R로 그리는 탐색 그래프 (exploratory plotting)

문서 컴파일

이 문서는 최초에 `Rmarkdown` 형태로 작성되었고 `knitr` 패키지로 `markdown` 번역한 뒤에 `marked2` 애플리케이션으로 pdf 형태로 최종 컴파일 하였다.

```
> # knit("graphing.Rmd", output = "graphing.md")
```

필요한 자료철

- anscombe : R 기본 패키지에 들어 있다.
- Orthodont : nlme 패키지에 있다.
- mtcars : 기본 패키지
- nycflights : 수업 깃허브의 RData 형태로 배포된 것
- bliss : 손으로 입력할 것

시작

앤스컴비가(Anscombe, F. J. (1973). “Graphs in Statistical Analysis”. American Statistician. 27 (1): 17–21) 만든 다음의 숫자들을 눈여겨보자. anscombe 자료철은 기본 패키지에 들어 있으므로 다음 명령을 내릴 필요조차 없지만:

```
> data(anscombe)
```

총 8개의 변수를 가지고 있으며, x1부터 x4까지 가로축에 그려질 숫자들 각각 11개씩과 y1부터 y4까지 세로축에 그려질 숫자 11개씩이 주어져 있다. 숫자들의 평균과 표준편차를 얻어 보면 x들이 모두 같은 요약값을 가지고 y들도 모두 같은 요약값을 가지고 있다.

```
> apply(anscombe, 2, function(x) round( mean(x), 2) )
```

```
##   x1   x2   x3   x4   y1   y2   y3   y4
## 9.0  9.0  9.0  9.0  7.5  7.5  7.5  7.5
```

```
> apply(anscombe, 2, function(x) round( sd(x), 2) )
```

```
##   x1   x2   x3   x4   y1   y2   y3   y4
## 3.32 3.32 3.32 3.32 2.03 2.03 2.03 2.03
```

하지만 이들을 (x1, y1), (x2, y2) … 방식으로 짝을 지워서 산포도만 그렸을 뿐임에도 수치들의 조합이 보여 주는 경이로운 차이가 시선을 압도한다.

```
> par(mfrow = c(2, 2))
> plot(y1 ~ x1, anscombe, main = "approximate linear relation")
> plot(y2 ~ x2, anscombe, main = "arc")
> plot(y3 ~ x3, anscombe, main = "perfectly linear with a leverage"
)
> plot(y4 ~ x3, anscombe, main = "two arcs with one outlier?")
```

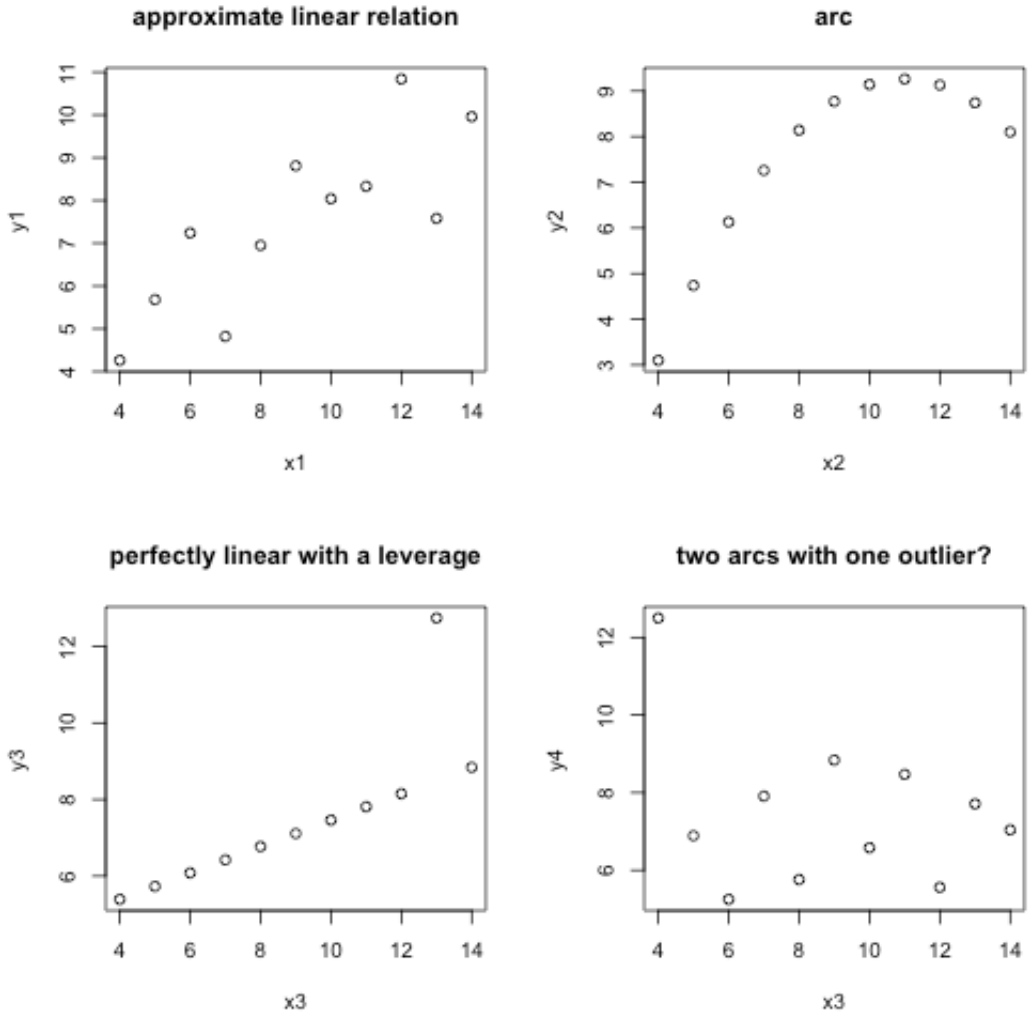


figure: The Anscombe Quartet

자료에 대한 본격적인 수치해석을 하는 것에 더불어, 탐색적인 그래프로 전반적인 조망을 하는 일도 중요하다는 강렬한 예시이다. 그래프를 그리지 않은 채 앤스컴비의 자료를 해석하려 대든다면 오류와 한계를 피하기 어려웠을 것이다.

탐색 그래프를 구분하는 법은 다양하다. 당장에 그리고자 하는 변수가 하나일 때는 히스토그램 말고는 딱히 그릴 기법이 없지만, 설명변수(explanatory variables; X)와 응답변수(response variables; Y)로 짝을 지워서 설명하고(X로 Y를 설명할 수 있을까? = Y는 X에 따라서 어떻게 변화하는가?) 그리고자 할 때엔 해당하는 변수들이 무슨 척도로 측정되었는가를 알고 시작하는 것이 필요하다.

간격척도 대 간격척도

`mtcars` 자료철에서 자동차의 마력(`hp`)과 연비(`mpg`)는 모두 간격척도로 측정되어 있으므로 다음처럼 관계를 그릴 수 있다. 일반적인 산포도의 형태를 띄고 있다.

```
> plot(hp ~ wt, mtcars)
```

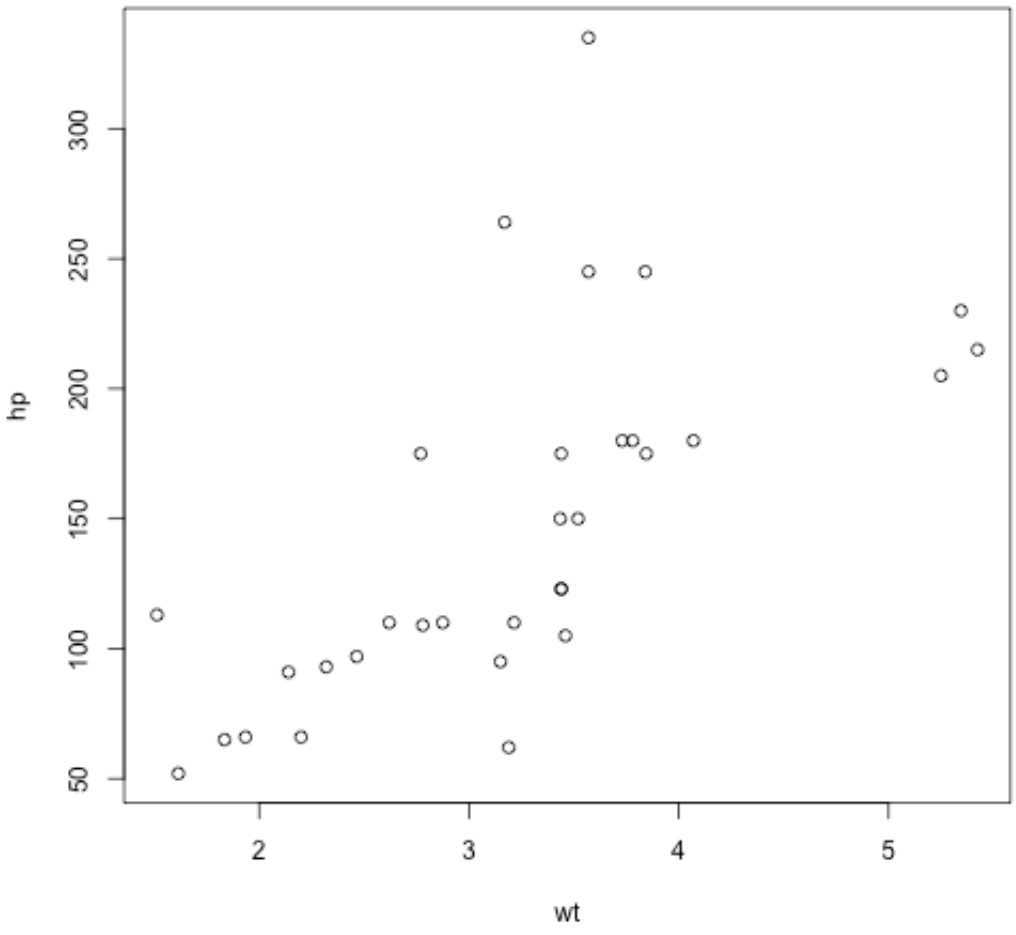
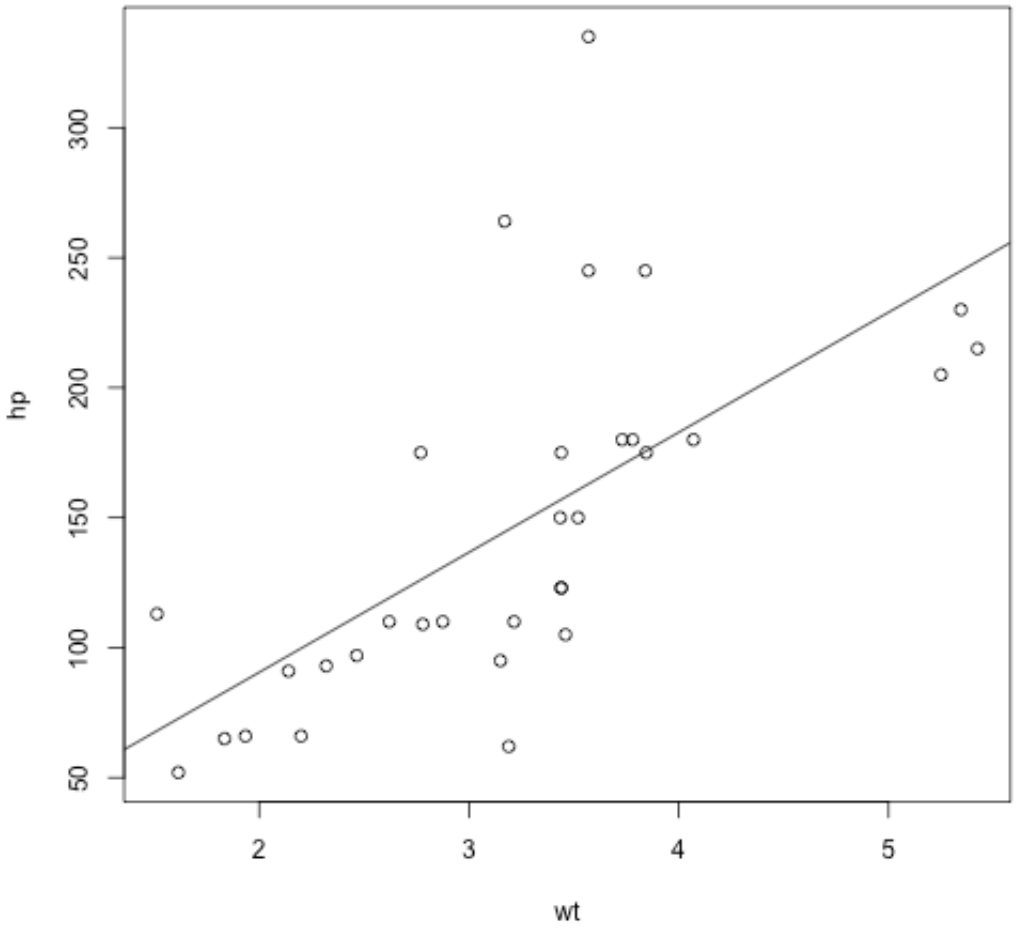


figure: Simple scatterplot. Y is explained by X.

그래프의 가운데 뾰족한 몇 개의 값을 고려사항에 넣더라도 hp와 mpg 사이에는 선형의 증가 관계가 있음을 알 수 있다. 선형회귀분석은 결론을 도출하는 통계분석에도 쓰이지만 보통은 탐색분석에 많이 쓰인다. 선형회귀분석을 행하는 R 함수는 `lm()`이다.

```
> plot(hp ~ wt, mtcars)
> cars.lm <- lm(hp ~ wt, mtcars)
> abline(cars.lm)
```



`plot()` 뒤의 두 행은 `hp ~ wt`관계를 선형으로 회귀하여 절편과 기울기를 얻은 후 그 값들을 기초로 하는 직선을 현재의 그래프에 더하는 `abline()` 함수에 넣은 것이다. 이해하기 쉽게 풀어서 쓰면 이렇게 된다.

```
> cars.lm <- lm(hp ~ wt, mtcars)
> summary(cars.lm) # intercept = -1.8, slope = 46.2
```

```
##
## Call:
## lm(formula = hp ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -83.430 -33.596 -13.587   7.913 172.030
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.821     32.325  -0.056    0.955
## wt            46.160      9.625   4.796 4.15e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.44 on 30 degrees of freedom
## Multiple R-squared:  0.4339, Adjusted R-squared:  0.4151
## F-statistic:    23 on 1 and 30 DF,  p-value: 4.146e-05
```

```
> # repeat
> plot(hp ~ wt, mtcars)
> abline(a = -1.8, b = 46.2, lwd = 1.5)
```

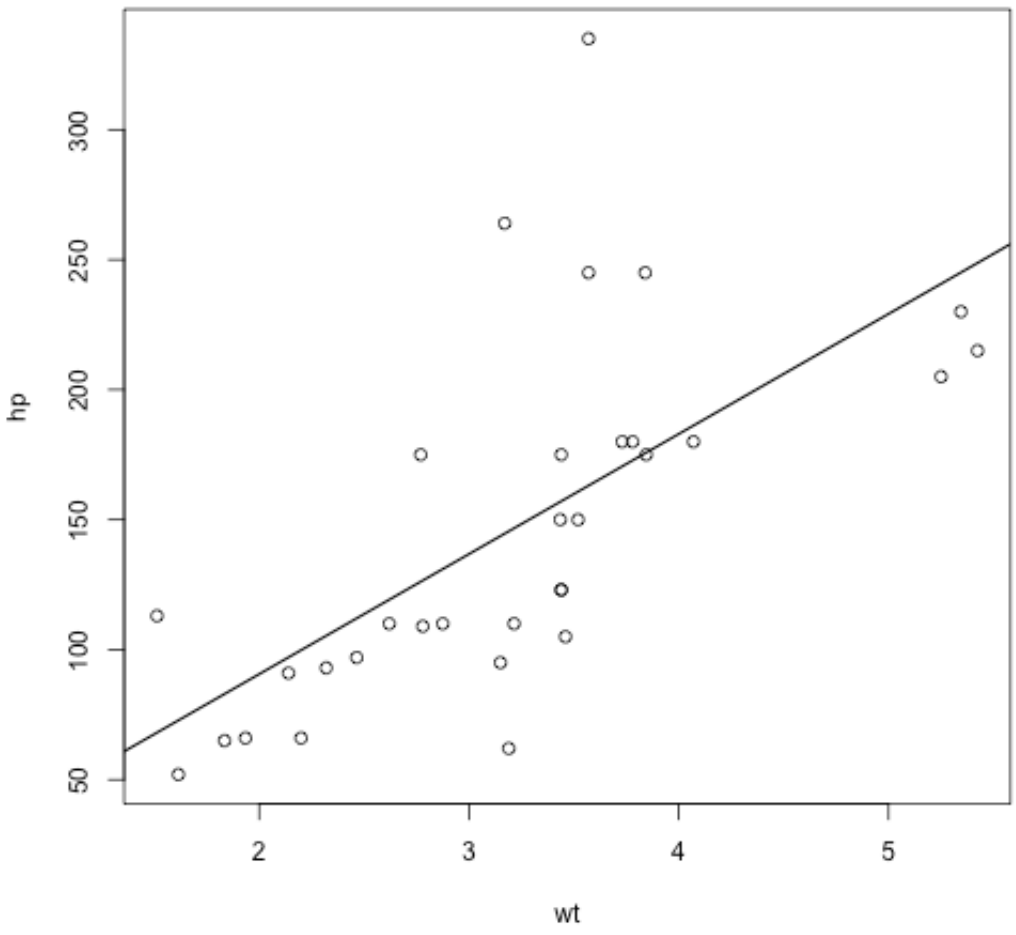


figure: Scatterplot with a line having regression coefficients input manually

마지막에 넣은 `abline()` 함수의 인수인 `lwd`는 R 그래프에서 선의 굵기를 배수의(0.5, 1, 1.3, 2, 3,) 형태로 지칭하는 공통약속이다.

간격척도 대 범주척도

Orthodont 자료철에서 성장 길이(distance)를 연령에 따라서 그리면 그림이 어떻게 보일까? 그리기 전에도 상상할 수 있지만 이 자료철에서 연령은 비록 간격척도로 측정되기는 하였지만 8, 10, 12, 14세라는 네 번만 측정되었을 뿐 그 사이에 측정된 값이 없으므로 그림을 그릴 때는 범주로 취급하는 것도 한편으로는 타당하다.

```
> data(Orthodont, package = "nlme")
> plot(distance ~ age, Orthodont)
```

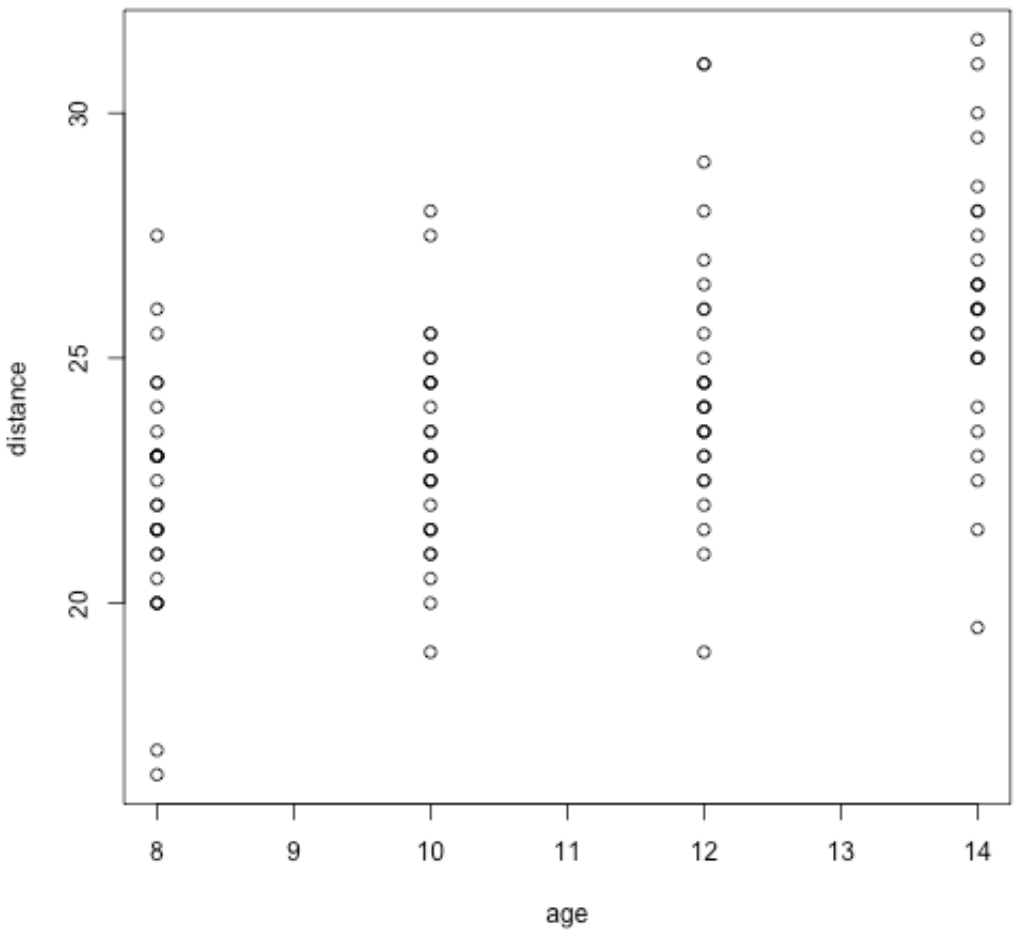


figure: Scatterplot of distances per ages

그러나 위와 같은 산포도로 숫자들의 분포를 알기는 어려워 보인다. X가 범주이고 Y가 간격척도일 때 무작정 탐색적으로 그려 보는 상자그림(boxplot)이 있다:

```
> boxplot(distance ~ age, Orthodont, whisker = F)
```

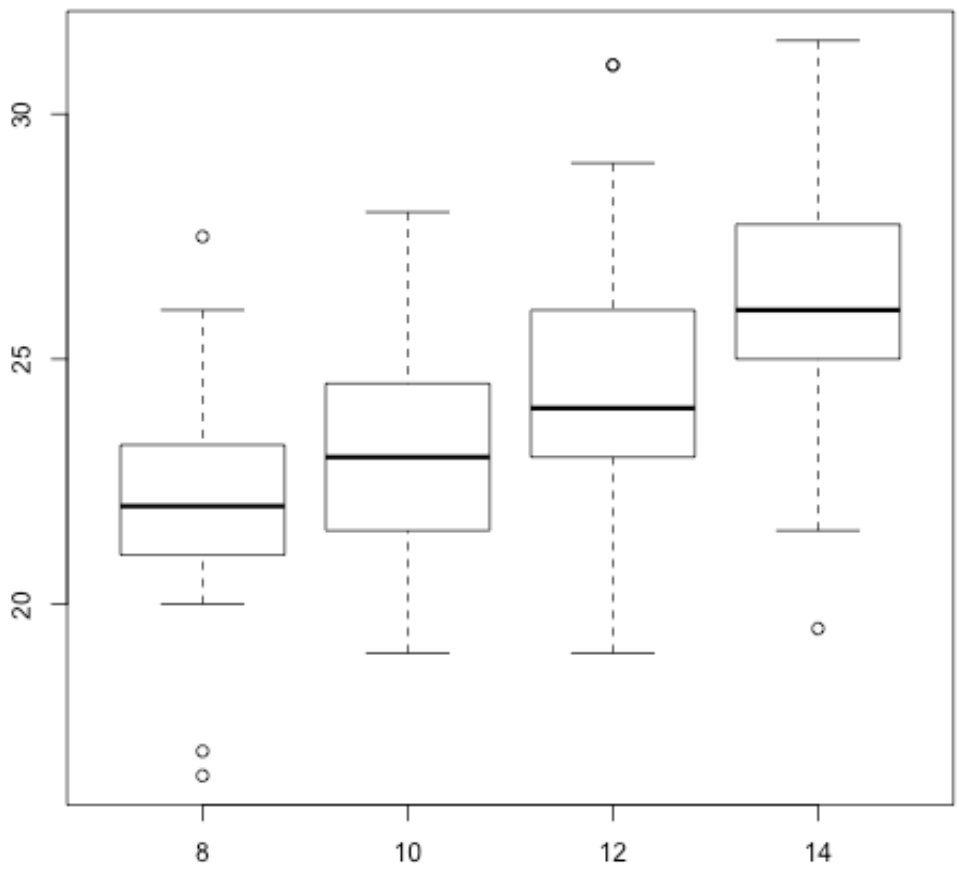


figure: Boxplot of distances per ages

상자그림의 정식 명칭은 상자수염그림(box-and-whisker plot)이다. 가운데의 네모가 상자이고 상자 안의 굵은 선은 중간값이고 장자 밖으로 그린 점선이 수염이고 수염 바깥의 동그라미는 외톨이이다. 각각의 뜻이 이름만으로 대변되는 것은 아니어서 풀어서 쓰면 이러하다:

- 상자: 자료의 하한선과 상한선이 상자의 길이를 반영하며 때에 따라서 상자의 폭이 관찰의 개수를 뜻하기도 한다(위 그림에서는 아니다). 하한선과 하한선은 제1사분위수(Q1), 상한선은 제3사분위수(Q3)이다.
- 상자 안의 굵은 선: 중간값, 즉 제2사분위수(Q2)이다.
- 수염: 사분범위(IQR = Q3 - Q1)의 1.5배를 계산해서 Q1으로부터 아래쪽 수염을, Q3로부터는 위쪽 수염을 그리되 실제 관찰된 최소값과 최대값이 범위 안에 있을 때는 축소해서 그린다.
- 외톨이: 수염 바깥에서 관찰된 값들이다.

비록 R이 자동으로 그려 주기는 하지만 수염을 그리는 좌표를 이해하는 것이 초심자들에게는 가장 어려울 것이다. Orthodont 자료로부터 계산하면 이렇다. 10세 자료만 따로 취하면서 시작한다:

```
> ortho10 <- subset(Orthodont, age == 10)
> summary(ortho10$distance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   21.50   23.00   23.17   24.50   28.00
```

```
> (IQR <- 24.5 - 21.5)
```

```
## [1] 3
```

중간값은 23, Q1은 21.5, Q3는 24.5이므로 IQR은 3이고 1.5배를 취하면 4.5이다. 따라서 10세 때 그려진 상자의 상한선은 24.5, 하한선은 21.5에 놓인다. 굵은 대표선은 23이다. 수염의 상한선은 Q3 + 4.5인 29여야 하지만 최대값이 28이므로 28까지만 그리고 수염의 하한선은 Q1 - 4.5인 17이 아니라 관찰된 최소값이 19이므로 거기까지만 그리게 된다.

```
> par(mar = c(5, 4, 4, 15))                    # margin width (bot
tom, left, top, right)
> boxplot(distance ~ age, ortho10)
> abline(h = 21.5, lty = 2)                    # lower border of t
he box#
> abline(h = 24.5, lty = 2)                    # upper border of t
he box
> abline(h = 23, lty = 2)                      # median
> abline(h = 28, lty = 2)                      # upper whisker
> abline(h = 19, lty = 2)                      #  lower whisker
> axis(4,
+       at = c(19, 21.5, 23, 24.5, 28),
+       label = c("Q1 - 1.5*IQR or minimum",
+                 "Q1",
+                 "Q2 (median)",
+                 "Q3",
+                 "Q3 + 1.5*IQR or maximum"),
+       las = 1                                 # tick-label axis (
1 = standing
+                                               # 2 = 90-de
g counterclockwise,
+                                               # 3 = upsid
e down
+                                               # 4 = 90-de
g clockwise ...)
+       ) #
```

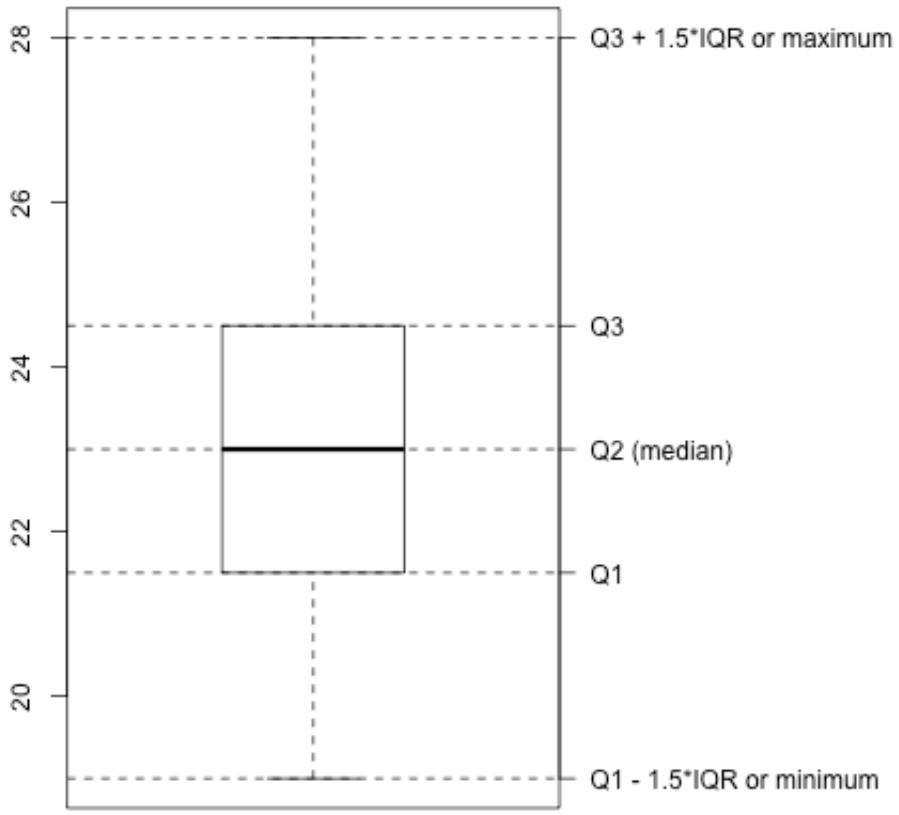


figure: Analytics of boxplot

`boxplot()` 함수는 다양한 인수를 조정함으로써 상자수염그림의 내재적인 다양성을 포괄할 수 있도록 되어 있으므로 함수 도움말을 숙지하는 것이 좋겠다. 방금의 설명 그래프에서 쓰인 `abline()` 함수에 사용된 그래프 인수 `lty`는 선의 형태, 즉 linetype이다. 지정하지 않으면 기본값인 1 (실선)이 대입된다.

차원 증강

Orthodont 자료철은 치아의 성장을 설명하는 피험자의 특성 가운데 성별도 측정되어 있으며 앞의 16명이 남자아이, 뒤의 11명이 여자아이이다. 위의 상자그림에 남녀별 차이를 가미하는 것을 차원을 증강한다고 말한다.

```
> boxplot(distance ~ age * Sex, Orthodont)
```

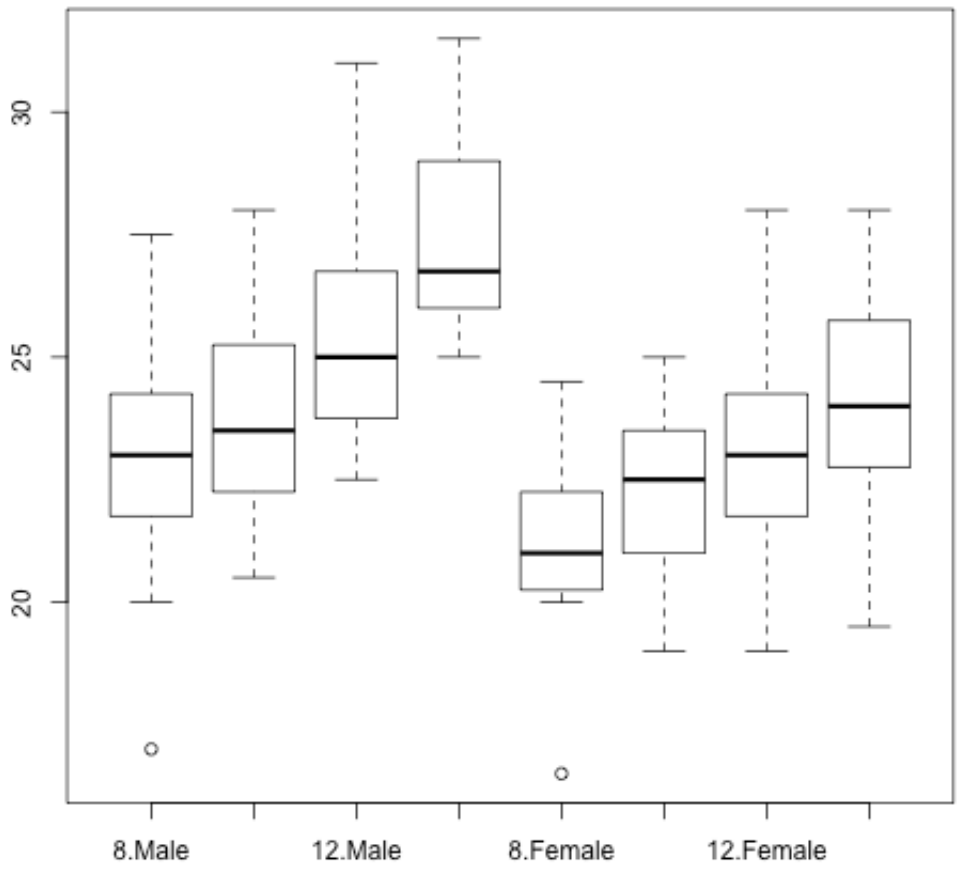


figure: Boxgraphs for age and sex

가로축의 틱 이름을 바꾸지 않으면 출판 용도로는 쓰지 못하겠지만 이것만으로도 탐색적인 용도로는 제값을 한다. 성별에 의한 차이가 눈에 잘 띄게 되었다. 덤으로 성별로 나누었더니 전체 그래프에서는 관측되었던 12세와 14세 때의 외톨이값들이 성별로 구분했더니 사라졌음을 주목할 수 있으며 8세 때의 하한구간의 외톨이값들은 성별로 구분해도 여전히 존재한다.

```
> boxplot(distance ~ Sex * age, Orthodont, col = c(0, 2))
```

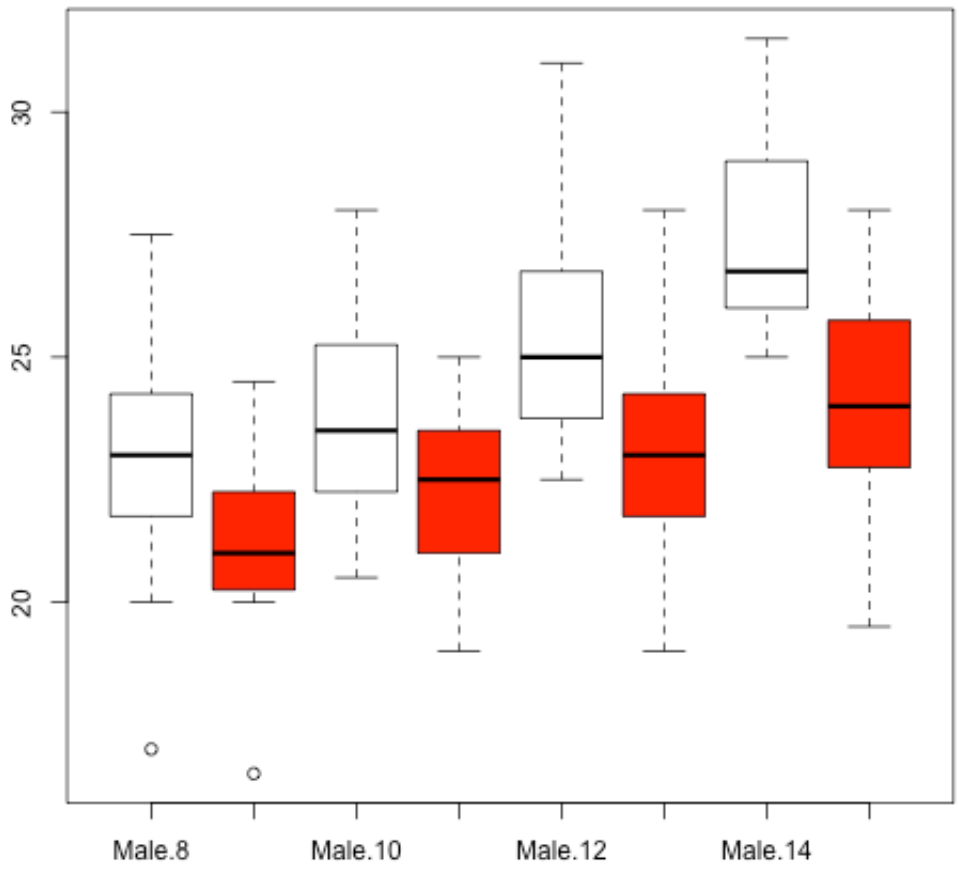


figure: Boxgraphs for sex and age

남자아이 상자의 색상을 투명(0)으로 두고 여자아이들의 색상을 빨강(2)으로 바꾸었다.

반복측정된 간격척도 대 범주척도

Orthodont 자료철은 한 피험자에서 4회의 측정이 이루어진 종적자료(longitudinal data)이므로 위의 방식으로 묘사할 경우에는 피험자-내 변동(within-subject variability)이 피험자-간 변동(between-subject variability) 속에 파묻혀서 식별할 수 없게 된다.

흔히 R에서 선그래프는 `plot(... type = "l")`로 그리기 마련인데 이를 단위 피험자에만 적용하는 것은 가능하지 않고 확장 도구인 `matplot()`을 쓰는 것이 편리하겠다. 행렬그래프라고 번역할 수 있지만 통용되는 말은 아니다. `matplot()`은 X를 범주로 인지하고 반복측정된 Y 행렬을 받아들인다. 여기에 Orthodont 자료철을 넣으려면 사전 가공이 필요하다. X에는 범주의 이름만, Y에는 반복측정된 값들을 X의 길이와 같은 행(row)을 가진 행렬에 넣는다.

```
> x <- c(8, 10, 12, 14)
> y <- matrix(Orthodont$distance,
+        nrow = 4)
> matplot(x, y, type = "l", col = 1, lty = 1,
```

```
+        ylab = "distances (mm)",
+        xlab = "age (year)")
```

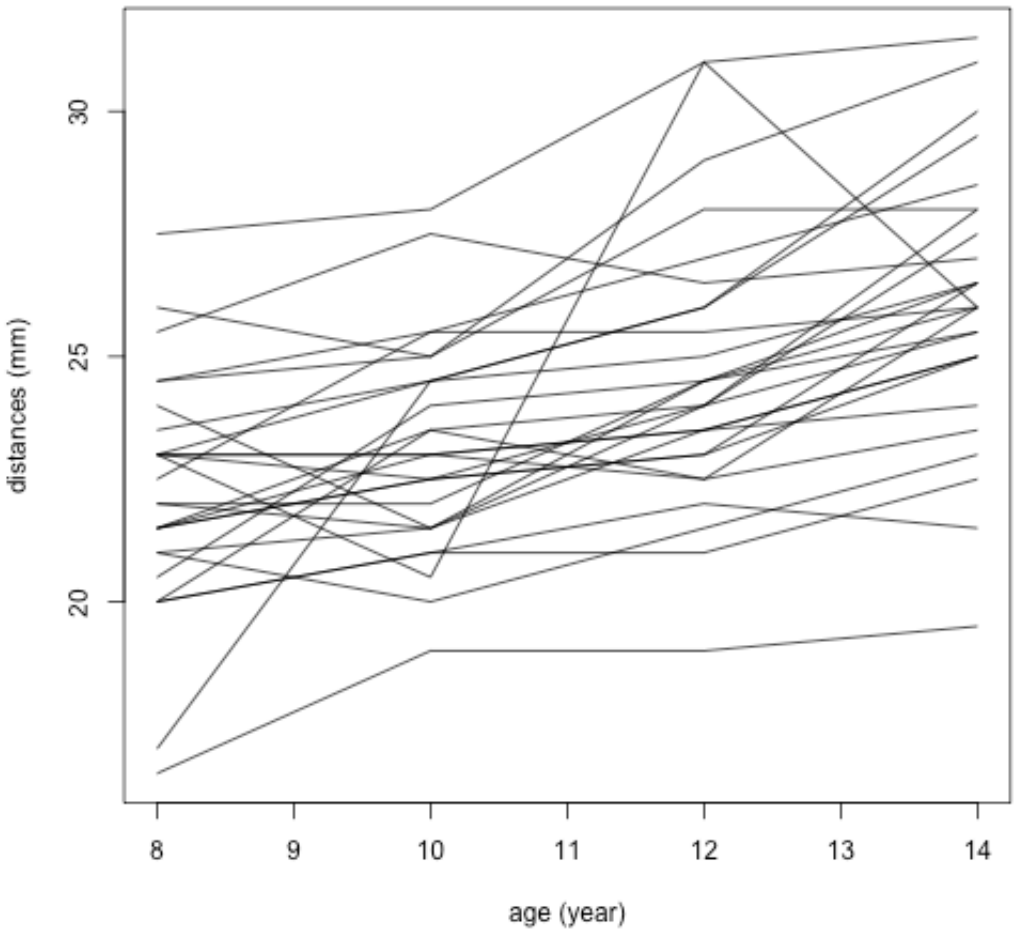


figure: Matrix graph showing within-subject changes

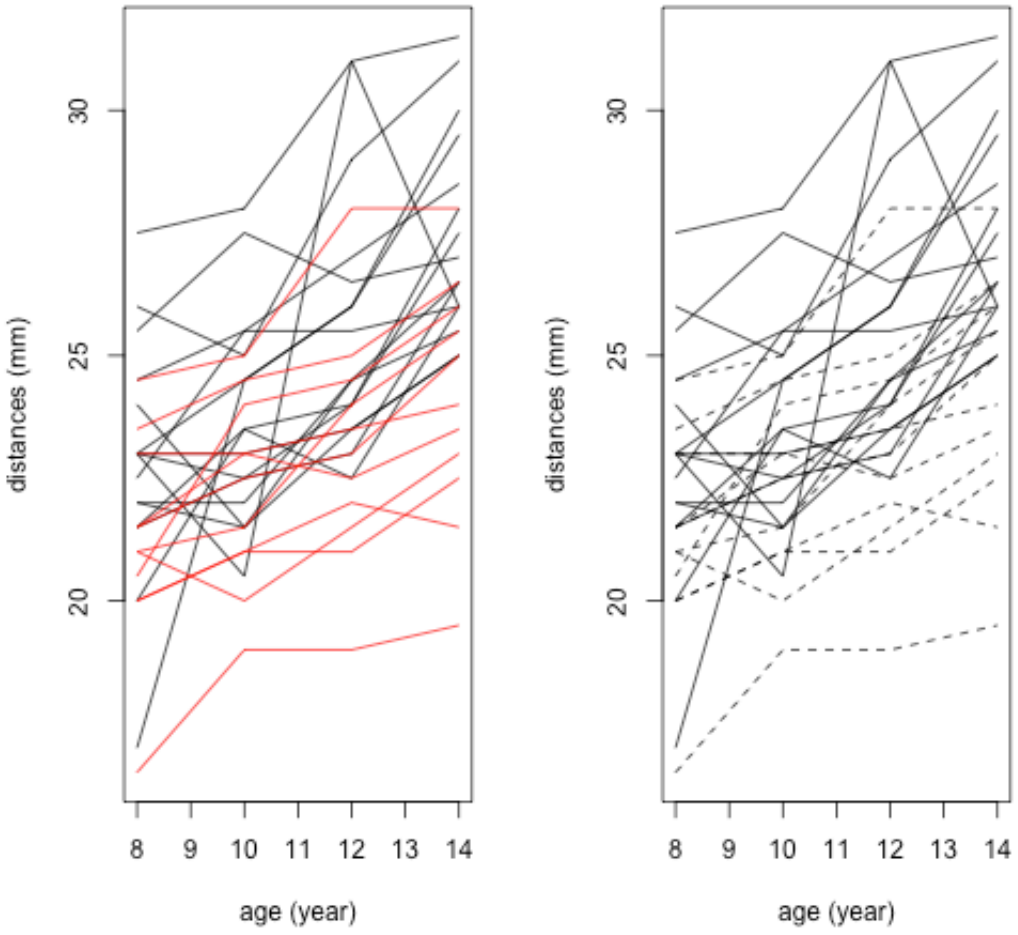
`matplot()`은 각각의 변화를 한눈에 구분하는 것을 목표로 제작된 함수이므로 `col = 1, lty = 1` 인수를 넣지 않으면 총천연색으로 그림을 그리는 것이 디폴트이다. 일단은 디폴트를 억제하면서 색깔은 검정(1), 선의 형태를 실선(1)으로 두었다.

차원 증강

위의 행렬그래프에 남녀 구분 없이 피험자 별로만 선을 그렸으므로 여기에 성별의 차이를 가미하면 차원이 증강된다. 아직까지는 단색의 선그래프이므로 색상으로 성별을 구분하거나 선의 형태로 성별을 구분하는 두 가지 형식을 대별하였다.

```
> par(mfrow = c(1, 2))
> matplot(x, y, type = "l",
+        col = c(rep(1, 16), rep(2, 11)),         # 1 = black, 2 = re
d
+        lty = 1,
```

```
+       ylab = "distances (mm)",
+       xlab = "age (year)")
> matplot(x, y, type = "l",
+       col = 1,
+       lty =  c(rep(1, 16), rep(2, 11)),       # 1 = solid, 2 = br
oken
+       ylab = "distances (mm)",
+       xlab = "age (year)")
```



Matrx graph with dimensional enforcement

전달한 인수의 차이와 보이는 결과의 차이를 어렵지 않게 파악할 수 있다. 그래프를 그리기 전에 설정한 `par(mfrow = ...)` 명령은 (앞에서도 잠깐 등장했었는데) 범용 그래프 인수를 바꾸는 구실을 해서 여기에서는 그래프가 놓이는 위치를 행렬처럼 지정한다. 즉, `c(1, 2)`는 1행 곱하기 2열 평면에 그래프를 그리라는 뜻으로서 첫 그래프가 왼쪽에 놓이고 둘째 그래프가 오른쪽에 놓인다. 이 인수의 값을 `c(2, 1)`로 바꾸었을 때 어떤 변화가 생길지 예측하여 보라.

범주척도 대 범주척도

설명변수와 응답변수가 모두 범주척도라면 (그릴 수 있는 방법이 없는 것은 아니지만) 그래프로 묘사하기에 적당하지 않다. 표를 그리는 것이 어울린다. nycflights 자료모음집에서 출발지연 여부를(`dep_delay > 0`) 항공사별로(`carrier`) 집계하려면 다음과 같은 표 하나면 충분하다.

```
> # load("nycflights.RData") # Its location depends on you.
> ( t <- table(flights$carrier, flights$dep_delay > 0) )
```

```
##
##      FALSE  TRUE
##   9E 10353  7063
##   AA 21931 10162
##   AS   486   226
##   B6 32724 21445
##   DL 32520 15241
##   EV 28217 23139
##   F9   341   341
##   FL  1533  1654
##   HA   273    69
##   MQ 17132  8031
##   OO    20     9
##   UA 30718 27261
##   US 15098  4775
##   VX  2906  2225
##   WN  5525  6558
##   YV   312   233
```

읽기 편하게 하기 위해서 위에 더해서 통상 적용하는 약간의 부가사항들을 가미하면 다음처럼 바뀐다:

```
> t <- cbind(t, apply(t, 1, sum))
> t <- cbind(t, apply(t, 1, function(x) round( x[2]/x[3] * 100, 1 )
) ) #
> colnames(t) <- c("On-time", "Delayed", "Sum", "Delayed (%)")
> t
```

```
##    On-time Delayed   Sum Delayed (%)
## 9E   10353    7063 17416        40.6
## AA   21931   10162 32093        31.7
## AS     486     226   712        31.7
## B6   32724   21445 54169        39.6
## DL   32520   15241 47761        31.9
## EV   28217   23139 51356        45.1
## F9     341     341   682        50.0
## FL    1533    1654  3187        51.9
```

```
## HA     273      69   342        20.2
## MQ   17132    8031 25163        31.9
## OO      20       9    29        31.0
## UA   30718   27261 57979        47.0
## US   15098    4775 19873        24.0
## VX    2906    2225  5131        43.4
## WN    5525    6558 12083        54.3
## YV     312     233   545        42.8
```

마지막으로 항공사별로 총 이륙 횟수에 따라 정렬하면 읽는 사람들에게 편리할 것이다.

```
> t[order(t[, 3], decreasing = TRUE),  ]
```

```
##    On-time Delayed   Sum Delayed (%)
## UA   30718   27261 57979        47.0
## B6   32724   21445 54169        39.6
## EV   28217   23139 51356        45.1
## DL   32520   15241 47761        31.9
## AA   21931   10162 32093        31.7
## MQ   17132    8031 25163        31.9
## US   15098    4775 19873        24.0
## 9E   10353    7063 17416        40.6
## WN    5525    6558 12083        54.3
## VX    2906    2225  5131        43.4
## FL    1533    1654  3187        51.9
## AS     486     226   712        31.7
## F9     341     341   682        50.0
## YV     312     233   545        42.8
## HA     273      69   342        20.2
## OO      20       9    29        31.0
```

표를 통해 알 수 있듯이 총 이륙의 수가 많다고 해서 출발 지연이 많은 것은 아닌 것처럼 보인다. 여기까지 가공했더니 자료에 대한 산포도(간격척도 대 간격척도!)를 그리는 것이 가능해졌다. 동그라미 대신에 항공사 기호를 찍어 보았다. 첫 행의 `plot()` 함수를 쓰면서 마치 그림을 그리는 것처럼 모든 인수를 다 제공하면서도 마지막의 인수 `type = "n"`을 통해 실제로는 점이 찍히는 것을 억제하였다. 같은 좌표 위에 `text()` 함수를 써서 항공사 이름을 지정한 레이블을 찍도록 우회한 방식이다. 방대한 그래프에서 외톨이를 찾을 때 유용하다.

```
> plot(t[, 4] ~ t[, 3], xlab = "Total number of departure",
+       ylab = "Number of delayed departure",
+       type = "n") #
> text(t[, 3], t[, 4], lab = rownames(t))
> abline(lm(t[, 4] ~ t[, 3]), lty = 2)
```

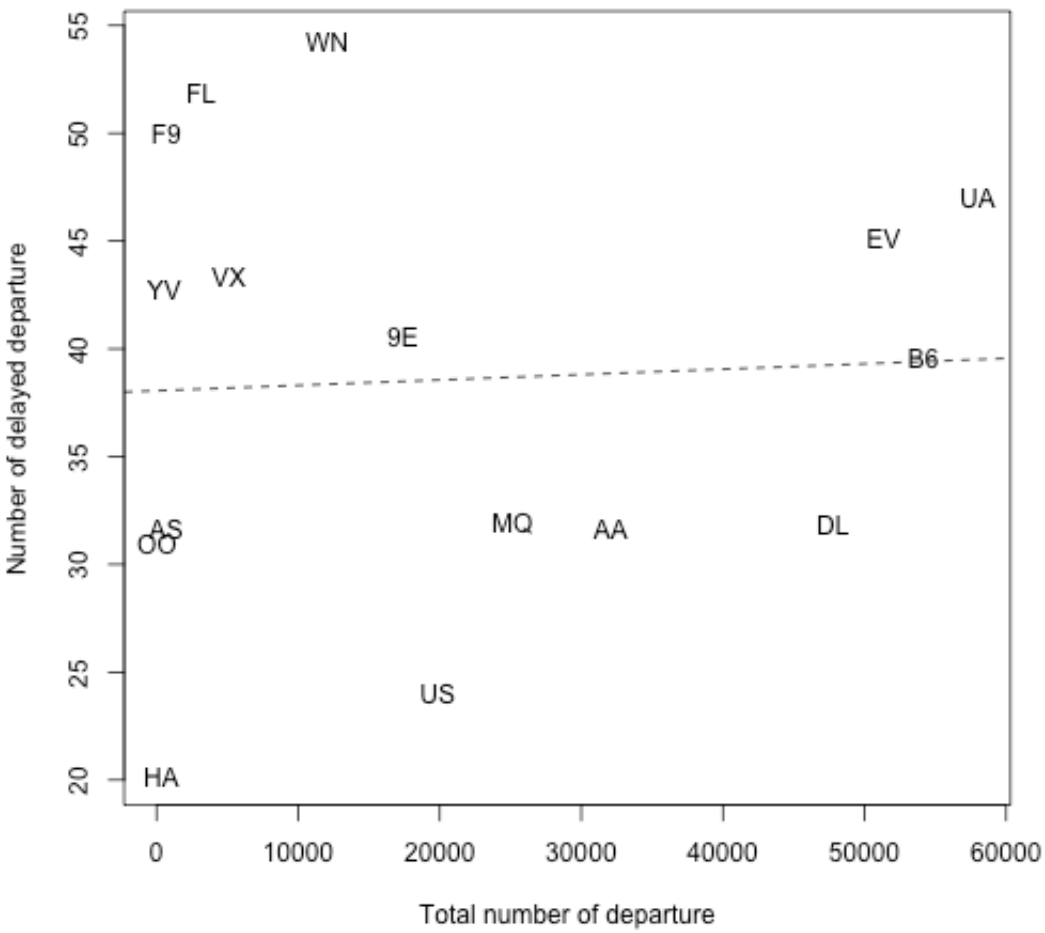


figure: Scattergraph for manupulated dataset

회귀 결과인 직선의 기울기가 0에 가까운 것으로 미루어 두 변수 사이의 연관은 낮아 보인다.

Bliss 자료

Bliss는 농도에 따른(0 ~ 4) 살충효과를 관찰하여 농도-치명률을 1935년에 발표하였다(Bliss. The calculation of the dosage-mortality curve. Ann Appl Biol 1935; 22: 134–67). 간단한 표이므로 손으로 입력해 본다. 자료철의 이름은 원문의 연구자 이름을 따서 bliss로 이름 붙였다:

```
> ( bliss <- data.frame(
+       dead = c(2, 8, 15, 23, 27),
+       alive = c(28, 22, 15, 7, 3),
+       conc = 0:4) )
```

```
##   dead alive conc
## 1    2    28    0
## 2    8    22    1
## 3   15    15    2
## 4   23     7    3
```

```
## 5    27      3     4
```

농도 1에서는 30마리의 해충 중에서 8마리가 죽었지만 농도 4에서는 30마리 중에서 27마리가 죽었다. 이 자료는 위와 같은 형태의 표로 작성하여 퍼센트를 계산해서 첨부해되 되지만 굳이 그래프로 그리려면 다음처럼 할 수도 있다. `barplot()` 함수인데 이 함수는 디폴트로만 그리면 내용을 식별하기 어려워서 처음 그릴 때부터 인수를 일일히 지정하는 것이 좋다.

```
> barplot(rbind(bliss$alive, bliss$dead), ylim = c(0, 40),
+         xlab = "Concentration of Insecticide",
+         ylab = "dead or alive (n)",
+         names.arg = 0:4,
+         legend.text = c("alive", "dead"))
```

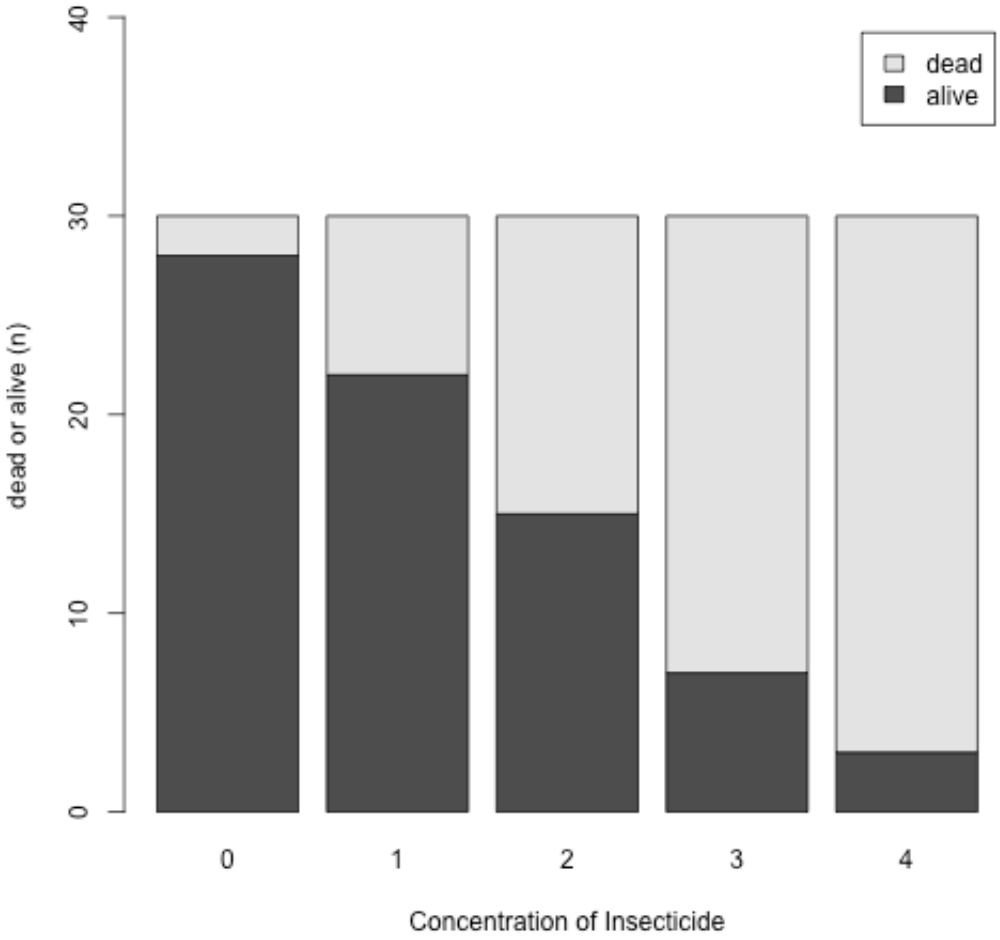


figure: Stacked bargraph for two categorical data

그리기는 그렸으되 차지하고 있는 지면에 비해서 우리에게 말해주는 내용은 저 앞에 그린 표에 비해서 얄팍하기 그지없다. 근래에는 아래처럼 그리는 그래프도(figure: Dead/Alive Scatterplot only) 유행하는 것 같다. 이 그래프가 말하는 것은 어렵지 않지만 bliss 자료철로부터 저 그래프를 그리기까지에는 상당한 수준의 개념 추상화와 자료의 변환능력이 요구된다. 각 농도 x를 0부터 4로 두고 사망여부 y에는 0과 1로 두는

150행짜리 긴 테이블로 바꾸는 것이다. 원자료가 짧기 때문에 손으로 직접 숫자를 넣어서 만들어도 무난하지만 코딩 시간이므로 코딩으로 해결해 보자. 다음부터 좀 더 축약할 수 있지만 더 줄이면 이해하기가 어려워진다:

```
> DEAD <- 1
> ALIVE <- 0
>
> x <- rep(0:4, each = 30)
> m <- bliss[, 1:2]
> y <- as.vector( apply(m, 1, function(x) c( rep(DEAD, x[1]), rep(A
LIVE, x[2]) ) ) )
> plot(jitter(x), jitter(y))
```

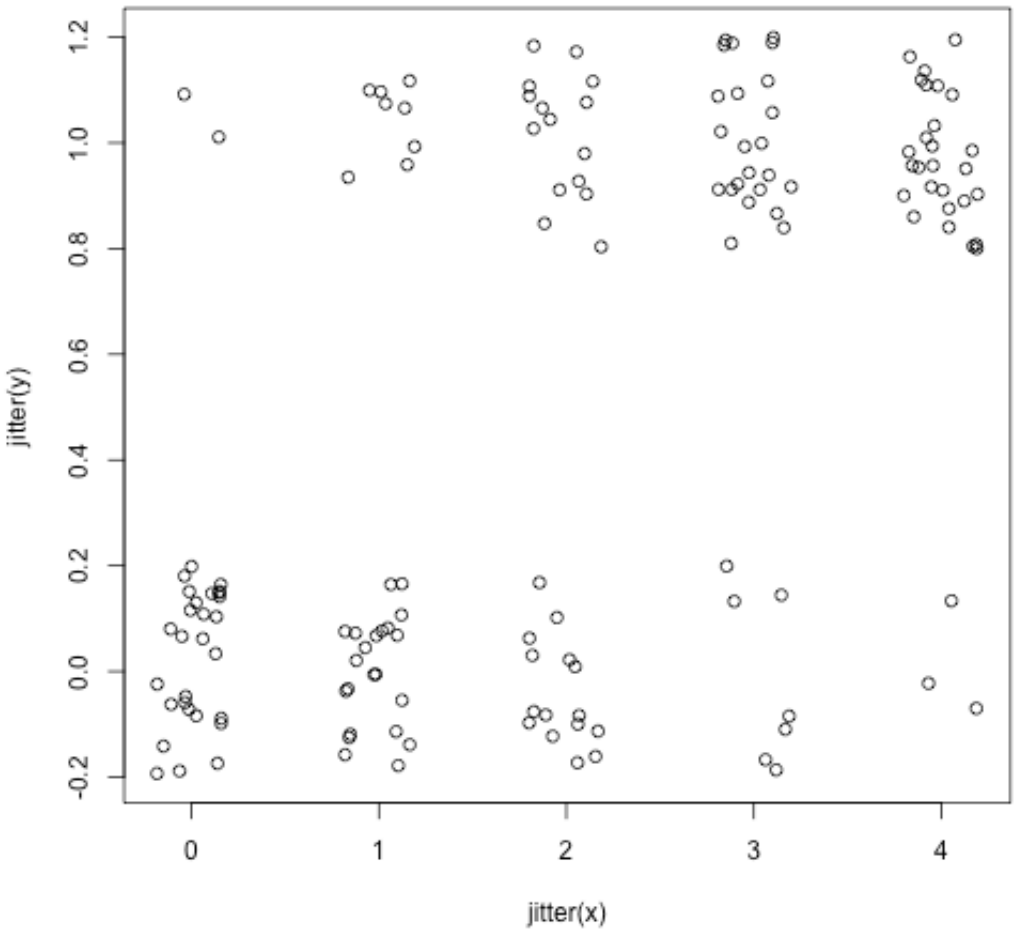


figure: Dead/Alive Scatterplot only

`plot()` 함수의 디폴트로 그린 위 그래프를 보았을 때 이 자료를 잘 알고 있는 우리에게는 무슨 뜻인지 와닿지만 과학 그래프를 이런 식으로 출판해서는 곤란한다. R의 그래프 변환 기능을 최소로 동원하여 모양을 바꾸어 보았다. 각 행의 인수들에는 설명을 붙였다:

```
> plot(0, 0,                                          # 좌표 0, 0에
 점을 찍는다
```

```
+       type = "n",                                  # 유형은 "그리지 않기(no
ne)"
+       ylim = c(-0.2, 1.2),                 # 세로축의 범위는 -0.2부터 1.2까지
 (위의 그래프로부터 따옴)
+       xlim = c(-0.5, 4.5),                 # 가로축의 범위
+       axes = FALSE,                                # 디폴트 축도 그리지 않는
다
+       xlab = "Concentration of Insecticide",
+       ylab = "")                                                  # 가로축 이름은
 ... 세로축 이름은 비운다.
> points(jitter(x), jitter(y))  # x, y 좌표에 동그라미를 그리되 좌표를 흔들어
서 그린다.
> axis(1, at = 0:4, label = 0:4)
       # 축 1 (아래)을 0부터 4까지(at =) 생성하고 이름을 붙인다.
> axis(2, at = 0:1, label = c("Alive", "Dead")) # 축 2 (왼쪽)를 생성하고
 이름을 붙인다.
> axis(3, at = 0:4, label = 0:4)
       # 축 3 (위)을 생성하고 이름을 붙인다.
```

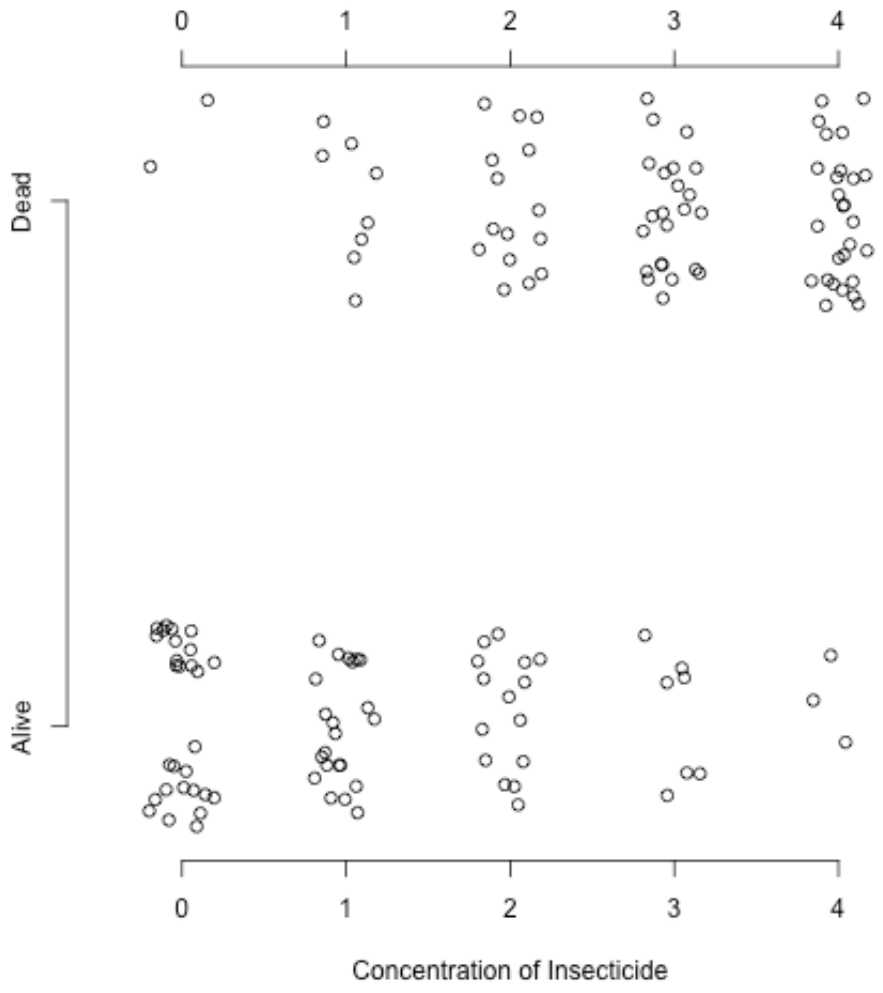


figure: Dead/Alive by Concentration of Insecticide

가로축은 간격척도로 측정된 농도였지만 연구에 적용된 농도를 5개의 절대구획으로 축소시킬 수 있으므로 범주척도와 다를 바 없으며 세로축은 사망/생존으로 측정된 집계자료이므로 자연스럽게 범주척도로 취급되었다. 위 그래프를 그릴 때 적용했던 기법에서 코드 외에도 주목할 핵심이 있다:

- 원재료로부터 가공된 자료로부터 그려졌다. 많은 경우에 원자료의 형식에 구애되지 않은 채 그래프로 묘사할 수 있어야 한다.
- 같은 좌표에 점이 연속으로 찍히는 경우에 자료의 정밀성을 훼손하지 않는 범위에서는 좌표를 흔들어서 찍으면 알아보기 좋다.
- R이라는 작업 환경 안에서 마주치는 모든 대상이 객체였던 것처럼 R로 작성한 그래프도 객체(object)이어서 한 객체는 여러 개의 객체로 이루어져 있다. 필요에 따라 모든 객체에 프로그래머가 원하는 값을 대입함으로써 원하는 최종 형태를 만들 수 있다.

값을 흔들 때 사용하는 함수 `jitter()`는 본디 자료에 정규 에러를 더할 때 사용하는 함수이다. 흔들리는 정도는 실행할 때마다 다르게 나오며 이 폭을 강제하기 위해서 `amount =` 인수를 조정할 수 있다.

```
> jitter(100)
```

```
## [1] 98.78621
```

```
> jitter(100)
```

```
## [1] 98.9074
```

```
> jitter(100, amount = 5)
```

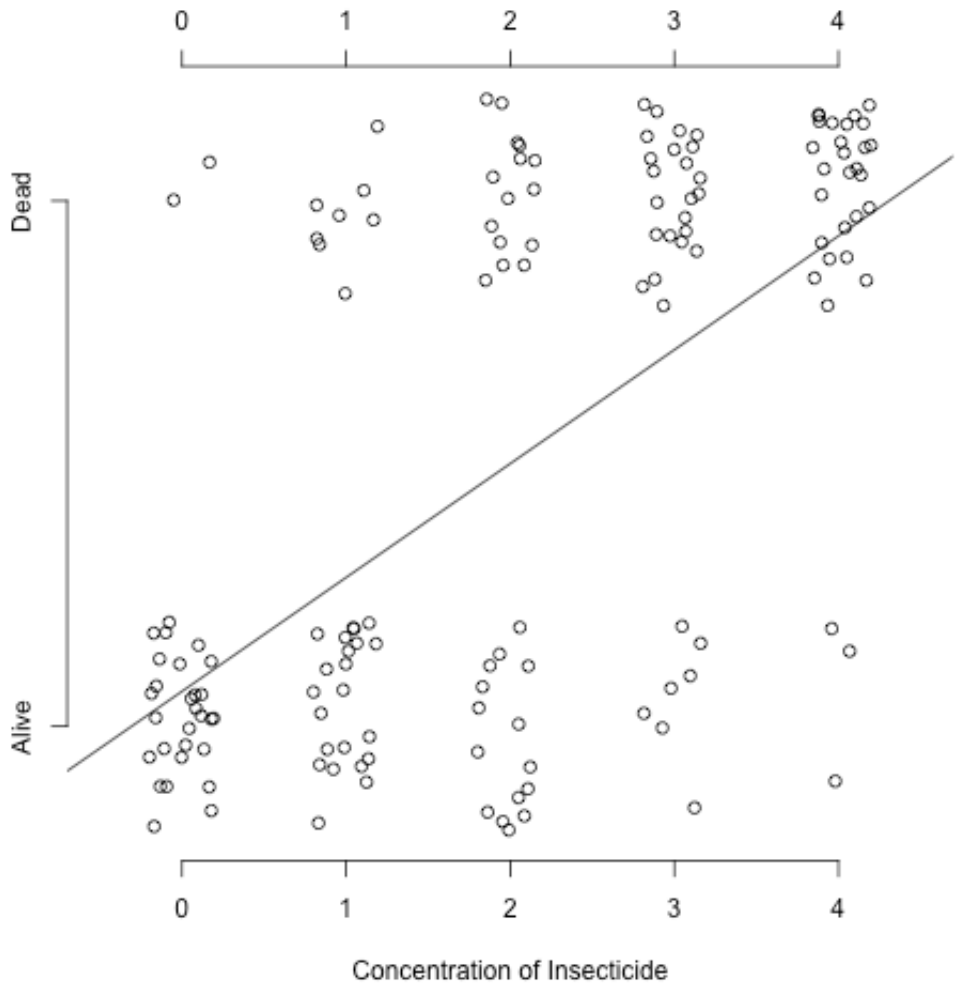
```
## [1] 98.83241
```

남은 지면

마지막에 응용한 bliss 자료철은 실제로 측정된 연구결과이며 통계학계에서는 흔히 로지스틱회귀분석을 연습할 때 모범 삼아 거론되는 자료이기도 하다. 이 자료를 흩뜨리고 산포도처럼 그려 놓고 맨 앞에서 했던 것처럼 선형 회귀직선을 그으면 다음처럼 보일 것이다(이런 분석은 엉터리이다):

```
> plot(0, 0,
+       type = "n",
+       ylim = c(-0.2, 1.2),
+       xlim = c(-0.5, 4.5),
+       axes = FALSE,
+       xlab = "Concentration of Insecticide",
```

```
+       ylab = "")
> points(jitter(x), jitter(y))
> axis(1, at = 0:4, label = 0:4)
> axis(2, at = 0:1, label = c("Alive", "Dead"))
> axis(3, at = 0:4, label = 0:4)
> abline(lm(y ~ x))
```



농도의 증가에 대해서 살충력이 증가하는 것처럼 보이는 직선이 그려졌지만 두 변수 사이의 관계를 일견하는 데 편리하다는 이상의 가치는 없다. 두 가지 질문에 답할 수 없다:

- 생존이 0이고 사망이 1이라고 코딩을 했었는데, 숫자와 의미 사이에는 아무 합리적 근거가 없다.
- 따라서 농도 2 부근에서 추정되는 응답으로 0.5쯤의 결과가 추정되지만 0과 1 외에는 의미를 부여하기 어렵다. 같은 이유로 농도를 (연구 범위보다) 더 높게 잡았을 때 계산되는 응답의 추정값인 dead 초과의 숫자에 부여할 수 있는 합리적인 이름이 존재하지 않는다(더 죽음?).

이분형 응답(0/1)을 일반화하여 설명변수의 증감에 따라 응답이 일어나는 확률분포를 다음의 복잡한 수식으로 묘사하는

$$expected\ response = logit^{-1} = \frac{1}{1 + e^{-(ax+b)}}$$

방법이 있다. x에는 설명변수(농도)를 넣고 a와 b 자리에는 다음의 glm() 결과로부터 얻은 계수를 넣으면 된다.

```
> ( bliss.glm <- glm(cbind(dead, alive) ~ conc, bliss, family = bin
omial) )
```

```
##
## Call:  glm(formula = cbind(dead, alive) ~ conc, family = binomia
l, data = bliss)
##
## Coefficients:
## (Intercept)         conc
##      -2.324        1.162
##
## Degrees of Freedom: 4 Total (i.e. Null);  3 Residual
## Null Deviance:           64.76
## Residual Deviance: 0.3787    AIC: 20.85
```

a 자리에 1.162, b 자리에 −2.324를 넣고 변수 x에 따라 반응을 돌려주는 함수(전통적으로 inverse logit 함수라고 부른다)를 만들면:

```
> ilogit <- function(x, a = 1.162, b = -2.324) return(1/(1 + exp(-(
a * x + b))) )
> ilogit(0:4)
```

```
## [1] 0.0891547 0.2383041 0.5000000 0.7616959 0.9108453
```

이 값을 (0:4가 아니라 조금 세분해서) 위의 그래프에 근사한 곡선이 더해진 모양이 완성되었다. 약리학에서 등장하는 S자형 약물 농도/반응 예측곡선에 실제 관찰된 점을 더해서 묘사된 형태로서 텍스트와 논문 등에서 자주 접할 수 있다.

```
> plot(0, 0,
+       type = "n",
+       ylim = c(-0.2, 1.2),
+       xlim = c(-0.5, 4.5),
+       axes = FALSE,
+       xlab = "Concentration of Insecticide",
+       ylab = "Expected probability of death")

> points(jitter(x), jitter(y, 0.3))                                  # 예
측곡선과 충돌하는 것을 줄이고자 세로쪽 jitter를 감소시켰다.
> axis(1, at = 0:4, label = 0:4)
```

```
> axis(2, at = 0:1, label = c("Alive", "Dead"))
> axis(3, at = 0:4, label = 0:4)
> x1 <- seq(0, 4, 0.1)
        # 곡선을 부드럽게 만들고자 간격을 좁혔다
> points(x1, ilogit(x1), type = "l", lwd = 2)    # 좌표 x1과 ilogit(x1
)에 곡선. 굵기는 두 배로
```

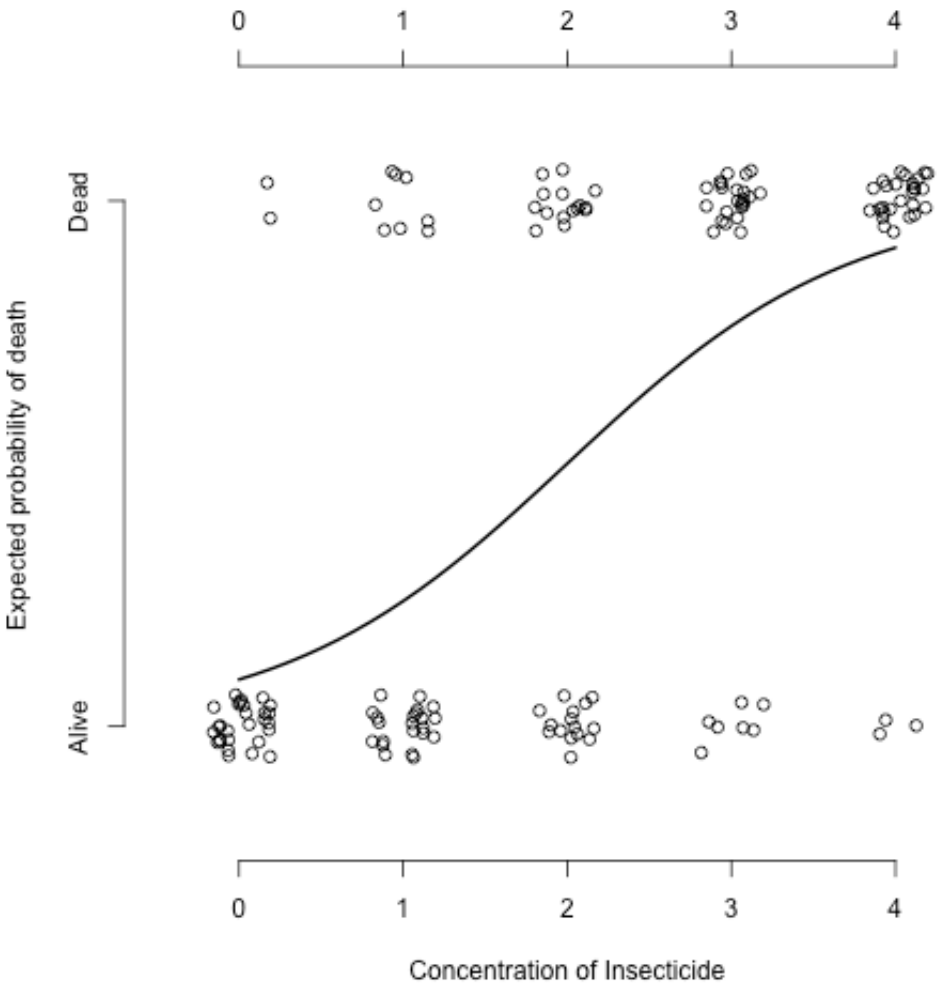


figure: S-shaped dose-death relationship with observed data points