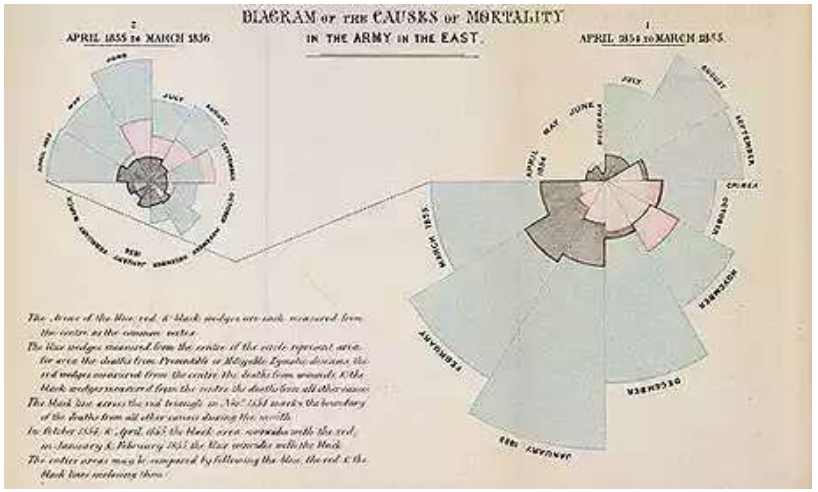
Charts & Graphs Basic

Graph

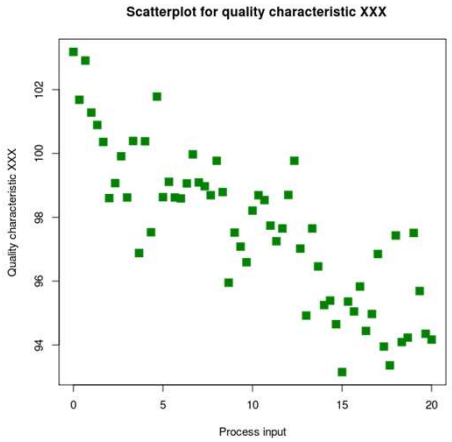
- 그래프 : 데이터를 보기 쉽게 그림으로 표현한 것
  - 데이터 원 자료나 통계표는 숫자와 문자로 구성되어 있어 의미를 한눈에 파악하기 어려움
- 그래프의 장점
  - 데이터 전체의 특성을 쉽게 이해할 수 있음
  - 데이터의 추세와 경향성을 파악할 수 있음
  - 그래프를 만드는 과정에서 새로운 패턴을 발견하기도 함
- 각 그래프의 종류, 특성과 용도를 잘 이용하여 사용하는 것이 중요
  - 어떤 그래프를 보여줄 것인가?
  - 그래프를 보여주기 위해 데이터를 어떻게 가공할 것인가?

- 참고) Graphics Parameter in R
  - https://www.statmethods.net/advgraphs/parameters.html

플로렌스 나이팅게일이 고안한 차트



Graph
: Scatter Plot



- 산점도(Scatter Plot)
  - x축과 y축에 데이터를 점으로 표현한 그래프
  - 연속 값으로 된 두 변수의 관계를 표현
- 기본 문법

```
plot(x, y, main, xlab, ylab, xlim, ylim, axes)
```

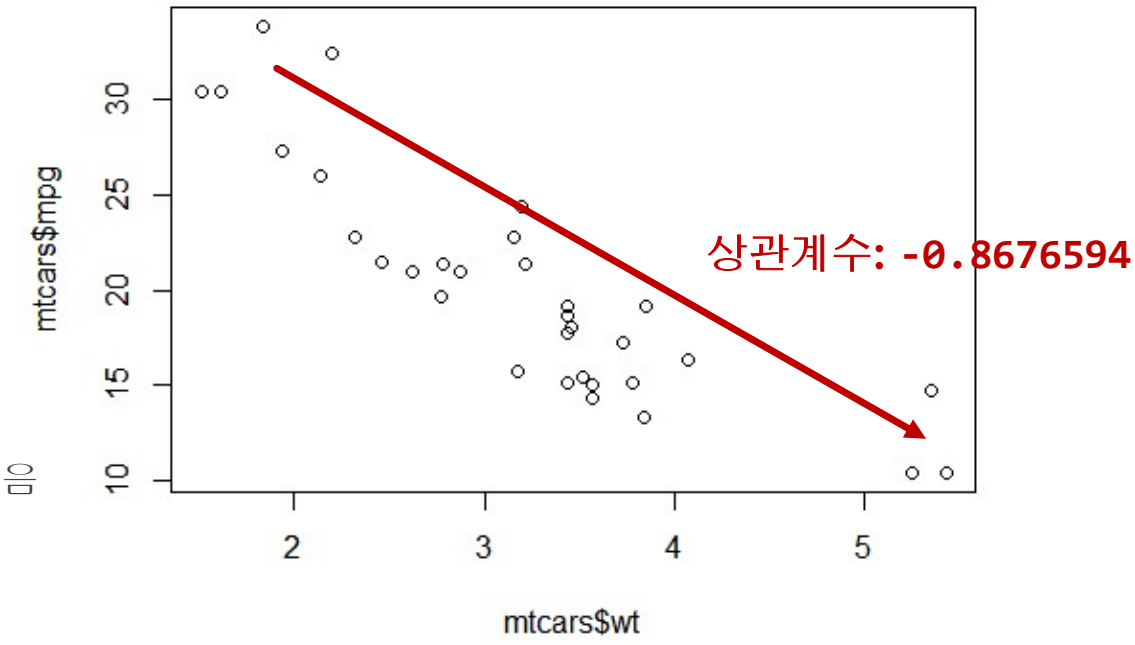
매개변수	설명
x	가로 축에 사용할 변수
y	세로 축에 사용할 변수
main	그래프의 제목
xlab	가로축에 표시할 라벨(텍스트)
ylab	세로축에 표시할 라벨(텍스트)
xlim	가로축 범위
ylim	세로축 범위

Graph
: Scatter Plot

- mtcars 데이터프레임을 이용, 중량(wt)과 연비(mpg)의 관계에 대해 알아봅시다
  - 기본 산점도 그래프의 생성

```
> head(mtcars)
> str(mtcars)
> plot(mtcars$wt, mtcars$mpg)
> cor(mtcars$wt, mtcars$mpg)
[1] -0.8676594
```

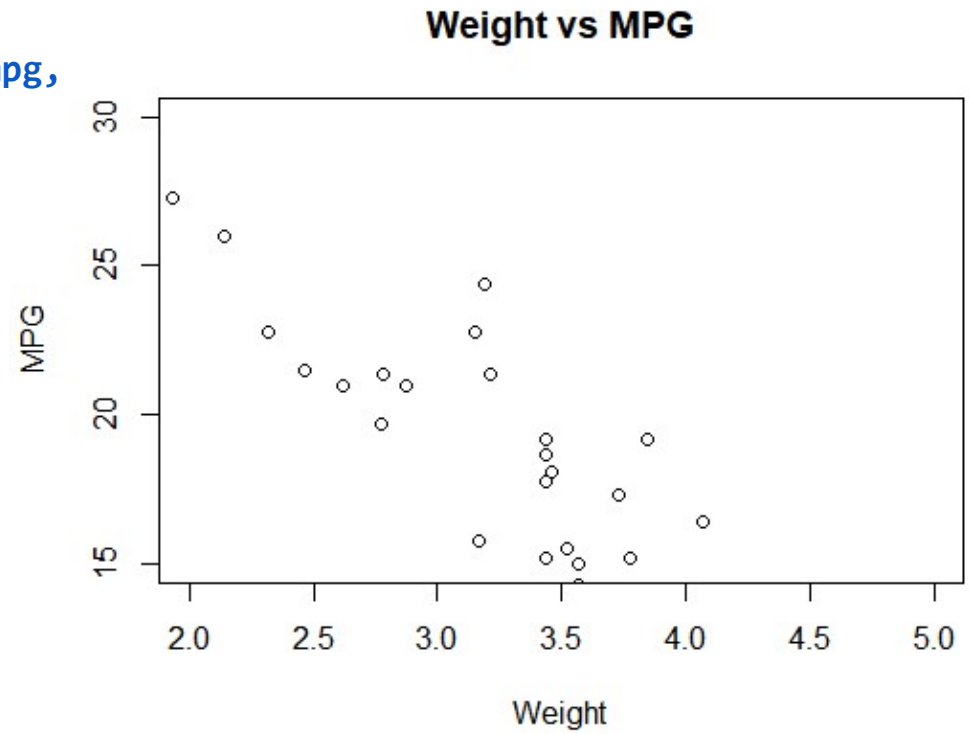
  - 산점도 그래프를 이용,
    전체적으로 중량이 무거울 수록
    연비가 낮다는 경향성을 확인할 수 있음



Graph
: Scatter Plot

- 타이틀과 라벨, 축 범위 제한의 추가

```
> plot(x = mtcars$wt, y = mtcars$mpg,
+      xlab = "Weight",
+      ylab = "MPG",
+      xlim = c(2, 5),
+      ylim = c(15, 30),
+      main = "Weight vs MPG")
```



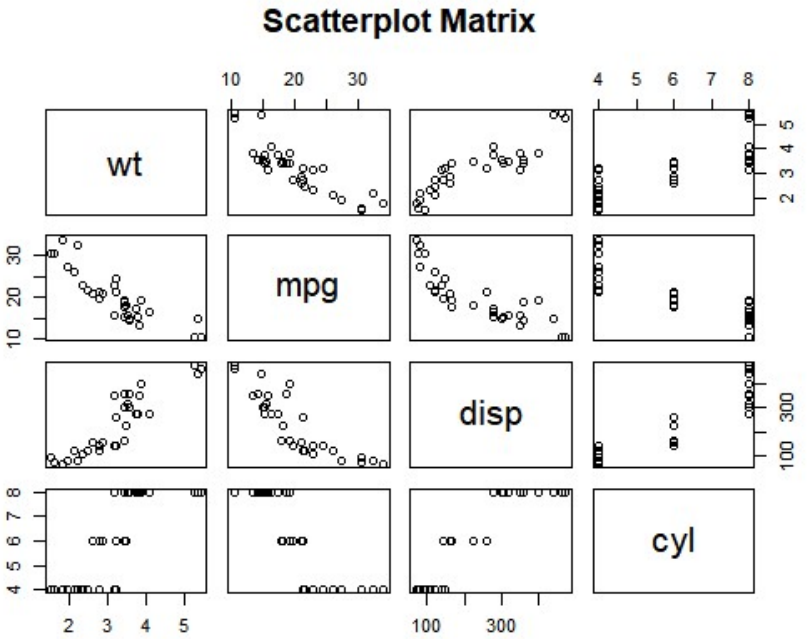
Graph
: Scatter Plot

- 산점도 매트릭스 (Scatterplot Matcix)
  - 두 개 이상의 변수가 있는 데이터에서 각 변수간 상관관계를 보여주는 그래프
- 기본 문법

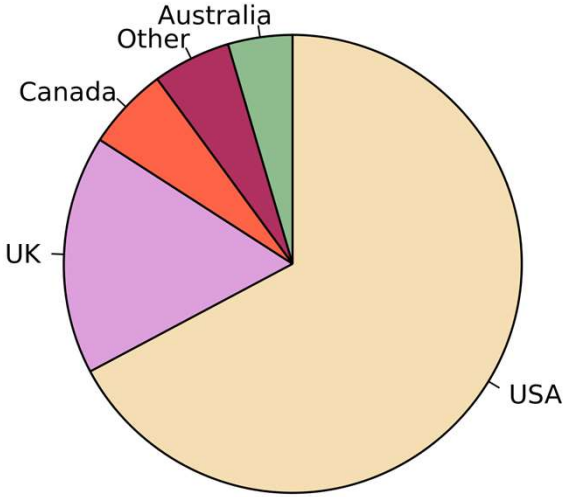
```
pairs(formula, data)
```

매개변수	설명
formula	쌍으로 사용될 일련의 변수
data	변수를 가져올 데이터 세트

```
> pairs(~wt+mpg+disp+cyl,
+       data = mtcars,
+       main = "Scatterplot Matrix")
```



Graph
: Pie Chart



- 전체에 대한 각 데이터의 기여도를 비율로 표현하고자 할 때 사용
  - 타 그래프에 비해 비율 파악에는 유리하지만, 정확한 수치 데이터를 파악할 때는 막대 그래프를 사용하는 것이 효과적
- 기본 문법

```
pie(x, labels, radius, main, col, clockwise)
```

매개변수	설명
x	파이 차트에 비율로 표시할 데이터의 벡터
labels	각 조각에 부여할 라벨(텍스트)
radius	전체 파이 그래프 영역 내에서의 반지름의 비율(-1 에서 1 사이)
main	그래프 제목
col	색상 팔레트 벡터
clockwise	조각 출력 방향 (T or F)

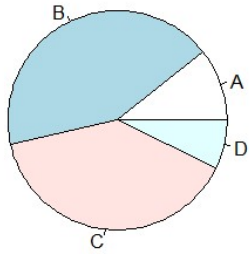
Graph
: Pie Chart

- 최근 시험 결과 다음과 같은 학점을 받은 데이터가 있다고 가정해 봅시다

A	B	C	D
3	12	11	2

- 데이터셋을 만들고 파이 차트로 비율을 시각화 해 봅시다

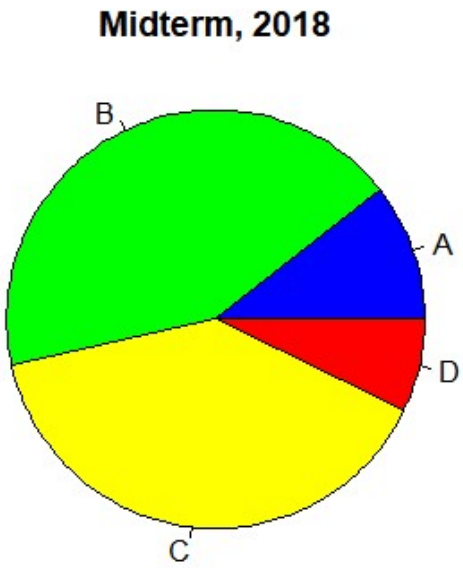
```
> midterm <- data.frame(grade = c("A", "B", "C", "D"),
+                       cnt = c(3, 12, 11, 2))
> pie(midterm$cnt, labels = midterm$grade)
```



Graph
: Pie Chart

- 메인 타이틀을 추가하고, 컬러 팔레트를 추가해 봅시다

```
pie(midterm$cnt,
    labels = midterm$grade,
    col = c("blue", "green", "yellow", "red"),
    radius = 1,
    main = "Midterm, 2018")
```



- 컬러 팔레트는 색상의 이름을 부여할 수도,
  16진수 컬러 코드를 부여할 수도 있습니다.

```
pie(midterm$cnt,
    labels = midterm$grade,
    col = c("#0000FF", "#00FF00", "#FFFF00", "#FF0000"),
    radius = 1,
    main = "Midterm, 2018")
```

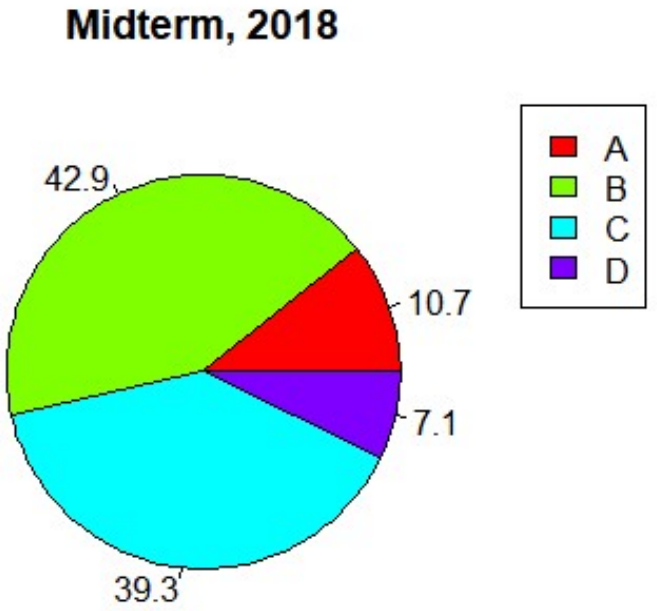
- rainbow 함수를 이용하면 r이 보유한 컬러 팔레트를 이용할 수 있습니다.

```
> rainbow(4)
[1] "#FF0000FF" "#80FF00FF" "#00FFFFFF" "#8000FFFF"
```

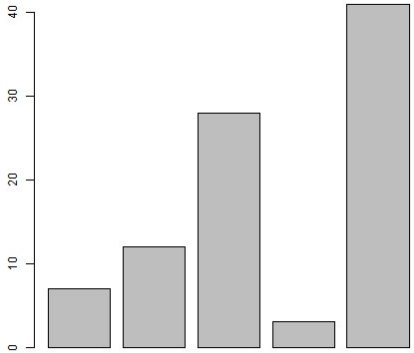
Graph
: Pie Chart

- 응용: 파이 차트 라벨을 비율로 대체하고 Legend(범례)를 넣어봅시다

```
> midterm$percent <- round(100 * midterm$cnt / sum(midterm$cnt), 1) # 비율 파생변수 생성
> midterm$percent
[1] 10.7 42.9 39.3  7.1
>
> pie(midterm$cnt,
+     labels = midterm$percent,
+     col = rainbow(length(midterm$cnt)),
+     main = "Midterm, 2018")
> legend("topright",
+        legend = midterm$grade,
+        fill = rainbow(length(midterm$grade)))
```



Graph
: Bar Chart



- 일정 기간 동안의 데이터 변화 혹은 항목간의 값을 비교하고자 할 때 사용
  각 값들은 면적을 가진 막대기 형태로 표시
- 기본 문법

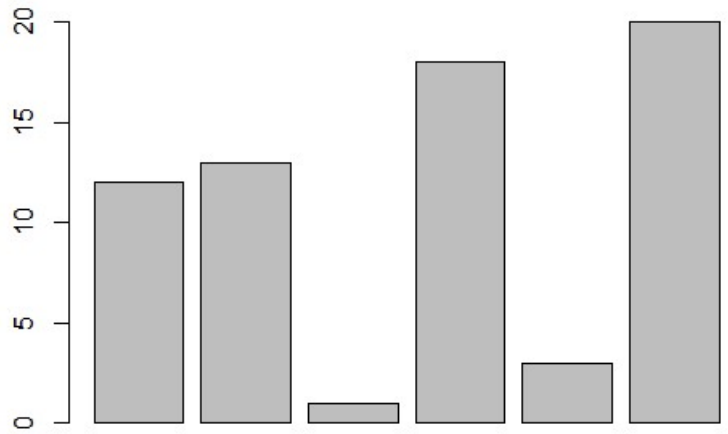
```
barplot(H,xlab,ylab,main, names.arg,col)
```

매개변수	설명
H	바 차트 내부에서 사용할 값들의 벡터
xlab	가로축 레이블(텍스트)
ylab	세로축 레이블(텍스트)
main	바 차트 제목
names.arg	각 막대기 하단에 표시할 이름의 벡터
col	컬러 팔레트 벡터

Graph
: Bar Chart

- 6개월 간의 revenue 데이터를 바차트로 시각화하여 변화 내용을 비교해 봅니다.

```
> rev <- sample(1:20, 6, rep=T)
> barplot(rev)
```



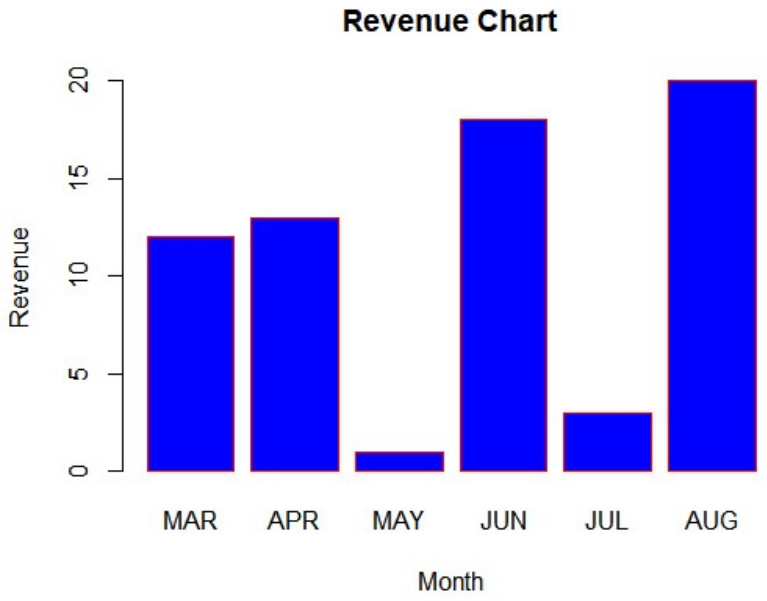
- sample 함수 : 데이터 내에서 무작위로 샘플을 추출해주는 함수. 모집합은 벡터가 될 수도, 데이터 프레임이 될 수도 있음

```
sample({모집합}, 추출 개수, rep={반복 여부 T or F}
```

Graph
: Bar Chart

- Bar Chart에 names.arg, 타이틀과 xlab, ylab, 컬러 팔레트를 추가해 봅니다

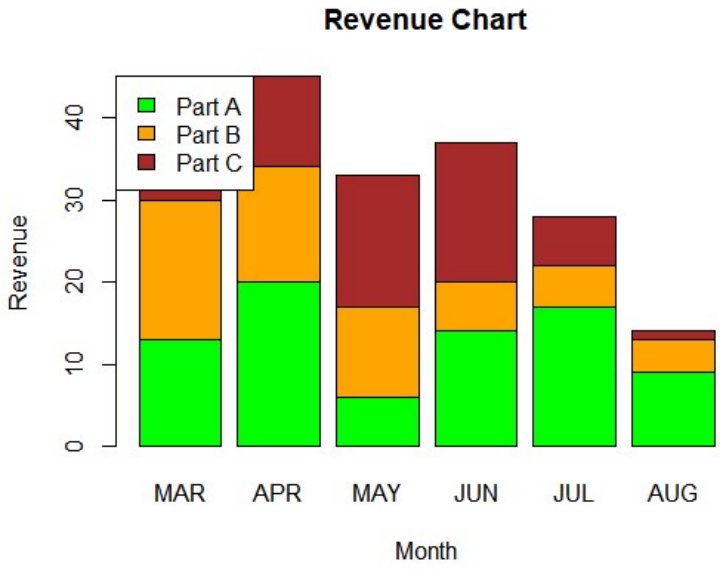
```
> names = c("MAR", "APR", "MAY", "JUN", "JUL", "AUG")
> barplot(rev,
+         names.arg = names,
+         xlab = "Month",
+         ylab = "Revenue",
+         main = "Revenue Chart",
+         col = "blue",
+         border = "red")
```



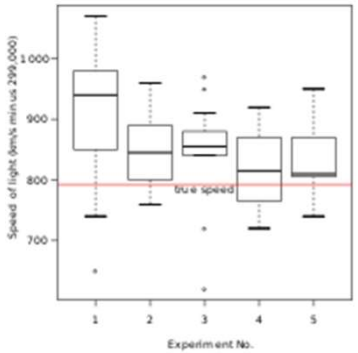
Graph
: Bar Chart

- 응용: 매트릭스를 응용하면 그룹 바차트(혹은 Stacked Bar Chart)를 만들어 낼 수 있음

```
> rev <- sample(1:20, 18, rep=TRUE)
> rev <- matrix(rev, byrow = T, nrow = 3)
> colors <- c("green", "orange", "brown")
> rev
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   13   20    6   14   17    9
[2,]   17   14   11    6    5    4
[3,]    9   11   16   17    6    1
> barplot(rev,
+         names.arg = names,
+         xlab = "Month",
+         ylab = "Revenue",
+         main = "Revenue Chart",
+         col = colors)
> legend("topleft",
+        c("Part A", "Part B", "Part C"),
+        fill = colors)
```



Graph
: Box Plot



- 데이터 셋의 데이터가 얼마나 잘 분포되어 있는가를 표시

- 기본 문법

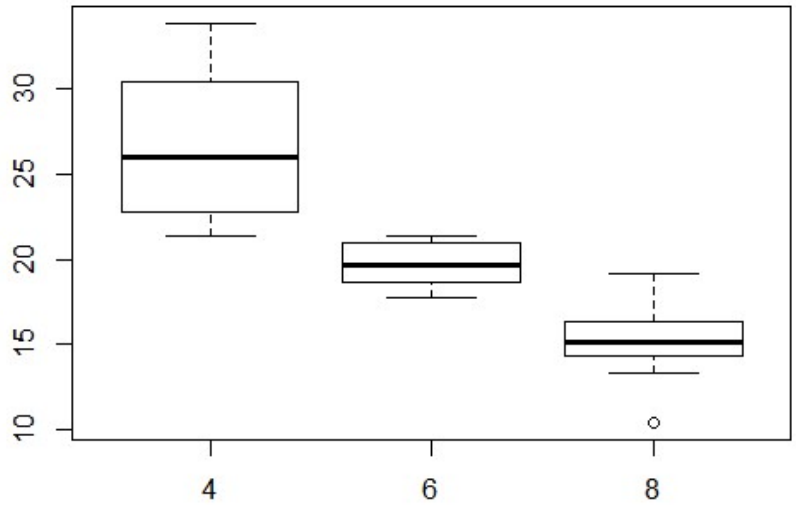
```
boxplot(x, data, notch, varwidth, names, main)
```

매개변수	설명
x	벡터 혹은 formula
data	데이터프레임
notch	TRUE 면 notch 형태로 표시
varwidth	TRUE면 sample 크기에 비례하여 너비를 그림
names	각 박스 그룹 하단에 표시할 그룹 레이블
main	박스 플롯 제목

Graph
: Box Plot

- 기본 데이터셋 mtcars를 이용하여 mpg ~ cyl의 박스 플롯을 그려 봅시다

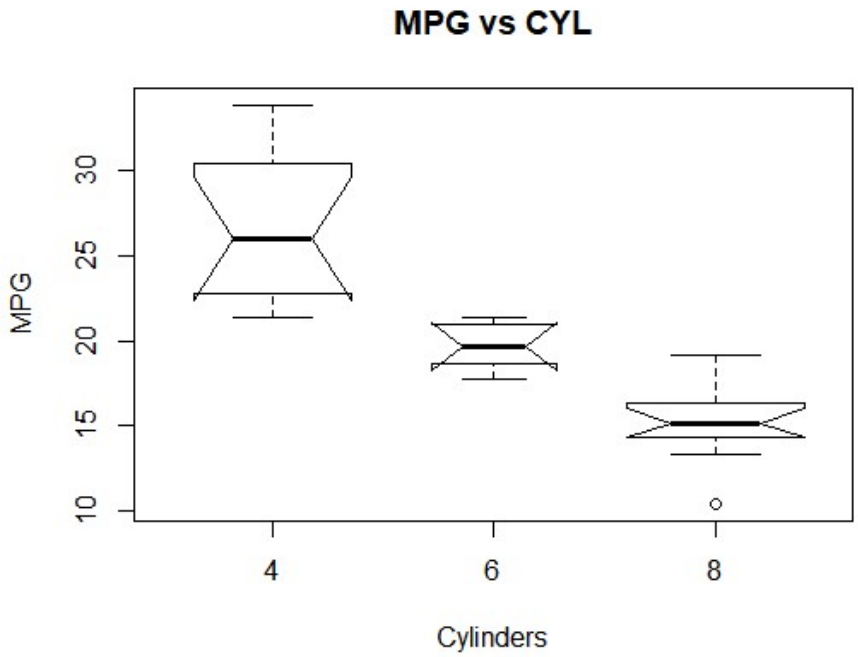
```
> bp.input <- mtcars[,c("mpg", "cyl")]
> head(bp.input)
                   mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
> boxplot(mpg ~ cyl, data = bp.input)
```



Graph
: Box Plot

- 타이틀과 xlab, ylab, notch, varwidth 등을 활용하여 플롯을 완성해 봅시다

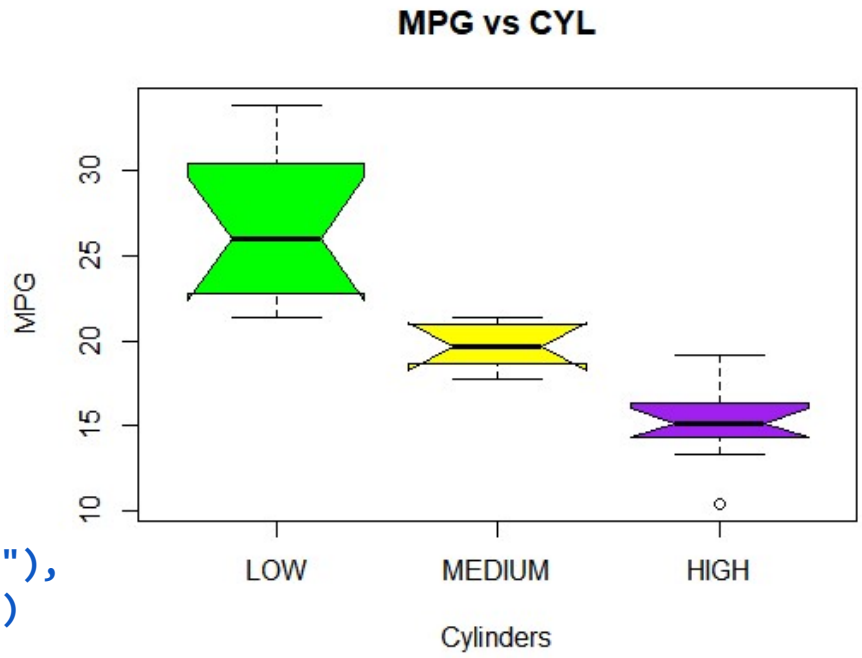
```
> bp.input <- mtcars[,c("mpg", "cyl")]
> head(bp.input)
                   mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
> boxplot(mpg ~ cyl, data = bp.input,
+         main = "MPG vs CYL",
+         xlab = "Cylinders",
+         ylab = "MPG",
+         notch = TRUE,
+         varwidth = TRUE)
Warning message:
In bxp(list(stats = c(21.4, 22.8, 26, 30.4, 33.9, 17.8, 18.65, 19.7,  :
  some notches went outside hinges ('box'): maybe set notch=FALSE
```



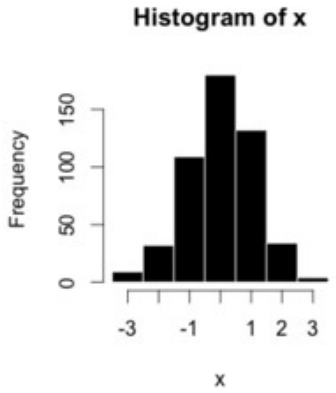
Graph
: Box Plot

- 컬러 팔레트, names 등을 활용하여 플롯을 꾸며 봅시다

```
> boxplot(mpg ~ cyl, data = bp.input,
+         main = "MPG vs CYL",
+         xlab = "Cylinders",
+         ylab = "MPG",
+         notch = TRUE,
+         col = c("green", "yellow", "purple"),
+         names = c("LOW", "MEDIUM", "HIGH"))
Warning message:
In bxp(list(stats = c(21.4, 22.8, 26, 30.4, 33.9, 17.8, 18.65, 19.7,  :
  some notches went outside hinges ('box'): maybe set notch=FALSE
```



Graph
: Histogram



- 범위로 묶은 변수의 빈도를 표현
  바 차트와 비슷하지만 히스토그램은 연속 범위를 그룹화
  히스토그램의 각 막대는 해당 범위 내에 있는 값의 수의 높이를 표현
- 기본 문법

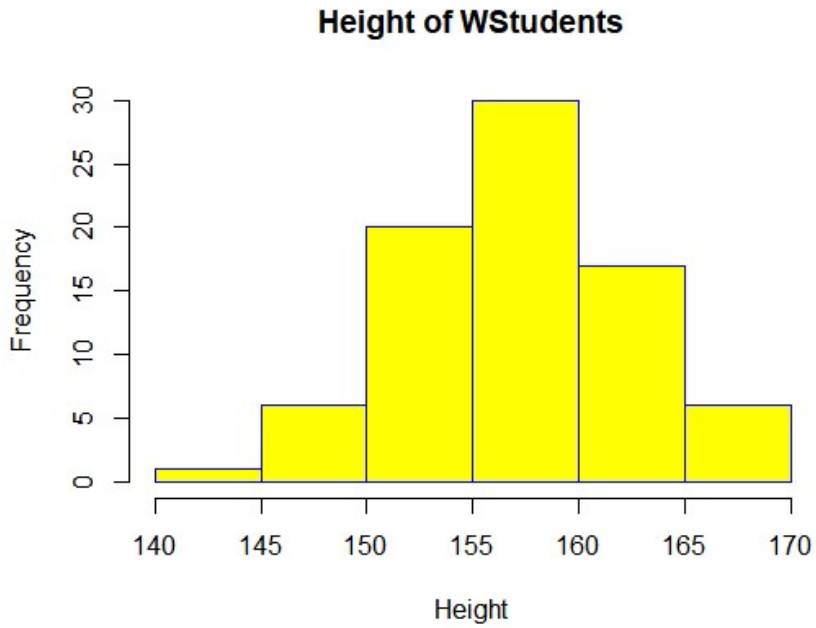
```
hist(v,main,xlab,xlim,ylim,breaks,col,border)
```

매개변수	설명
v	히스토그램에 사용할 수치 값의 벡터
main, xlab	메인 타이틀, 가로축 라벨
xlim, ylim	가로축, 세로축 제한 범위(벡터)
breaks	데이터를 그리기 위해 데이터를 어떻게 나눌 것인지를 결정
col	막대의 컬러 팔레트
border	막대의 윤곽선 컬러 팔레트
freq	TRUE 혹은 NULL이면 빈도, FALSE 면 확률 밀도

Graph
: Histogram

- wstudent 데이터의 height 변수를 이용하여 히스토그램을 그려 봅시다

```
> hist(wstudents$height,
+      main = "Height of WStudents",
+      xlab = "Height",
+      col = "yellow",
+      border = "blue")
```



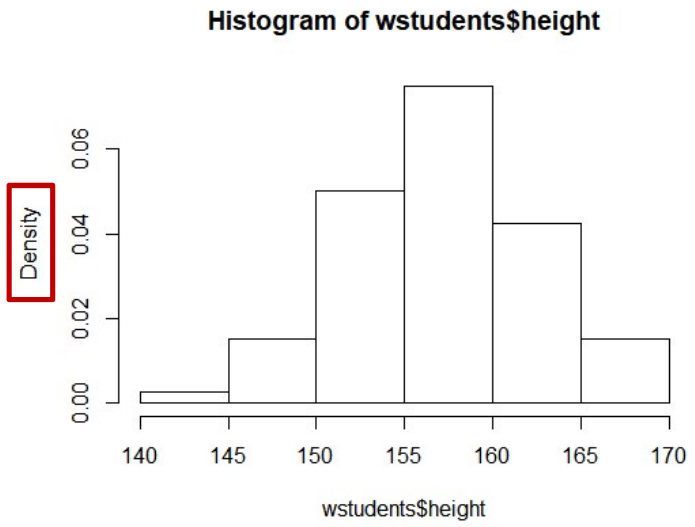
- xlim, ylim을 이용하여 표시 영역을 제한해 봅시다

```
> hist(wstudents$height,
+      main = "Height of WStudents",
+      xlab = "Height",
+      col = "yellow",
+      border = "blue",
+      xlim = c(150, 165),
+      ylim = c(0, 20))
```

Graph
: Histogram

- R의 Histogram 그래프는 단순히 그래프를 그리는데 그치지 않고 해당 그래프를 그리는데 계산된 통계 값을 반환
  - 계산된 통계 값은 그래프 객체에 리스트로 저장되어 있어 그래프를 그린 이후에 통계 값을 활용할 수 있음
  - freq = FALSE로 부여하면 빈도 히스토그램이 아니라 확률 밀도 히스토그램을 취할 수 있음

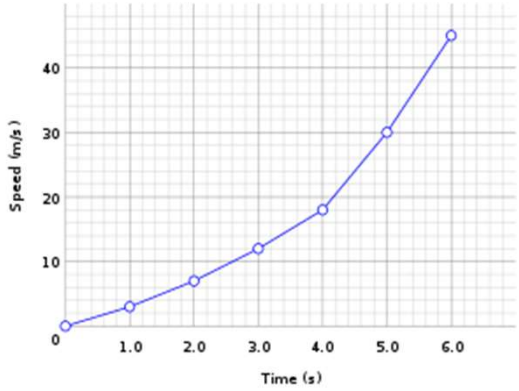
```
> (stat <- hist(wstudents$height, freq = FALSE))
$`breaks`
[1] 140 145 150 155 160 165 170
...
```



  - 확률 밀도의 경우, 바 너비와 density를 계산하면 그 총 합은 1이 된다

```
> sum(stat$density * 5)
[1] 1
```

Graph
: Line Graph



- 일련의 점들 사이를 선으로 연결하여 표현한 그래프
  주로 시간의 흐름에 따른 추세 혹은 트렌드 변화를 식별하고자 할 때 사용
- 기본 문법

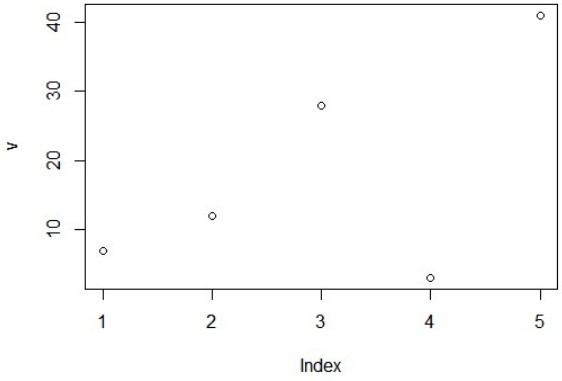
```
plot(v, type, col, main, xlab, ylab)
```

매개변수	설명
v	라인 그래프에 표시할 수치 데이터 벡터
type	그래프의 형태 - p : 점만 그림 - l : 선만 그림 - o : 점과 선 모두 그림
xlab	가로축 레이블(텍스트)
ylab	세로축 레이블(텍스트)

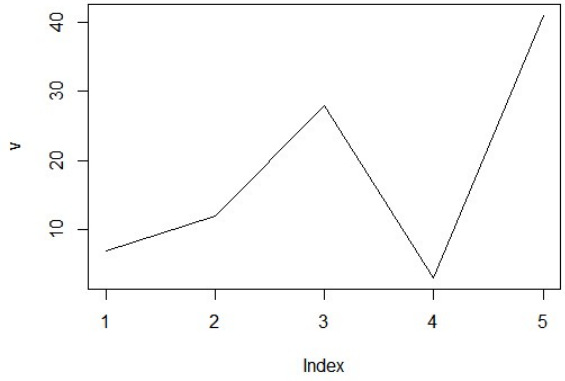
Graph
: Line Graph

- 라인 그래프에 설정할 벡터를 정의하고 type 을 변경하며 그래프의 변화를 살펴봅시다

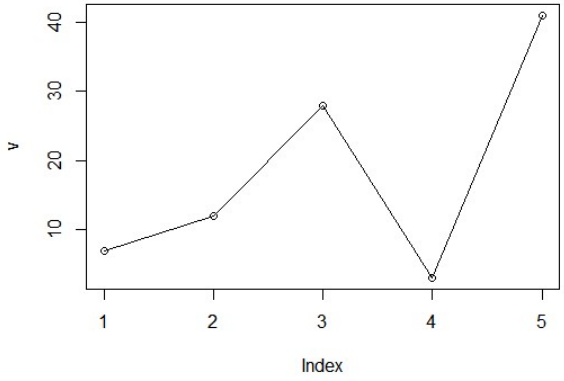
```
> v <- c(7, 12, 28,3, 41)
> plot(v, type = "p")
> plot(v, type = "l")
> plot(v, type = "o")
```



type = "p"



type = "l"

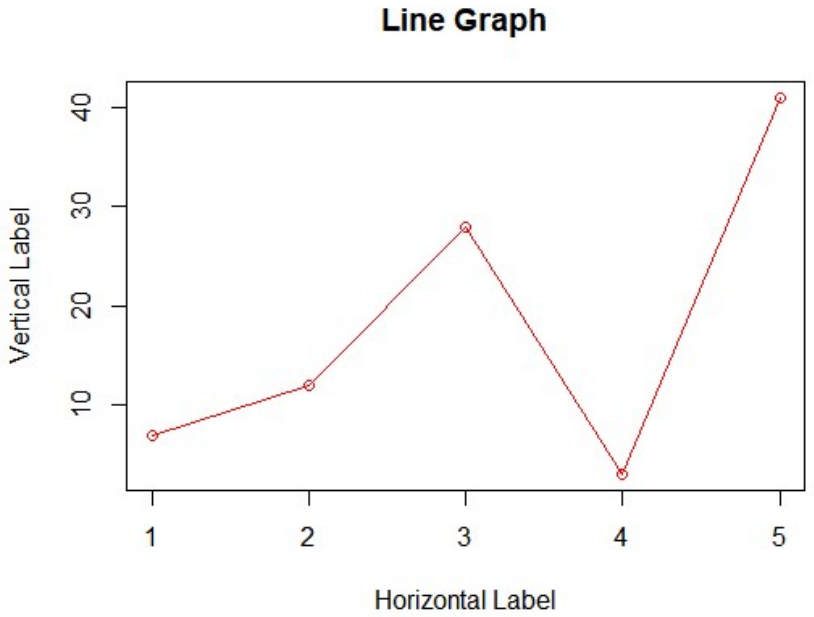


type = "o"

Graph
: Line Graph

- Title과 col, xlab, ylab 등을 추가하여 라인 그래프를 완성해 봅시다

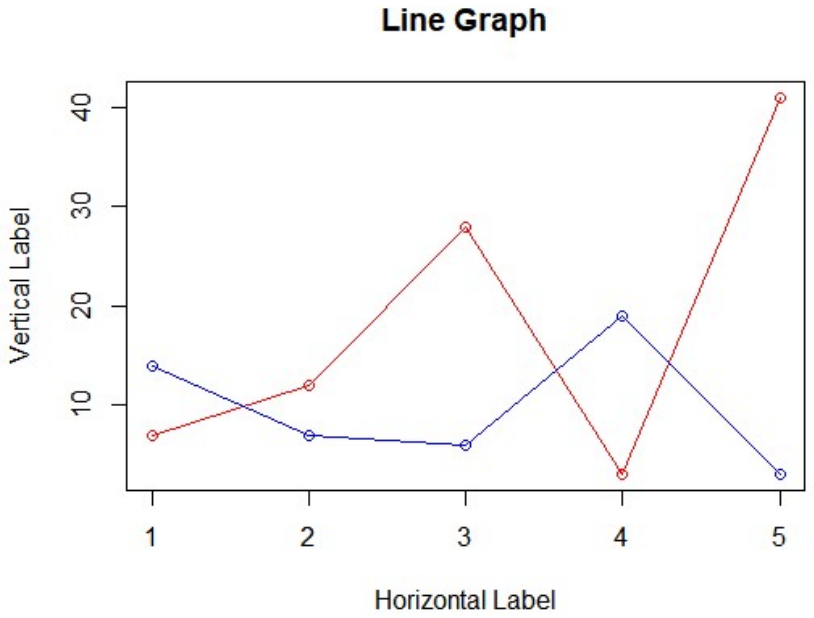
```
> v <- c(7, 12, 28, 3, 41)
> plot(v, type = "o",
+      col = "red",
+      main = "Line Graph",
+      xlab = "Horizontal Label",
+      ylab = "Vertical Label")
```



Graph
: Line Graph

- 다중 라인 그래프
  - lines, points 함수를 이용하면 기존 라인 그래프에 다중 그래프를 그릴 수 있다

```
> v <- c(7, 12, 28, 3, 41)
> t <- c(14, 7, 6, 19, 3)
> plot(v, type = "o",
+      col = "red",
+      main = "Line Graph",
+      xlab = "Horizontal Label",
+      ylab = "Vertical Label")
> lines(t,
+       type = "o",
+       col = "blue",)
```



Data Visualization

with GGPLOT2

Data Visualization

- 데이터 시각화(Data Visualization) : 데이터 분석의 결과를 쉽게 이해할 수 있도록 시각적으로 표현하고 전달하는 과정
  - 주요 목적은 도표(Graph)라는 수단을 통해 정보를 명확하고 효과적으로 전달하는 것

- 비정형, 대량의 데이터가 중심인 빅데이터 기술 시장의 성장과 함께 시각화 기술도 중요한 기술요소로 자리잡고 있음
  - 빅데이터는 모든 데이터를 살펴보는 것에 제약이 따르기 때문에 시각화의 기술적 요소와 더불어 데이터를 요약하고, 한 눈에 살펴볼 수 있도록 돕는 시각화 방법론적 요소의 중요성이 커지고 있음

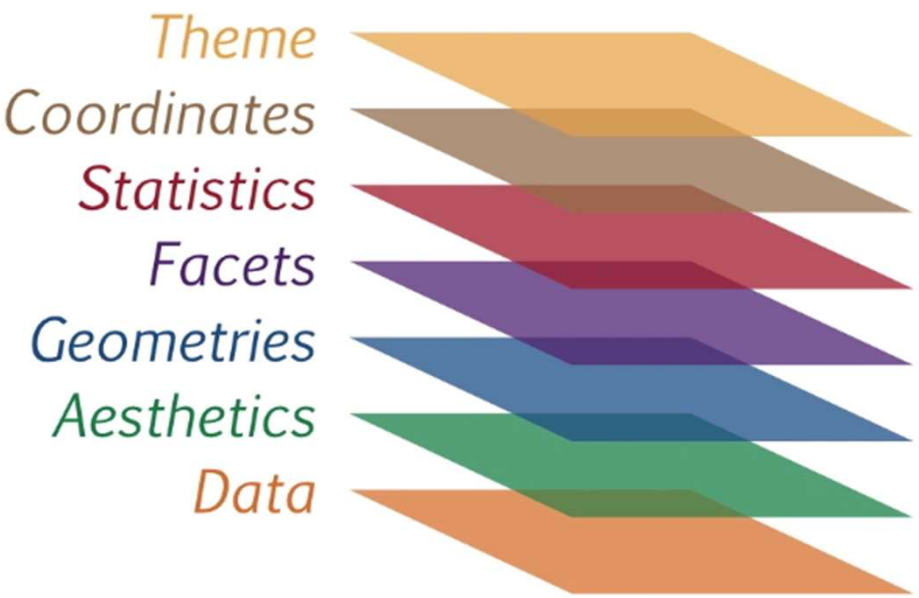
- 데이터 시각화에서는 미적 형태와 기능성 두 가지를 모두 충족해야 한다

GGPlot2

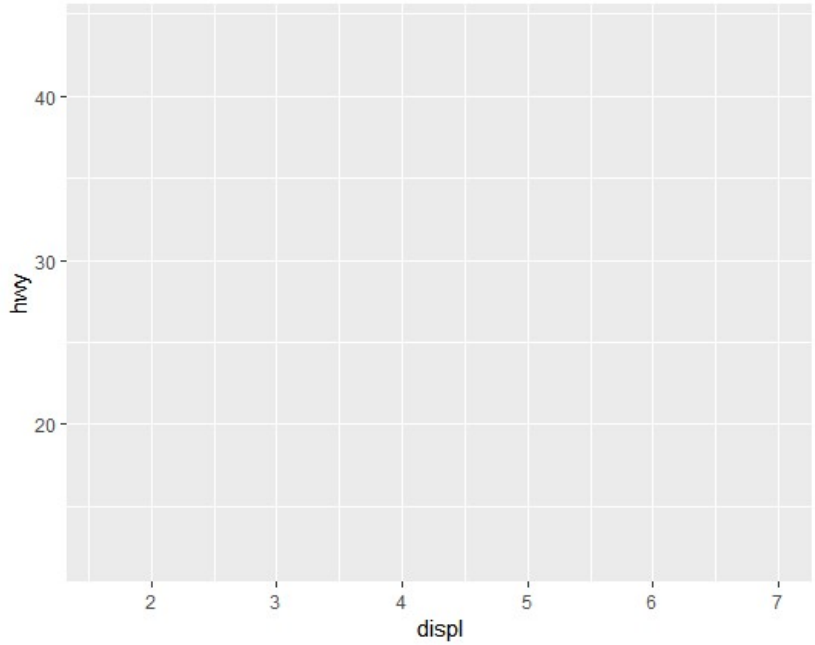
- Wickham 교수가 만든 데이터 시각화용 패키지
  - R은 다양한 기본 그래프 기능을 제공하고 있지만, 사용법이 제각각이고 전통적인 그래프 영역에 머물러 있다.
  - 표준화된 Grammer of Graphics 기초 문법을 기반으로 직관적인 함수를 제공하고 있으며 그래픽 시스템의 완성도가 높음
  - 레이어 추가 방식으로 레이어를 추가 혹은 교체해 가면서 그래프를 완성시켜 나가는 방식을 도입

- 주요 3 Layers
  - 배경 설정 : 축
  - 그래프 추가 : 점, 선, 막대 등
  - 추가 설정 : 축 범위, 색, 표식 등

  - 각 레이어는 + 를 이용하여 연결한다



GGPlot2
: 설치 및 사용



- GGPlot2 설치와 로드

```
> install.packages("ggplot2")
> library(ggplot2)
```

- 내장된 데이터셋의 확인

```
> data(package = "ggplot2")
```

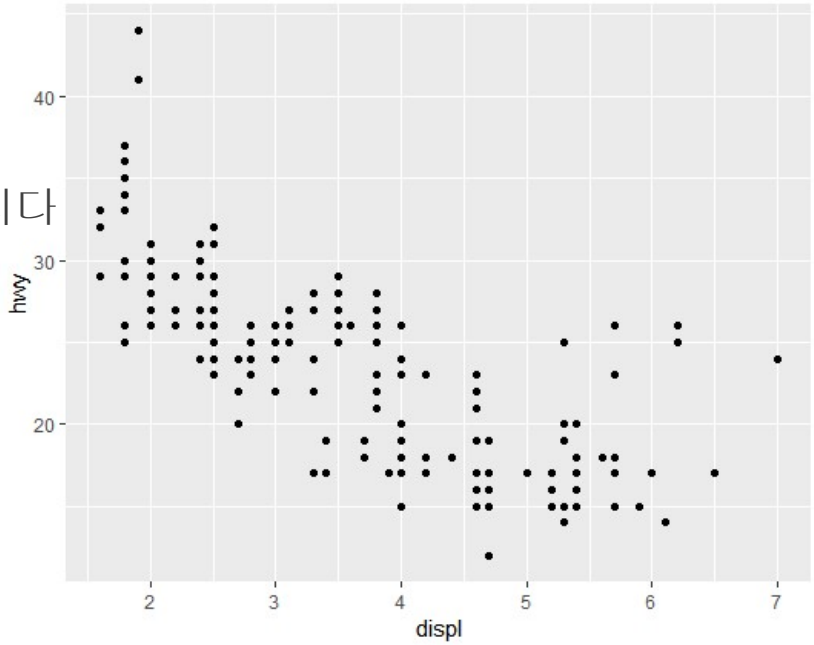
- Step 1. 내장된 데이터셋 mpg를 이용하여 ggplot 배경을 설정합니다

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

- ggplot() 함수에 직접 aes(좌표계)를 추가할 수도 있습니다.

```
> ggplot(data = mpg, aes(x = displ, hwy))
```

GGPlot2
: 그래프 레이어와 상세 설정



- Step 2: 기 설정된 ggplot base에 그래프 레이어를 얹어 봅시다

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

- Step 3: 세부 설정의 추가

```
> ggplot(data = mpg) +
+   geom_point(aes(x = displ, y = hwy)) +
+   xlim(3, 6) + ylim(10, 40)
```

  - xlab, ylab을 이용하여 가로축과 세로축 레이블을 설정해 봅시다

- mpg를 이용한 배경을 별도의 변수에 담아 봅시다

```
> (plt <- ggplot(data = mpg))
```

GGPlot2
: 레이어의 변경

- 기 작성된 ggplot 플롯 객체를 이용,
  그래프의 종류를 바꿔 봅니다.

```
> plt + geom_line(aes(x = displ, y = hwy)) +
+     xlab("배기량") + ylab("고속도로 연비")
```

- aes 설정은 ggplot에서 해도, geom_ 쪽에서 해도 무방