Part I. Explore

Spring 2019 - Statistical Graphics

1. Data visualisation wigh ggplot2

Prerequisites

- ggplot2 와 R4DS 의모든 package 들을 모아놓은 tidyverse 가 필요함

```
library(tidyverse)
```

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

## √ ggplot2 3.1.0     √ purrr   0.2.5
## √ tibble  1.4.2     √ dplyr   0.7.8
## √ tidyr   0.8.2     √ stringr 1.3.1
## √ readr   1.2.1     √ forcats 0.3.0

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

mpg 자료

```
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl    cla~
##    <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <ch>
##  1 audi         a4      1.8  1999     4 auto~ f        18    29 p     com~
##  2 audi         a4      1.8  1999     4 manu~ f        21    29 p     com~
##  3 audi         a4      2    2008     4 manu~ f        20    31 p     com~
##  4 audi         a4      2    2008     4 auto~ f        21    30 p     com~
##  5 audi         a4      2.8  1999     6 auto~ f        16    26 p     com~
##  6 audi         a4      2.8  1999     6 manu~ f        18    26 p     com~
##  7 audi         a4      3.1  2008     6 auto~ f        18    27 p     com~
##  8 audi         a4 q~   1.8  1999     4 manu~ 4        18    26 p     com~
##  9 audi         a4 q~   1.8  1999     4 auto~ 4        16    25 p     com~
## 10 audi         a4 q~   2    2008     4 manu~ 4        20    28 p     com~
## # ... with 224 more rows
```

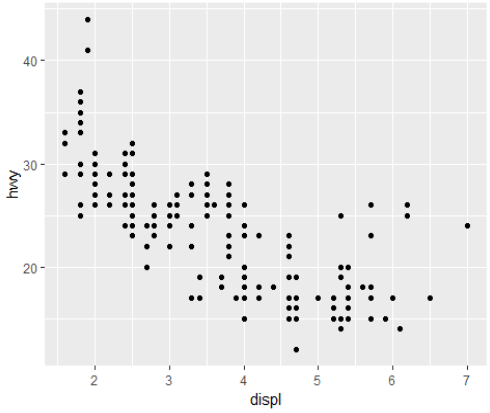
- mpg 자료의 변수들
- manufacturer: 자동차 제조사
- model : 자동차 모델명

- `displ` : 엔진크기 (liter)
- `year` : 연식
- `cyl`: 실린더 수
- `trans`: transmission 타입
- `drv`: f= front-wheel drive, r = rear wheel drive, 4 = 4wd
- `cty` : 도심도로 마일리지 (mile/gallon)
- `hwy` : 고속도로 마일리지 (mile/gallon)
- `fl` : fuel type
- `class` : 자동차 타입

Creating a ggplot

- displ 과 hwy 의 산점도

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

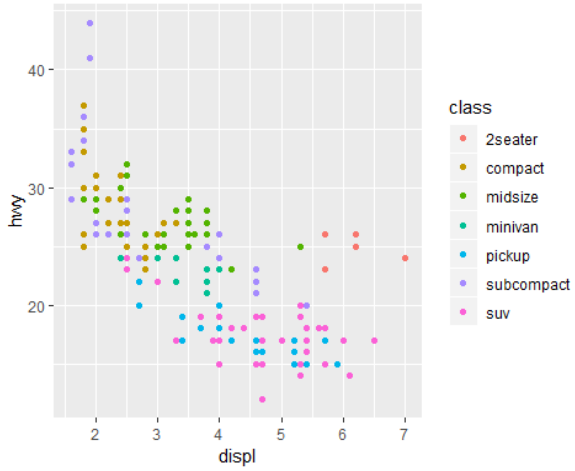



```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Aesthetic mappings

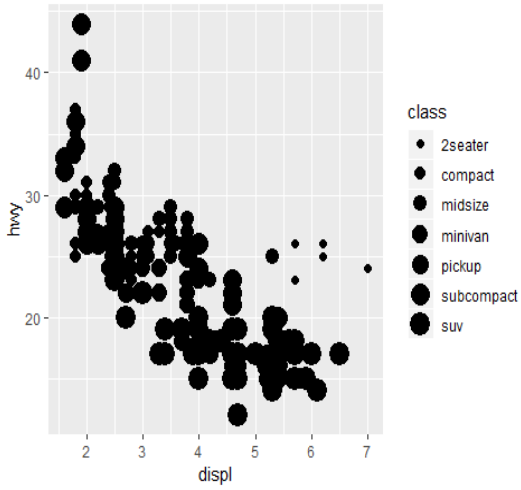
- class 마다 다른 색의 점으로 산점도 그리기

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

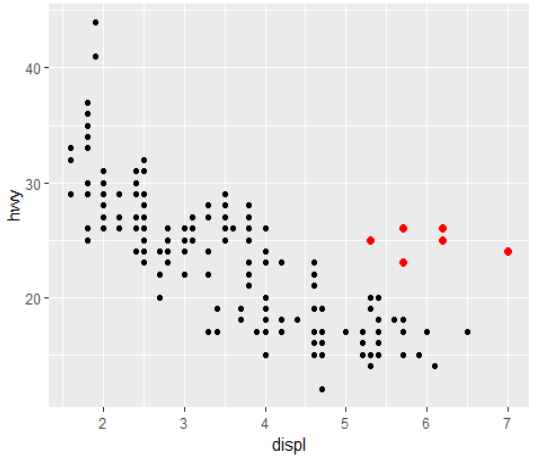


- class 마다 크기를 달리하는 점으로 산점도 그리기

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

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

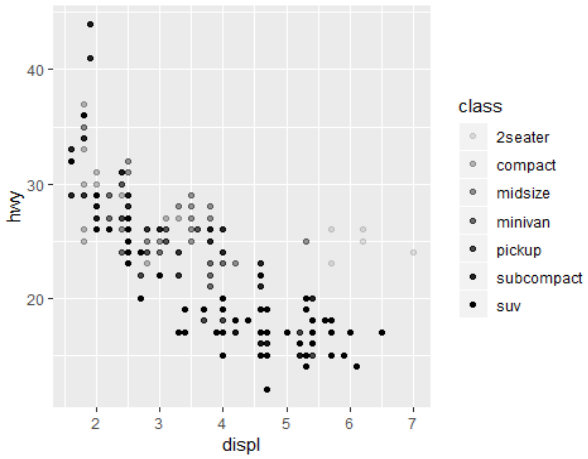


- 특정 범위의 자료에 대한 색을 달리하는 산점도 그리기



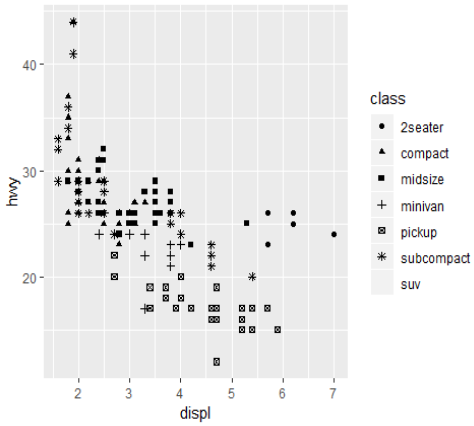
- class 마다 투명도를 달리하는 점으로 산점도 그리기

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

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



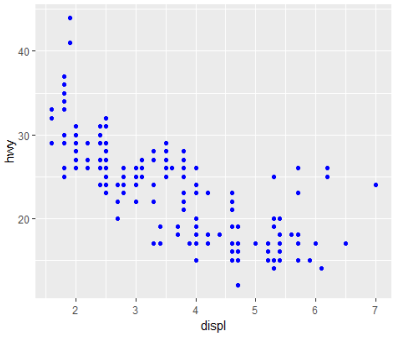
- class 마다 모양을 달리하는 점으로 산점도 그리기

```
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).
```

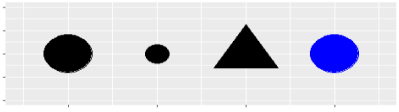


- 점 전체의 색을 “blue”로 바꾸기

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

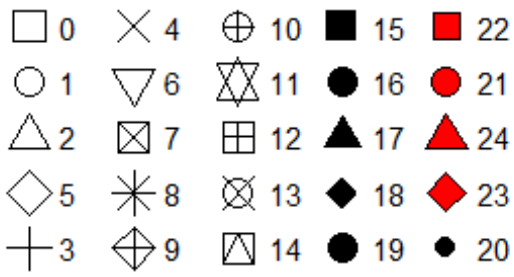


- 다양한 aesthetic mapping 의 예



- shape 의 다양한 점 모양

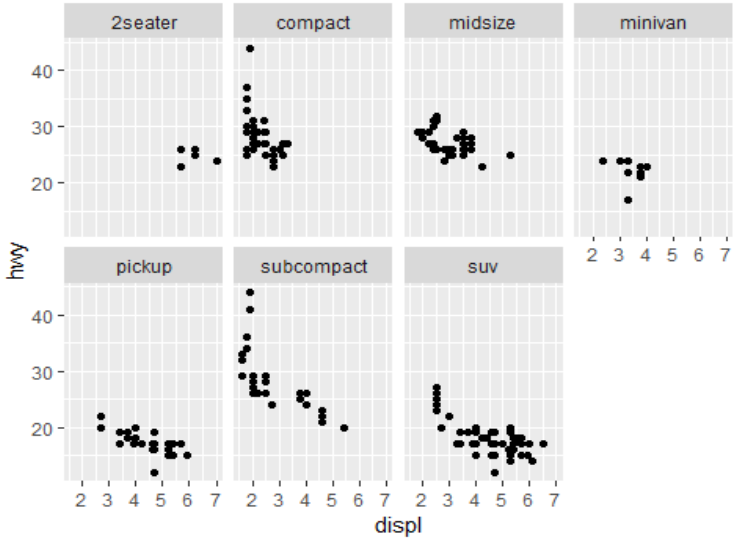
```
shapes <- tibble(
  shape = c(0, 1, 2, 5, 3, 4, 6:19, 22, 21, 24, 23, 20),
  x = (0:24 %/% 5) / 2,
  y = (-(0:24 %% 5)) / 4
)
ggplot(shapes, aes(x, y)) +
  geom_point(aes(shape = shape), size = 5, fill = "red") +
  geom_text(aes(label = shape), hjust = 0, nudge_x = 0.15) +
  scale_shape_identity() +
  expand_limits(x = 4.1) +
  scale_x_continuous(NULL, breaks = NULL) +
  scale_y_continuous(NULL, breaks = NULL, limits = c(-1.2, 0.2)) +
  theme_minimal() +
  theme(aspect.ratio = 1/2.75)
```



Facets

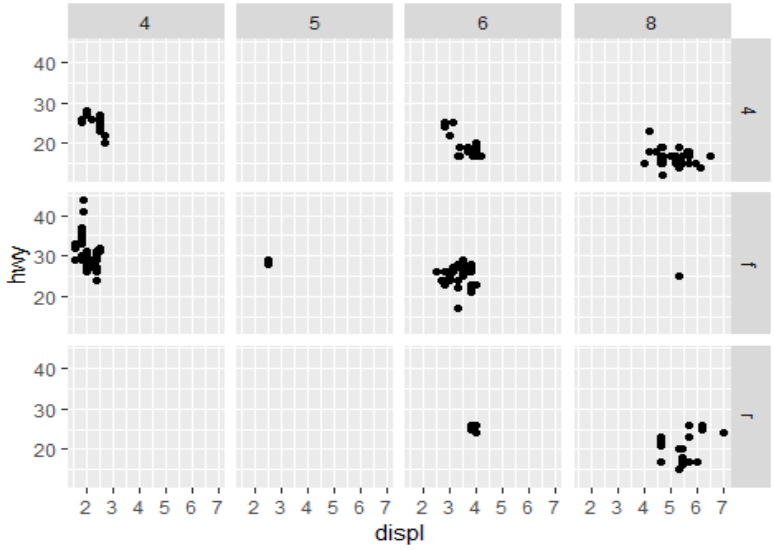
- facet_wrap: 하나의 범주형 변수를 이용, 범주마다 나눠서 그림 그리기

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



- facet_grid: 두개의 범주형 변수를 이용, 범주마다 나눠서 그림 그리기

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

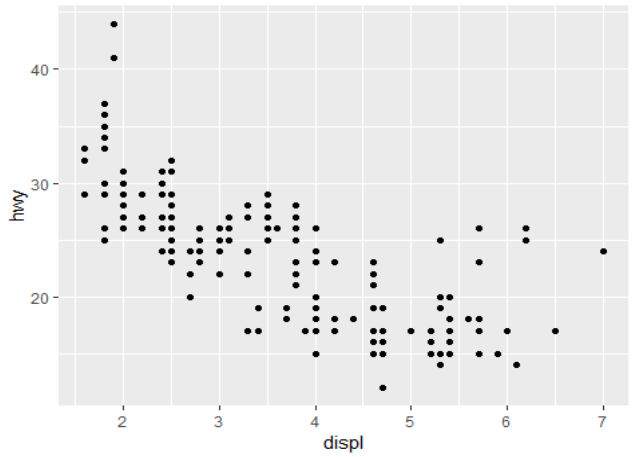


Geometric objects

- 다양한 geom 에서 aesthetic mapping 이용하기

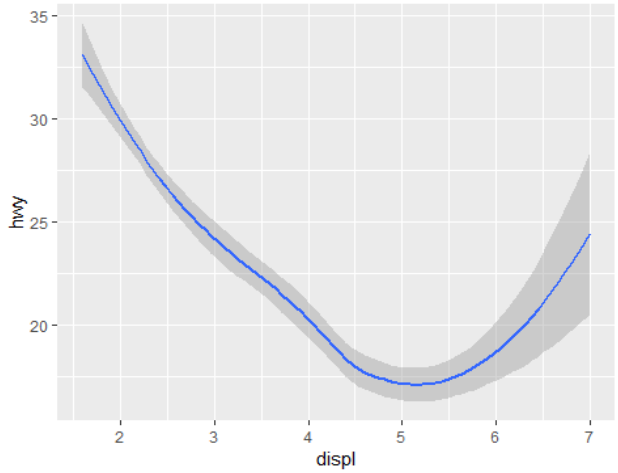
- geom_point

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



- geom_smooth

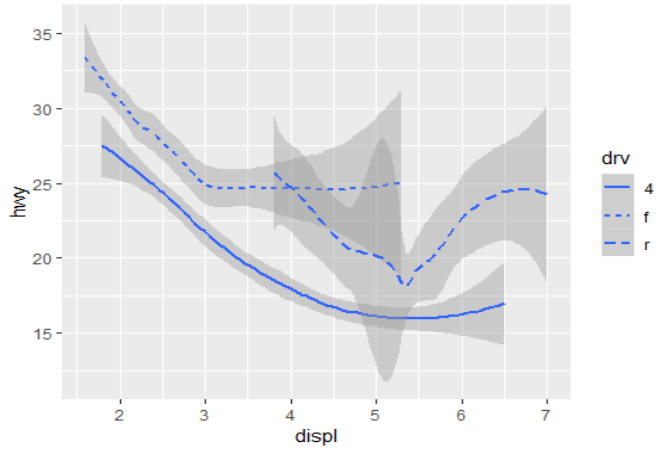
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))

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



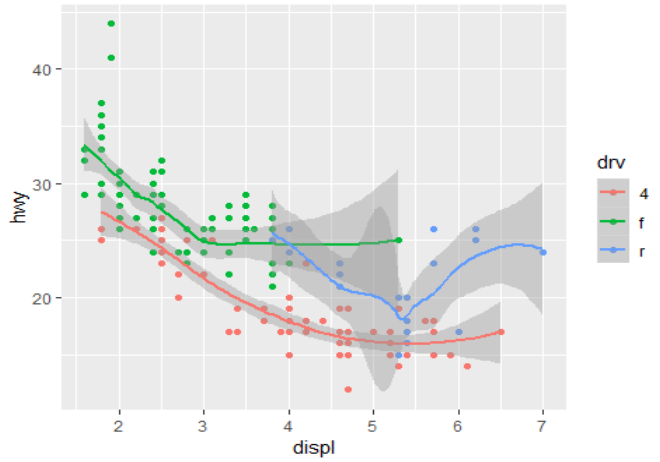
- drv 별로 smooth line 따로 그리기

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

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

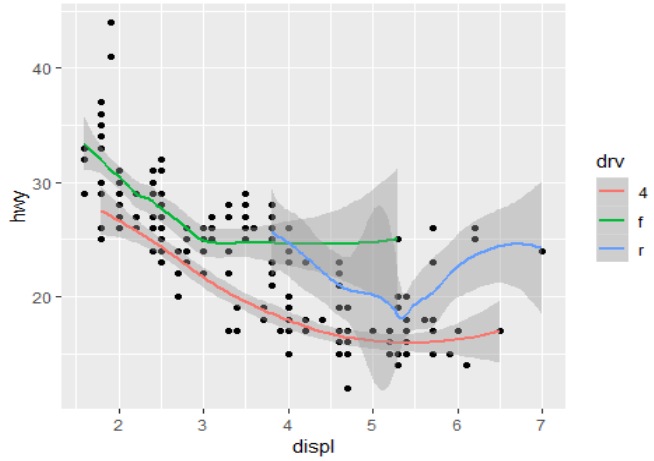


- geom_point 와 geom_smooth 를 함께 이용하는 경우 drv 별로 smooth line 을 따로 그리기
- `ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +`
- `  geom_point() + geom_smooth()`



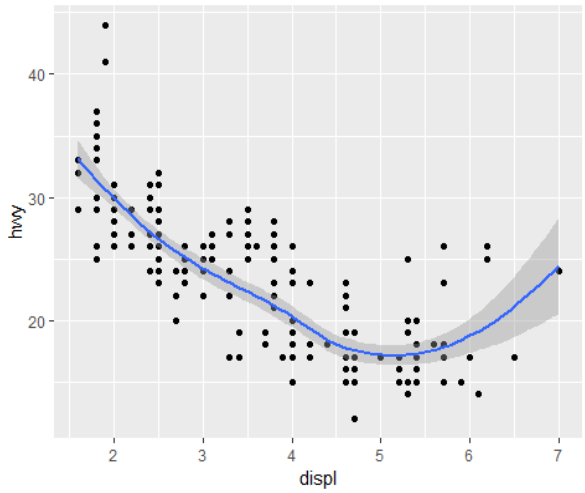
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy, color=drv))
```

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

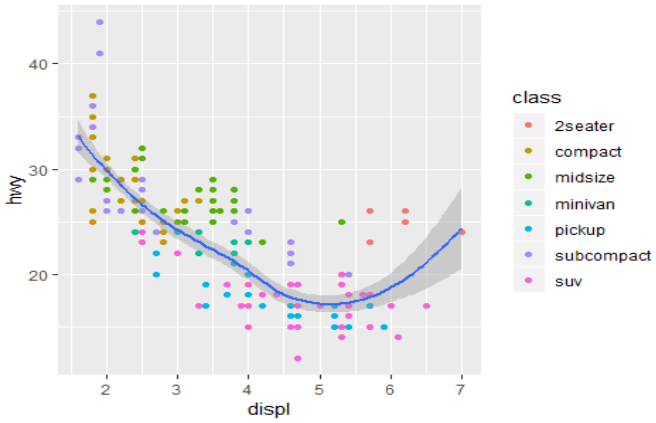



```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

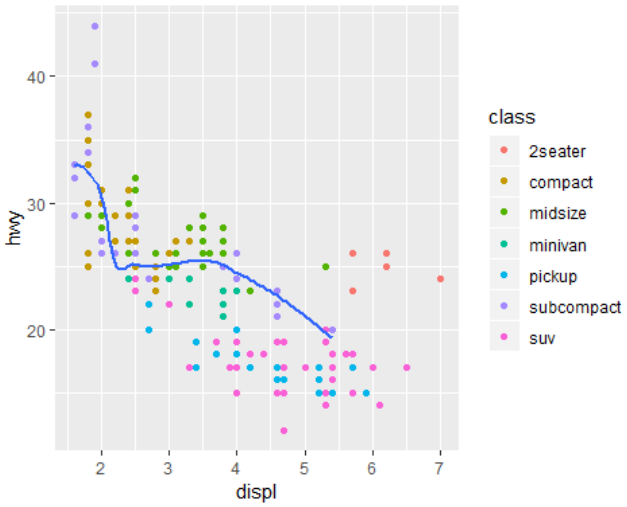
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```
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'
```



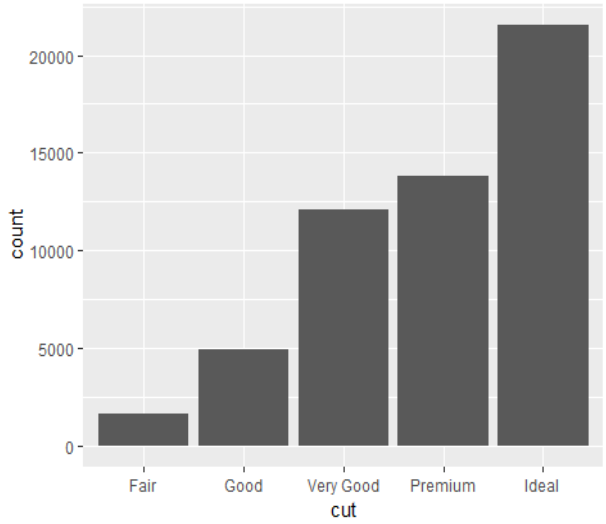

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)

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



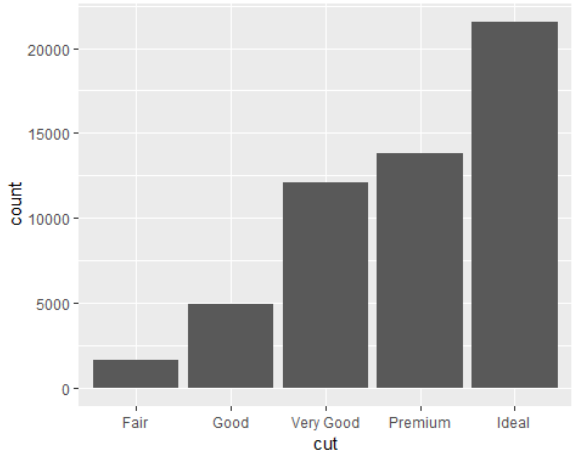
Statistical transformations

- geom_bar 를 그리기 위해 stat_count 를 이용하여 자료를 변환

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

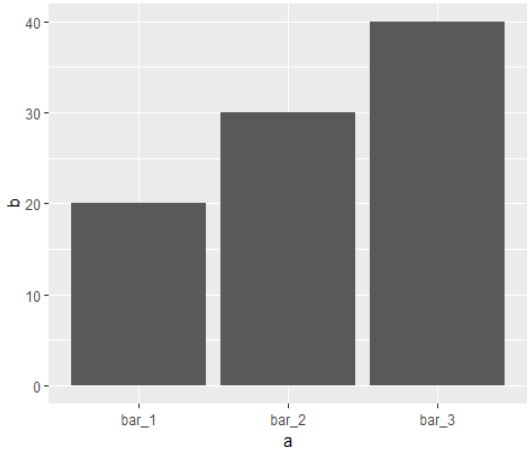



```
ggplot(data = diamonds) +
  stat_count(mapping = aes(x = cut))
```



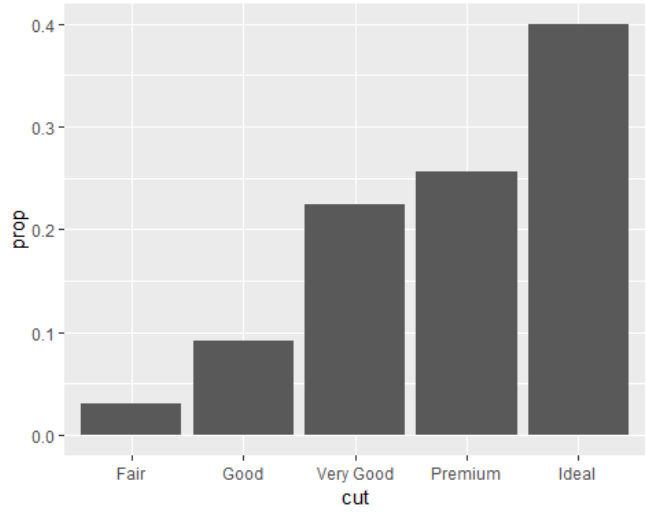
- table 로 정리된 자료를 이용하여 bar chart 를 그리는 경우 `stat="identity"`옵션을 이용

```
demo <- tribble(
  ~a,      ~b,
  "bar_1", 20,
  "bar_2", 30,
  "bar_3", 40
)

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



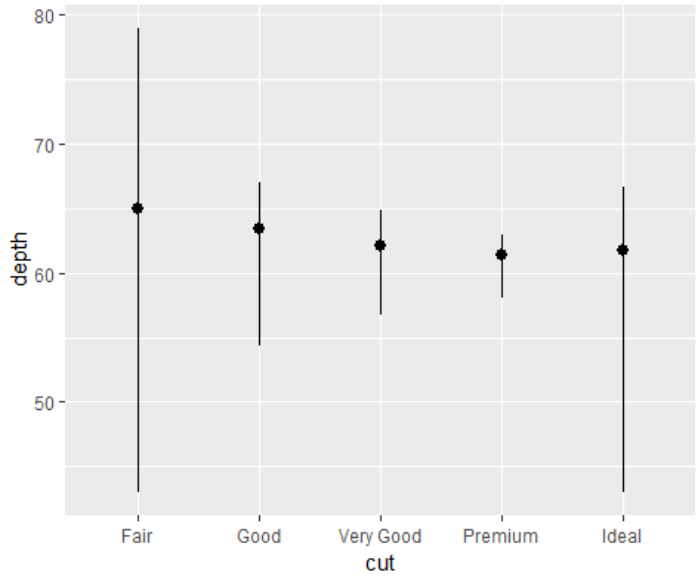
- y 축을 count 대신 proportion 으로 표현하여 그리기

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



- stat_summary 를 이용하여 다양한 통계량을 그림에 표현하기

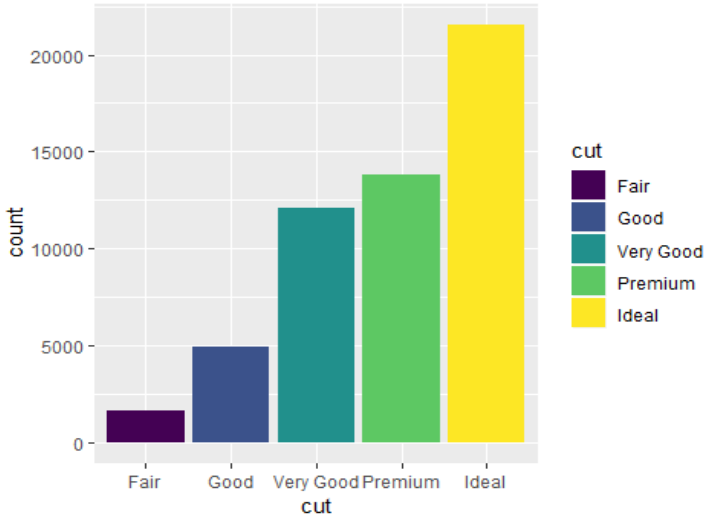
```
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
  )
```



Position adjustments

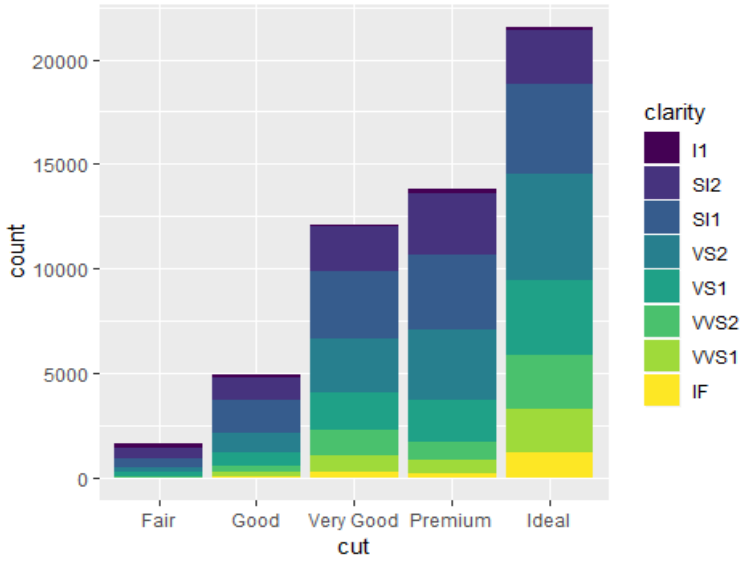
- geom_bar 의 fill 옵션 이용하기

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



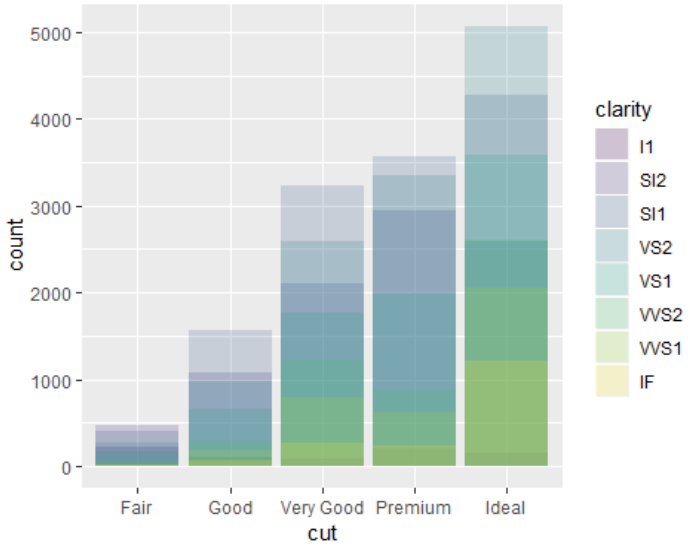
- fill=clarity 를 이용하여 각 cut 별로 clarity 비율 표현하기

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



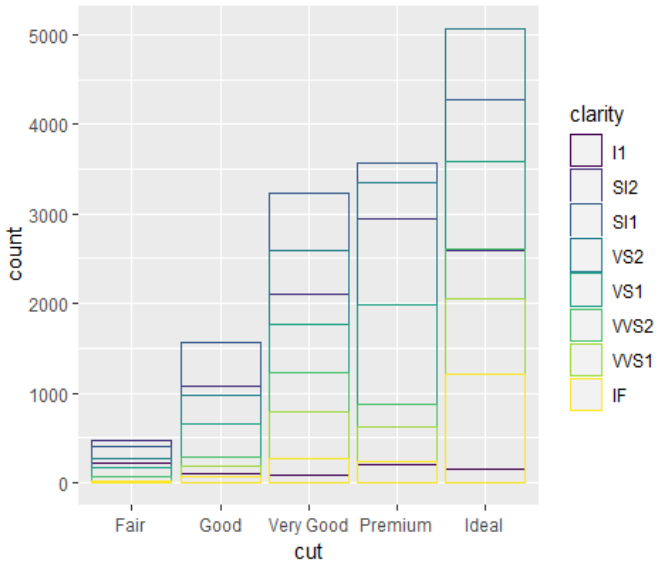
* position="identity"를 이용하여 각 cut 내에서 clarity 별 막대를 겹쳐그리기

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



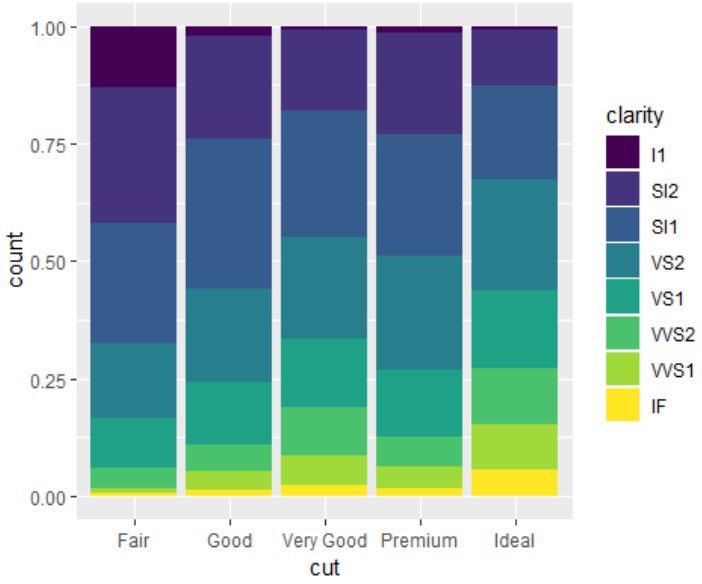
* fill = clarity 대신 colour = clarity 를 이용하여 각 cut 내에서 clarity 별 막대의 가장자리 선만 그리기

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



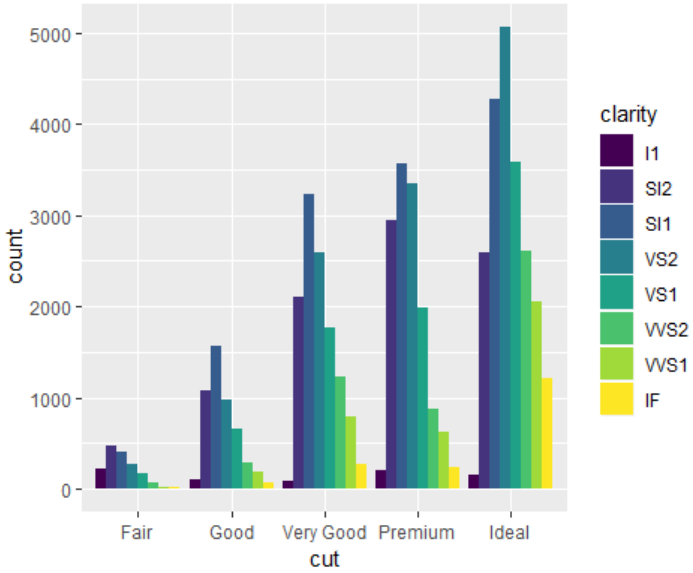
- `fill = clarity` 을 aesthetic mapping 밖에 지정하여 cut 별 막대를 1 로 했을 때의 clarity 비율 표현하기

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



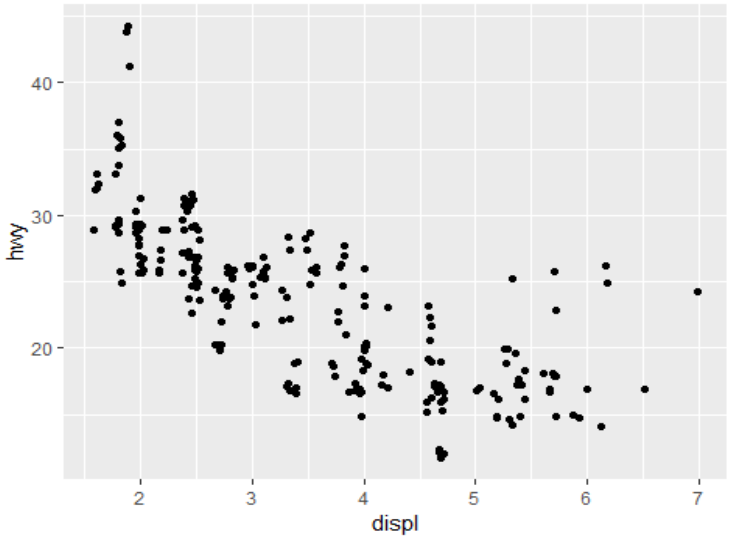
- `position = "dodge"`를 이용하여 각 cut 내에서 clarity 별 막대를 옆으로 나란히 그리기

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



- position = "jitter"을 이용하여 겹쳐지는 점을 흩뿌려 그리기

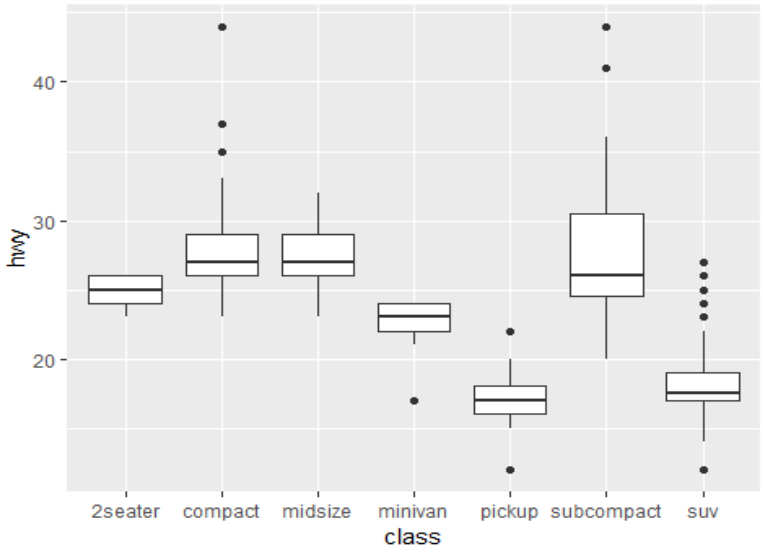
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), position = "jitter")
```



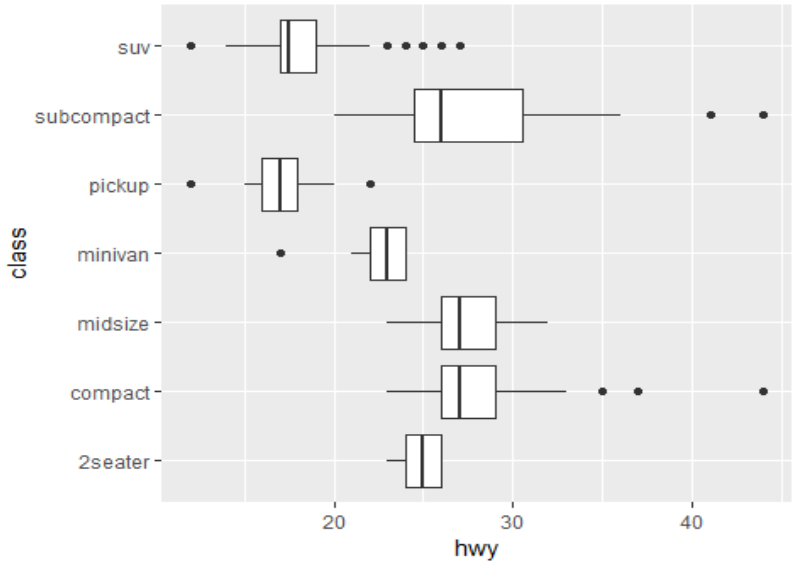
Coordinate systems

- coord_flip()을 이용하여 x 와 y 축 바꾸기

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

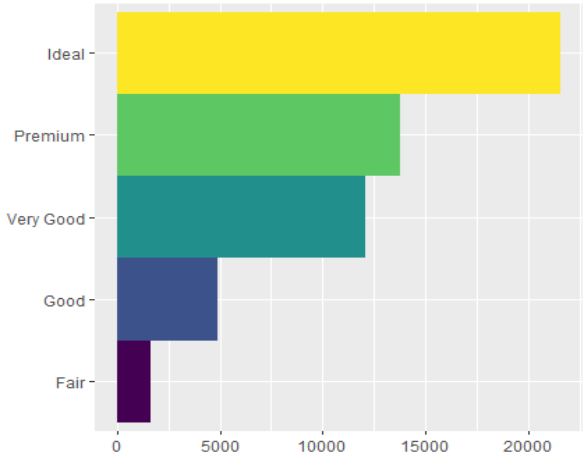



```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```

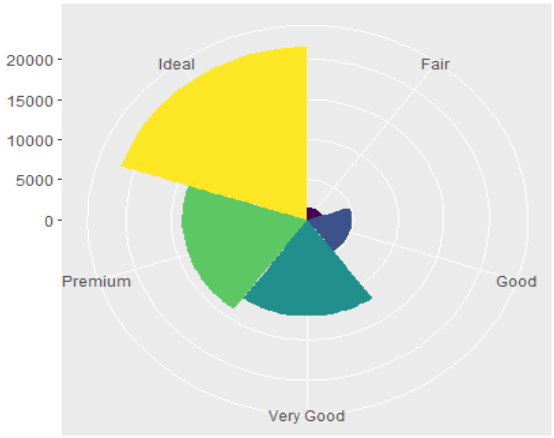


- coord_polar()를 이용하여 극좌표로 변환하기

```
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()
```



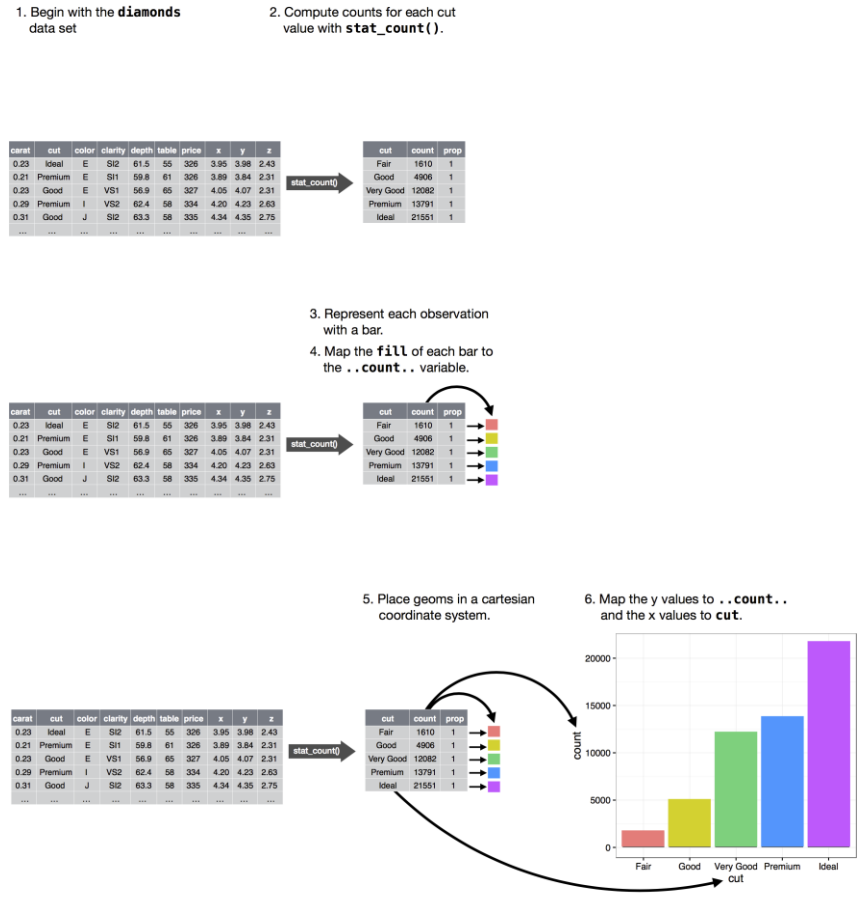

```
bar + coord_polar()
```



The layered grammar of graphics

- ggplot 의 일반적 사용법

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,
     position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION>
```



3. Data Transformation with dplyr

예제자료: nycflights13

- 2013 년 New York 으로부터 출발하는 비행기에 대한 자료로 `nycflights13` library 의 자료 중 `flights` 를 이용.
- `flights` 는 tibble 형태로 저장되어 있음.

```
library(nycflights13)
library(tidyverse)
flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

- tibble 에서 나타내는 변수의 형태
  - `int`: integers.
  - `dbl`: doubles, real numbers.
  - `chr`: character vectors, or strings.
  - `dttm`: date-times (a date + a time).
  - `lgl`: logical (`TRUE`/`FALSE`).
  - `fctr`: factors.
  - `date`: dates.

dplyr 의 기본 함수

- `filter()`: 조건에 맞는 관측 선택하기.
- `arrange()`: 자료에서 관측의 순서 바꾸기.
- `select()`: 변수 선택.

- `mutate()`: 새로운 변수 만들기.
- `summarise()`: 자료 요약하기.

Filter rows with `filter()`

- >, >=, <, <=, != (not equal), and == (equal)의 논리연산자를 이용하여 자료 선택
- 예: 1 월 1 일 ~ 1 월 2 일 자료 선택

```
filter(flights, month == 1, day <= 2)
```

```
## # A tibble: 1,785 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 1,775 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

- 수치연산의 비교

```
sqrt(2) ^ 2 == 2
```

```
## [1] FALSE
```

```
1/49 * 49 == 1
```

```
## [1] FALSE
```

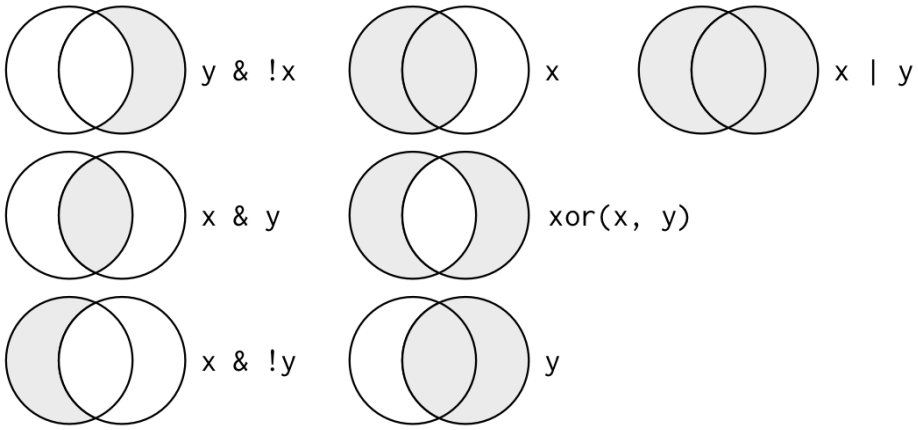
```
near(sqrt(2) ^ 2,  2)
```

```
## [1] TRUE
```

```
near(1 / 49 * 49, 1)
```

```
## [1] TRUE
```

- &, |, ! 이용



Complete set of boolean operations. x is the left-hand circle, y is the right-hand circle, and the shaded region show which parts each operator selects.

-예: 11 월, 12 월 자료 추출

```
filter(flights, month == 11 | month == 12)

nov_dec <- filter(flights, month %in% c(11, 12))
```

Missing values

```
NA > 5

## [1] NA

10 == NA

## [1] NA

NA + 10

## [1] NA

NA / 2

## [1] NA

NA == NA

## [1] NA

x = c(1, NA, 3)
is.na(x)

## [1] FALSE  TRUE FALSE

df <- tibble(x = c(1, NA, 3))
filter(df, x > 1)
```

```
## # A tibble: 1 x 1
##       x
##   <dbl>
## 1     3
```

```
filter(df, is.na(x) | x > 1)
```

```
## # A tibble: 2 x 1
##       x
##   <dbl>
## 1    NA
## 2     3
```

Arrange rows with `arrange()`

- 지정한 변수의 크기 순으로 자료를 정렬하기

```
arrange(flights, year, month, day)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

- 내림차순은 `desc()` 이요

```
arrange(flights, desc(arr_delay))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     9      641            900      1301     1242
##  2  2013     6    15     1432           1935      1137     1607
##  3  2013     1    10     1121           1635      1126     1239
##  4  2013     9    20     1139           1845      1014     1457
##  5  2013     7    22      845           1600      1005     1044
##  6  2013     4    10     1100           1900       960     1342
##  7  2013     3    17     2321            810       911      135
```

```
##  8  2013     7    22     2257            759       898      121
##  9  2013    12     5      756           1700       896     1058
## 10  2013     5     3     1133           2055       878     1250
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

- NA 값은 항상 마지막순서로.

```
df <- tibble(x = c(5, 2, NA))
arrange(df, x)
```

```
## # A tibble: 3 x 1
##       x
##   <dbl>
## 1     2
## 2     5
## 3    NA
```

```
arrange(df, desc(x))
```

```
## # A tibble: 3 x 1
##       x
##   <dbl>
## 1     5
## 2     2
## 3    NA
```

Select columns with `select()`

- 나열한 변수를 선택

```
# Select columns by name
select(flights, year, month, day)
```

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  1  2013     1     1
##  2  2013     1     1
##  3  2013     1     1
##  4  2013     1     1
##  5  2013     1     1
##  6  2013     1     1
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # ... with 336,766 more rows
```

```r
# Select all columns between year and day (inclusive)
select(flights, year:day)
```

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  1  2013     1     1
##  2  2013     1     1
##  3  2013     1     1
##  4  2013     1     1
##  5  2013     1     1
##  6  2013     1     1
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # ... with 336,766 more rows
```

```r
# Select all columns except those from year to day (inclusive)
select(flights, -(year:day))
```

```
## # A tibble: 336,776 x 16
##    dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
##       <int>          <int>     <dbl>    <int>          <int>     <dbl>
##  1      517            515         2      830            819        11
##  2      533            529         4      850            830        20
##  3      542            540         2      923            850        33
##  4      544            545        -1     1004           1022       -18
##  5      554            600        -6      812            837       -25
##  6      554            558        -4      740            728        12
##  7      555            600        -5      913            854        19
##  8      557            600        -3      709            723       -14
##  9      557            600        -3      838            846        -8
## 10      558            600        -2      753            745         8
## # ... with 336,766 more rows, and 10 more variables: carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- `starts_with("abc")`: “abc”로 시작하는 이름의 변수 선택
- `ends_with("xyz")`: “xyz”로 끝나는 이름의 변수 선택.
- `contains("ijk")`: “ijk”가 들어있는 이름의 변수 선택.
- `matches("(.)\\1")`: regular expression 이용
- `num_range("x", 1:3)`: x1, x2, x3 선택
- `rename()`: 변수 이름 바꾸기

```
rename(flights, tail_num = tailnum)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tail_num <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

- everything(): 나머지 모든 변수를 의미

```
select(flights, time_hour, air_time, everything())
```

```
## # A tibble: 336,776 x 19
##    time_hour           air_time  year month   day dep_time sched_dep_time
##    <dttm>                 <dbl> <int> <int> <int>    <int>          <int>
##  1 2013-01-01 05:00:00      227  2013     1     1      517            515
##  2 2013-01-01 05:00:00      227  2013     1     1      533            529
##  3 2013-01-01 05:00:00      160  2013     1     1      542            540
##  4 2013-01-01 05:00:00      183  2013     1     1      544            545
##  5 2013-01-01 06:00:00      116  2013     1     1      554            600
##  6 2013-01-01 05:00:00      150  2013     1     1      554            558
##  7 2013-01-01 06:00:00      158  2013     1     1      555            600
##  8 2013-01-01 06:00:00       53  2013     1     1      557            600
##  9 2013-01-01 06:00:00      140  2013     1     1      557            600
## 10 2013-01-01 06:00:00      138  2013     1     1      558            600
## # ... with 336,766 more rows, and 12 more variables: dep_delay <dbl>,
## #   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, distance <dbl>,
## #   hour <dbl>, minute <dbl>
```

Add new variables with `mutate()`

- 새로운 변수를 만들어 기존 자료에 추가

```
flights_sml <- select(flights,
  year:day,
  ends_with("delay"),
  distance,
  air_time
```

```
)

mutate(flights_sml,
  gain = arr_delay - dep_delay,
  hours = air_time / 60,
  gain_per_hour = gain / hours
)
```

```
## # A tibble: 336,776 x 10
##     year month   day dep_delay arr_delay distance air_time  gain hours
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl> <dbl>
##  1  2013     1     1         2        11     1400      227     9 3.78
##  2  2013     1     1         4        20     1416      227    16 3.78
##  3  2013     1     1         2        33     1089      160    31 2.67
##  4  2013     1     1        -1       -18     1576      183   -17 3.05
##  5  2013     1     1        -6       -25      762      116   -19 1.93
##  6  2013     1     1        -4        12      719      150    16 2.5
##  7  2013     1     1        -5        19     1065      158    24 2.63
##  8  2013     1     1        -3       -14      229       53   -11 0.883
##  9  2013     1     1        -3        -8      944      140    -5 2.33
## 10  2013     1     1        -2         8      733      138    10 2.3
## # ... with 336,766 more rows, and 1 more variable: gain_per_hour <dbl>
```

- transmute(): 새로 만든 변수만을 가지는 자료를 만듦,

```
transmute(flights,
  gain = arr_delay - dep_delay,
  hours = air_time / 60,
  gain_per_hour = gain / hours
)
```

```
## # A tibble: 336,776 x 3
##     gain hours gain_per_hour
##    <dbl> <dbl>         <dbl>
##  1     9 3.78           2.38
##  2    16 3.78           4.23
##  3    31 2.67          11.6
##  4   -17 3.05          -5.57
##  5   -19 1.93          -9.83
##  6    16 2.5            6.4
##  7    24 2.63           9.11
##  8   -11 0.883        -12.5
##  9    -5 2.33          -2.14
## 10    10 2.3            4.35
## # ... with 336,766 more rows
```

Useful creation functions

- Arithmetic operators: +, -, *, /, ^.
- Modular arithmetic

  - %/% (integer division)
  - %% (remainder), `x == y * (x %/% y) + (x %% y)`.

```
transmute(flights,
  dep_time,
  hour = dep_time %/% 100,
  minute = dep_time %% 100
)
```

```
## # A tibble: 336,776 x 3
##    dep_time  hour minute
##       <int> <dbl>  <dbl>
##  1      517     5     17
##  2      533     5     33
##  3      542     5     42
##  4      544     5     44
##  5      554     5     54
##  6      554     5     54
##  7      555     5     55
##  8      557     5     57
##  9      557     5     57
## 10      558     5     58
## # ... with 336,766 more rows
```

- Logs: `log()`, `log2()`, `log10()`
- Offsets: `lead()`, `lag()`

```
(x <- 1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
lag(x)
```

```
##  [1] NA  1  2  3  4  5  6  7  8  9
```

```
lead(x)
```

```
##  [1]  2  3  4  5  6  7  8  9 10 NA
```

- Cumulative and rolling aggregates: `cumsum()`, `cumprod()`, `cummin()`, `cummax()`m `cummean()`

```
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
cumsum(x)
```

```
##  [1]  1  3  6 10 15 21 28 36 45 55
```

```
cummean(x)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

- Logical comparisons, <, <=, >, >=, !=
- Ranking: min_rank(),desc(x)

```
y <- c(1, 2, 2, NA, 3, 4)
min_rank(y)
```

```
## [1]  1  2  2 NA  4  5
```

```
min_rank(desc(y))
```

```
## [1]  5  3  3 NA  2  1
```

*row_number(), dense_rank(), percent_rank(), cume_dist(), ntile().

```
row_number(y)
```

```
## [1]  1  2  3 NA  4  5
```

```
dense_rank(y)
```

```
## [1]  1  2  2 NA  3  4
```

```
percent_rank(y)
```

```
## [1] 0.00 0.25 0.25   NA 0.75 1.00
```

```
cume_dist(y)
```

```
## [1] 0.2 0.6 0.6  NA 0.8 1.0
```

Grouped summaries with summarise()

- 요약 통계량 계산

```
summarise(flights, delay = mean(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   delay
##   <dbl>
## 1  12.6
```

- group_by()와 함께 사용할 경우 유용.

```
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE))
```

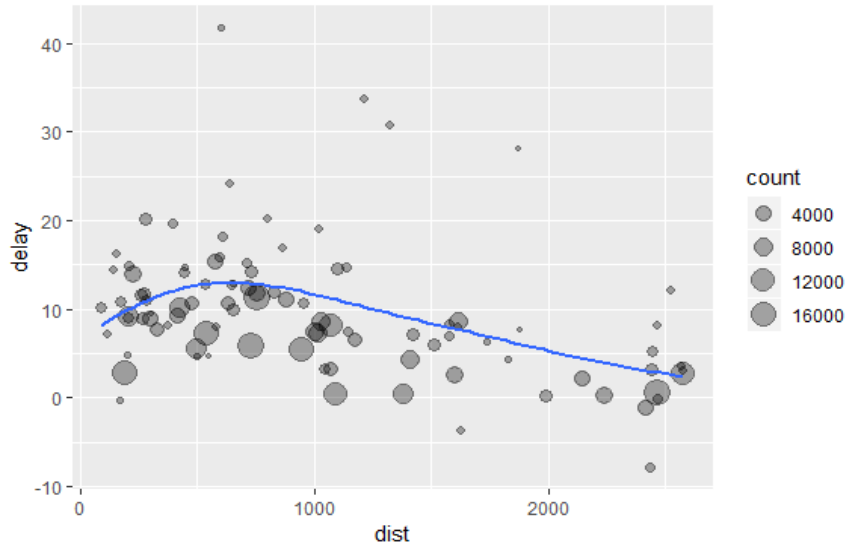
```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day delay
##    <int> <int> <int> <dbl>
##  1  2013     1     1 11.5
##  2  2013     1     2 13.9
##  3  2013     1     3 11.0
##  4  2013     1     4  8.95
```

```
##  5  2013     1     5  5.73
##  6  2013     1     6  7.15
##  7  2013     1     7  5.42
##  8  2013     1     8  2.55
##  9  2013     1     9  2.28
## 10  2013     1    10  2.84
## # ... with 355 more rows
```

Combining multiple operations with the pipe

- 각 지역에서 distance 와 average delay 의 관계 살펴보기.

```
by_dest <- group_by(flights, dest)
delay <- summarise(by_dest,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE)
)
delay <- filter(delay, count > 20, dest != "HNL")

ggplot(data = delay, mapping = aes(x = dist, y = delay)) +
  geom_point(aes(size = count), alpha = 1/3) +
  geom_smooth(se = FALSE)

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



- 위의 그림을 그리기 위한 자료 준비 단계

1. flights 자료를 destination 에 따라 그룹으로 나누기
2. 각 그룹별로 distance, average delay, 그리고 flight 수 구하기
3. 가장 먼 Honolulu 와 비행건수가 20 이하인 자료 제거

- 자료준비를 위해 pipe %>%를 이용하면 간편

```
delays <- flights %>%
  group_by(dest) %>%
  summarise(
    count = n(),
    dist = mean(distance, na.rm = TRUE),
    delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  filter(count > 20, dest != "HNL")
```

Missing values

- na.rm=TRUE 옵션은 통계량 계산 전에 NA 를 모두 제거한 후 계산하기 위한 것

```
flights %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day  mean
##    <int> <int> <int> <dbl>
##  1  2013     1     1    NA
##  2  2013     1     2    NA
##  3  2013     1     3    NA
##  4  2013     1     4    NA
##  5  2013     1     5    NA
##  6  2013     1     6    NA
##  7  2013     1     7    NA
##  8  2013     1     8    NA
##  9  2013     1     9    NA
## 10  2013     1    10    NA
## # ... with 355 more rows
```

```
flights %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day  mean
##    <int> <int> <int> <dbl>
##  1  2013     1     1 11.5
##  2  2013     1     2 13.9
##  3  2013     1     3 11.0
##  4  2013     1     4  8.95
##  5  2013     1     5  5.73
##  6  2013     1     6  7.15
##  7  2013     1     7  5.42
##  8  2013     1     8  2.55
```

```
##  9  2013     1     9  2.28
## 10  2013     1    10  2.84
## # ... with 355 more rows
```

- 아래와 같이 계산해도 같은 결과

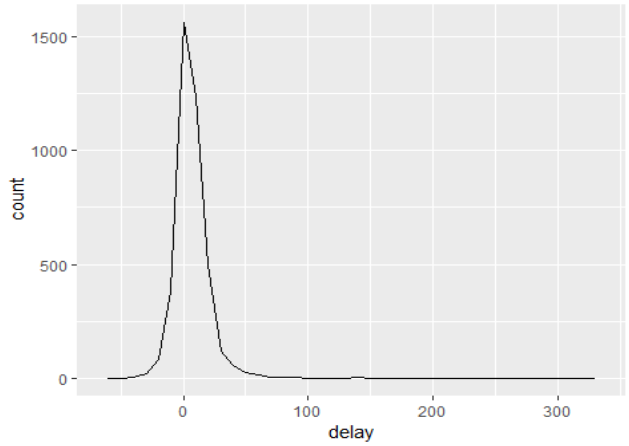
```
not_cancelled <- flights %>%
  filter(!is.na(dep_delay), !is.na(arr_delay))

not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day  mean
##    <int> <int> <int> <dbl>
##  1  2013     1     1 11.4
##  2  2013     1     2 13.7
##  3  2013     1     3 10.9
##  4  2013     1     4  8.97
##  5  2013     1     5  5.73
##  6  2013     1     6  7.15
##  7  2013     1     7  5.42
##  8  2013     1     8  2.56
##  9  2013     1     9  2.30
## 10  2013     1    10  2.84
## # ... with 355 more rows
```

Counts

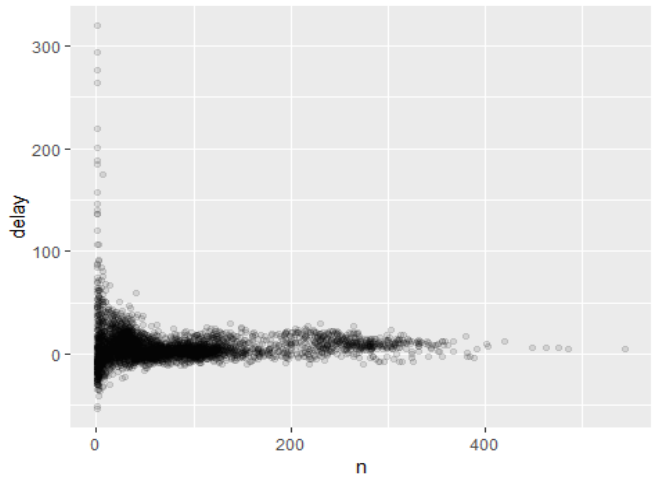
- `n()`: 자료의 갯수
- `sum(!is.na(x))`: missing 이 아닌 자료의 갯수
- 예: delay 분포 살펴보기

```
delays <- not_cancelled %>%
  group_by(tailnum) %>%
  summarise(
    delay = mean(arr_delay)
  )

ggplot(data = delays, mapping = aes(x = delay)) +
  geom_freqpoly(binwidth = 10)
```



- 300 분 이상의 평균 delay 인 비행기가 있음. 좀 더 자세히 살펴보기 위해 아래와 같이 delay 와 비행건수 간의 산점도를 그림

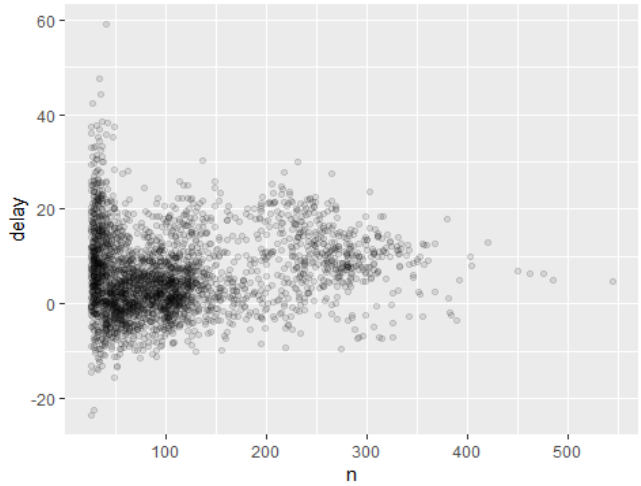
```
delays <- not_cancelled %>%
  group_by(tailnum) %>%
  summarise(
    delay = mean(arr_delay, na.rm = TRUE),
    n = n()
  )

ggplot(data = delays, mapping = aes(x = n, y = delay)) +
  geom_point(alpha = 1/10)
```



- 평균 300 이상이긴 하지만 비행건수가 매우 작음을 알 수 있음.
- 비행건수 25 이상만을 그림.

```
delays %>%
  filter(n > 25) %>%
```

```
  ggplot(mapping = aes(x = n, y = delay)) +
    geom_point(alpha = 1/10)
```



Useful summary functions

- mean(x), median(x)

```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(
    avg_delay1 = mean(arr_delay),
    avg_delay2 = mean(arr_delay[arr_delay > 0])
  )
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [?]
##     year month   day avg_delay1 avg_delay2
##    <int> <int> <int>      <dbl>      <dbl>
##  1  2013     1     1     12.7         32.5
##  2  2013     1     2     12.7         32.0
##  3  2013     1     3      5.73        27.7
##  4  2013     1     4     -1.93        28.3
##  5  2013     1     5     -1.53        22.6
##  6  2013     1     6      4.24        24.4
##  7  2013     1     7     -4.95        27.8
##  8  2013     1     8     -3.23        20.8
##  9  2013     1     9     -0.264       25.6
## 10  2013     1    10     -5.90        27.3
## # ... with 355 more rows
```

- sd(x), IQR(x), mad(x)

```
not_cancelled %>%
  group_by(dest) %>%
  summarise(distance_sd = sd(distance)) %>%
  arrange(desc(distance_sd))
```

```
## # A tibble: 104 x 2
##    dest  distance_sd
##    <chr>       <dbl>
##  1 EGE         10.5
##  2 SAN         10.4
##  3 SFO         10.2
##  4 HNL         10.0
##  5 SEA          9.98
##  6 LAS          9.91
##  7 PDX          9.87
##  8 PHX          9.86
##  9 LAX          9.66
## 10 IND          9.46
## # ... with 94 more rows
```

- `min(x), quantile(x, 0.25), max(x)`
- 일별로 가장 먼저, 그리고 가장 마지막에 비행기가 출발하는 시간은?

```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(
    first = min(dep_time),
    last = max(dep_time)
  )
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [?]
##     year month   day first  last
##    <int> <int> <int> <dbl> <dbl>
##  1  2013     1     1   517  2356
##  2  2013     1     2    42  2354
##  3  2013     1     3    32  2349
##  4  2013     1     4    25  2358
##  5  2013     1     5    14  2357
##  6  2013     1     6    16  2355
##  7  2013     1     7    49  2359
##  8  2013     1     8   454  2351
##  9  2013     1     9     2  2252
## 10  2013     1    10     3  2320
## # ... with 355 more rows
```

- `first(x), nth(x, 2), last(x)`

```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(
    first_dep = first(dep_time),
    last_dep = last(dep_time)
  )
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [?]
##     year month   day first_dep last_dep
##    <int> <int> <int>     <int>    <int>
##  1  2013     1     1       517     2356
##  2  2013     1     2        42     2354
##  3  2013     1     3        32     2349
##  4  2013     1     4        25     2358
##  5  2013     1     5        14     2357
##  6  2013     1     6        16     2355
##  7  2013     1     7        49     2359
##  8  2013     1     8       454     2351
##  9  2013     1     9         2     2252
## 10  2013     1    10         3     2320
## # ... with 355 more rows
```

- 위와 비교

```
not_cancelled %>%
  group_by(year, month, day) %>%
  mutate(r = min_rank(desc(dep_time))) %>%
  filter(r %in% range(r))
```

```
## # A tibble: 770 x 20
## # Groups:   year, month, day [365]
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1     2356           2359        -3      425
##  3  2013     1     2       42           2359        43      518
##  4  2013     1     2     2354           2359        -5      413
##  5  2013     1     3       32           2359        33      504
##  6  2013     1     3     2349           2359       -10      434
##  7  2013     1     4       25           2359        26      505
##  8  2013     1     4     2358           2359        -1      429
##  9  2013     1     4     2358           2359        -1      436
## 10  2013     1     5       14           2359        15      503
## # ... with 760 more rows, and 13 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, r <int>
```

- `n()`, `sum(!is.na(x))`, `n_distinct(x)`.
- 가장 다양한 항공사에서 노선을 제공하고 있는 목적지는?

```
not_cancelled %>%
  group_by(dest) %>%
  summarise(carriers = n_distinct(carrier)) %>%
  arrange(desc(carriers))
```

```
## # A tibble: 104 x 2
##    dest  carriers
##    <chr>    <int>
##  1 ATL          7
##  2 BOS          7
##  3 CLT          7
##  4 ORD          7
##  5 TPA          7
##  6 AUS          6
##  7 DCA          6
##  8 DTW          6
##  9 IAD          6
## 10 MSP          6
## # ... with 94 more rows
```

- 가장 많은 비행기가 제공되는 목적지는?

```
not_cancelled %>%
  count(dest)
```

```
## # A tibble: 104 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
##  3 ALB     418
##  4 ANC       8
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # ... with 94 more rows
```

- weighted count 도 가능

```
not_cancelled %>%
  count(tailnum, wt = distance)
```

```
## # A tibble: 4,037 x 2
##    tailnum      n
##    <chr>    <dbl>
##  1 D942DN    3418
##  2 N0EGMQ  239143
##  3 N10156  109664
##  4 N102UW   25722
##  5 N103US   24619
##  6 N104UW   24616
##  7 N10575  139903
```

```
##  8 N105UW   23618
##  9 N107US   21677
## 10 N108UW   32070
## # ... with 4,027 more rows
```

- `sum(x > 10)`: 횟수
- `mean(y == 0)`: 비율
- 5 시 이전에 출발하는 비행기는 몇대?

```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(n_early = sum(dep_time < 500))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day n_early
##    <int> <int> <int>   <int>
##  1  2013     1     1       0
##  2  2013     1     2       3
##  3  2013     1     3       4
##  4  2013     1     4       3
##  5  2013     1     5       3
##  6  2013     1     6       2
##  7  2013     1     7       2
##  8  2013     1     8       1
##  9  2013     1     9       3
## 10  2013     1    10       3
## # ... with 355 more rows
```

- 1 시간 이상 지연된 비행기 비율은?

```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(hour_perc = mean(arr_delay > 60))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day hour_perc
##    <int> <int> <int>     <dbl>
##  1  2013     1     1    0.0722
##  2  2013     1     2    0.0851
##  3  2013     1     3    0.0567
##  4  2013     1     4    0.0396
##  5  2013     1     5    0.0349
##  6  2013     1     6    0.0470
##  7  2013     1     7    0.0333
##  8  2013     1     8    0.0213
##  9  2013     1     9    0.0202
```

```
## 10  2013     1    10    0.0183
## # ... with 355 more rows
```

- 여러 변수를 이용한 그룹 지정

```
daily <- group_by(flights, year, month, day)
(per_day   <- summarise(daily, flights = n()))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day flights
##    <int> <int> <int>   <int>
##  1  2013     1     1     842
##  2  2013     1     2     943
##  3  2013     1     3     914
##  4  2013     1     4     915
##  5  2013     1     5     720
##  6  2013     1     6     832
##  7  2013     1     7     933
##  8  2013     1     8     899
##  9  2013     1     9     902
## 10  2013     1    10     932
## # ... with 355 more rows
```

```
(per_month <- summarise(per_day, flights = sum(flights)))
```

```
## # A tibble: 12 x 3
## # Groups:   year [?]
##     year month flights
##    <int> <int>   <int>
##  1  2013     1   27004
##  2  2013     2   24951
##  3  2013     3   28834
##  4  2013     4   28330
##  5  2013     5   28796
##  6  2013     6   28243
##  7  2013     7   29425
##  8  2013     8   29327
##  9  2013     9   27574
## 10  2013    10   28889
## 11  2013    11   27268
## 12  2013    12   28135
```

```
(per_year  <- summarise(per_month, flights = sum(flights)))
```

```
## # A tibble: 1 x 2
##    year flights
##   <int>   <int>
## 1  2013  336776
```

- ungroup()을 이용하여 그룹 해제

```
daily %>%
  ungroup() %>%             # no longer grouped by date
  summarise(flights = n())  # all flights
```

```
## # A tibble: 1 x 1
##   flights
##     <int>
## 1  336776
```

각 그룹에서 변수 생성

- 각 그룹에서 최하위 찾기 :

```
flights_sml %>%
  group_by(year, month, day) %>%
  filter(rank(desc(arr_delay)) < 10)
```

```
## # A tibble: 3,306 x 7
## # Groups:   year, month, day [365]
##     year month   day dep_delay arr_delay distance air_time
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl>
##  1  2013     1     1       853       851      184       41
##  2  2013     1     1       290       338     1134      213
##  3  2013     1     1       260       263      266       46
##  4  2013     1     1       157       174      213       60
##  5  2013     1     1       216       222      708      121
##  6  2013     1     1       255       250      589      115
##  7  2013     1     1       285       246     1085      146
##  8  2013     1     1       192       191      199       44
##  9  2013     1     1       379       456     1092      222
## 10  2013     1     2       224       207      550       94
## # ... with 3,296 more rows
```

- 기준 조건 이상의 그룹 찾기 :

```
popular_dests <- flights %>%
  group_by(dest) %>%
  filter(n() > 365)
popular_dests
```

```
## # A tibble: 332,577 x 19
## # Groups:   dest [77]
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
```

```
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 332,567 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

- 각 그룹마다 새로운 변수 생성하기

```
popular_dests %>%
  filter(arr_delay > 0) %>%
  mutate(prop_delay = arr_delay / sum(arr_delay)) %>%
  select(year:day, dest, arr_delay, prop_delay)
```

```
## # A tibble: 131,106 x 6
## # Groups:   dest [77]
##     year month   day dest  arr_delay prop_delay
##    <int> <int> <int> <chr>     <dbl>      <dbl>
##  1  2013     1     1 IAH          11  0.000111
##  2  2013     1     1 IAH          20  0.000201
##  3  2013     1     1 MIA          33  0.000235
##  4  2013     1     1 ORD          12  0.0000424
##  5  2013     1     1 FLL          19  0.0000938
##  6  2013     1     1 ORD           8  0.0000283
##  7  2013     1     1 LAX           7  0.0000344
##  8  2013     1     1 DFW          31  0.000282
##  9  2013     1     1 ATL          12  0.0000400
## 10  2013     1     1 DTW          16  0.000116
## # ... with 131,096 more rows
```

5. Exploratory Data Analysis

- EDA 의 단계

1. 자료에 대하여 궁금한 질문 사항들 정리
2. 자료 시각화, 변형, 그리고 모델링등의 탐색을 통해 질문들에 대한 답을 찾기
3. 탐색 결과를 이용하여 질문 사항들을 구체화하거나 새로운 질문 사항들 만들기

- EDA 과정은 자료분석의 가장 중요한 단계
- 자료정리, 자료 시각화, 자료 변형, 자료 모형화 등이 포함

```
library(tidyverse)
```

Questions

"There are no routine statistical questions, only questionable statistical routines." — Sir David Cox

"Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise." — John Tukey

- EDA 의 목표는 자료에 대한 이해를 위한 것.
- EDA 를 위한 일반적인 질문
  - What type of variation occurs within my variables?
  - What type of covariation occurs between my variables?
- variable: 측정할 수 있는 것, 변수.
- value: 측정한 값
- observation: 비슷한 환경에서 하나의 개체로 부터 측정된 값들의 집합.
- Tabular data: 변수와 관측으로 이루어진 값들의 집합.

- tidy : column 은 변수를, row 는 관측을 의미하며 column 과 row 로 구성되는 cell 에는 값을 가지고 있는 형태로 정리된 tabular data

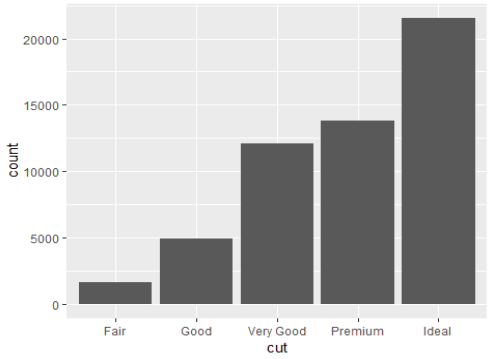
Variation

- 변수가 관측마다 변하는 정도

분포 시각화

- 범주형 변수: bar chart 이용

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

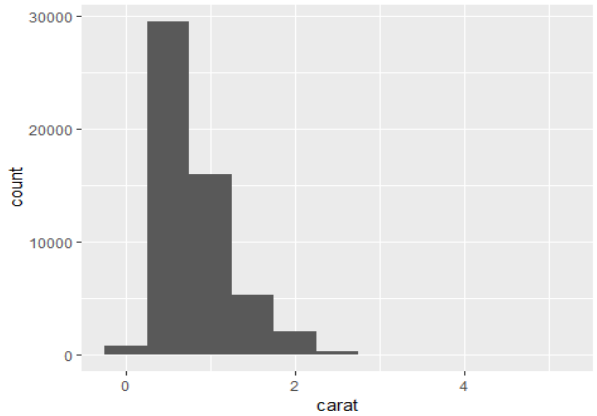


- dplyr::count()를 이용하여 bar chart 의 높이 계산 가능

```
diamonds %>%
  count(cut)

## # A tibble: 5 x 2
##   cut           n
##   <ord>     <int>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

- 연속형 변수: histogram 이용

```
ggplot(data = diamonds) +
  geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```



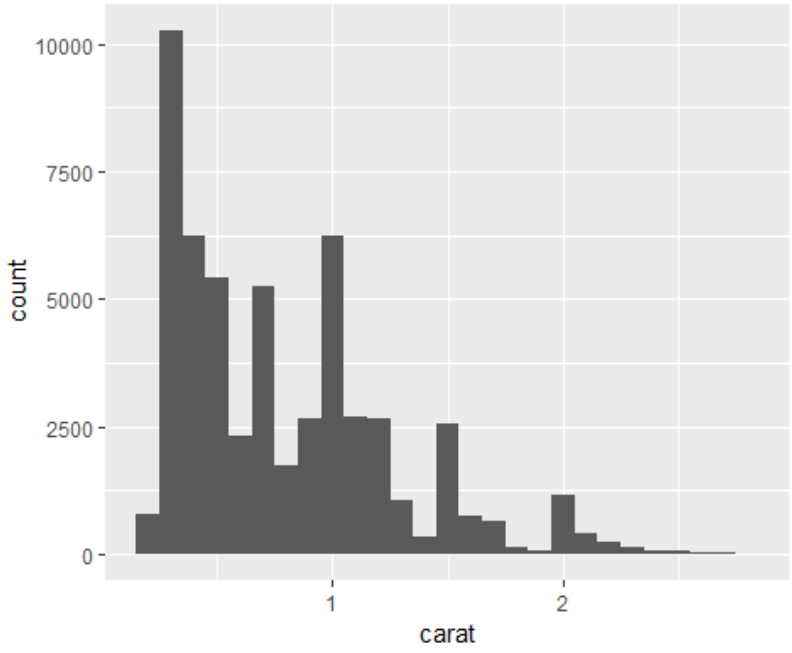
- `ggplot2::cut_width()`로 범주화 한후 `dplyr::count()`를 이용하여 histogram 의 높이 계산 가능

```
diamonds %>%
  count(cut_width(carat, 0.5))
```

```
## # A tibble: 11 x 2
##    `cut_width(carat, 0.5)`     n
##    <fct>                   <int>
##  1 [-0.25,0.25]              785
##  2 (0.25,0.75]             29498
##  3 (0.75,1.25]             15977
##  4 (1.25,1.75]              5313
##  5 (1.75,2.25]              2002
##  6 (2.25,2.75]               322
##  7 (2.75,3.25]                32
##  8 (3.25,3.75]                 5
##  9 (3.75,4.25]                 4
## 10 (4.25,4.75]                 1
## 11 (4.75,5.25]                 1
```

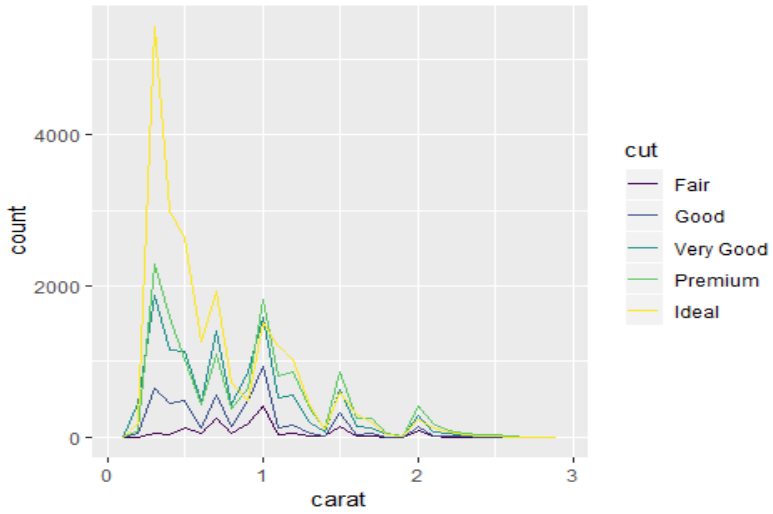
- zooming: `filter()` 와 `binwidth` 이용.

```
smaller <- diamonds %>%
  filter(carat < 3)

ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.1)
```



- 연속형 변수: `freqpoly()` 이용

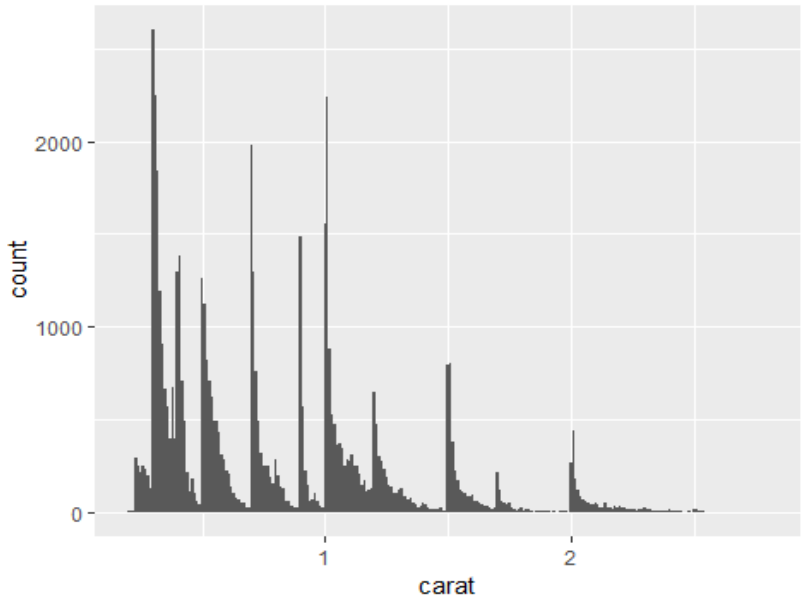
```
ggplot(data = smaller, mapping = aes(x = carat, colour = cut)) +
  geom_freqpoly(binwidth = 0.1)
```



Typical values

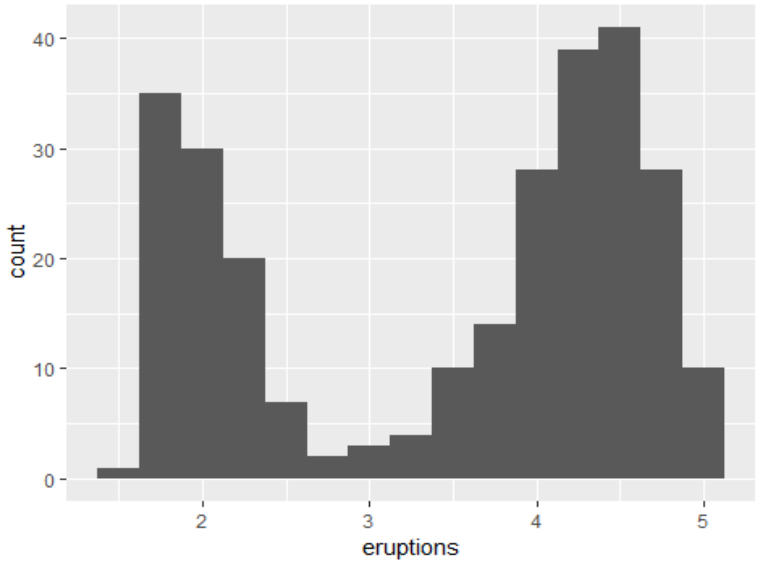
- 일반적 질문
  - Which values are the most common? Why?
  - Which values are rare? Why? Does that match your expectations?
  - Can you see any unusual patterns? What might explain them?

```
ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.01)
```



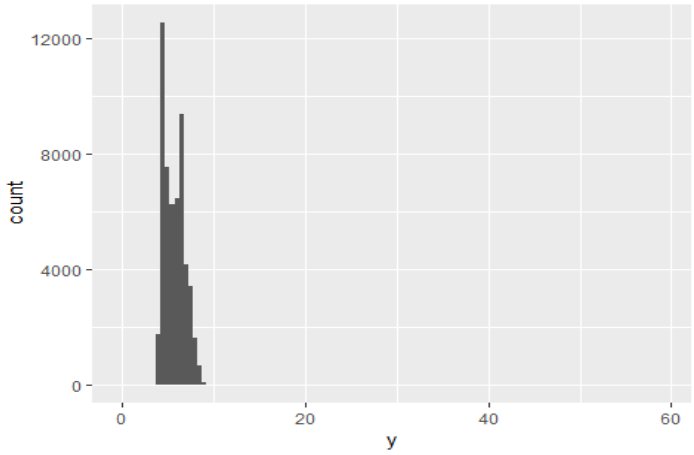
- 위의 histogram 에 대한 질문들
  - Why are there more diamonds at whole carats and common fractions of carats?
  - Why are there more diamonds slightly to the right of each peak than there are slightly to the left of each peak?
  - Why are there no diamonds bigger than 3 carats?
- Cluster 는 subgroup 에 대한 가능성을 의미함. 이에 대한 질문들
  - How are the observations within each cluster similar to each other?
  - How are the observations in separate clusters different from each other? -How can you explain or describe the clusters?
  - Why might the appearance of clusters be misleading?

```
ggplot(data = faithful, mapping = aes(x = eruptions)) +
  geom_histogram(binwidth = 0.25)
```



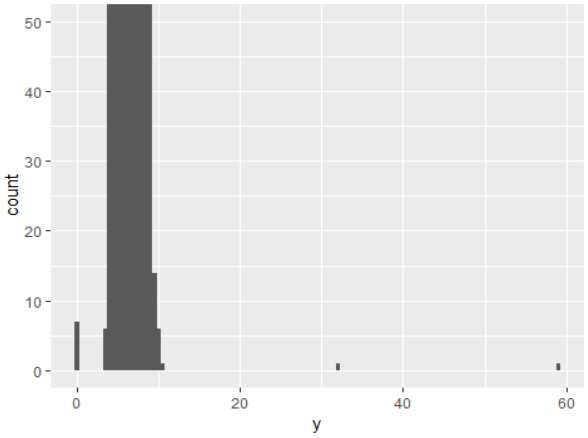
이상점

```
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5)
```



- y 의 범위가 60 까지 되어 있으나 10 에서 60 사이에는 자료가 거의 보이지 않음.
  *coord_cartesian()의 ylim 옵션을 이용하여 zooming 하여 살펴보기

```
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5) +
  coord_cartesian(ylim = c(0, 50))
```



- 0, 30 그리고 60 근처에서 관측 발견.

```
unusual <- diamonds %>%
  filter(y < 3 | y > 20) %>%
  arrange(y)
unusual
```

```
## # A tibble: 9 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
```

```
## 1  1    Very Good H     VS2      63.3    53  5139  0      0    0
## 2  1.14 Fair      G     VS1      57.5    67  6381  0      0    0
## 3  1.56 Ideal     G     VS2      62.2    54 12800  0      0    0
## 4  1.2  Premium   D     VVS1     62.1    59 15686  0      0    0
## 5  2.25 Premium   H     SI2      62.8    59 18034  0      0    0
## 6  0.71 Good      F     SI2      64.1    60  2130  0      0    0
## 7  0.71 Good      F     SI2      64.1    60  2130  0      0    0
## 8  0.51 Ideal     E     VS1      61.8    55  2075  5.15  31.8  5.12
## 9  2    Premium   H     SI2      58.9    57 12210  8.09  58.9  8.06
```

Missing values

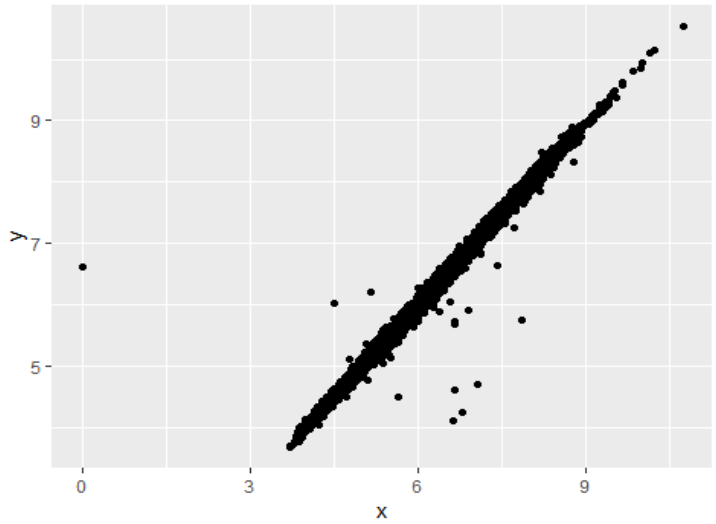
- 이상점 처리 방법

1. 이상점이 있는 관측을 모두 삭제 - 바람직하지 않음

```
diamonds2 <- diamonds %>%
  filter(between(y, 3, 20))
```

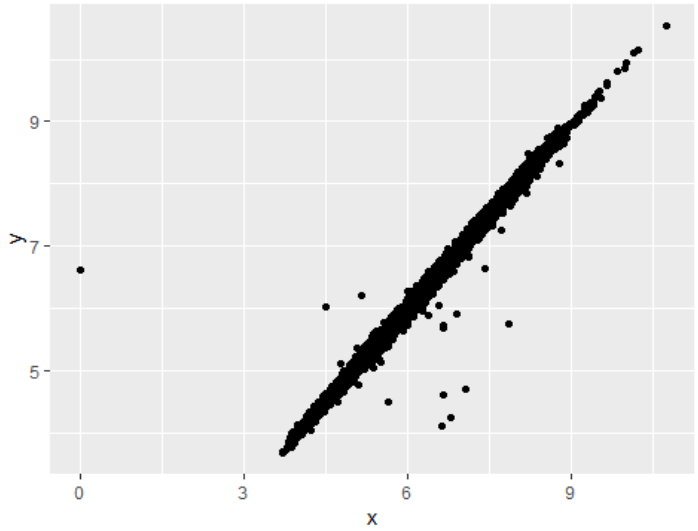
2. 이상한 값만 NA 로 처리

```
diamonds2 <- diamonds %>%
  mutate(y = ifelse(y < 3 | y > 20, NA, y))

ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
  geom_point()

## Warning: Removed 9 rows containing missing values (geom_point).
```



- na.rm = TRUE 옵션 이용:

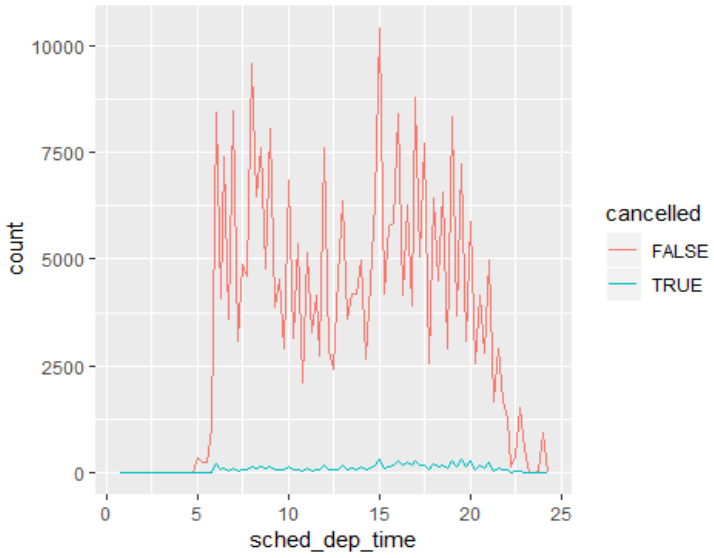
```
ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
  geom_point(na.rm = TRUE)
```



- cancel 된 비행기와 cancel 되지 않은 비행기의 scheduled departure time 별 비교

```
nycflights13::flights %>%
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/% 100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + sched_min / 60
  ) %>%
  ggplot(mapping = aes(sched_dep_time)) +
    geom_freqpoly(mapping = aes(colour = cancelled), binwidth = 1/4)
```

Mutate를 이용한 변수 생성



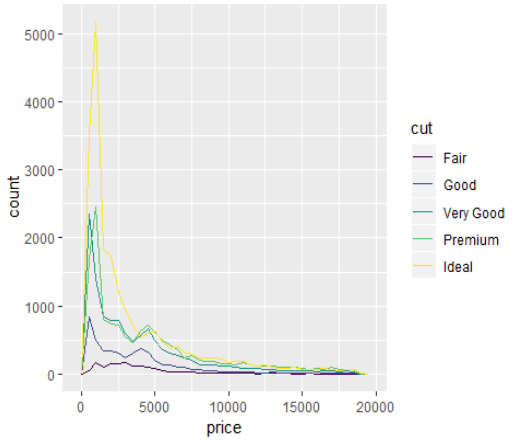
Covariation

- 변수들 간의 관계를 나타내는 것으로 둘 이상의 변수들이 함께 변하는 경향을 파악하는 것이 필요

범주형과 연속형 변수 간의 관계

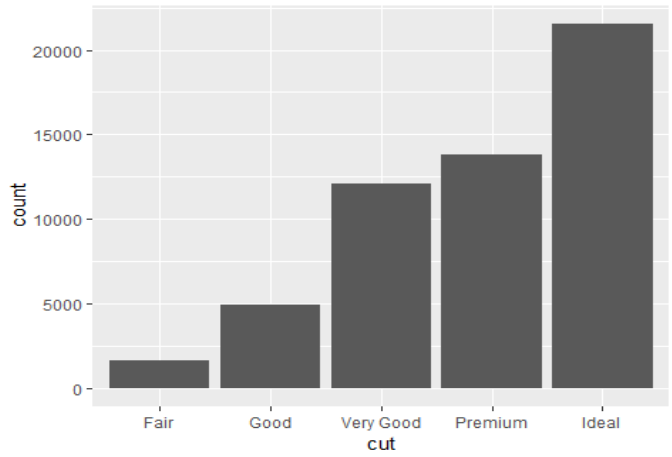
- 범주의 수준별로 나누어 연속형 변수의 분포 살펴보기
- geom_freqpoly 를 이용하는 경우

```
ggplot(data = diamonds, mapping = aes(x = price)) +
  geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```



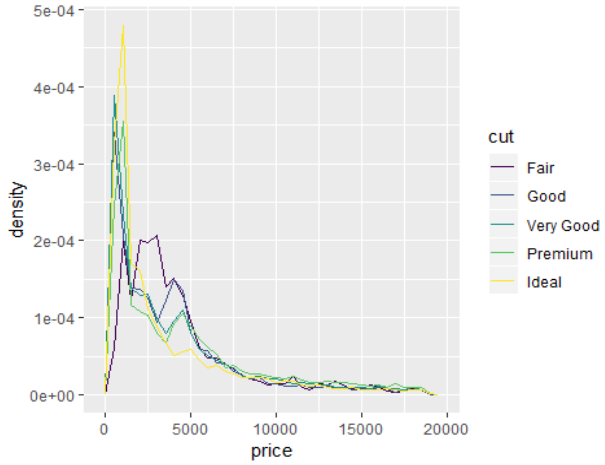
- 위의 그림에서 범주의 수준간 차이를 알아보기 힘든 이유는 수준간 관측수가 다르기 때문.

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



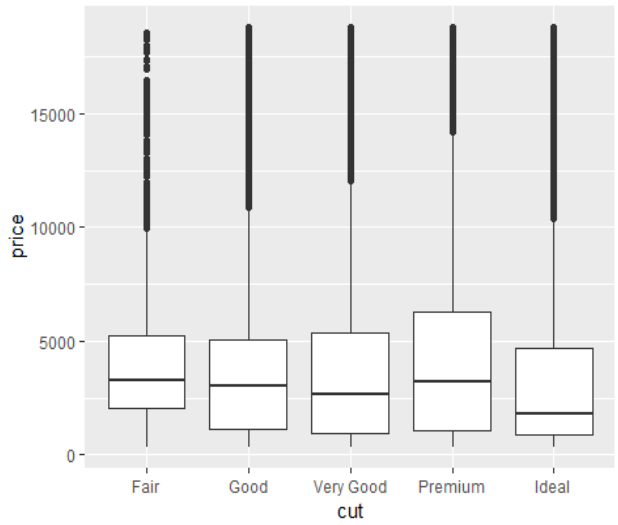
- geom_density 이용하여 비교하기

```
ggplot(data = diamonds, mapping = aes(x = price, y = ..density..)) +
  geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```



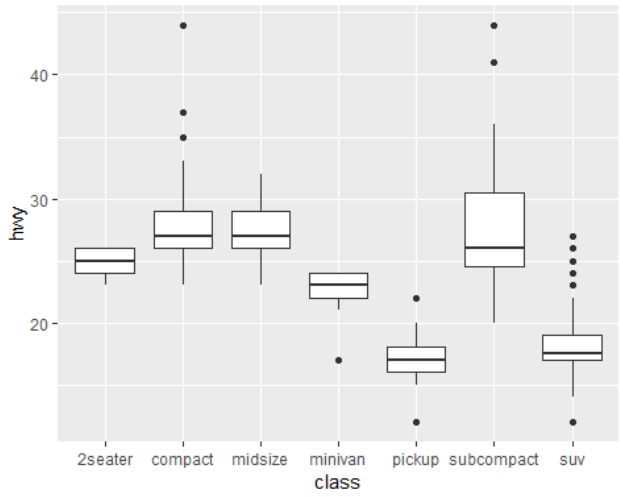
- geom_boxplot()으로 비교하기

```
ggplot(data = diamonds, mapping = aes(x = cut, y = price)) +
  geom_boxplot()
```



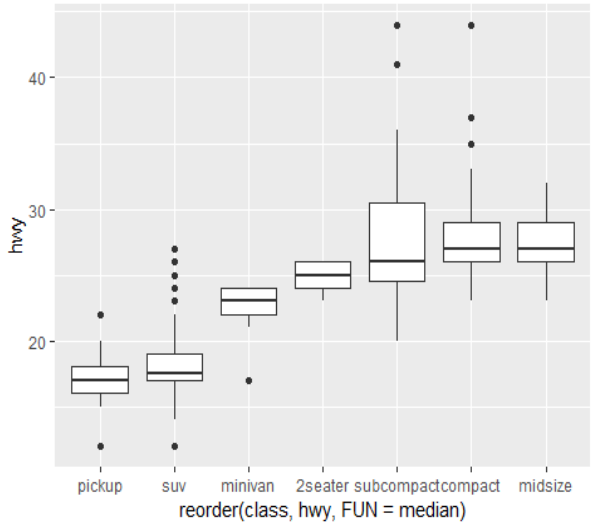
- `reorder()` 를 이용하여 범주 순서를 y 값의 크기에 따라 바꾸어 그리기

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



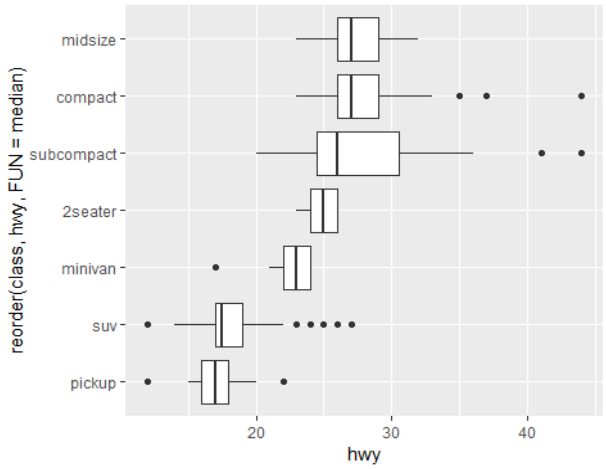
boxplot을 그리는 경우 median기준 sorting이 better(mean 보다)

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



- coord_flip()을 이용하여 축 위치 바꾸기

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



두 범주형 변수의 관계

- geom_count()를 이용하여 관측수를 점 크기로 표시

```
ggplot(data = diamonds) +
  geom_count(mapping = aes(x = cut, y = color))
```

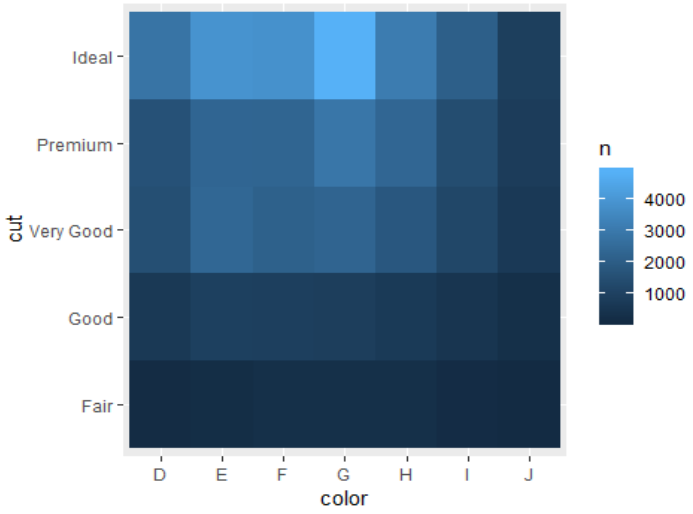
- dplyr::count 를 이용하여 계산하기

```
diamonds %>%
  count(color, cut)

## # A tibble: 35 x 3
##    color cut           n
```

```
##    <ord> <ord>     <int>
##  1 D     Fair        163
##  2 D     Good        662
##  3 D     Very Good  1513
##  4 D     Premium    1603
##  5 D     Ideal      2834
##  6 E     Fair        224
##  7 E     Good        933
##  8 E     Very Good  2400
##  9 E     Premium    2337
## 10 E     Ideal      3903
## # ... with 25 more rows
```

- geom_tile()과 fill aesthetic 을 이용하여 표현하기

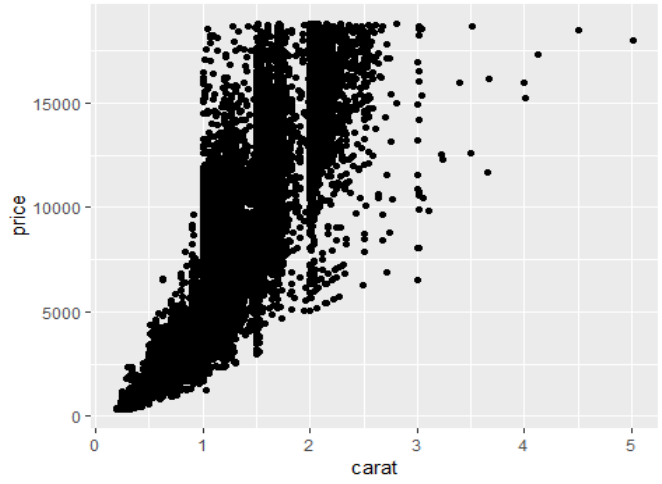
```
diamonds %>%
  count(color, cut) %>%
  ggplot(mapping = aes(x = color, y = cut)) +
    geom_tile(mapping = aes(fill = n))
```



두 연속변수의 관계

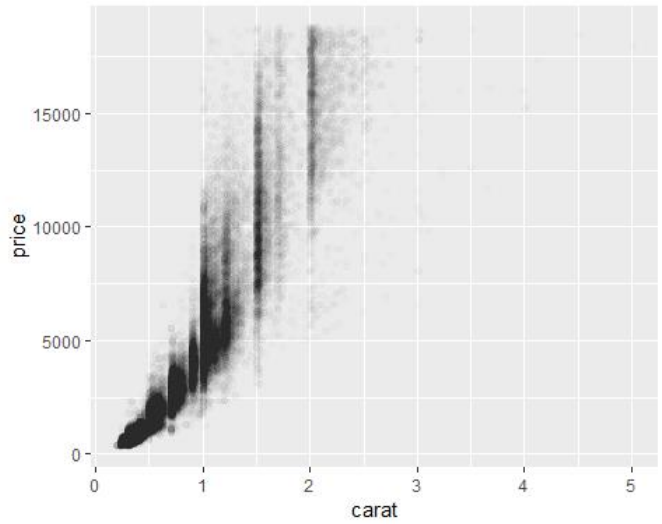
- geom_point()를 이용한 산점도

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



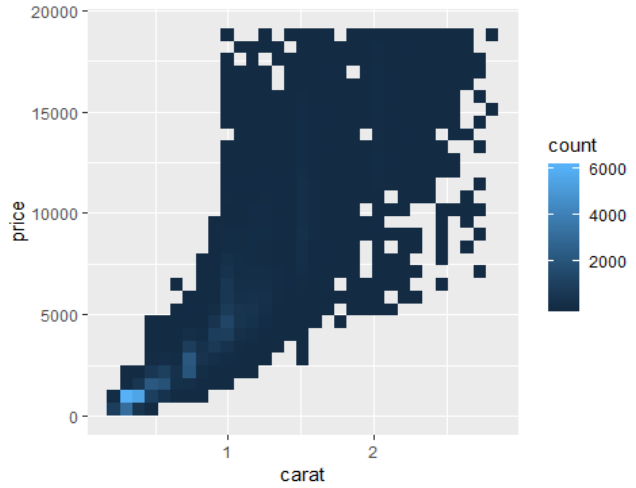
- alpha 옵션을 이용하여 투명도 조정

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price), alpha = 1 / 100)
```

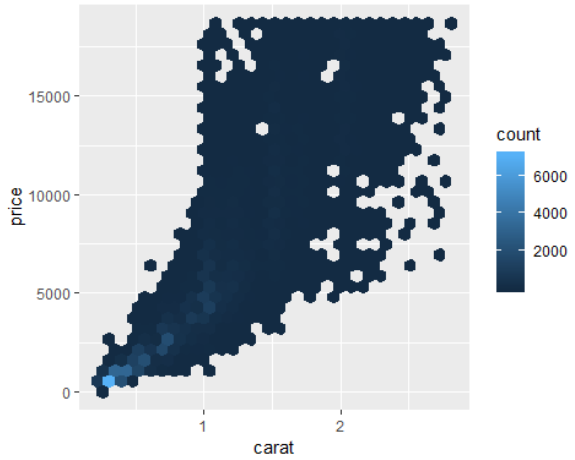


- bin 을 이용 : geom_bin2d(), geom_hex()

```
ggplot(data = smaller) +
  geom_bin2d(mapping = aes(x = carat, y = price))
```

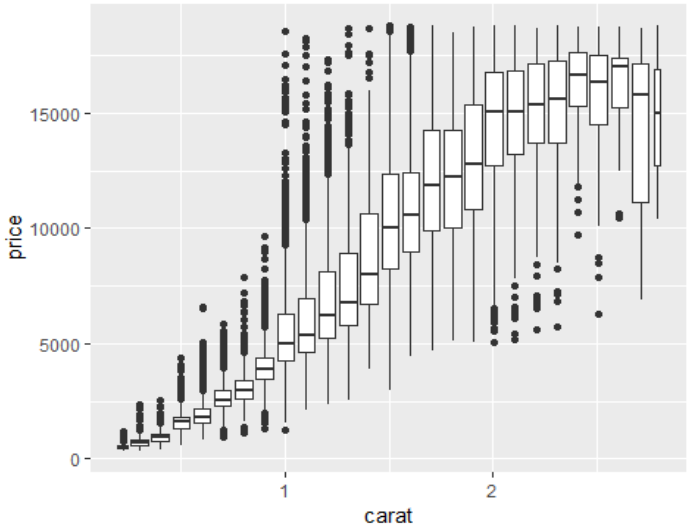



```
# install.packages("hexbin")
ggplot(data = smaller) +
  geom_hex(mapping = aes(x = carat, y = price))

## Warning: package 'hexbin' was built under R version 3.5.2
```



- 한 연속변수를 cut_width 로 범주화하여 geom_boxplot 이용

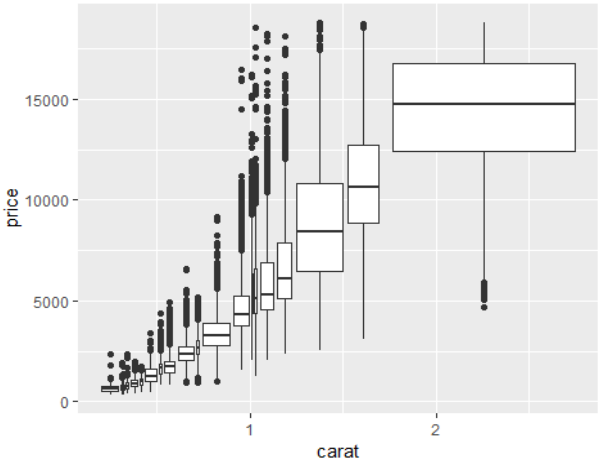
```
ggplot(data = smaller, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_width(carat, 0.1)))
```



- cut_number()를 이용하여 범주화

```
ggplot(data = smaller, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_number(carat, 20)))
```

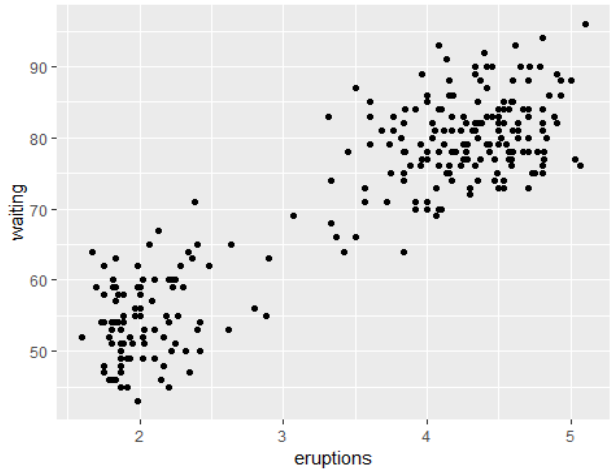
obs의 갯수를 지정



Patterns and models

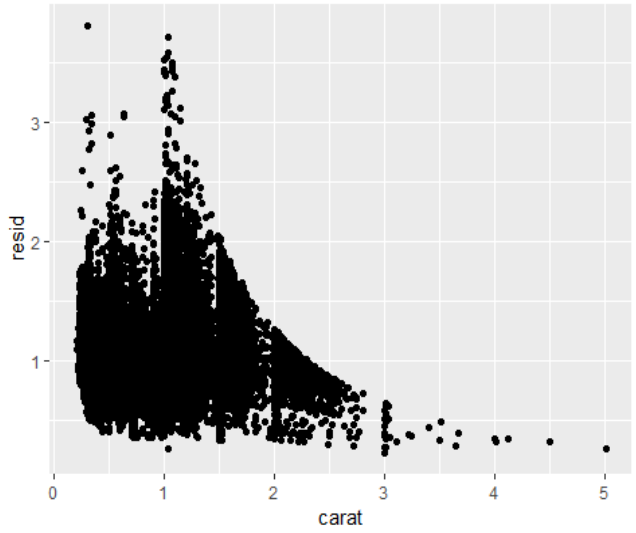
- 자료에서 특정 패턴을 발견한 경우 해야하는 질문들
  - Could this pattern be due to coincidence (i.e. random chance)?
  - How can you describe the relationship implied by the pattern?
  - How strong is the relationship implied by the pattern?
  - What other variables might affect the relationship?
  - Does the relationship change if you look at individual subgroups of the data?
- faithful 자료에서 2 개의 cluster 발견

```
ggplot(data = faithful) +
  geom_point(mapping = aes(x = eruptions, y = waiting))
```



- model 을 이용한 패턴 분석
  - diamonds 자료에서 cut 과 price 의 관계의 파악이 어려움
  - cut 과 carat, carat 과 price 간의 관계 때문
  - price 와 carat 의 관계를 제외하고 cut 과 price 관계를 파악

```r
library(modelr)

mod <- lm(log(price) ~ log(carat), data = diamonds)

diamonds2 <- diamonds %>%
  add_residuals(mod) %>%
  mutate(resid = exp(resid))

ggplot(data = diamonds2) +
  geom_point(mapping = aes(x = carat, y = resid))
```



- cut 이 좋아질수록 price(carat 보정후)가 높아짐

```r
ggplot(data = diamonds2) +
  geom_boxplot(mapping = aes(x = cut, y = resid))
```