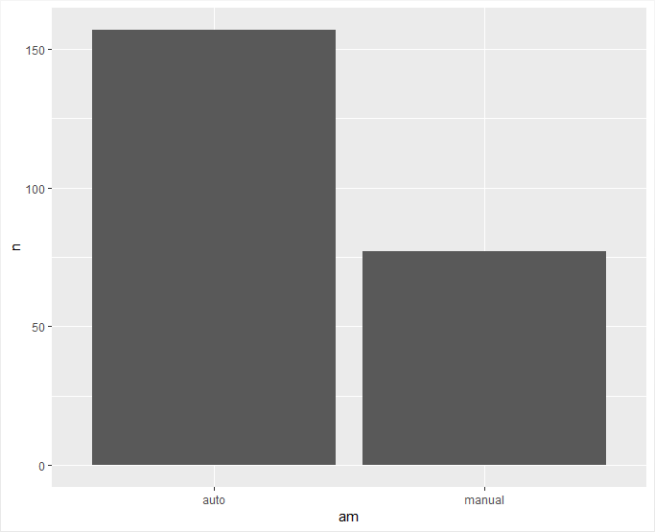
  - 통합된 범주의 도수분포 tibble 작성

```
> mpg_am <- mpg %>%
+   mutate(am=substr(trans, 1, nchar(trans)-4)) %>%
+   group_by(am) %>%
+   summarize(n=n())
```

```
> mpg_am
# A tibble: 2 x 2
  am          n
  <chr>  <int>
1 auto     157
2 manual    77
```

- 도수분포표로 막대 그래프 작성

```
> ggplot(data=mpg_am) +
    geom_bar(mapping=aes(x=am, y=n), stat="identity")
```




```
> mpg_am
# A tibble: 2 x 2
  am          n
  <chr>   <int>
1 auto      157
2 manual     77
```

6. 위치 조정: Position adjustments

- 그래프 요소들의 위치 조정
  - 연속형 자료: 산점도의 점이 겹쳐지는 경우
  - 범주형 자료: 이변량 막대 그래프 작성

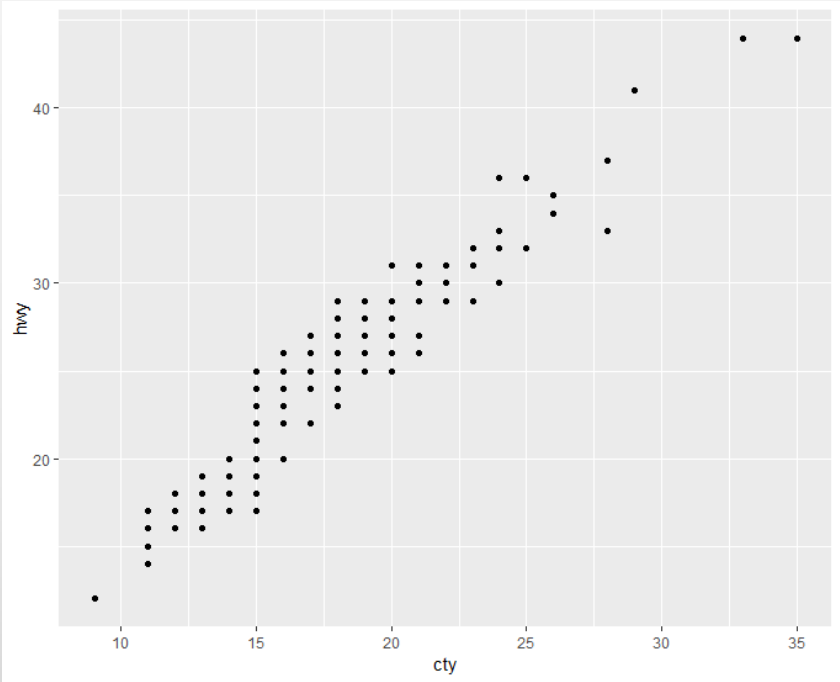
- 산점도의 점이 겹치는 문제
  - 산점도 작성의 가장 큰 문제
  - 해결 방안
    - 반올림 처리 등으로 같은 값이 많은 자료의 경우: 자료에 약간의 난수를 더해 점의 위치 조정(jittering)
    - 대규모의 자료가 좁은 구역에 몰려서 한 무리를 형성하는 경우: 추후에 다룰 예정

- 이변량 막대 그래프
  - 쌓아 올린 막대 그래프 / 옆으로 붙여 놓은 막대 그래프 / mosaic plot

- 산점도에서 점이 겹쳐지는 문제 해결

  - 예: mpg에서 변수 cty와 hwy의 산점도 작성

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



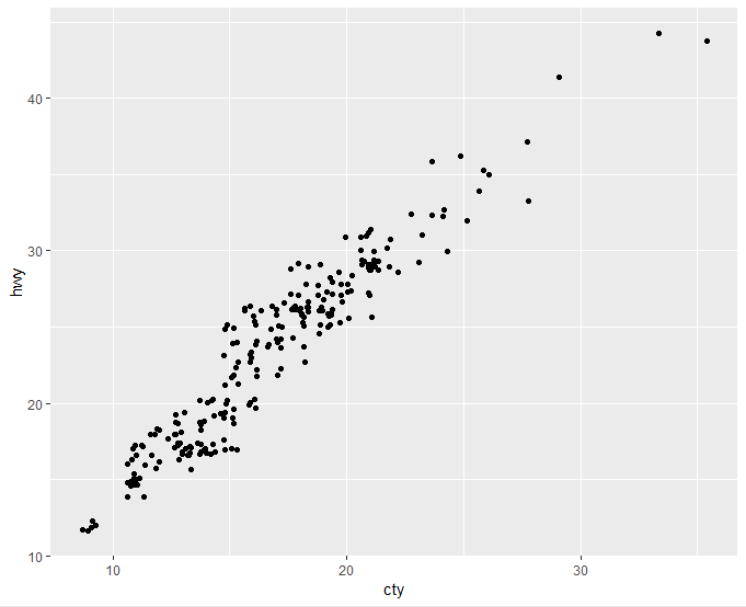
- 산점도에 나타난 점의 개수가 전체 데이터 개수인 234개에 훨씬 못 미쳐 보임

- 두 변수의 값이 반올림 처리 되어 같은 값이 많아짐

jittering
- 자료에 약간의 난수 추가
- $(x, y) \rightarrow (x + \varepsilon,\ y + \varepsilon)$
  $\varepsilon \sim Unif(-\alpha, \alpha)$

- jittering 실시

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

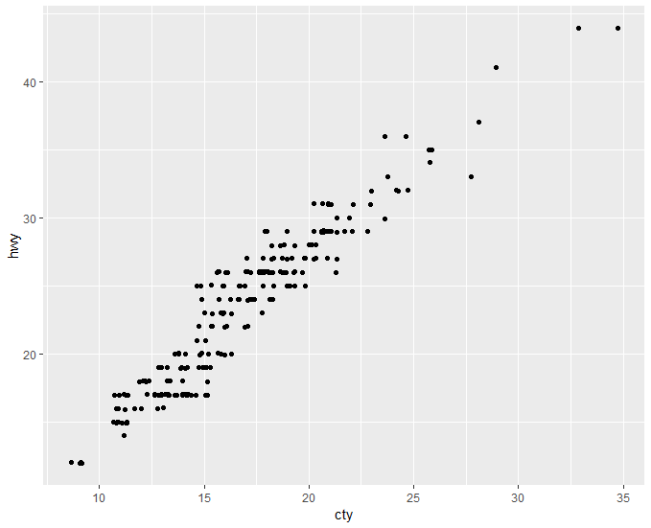


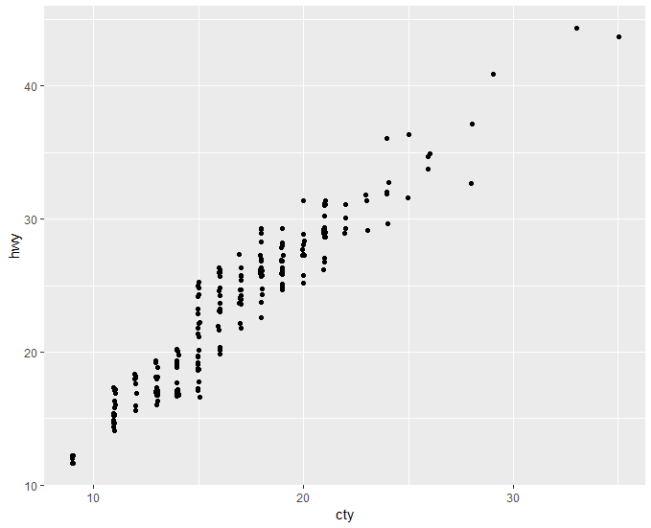
- 작성되는 그래프마다 미세한 차이 발생
- 추가되는 난수의 크기를 조절하고자 하는 경우에는 함수 geom_jitter( ) 사용

- 함수 geom_jitter() 사용

```
> ggplot(data=mpg, mapping=aes(x=cty,y=hwy))+
    geom_jitter(width=0.4, height=0.05)
```

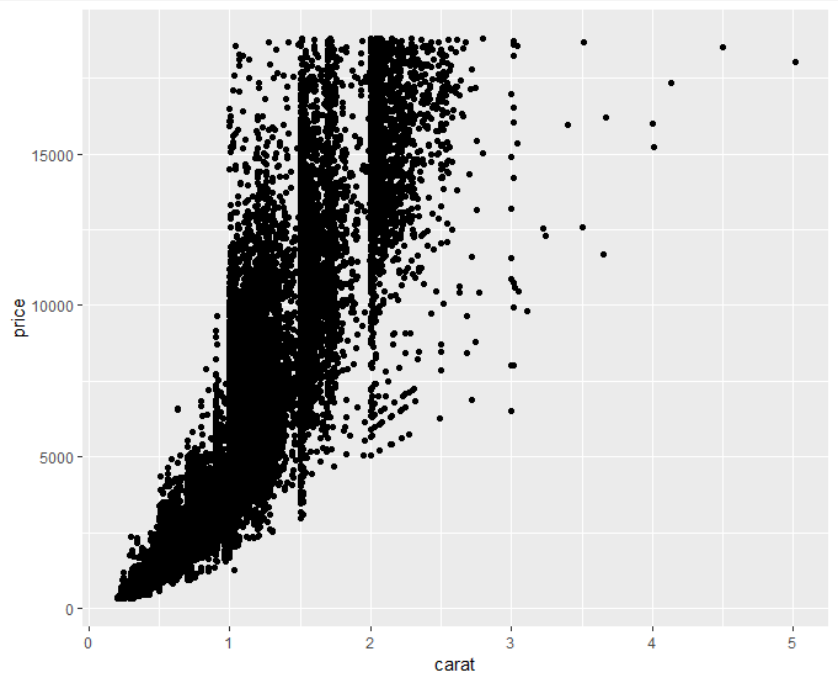
```
> ggplot(data=mpg, mapping=aes(x=cty,y=hwy))+
    geom_jitter(width=0.05, height=0.4)
```





- 예: diamonds에서 변수 carat과 price의 산점도

```
> ggplot(data=diamonds) +
+   geom_point(mapping=aes(x=carat,y=price))
```



- 너무 많은 점들이 밀집되어 있는 상황
- 두 변수의 정확한 관계 파악이 어려운 그래프
- Jittering으로는 문제 해결이 불가능
- 다른 방법은?

- 이변량 막대 그래프 작성
  - 막대 그래프 작성: geom_bar( )
  - 이변량 막대 그래프: 함수 geom_bar( )에 시각적 요소 x와 fill, position 사용

• 예제: mpg에서 trans의 범주를 auto와 manual로 통합한 변수 am 생성
변수 cyl이 5인 케이스 제거 후 am과 cyl의 이변량 막대 그래프 작성

- 자료 준비

```
> mpg_1 <- mpg %>%
    mutate(am=substr(trans,1,nchar(trans)-4)) %>%
    filter(cyl!=5)
```

- 쌓아 올린 막대 그래프와 옆으로 붙여 놓은 막대 그래프 작성

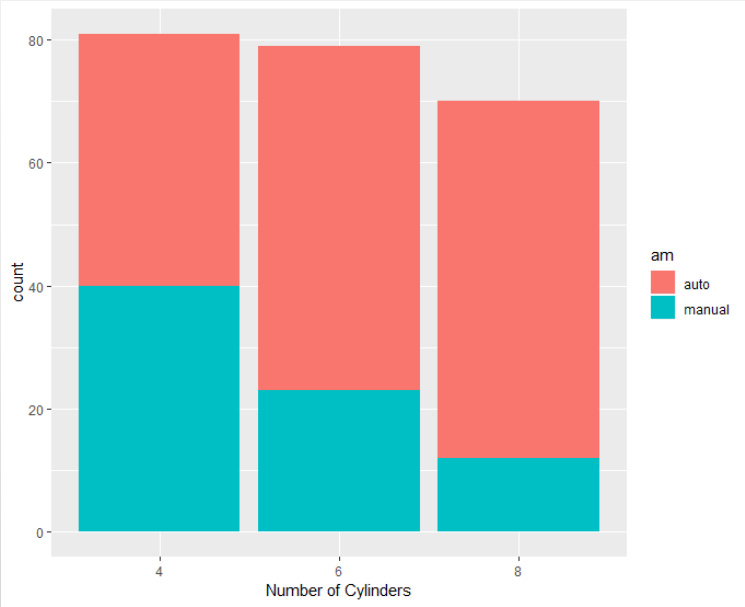
```
> p_1 <- ggplot(data=mpg_1,
          mapping=aes(x=as.factor(cyl), fill=am)) +
    xlab("Number of Cylinders")
```

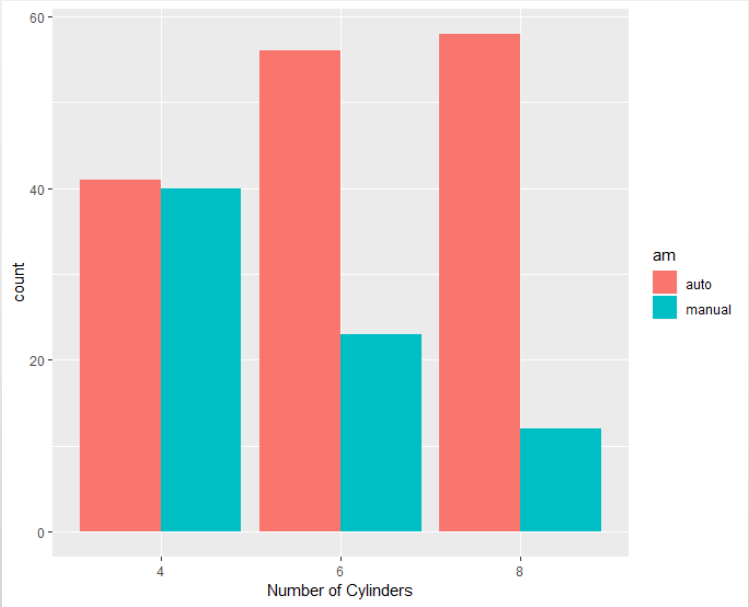
디폴트
position="stack"

```
> p_1 + geom_bar()
```

ggplot(mpg_1, aes(x=cyl, fill=am))을 실행하면 어떤 문제가 있는가?

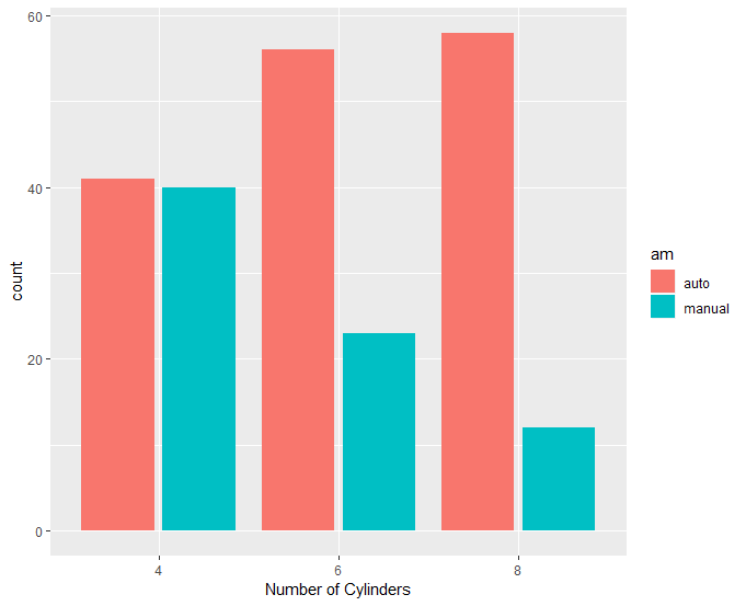
```
> p_1 + geom_bar(position="dodge")
```

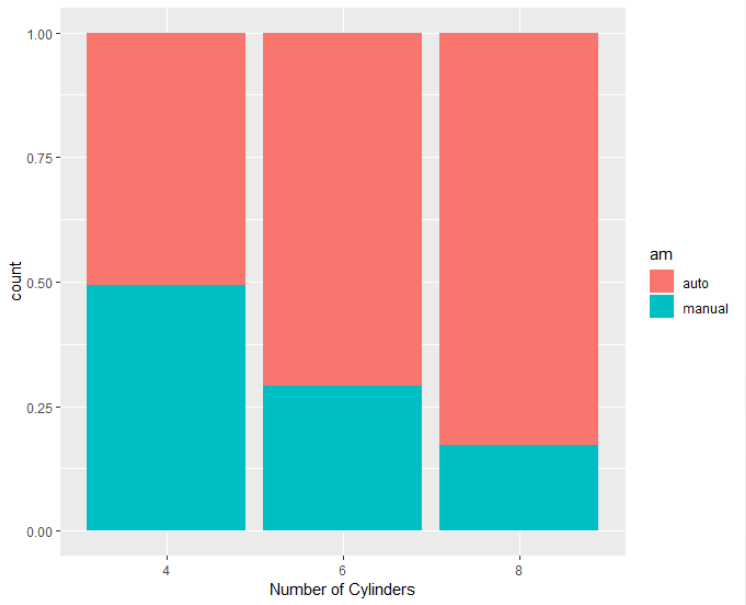




```
> p_1 + geom_bar(position="dodge2")
```

```
> p_1 + geom_bar(position="fill")
```





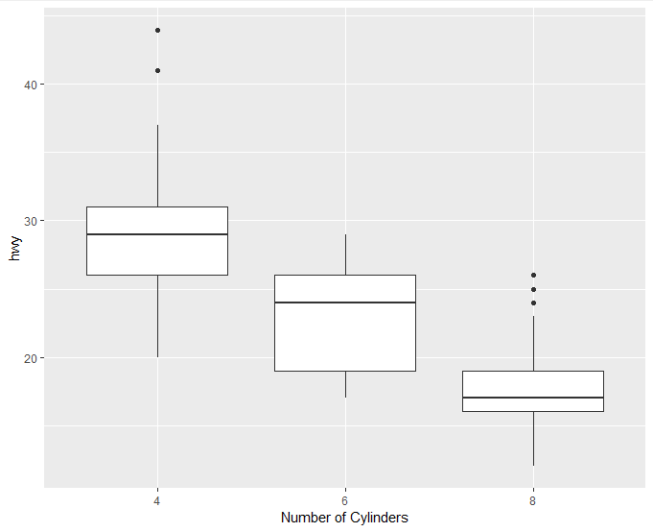
cyl을 조건으로 하는 cyl과 am의 조건부 확률

```
    am
cyl      auto    manual
  4 0.5061728 0.4938272
  6 0.7088608 0.2911392
  8 0.8285714 0.1714286
```

- 나란히 서 있는 상자그림

  - geom_boxplot( )
  - 필요한 시각적 요소: x=그룹을 구성하는 변수(요인)
    y=연속형 변수

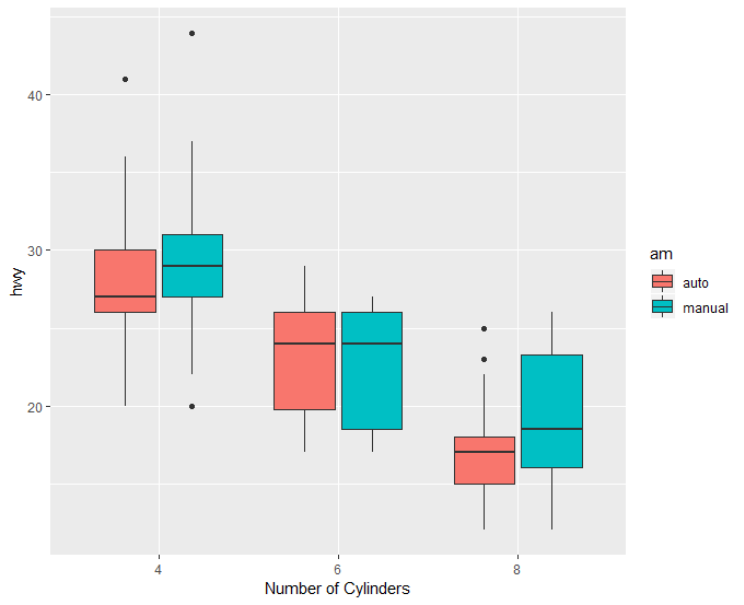
```
> ggplot(data=mpg_1, mapping=aes(x=as.factor(cyl),y=hwy)) +
    geom_boxplot() +
    xlab("Number of Cylinders")
```



- 시각적 요소 x에 요인이 아닌 변수를 매핑하면?
- 그룹을 구성하는 변수가 두 개이면?

- 그룹을 구성하는 변수가 두 개인 경우의 상자그림

```
> ggplot(data=mpg_1, mapping=aes(x=as.factor(cyl),y=hwy)) +
    geom_boxplot(mapping=aes(fill=am)) +
    xlab("Number of Cylinders")
```



- 변수 am에 따라 다른 색이 채워져 있고 두 상자그림이 옆에 붙어 있다
- 필요한 시각적 요소: x, y, fill, position
- position="dodge"가 디폴트로 적용됨