

Student 3000 Criminal Data : ggplot

coop711

2025-10-08

Working Data Loading

```
library(magrittr)
load("./crimtab.RData")
ls()

## [1] "crimtab_2"          "crimtab_df"        "crimtab_long"      "crimtab_long_df"

ls.str()

## crimtab_2 : 'table' int [1:42, 1:22] 0 0 0 0 0 0 1 0 0 0 ...
## crimtab_df : 'data.frame': 924 obs. of 3 variables:
##   $ finger: num  9.4 9.5 9.6 9.7 9.8 9.9 10 10.1 10.2 10.3 ...
##   $ height: num  56 56 56 56 56 56 56 56 56 56 ...
##   $ Freq  : int  0 0 0 0 0 0 1 0 0 0 ...
## crimtab_long : num [1:3000, 1:2] 10 10.3 9.9 10.2 10.2 10.3 10.4 10.7 10 10.1 ...
## crimtab_long_df : 'data.frame': 3000 obs. of 2 variables:
##   $ finger: num  10 10.3 9.9 10.2 10.2 10.3 10.4 10.7 10 10.1 ...
##   $ height: num  56 57 58 58 58 58 58 59 59 ...

head(crimtab_long_df)

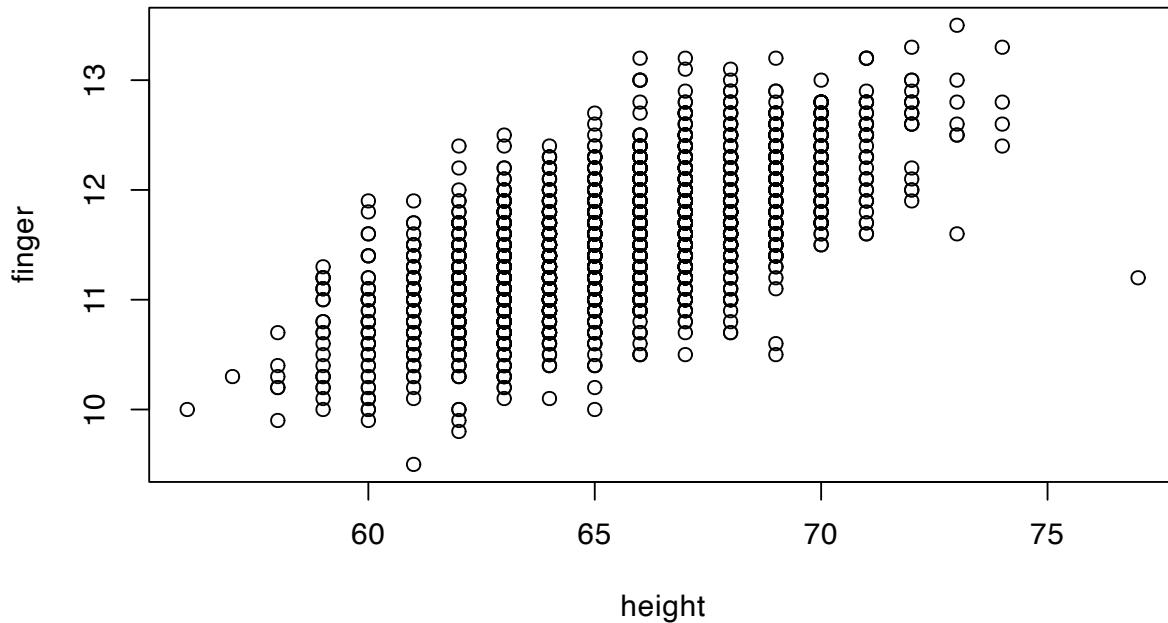
##   finger height
## 1    10.0     56
## 2    10.3     57
## 3     9.9     58
## 4    10.2     58
## 5    10.2     58
## 6    10.3     58
```

Graphic Representation

Base Graphics

- 키와 손가락길이의 산점도

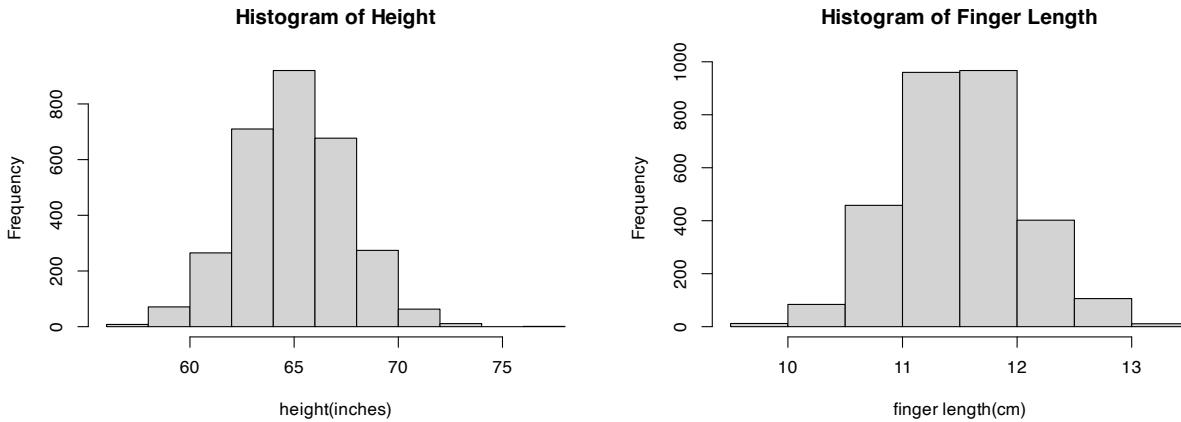
```
# plot(finger ~ height, data = crimtab_long_df)
crimtab_long_df[2:1] %>%
  plot
```



```
plot(crimtab_long_df[, 2:1])
```

- 변수 각각의 히스토그램은?

```
par(mfrow = c(1, 2))
hist(crimtab_long_df$height,
     main = "Histogram of Height",
     xlab = "height(inches)")
hist(crimtab_long_df$finger,
     main = "Histogram of Finger Length",
     xlab = "finger length(cm)")
```



```
# hist(crimtab_long_df["height"],
#       main="Histogram of Height",
#       xlab="height(inches)")
# hist(crimtab_long_df["finger"],
#       main="Histogram of Finger Length",
#       xlab= "finger length(cm)")
```

- 평균과 표준편차를 한번에 구하려면 다음과 같이 anonymous function을 작성하고 mapply() 또는 sapply()를 이용하는 게 편함. 이를 모수로 하는 정규곡선을 덧씌워 볼 것.
 - mean_sd()도 anonymous function으로 평균과 표준편차를 계산해서 출력하는 함수임. 이와 같은 함수를 저장해 놓으려면 dump()를 이용함.
 - 이와 같이 계산한 평균과 표준편차를 모수로 하는 정규곡선을 덧씌워 볼 것.

```
mean_sd <- function(x) {
  mean <- mean(x, na.rm = TRUE)
  sd <- sd(x)
  c(mean = mean, sd = sd)
}
dump("mean_sd", file = "mean_sd.R")
```

```
# crimtab_long_df %>%
#   sapply(FUN = mean_sd)
crimtab_stat <- sapply(crimtab_long_df, FUN = mean_sd)
# crimtab_stat <- mapply(mean_sd, crimtab_long_df)
# apply(crimtab_long, 2, mean)
# apply(crimtab_long, 2, sd)
str(crimtab_stat)
```

```
##  num [1:2, 1:2] 11.547 0.549 65.473 2.558
##  - attr(*, "dimnames")=List of 2
##    ..$ : chr [1:2] "mean" "sd"
##    ..$ : chr [1:2] "finger" "height"
```

- crimtab_stat의 어떤 성격을 갖는지 다음 질문과 추출 작업을 통해서 알아보자.

```
is.matrix(crimtab_stat)

## [1] TRUE

is.table(crimtab_stat)

## [1] FALSE

is.list(crimtab_stat)

## [1] FALSE

is.data.frame(crimtab_stat)

## [1] FALSE

crimtab_stat[, 1]

##      mean      sd
## 11.5473667 0.5487137

crimtab_stat[, "finger"]

##      mean      sd
## 11.5473667 0.5487137

crimtab_stat[, "finger"] [1]

##      mean
## 11.54737

crimtab_stat[, "finger"] [[1]]

## [1] 11.54737

crimtab_stat[1]

## [1] 11.54737

crimtab_stat[2:3]

## [1] 0.5487137 65.4730000
```

```
# crimtab_stat["finger"]
# crimtab_stat$finger
```

matrix 을 data frame 으로 변환하면

```
(crimtab_stat_df <- data.frame(crimtab_stat))
```

```
##          finger    height
##  mean  11.5473667 65.473000
##  sd    0.5487137  2.557757
```

```
is.matrix(crimtab_stat_df)
```

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

```
is.table(crimtab_stat_df)
```

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

```
is.list(crimtab_stat_df)
```

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

```
is.data.frame(crimtab_stat_df)
```

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

```
crimtab_stat_df[, 1]
```

```
## [1] 11.5473667 0.5487137
```

```
str(crimtab_stat_df[, 1])
```

```
## num [1:2] 11.547 0.549
```

```
crimtab_stat_df[, "finger"]
```

```
## [1] 11.5473667 0.5487137
```

```
str(crimtab_stat_df[, "finger"])
```

```
## num [1:2] 11.547 0.549
```

```
crimtab_stat_df[, "finger"] [1]
```

```
## [1] 11.54737
```

```

str(crimtab_stat_df[, "finger"] [1])

## num 11.5

crimtab_stat_df[, "finger"] [[1]]

## [1] 11.54737

str(crimtab_stat_df[, "finger"] [[1]]))

## num 11.5

crimtab_stat_df [1]

##          finger
## mean 11.5473667
## sd    0.5487137

str(crimtab_stat_df [1])

## 'data.frame':   2 obs. of  1 variable:
## $ finger: num  11.547 0.549

crimtab_stat_df ["finger"]

##          finger
## mean 11.5473667
## sd    0.5487137

str(crimtab_stat_df ["finger"])

## 'data.frame':   2 obs. of  1 variable:
## $ finger: num  11.547 0.549

crimtab_stat_df ["finger"] [1]

##          finger
## mean 11.5473667
## sd    0.5487137

str(crimtab_stat_df ["finger"] [1])

## 'data.frame':   2 obs. of  1 variable:
## $ finger: num  11.547 0.549

```

```

crimtab_stat_df["finger"][[1]]

## [1] 11.5473667 0.5487137

str(crimtab_stat_df["finger"][[1]])

## num [1:2] 11.547 0.549

crimtab_stat_df$finger

## [1] 11.5473667 0.5487137

str(crimtab_stat_df$finger)

## num [1:2] 11.547 0.549

crimtab_stat_df$finger[1]

## [1] 11.54737

str(crimtab_stat_df$finger[1])

## num 11.5

crimtab_stat_df$finger[[1]]

## [1] 11.54737

str(crimtab_stat_df$finger[[1]])

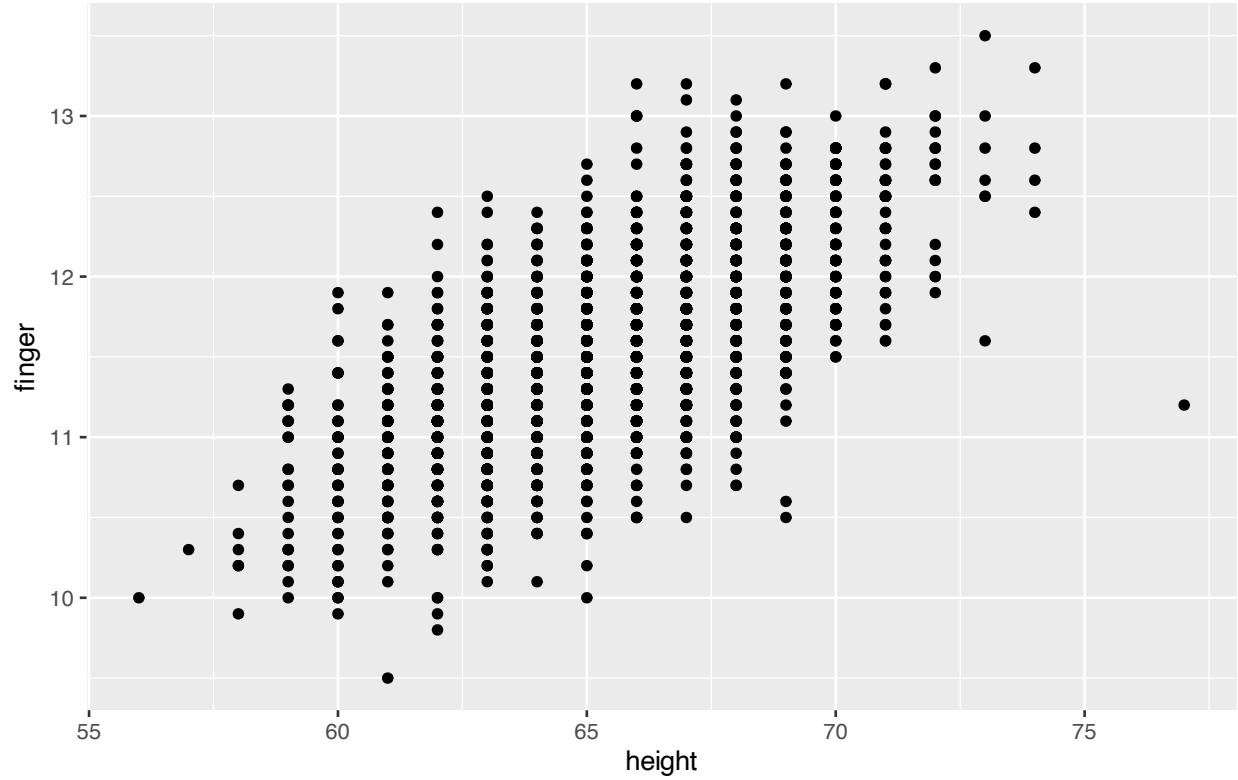
## num 11.5

ggplot

• 키와 손가락 길이의 산점도

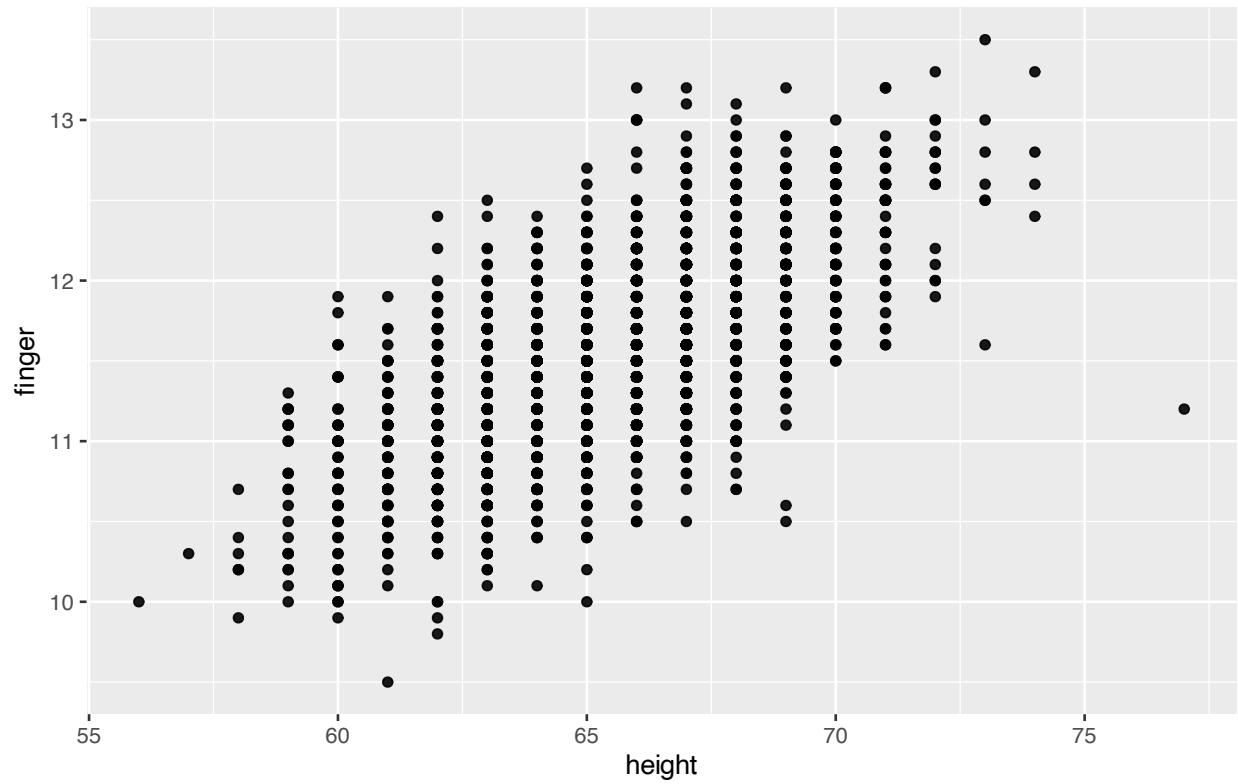
library(ggplot2)
g1 <- ggplot(data = crimtab_long_df,
             mapping = aes(x = height, y = finger))
g2 <- g1 +
  geom_point()
g2

```



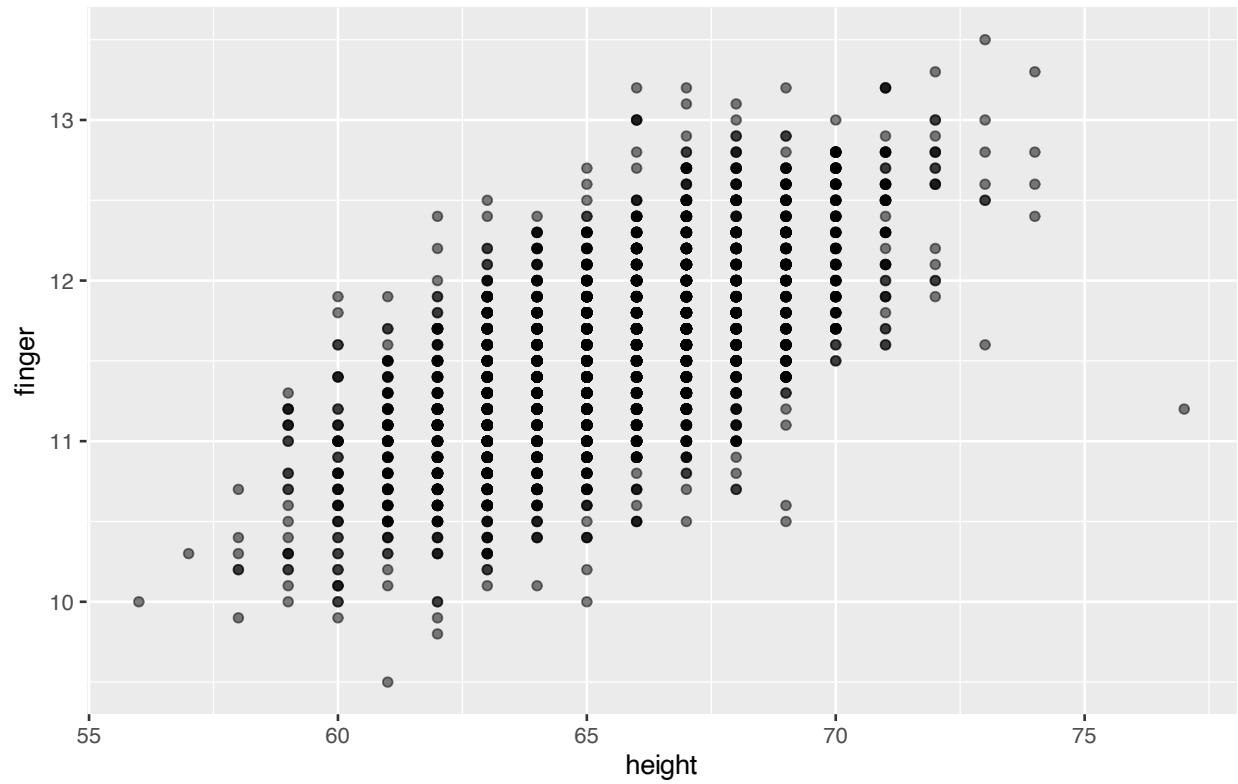
- 투명도 변경 : $\text{alpha} = 0.9$

```
g2_2 <- g1 +
  geom_point(alpha = 0.9)
g2_2
```



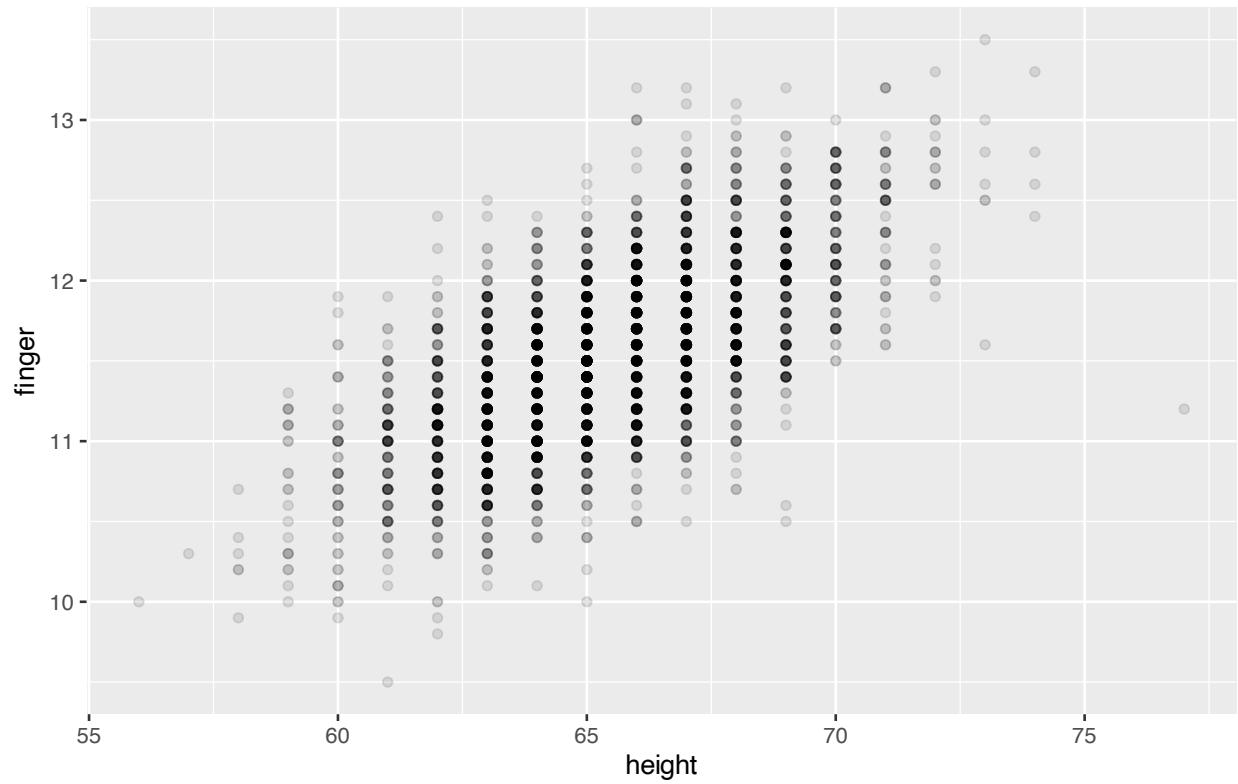
- 투명도 변경 : alpha = 0.5

```
g2_3 <- g1 +
  geom_point(alpha = 0.5)
g2_3
```



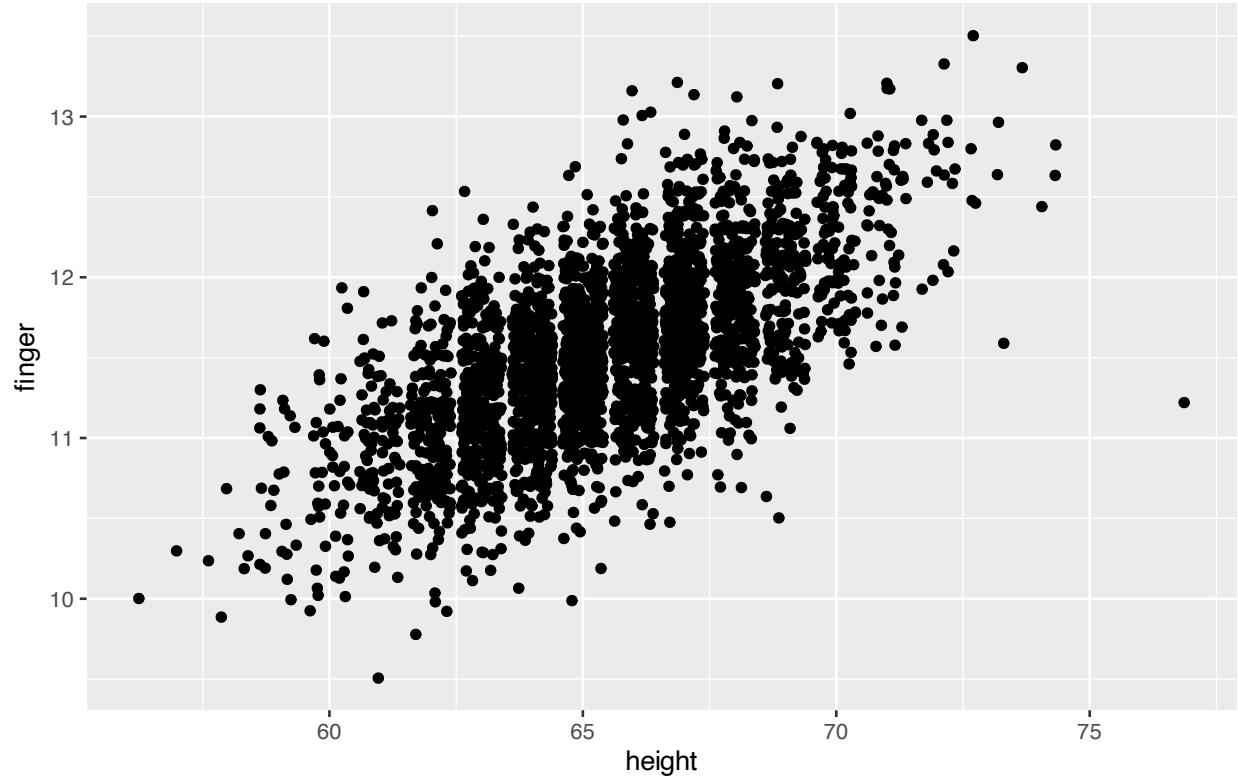
- 투명도 변경 : $\text{alpha} = 0.1$

```
g2_4 <- g1 +
  geom_point(alpha = 0.1)
g2_4
```



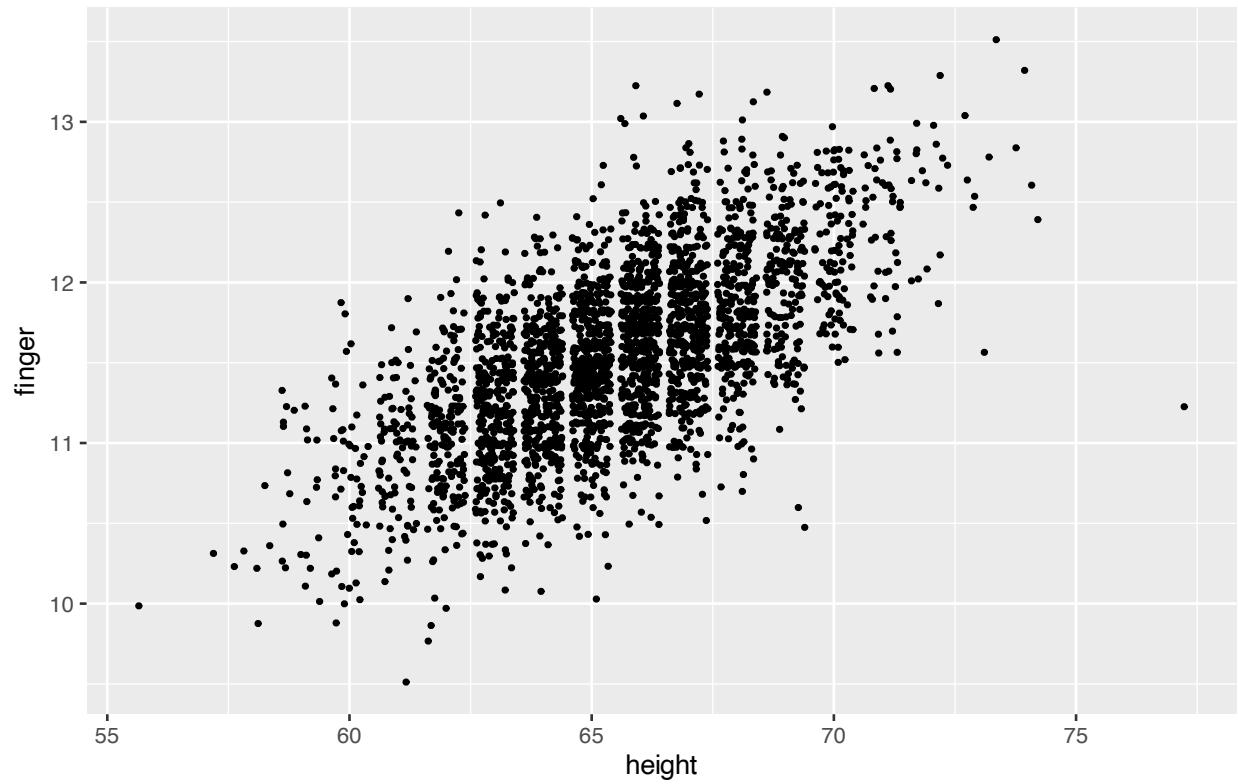
- 중복점 흐트러놓기 : position = jitter

```
g2_5 <- g1 +
  geom_point(position = "jitter")
g2_5
```



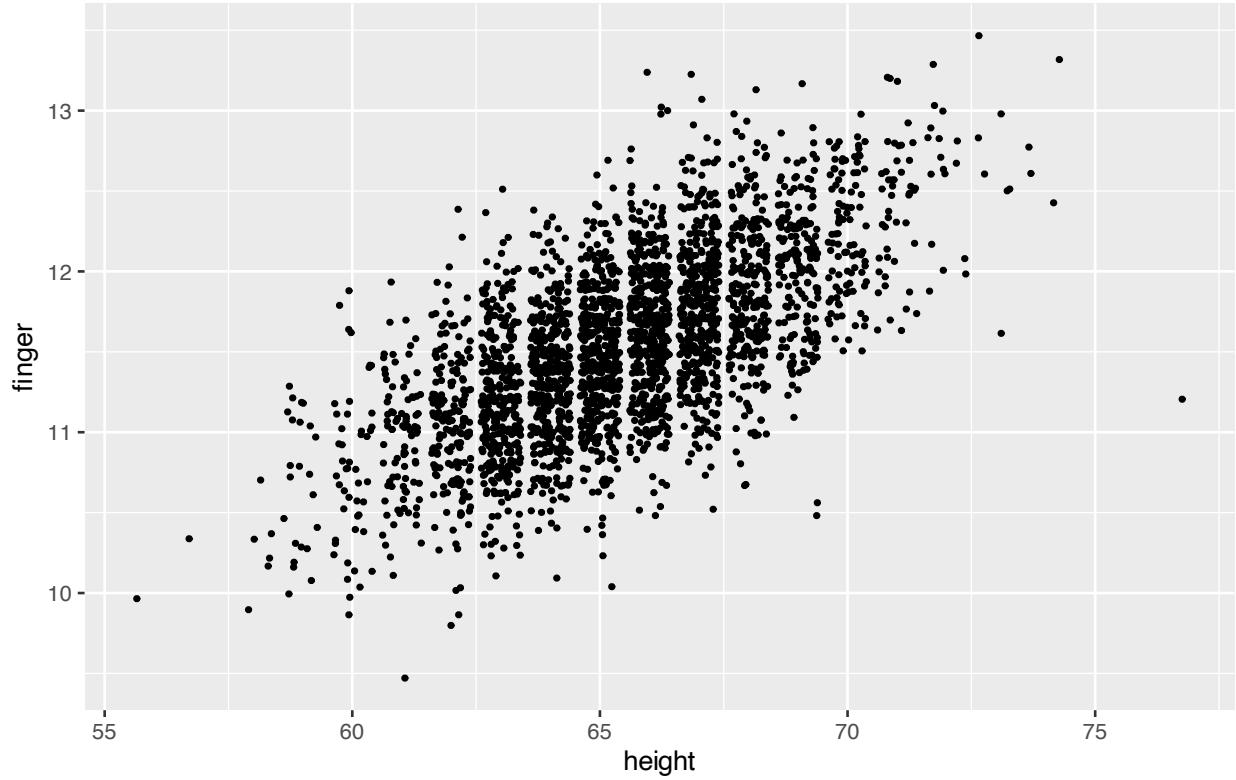
- 점의 크기를 줄이고 중복점 흐트러놓기 : position = jitter, size = 0.7

```
g2_6 <- g1 +
  geom_point(position = "jitter", size = 0.7)
g2_6
```



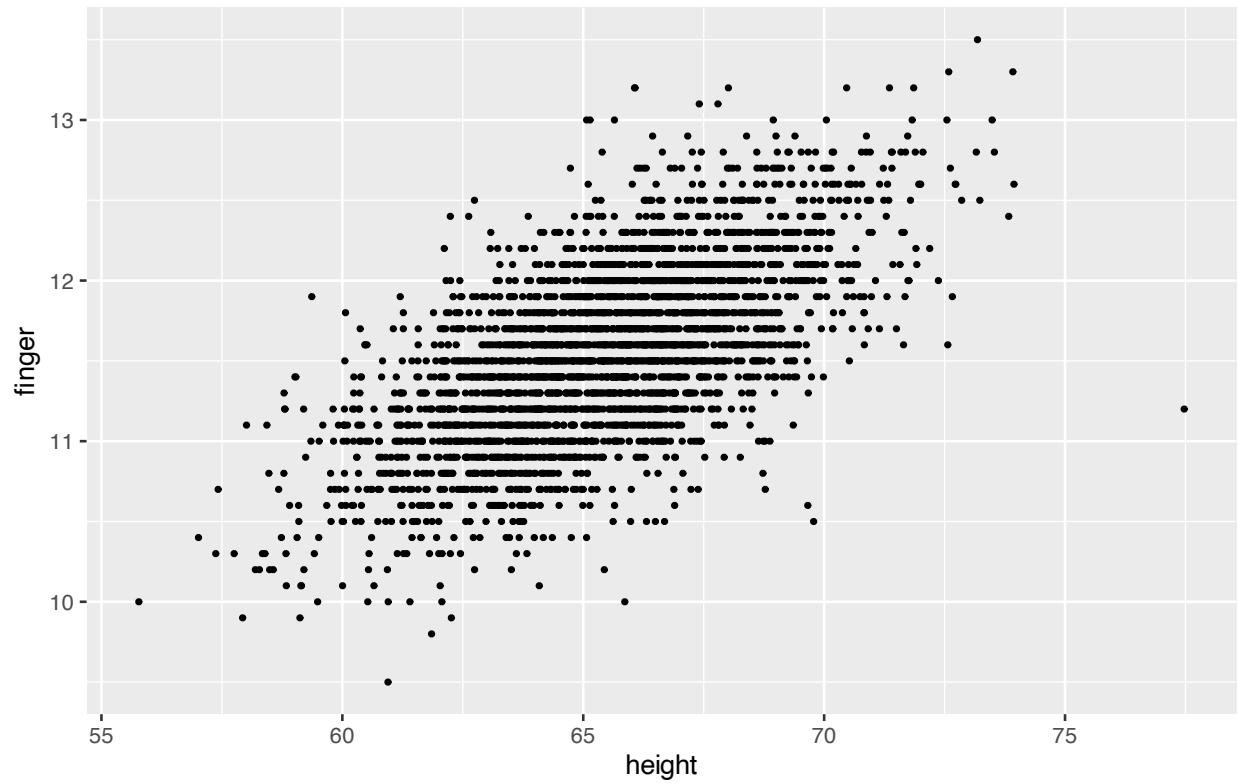
- 동일한 효과 : `position = position_jitter()`, `size = 0.7`

```
g2_7 <- g1 +
  geom_point(position = position_jitter(), size = 0.7)
g2_7
```



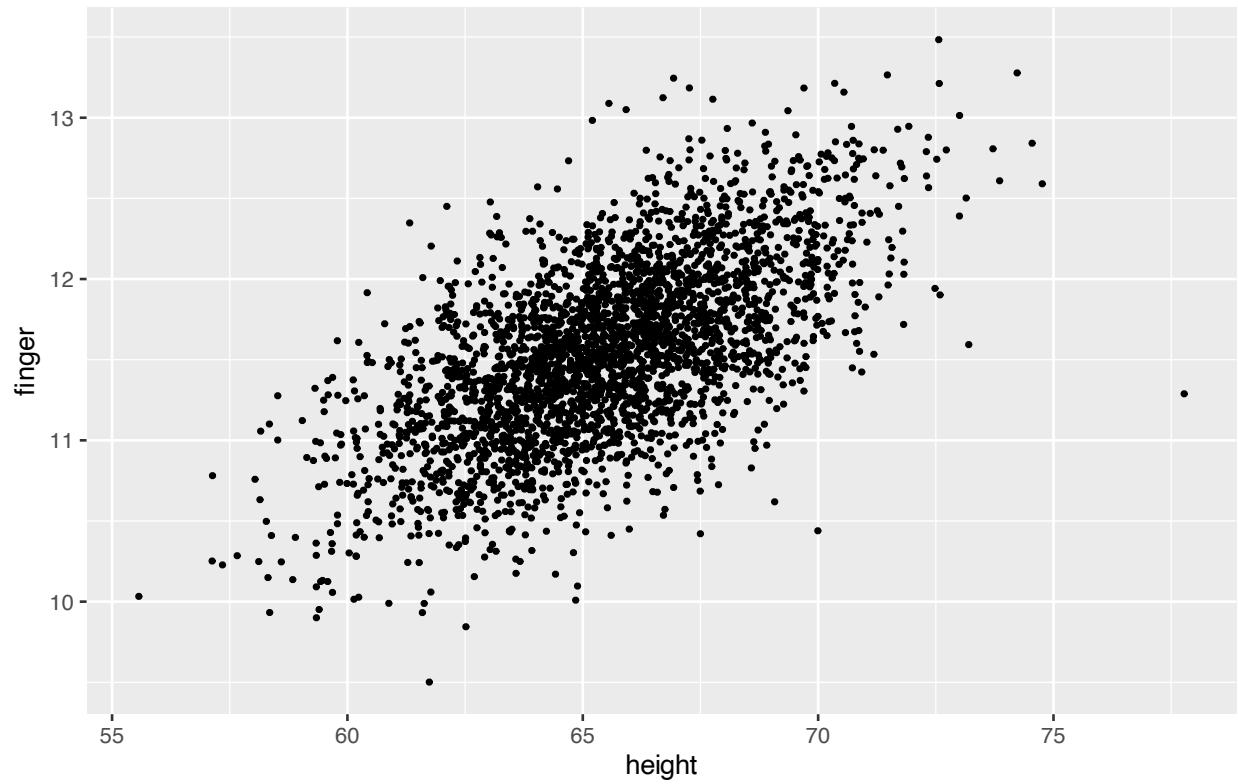
- 흐트러놓는 폭 조절 : width = 1, height = 0, size = 0.7

```
g2_8 <- g1 +
  geom_point(position = position_jitter(width = 1, height = 0),
             size = 0.7)
g2_8
```



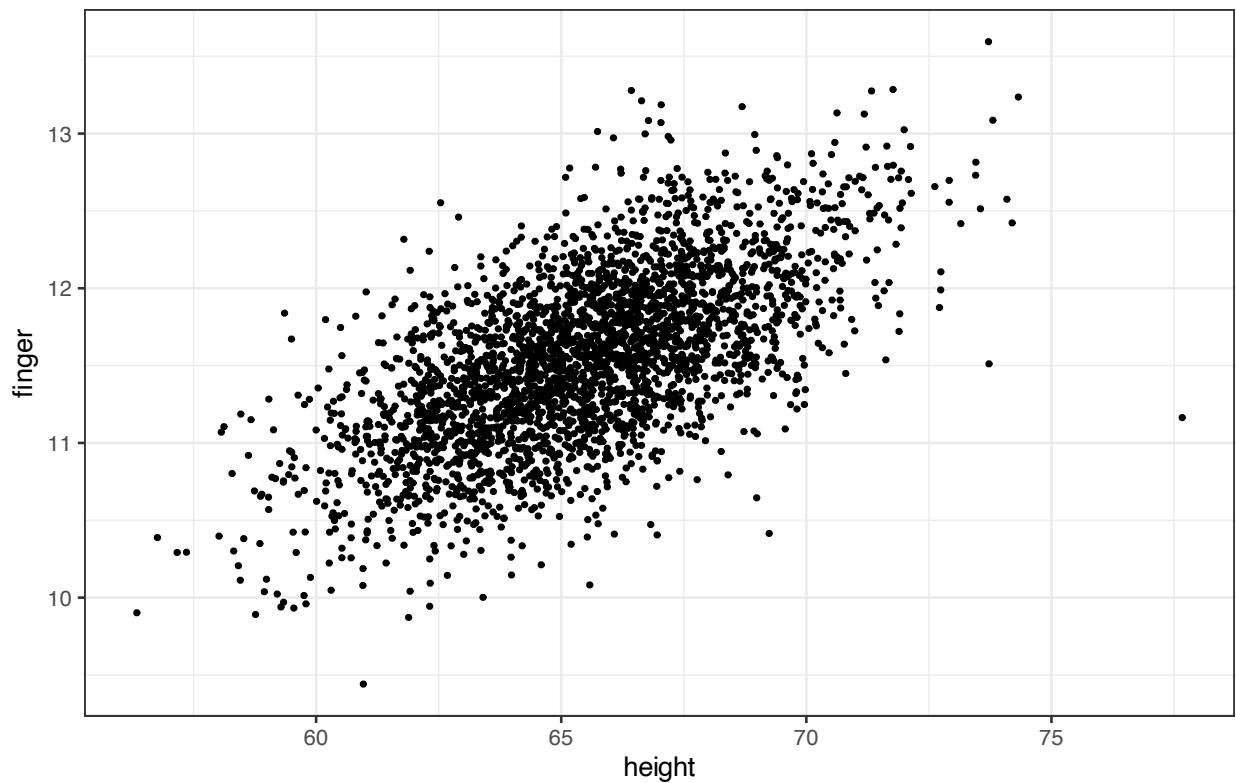
- 흐트러놓는 폭과 높이 조절 : width = 1, height = 0.1, size = 0.7

```
g2_9 <- g1 +
  geom_point(position = position_jitter(width = 1, height = 0.1),
             size = 0.7)
g2_9
```



- 흑백 테마 : theme_bw()

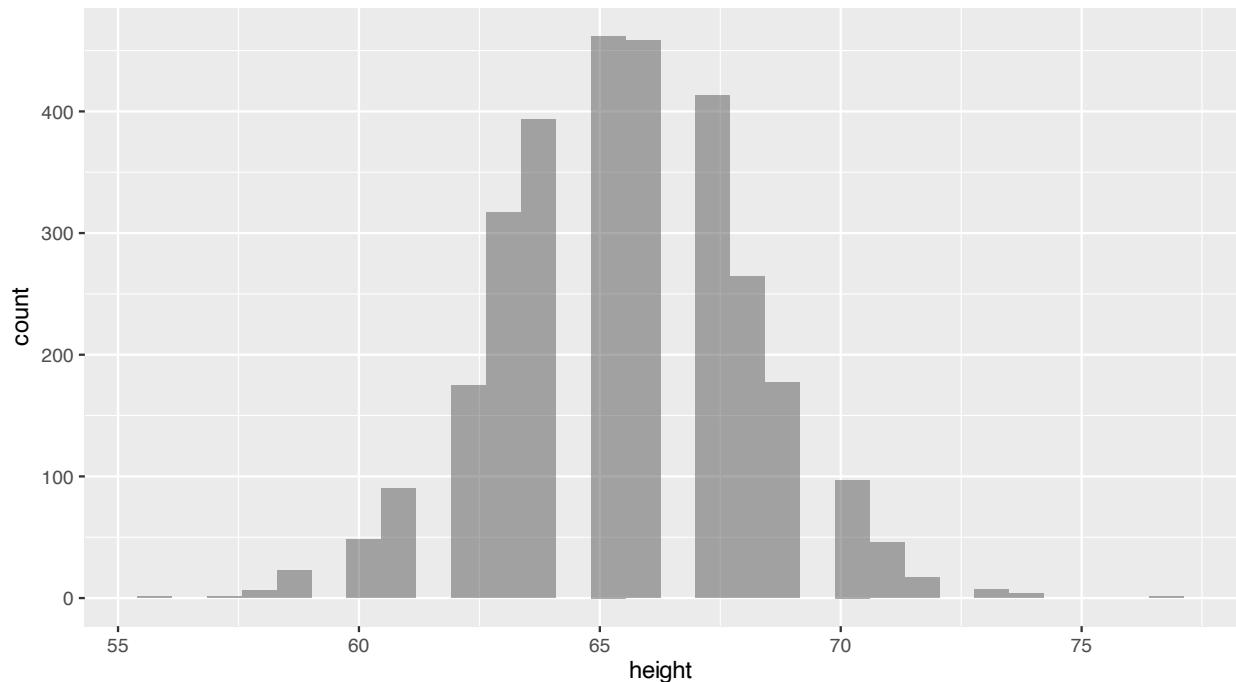
```
g3 <- g2_9 +  
  theme_bw()  
g3
```



히스토그램

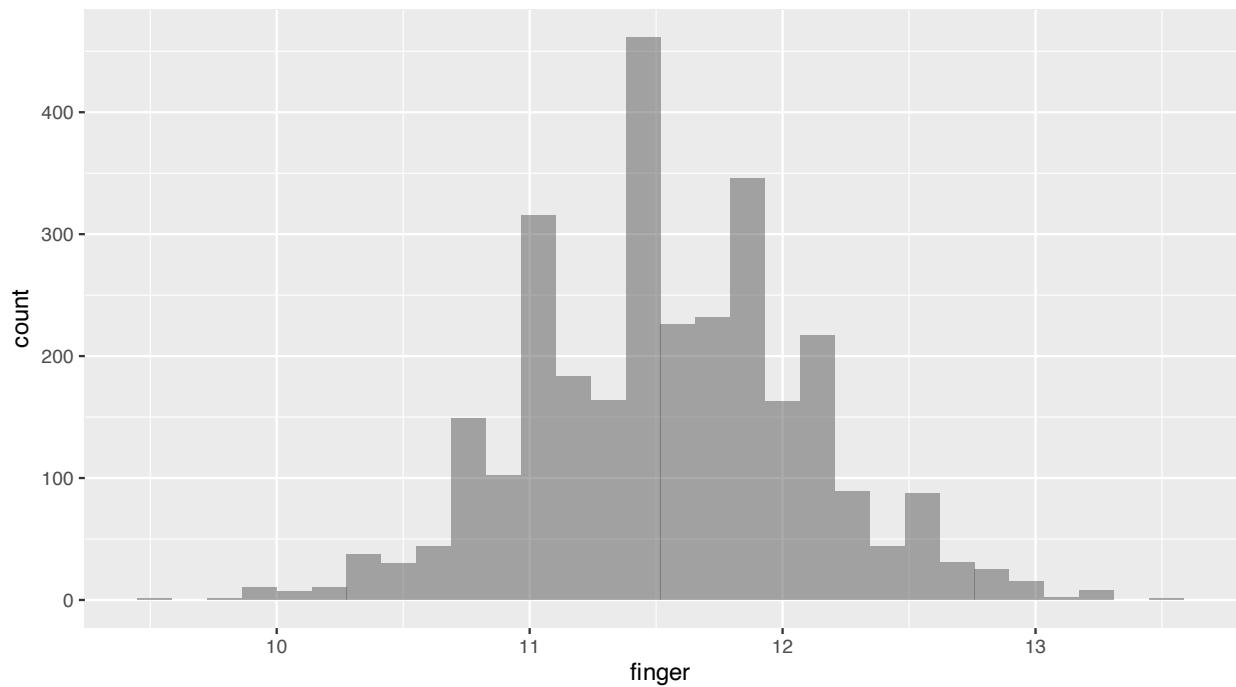
```
h1 <- ggplot(data = crimtab_long_df,
               mapping = aes(x = height))
h1 + geom_histogram(alpha = 0.5)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

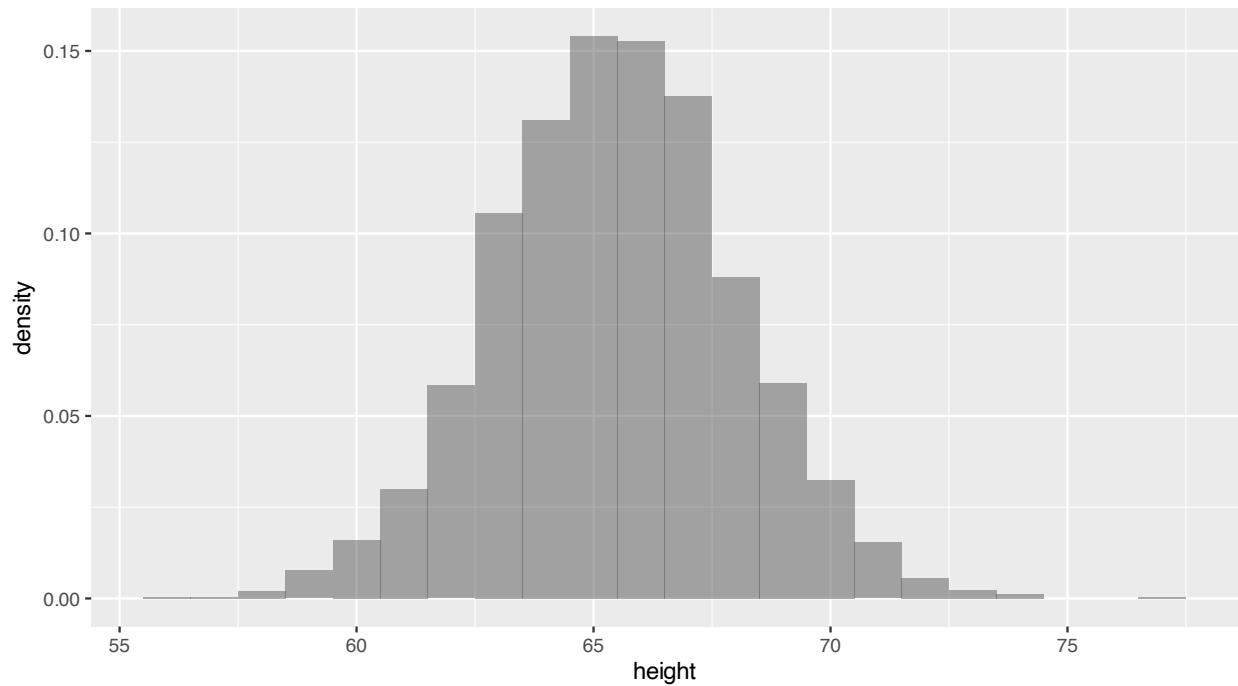


```
f1 <- ggplot(data = crimtab_long_df,
               mapping = aes(x = finger))
f1 + geom_histogram(alpha = 0.5)
```

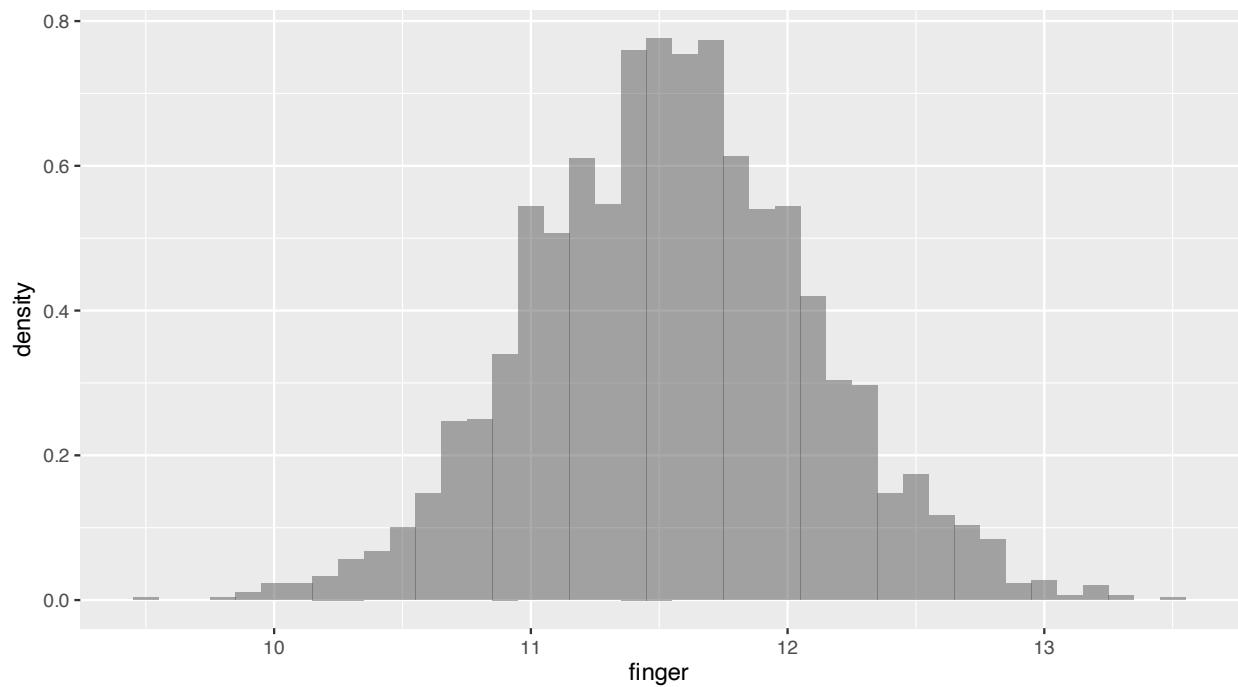
`stat_bin()` using `bins = 30` . Pick better value with `binwidth` .



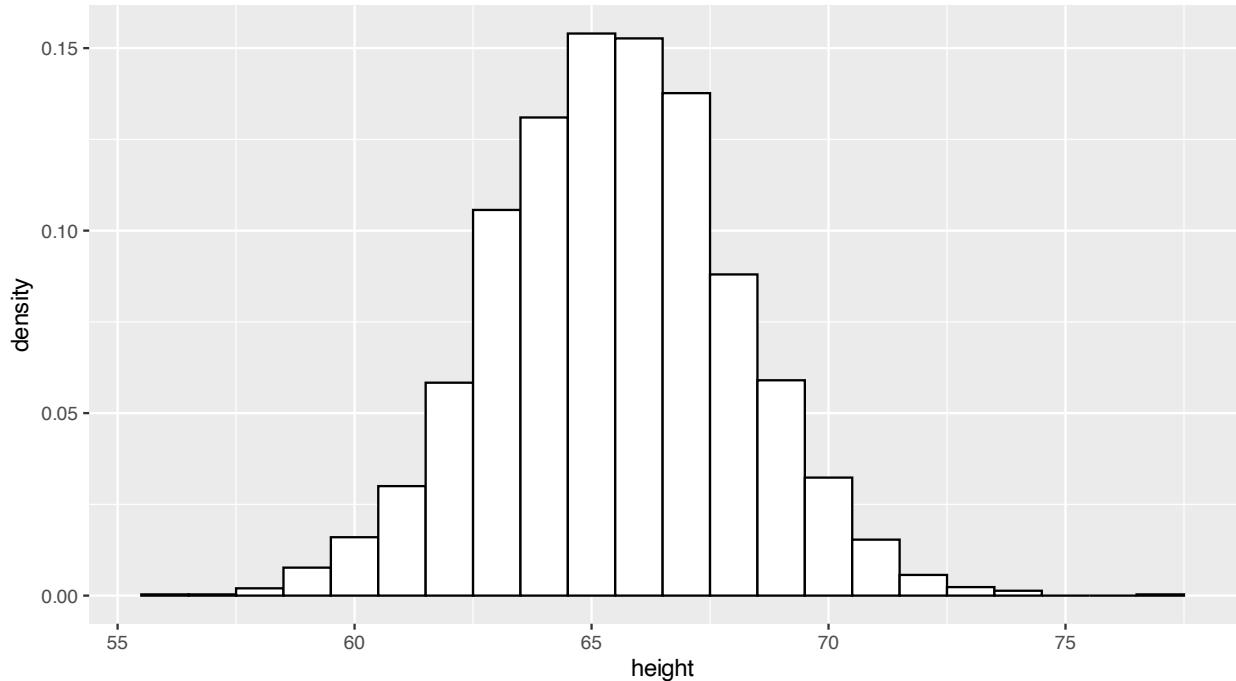
```
h1 + geom_histogram(aes(y = after_stat(density)),  
                    binwidth = 1,  
                    alpha = 0.5)
```



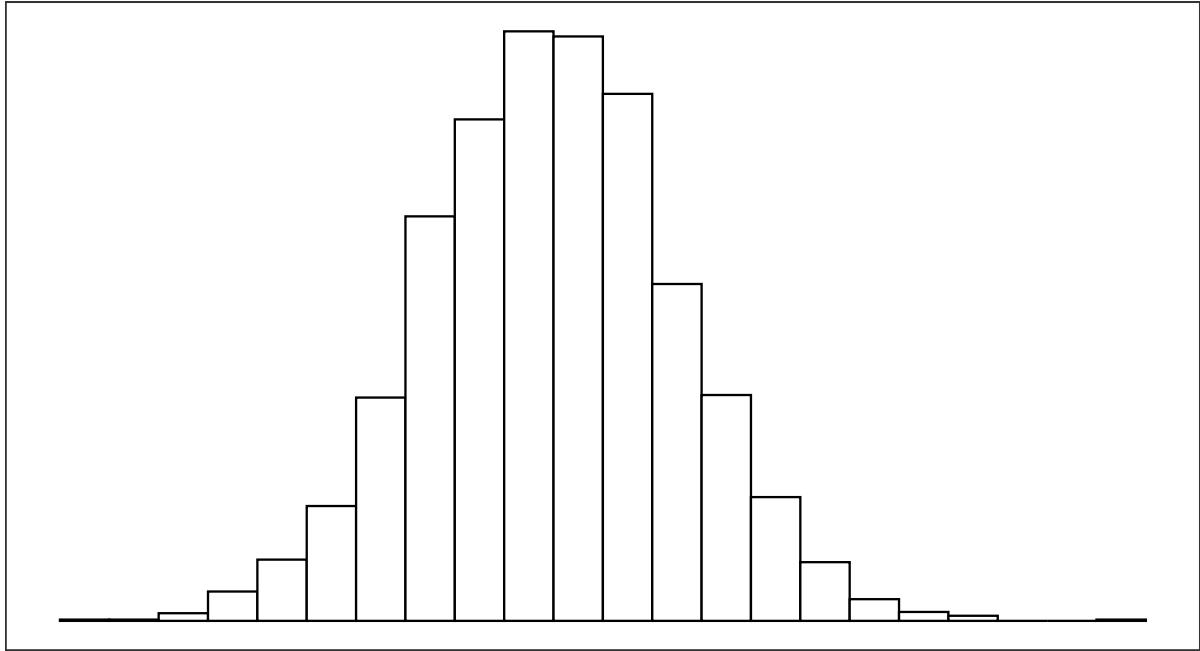
```
f1 + geom_histogram(aes(y = after_stat(density)),  
                    binwidth = 0.1,  
                    alpha = 0.5)
```



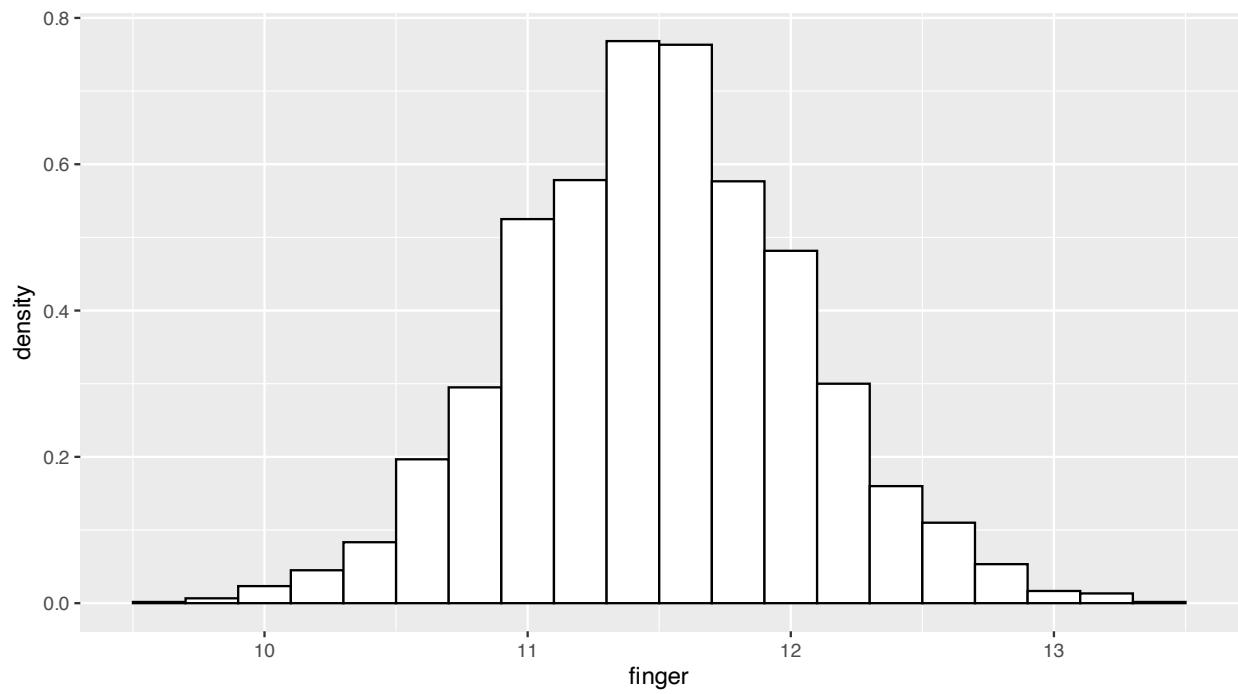
```
(g_h_1 <- h1 +
  geom_histogram(aes(y = after_stat(density)),
    binwidth = 1,
    fill = "white",
    colour = "black"))
```



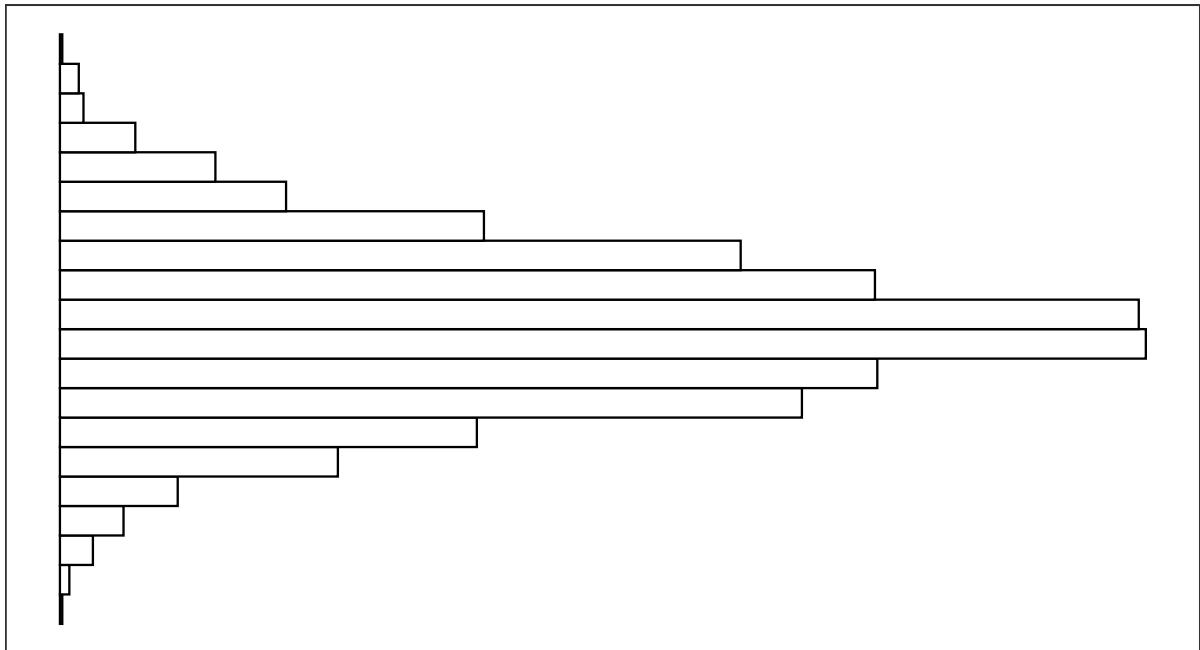
```
(g_h <- g_h_1 +
  theme_bw() +
  scale_x_continuous(name = "",
                     breaks = NULL) +
  scale_y_continuous(name = "",
                     breaks = NULL))
```



```
(g_f_1 <- f1 +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 0.2,
                 fill = "white",
                 colour = "black"))
```



```
(g_f <- g_f_1 +
  theme_bw() +
  scale_x_continuous(name = "",
                     breaks = NULL) +
  scale_y_continuous(name = "",
                     breaks = NULL) +
  coord_flip())
```



평균 위치를 화살표로 나타내려면

```
library(grid)
(mean_finger <- crimtab_stat[, 1][[1]])

## [1] 11.54737

(sd_finger <- crimtab_stat[, 1][[2]])

## [1] 0.5487137

(mean_height <- crimtab_stat[, 2][[1]])

## [1] 65.473

(sd_height <- crimtab_stat[, 2][[2]])

## [1] 2.557757
```

```

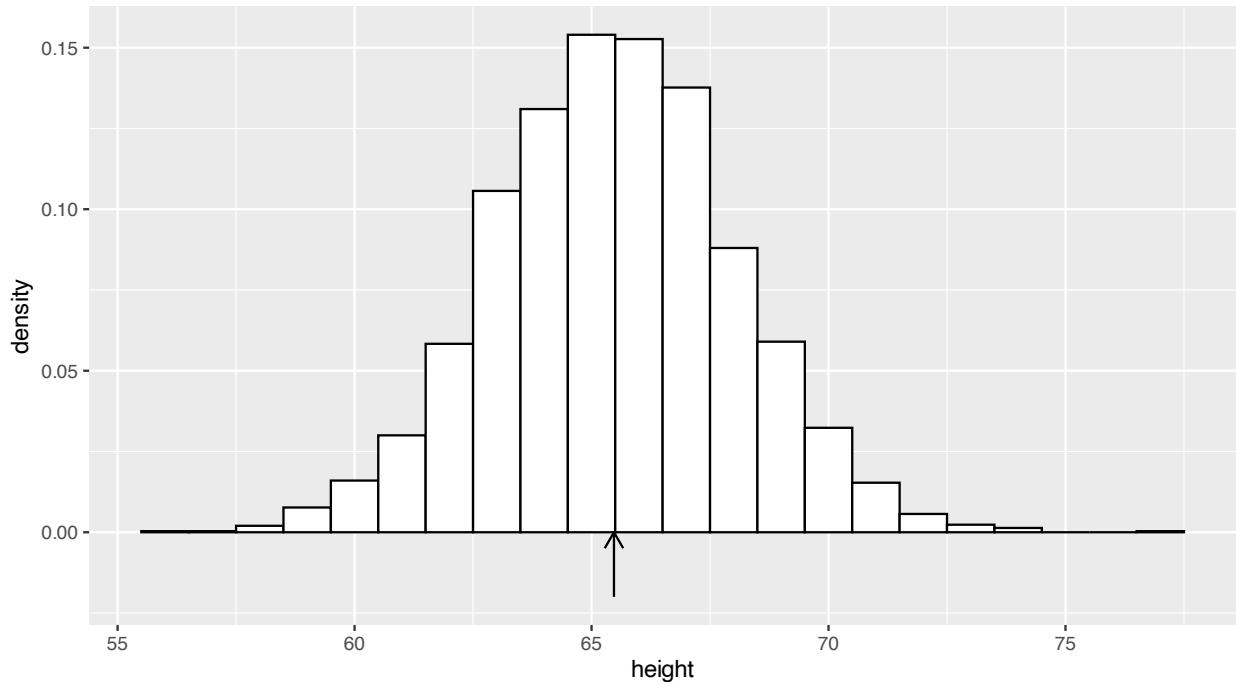
x_finger <- seq(9.5, 13.5,
                 length.out = 3000)
y_finger <- dnorm(x_finger,
                   mean = mean_finger,
                   sd = sd_finger)
x_height <- seq(56, 77,
                  length.out = 3000)
y_height <- dnorm(x_height,
                   mean = mean_height,
                   sd = sd_height)

```

```

(g_h_2 <- g_h_1 +
  annotate("segment",
           x = mean_height,
           xend = mean_height,
           y = -0.02,
           yend = 0,
           arrow = arrow(length = unit(0.3, "cm"))))

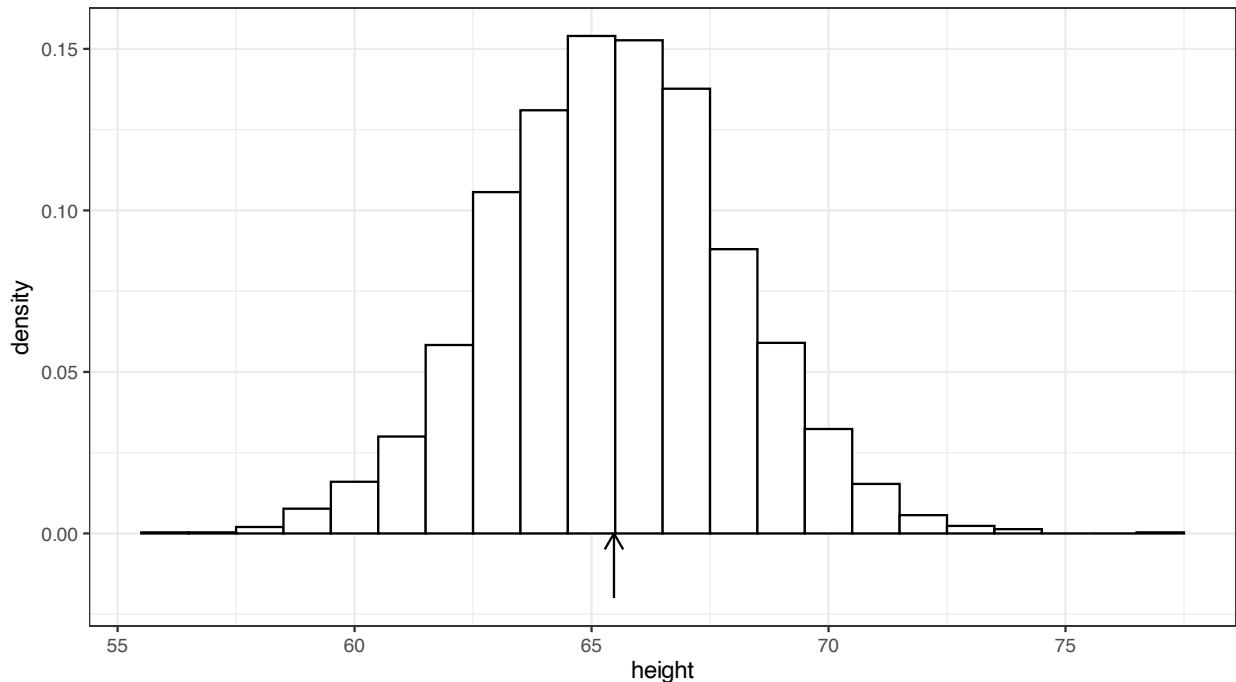

```



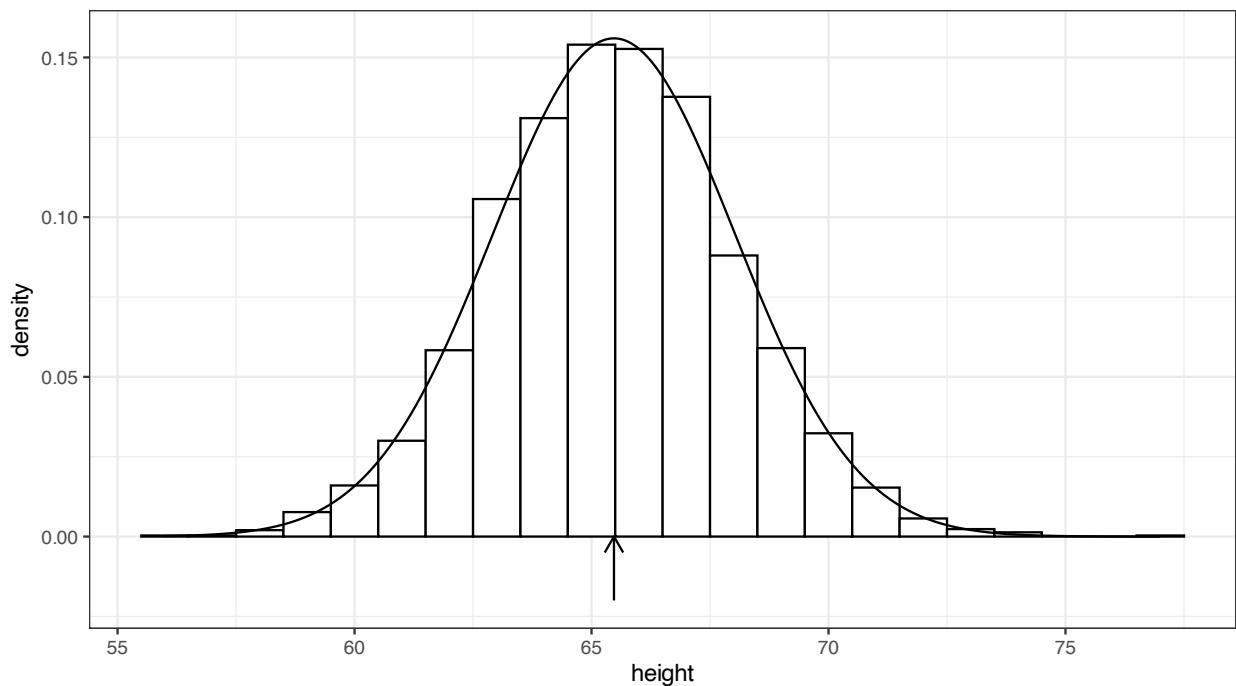
```

(g_h_3 <- g_h_2 +
  theme_bw())

```

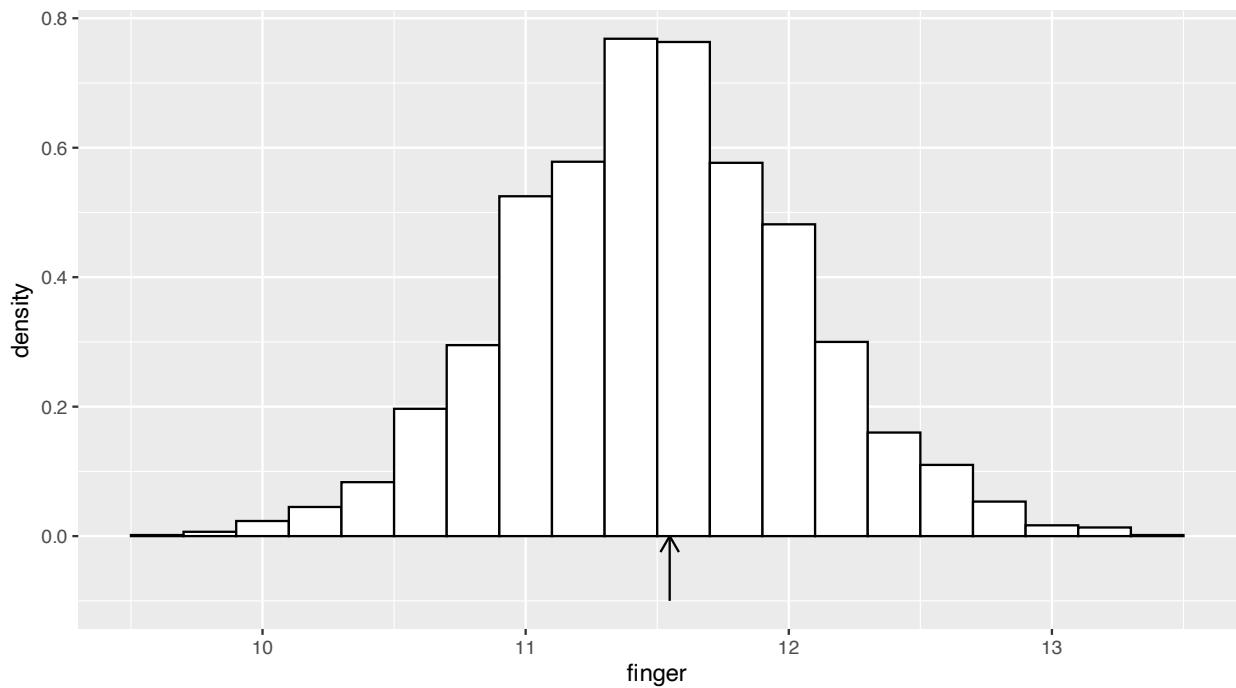


```
(g_h_4 <- g_h_3 +
  geom_line(aes(x = x_height, y = y_height)))
```

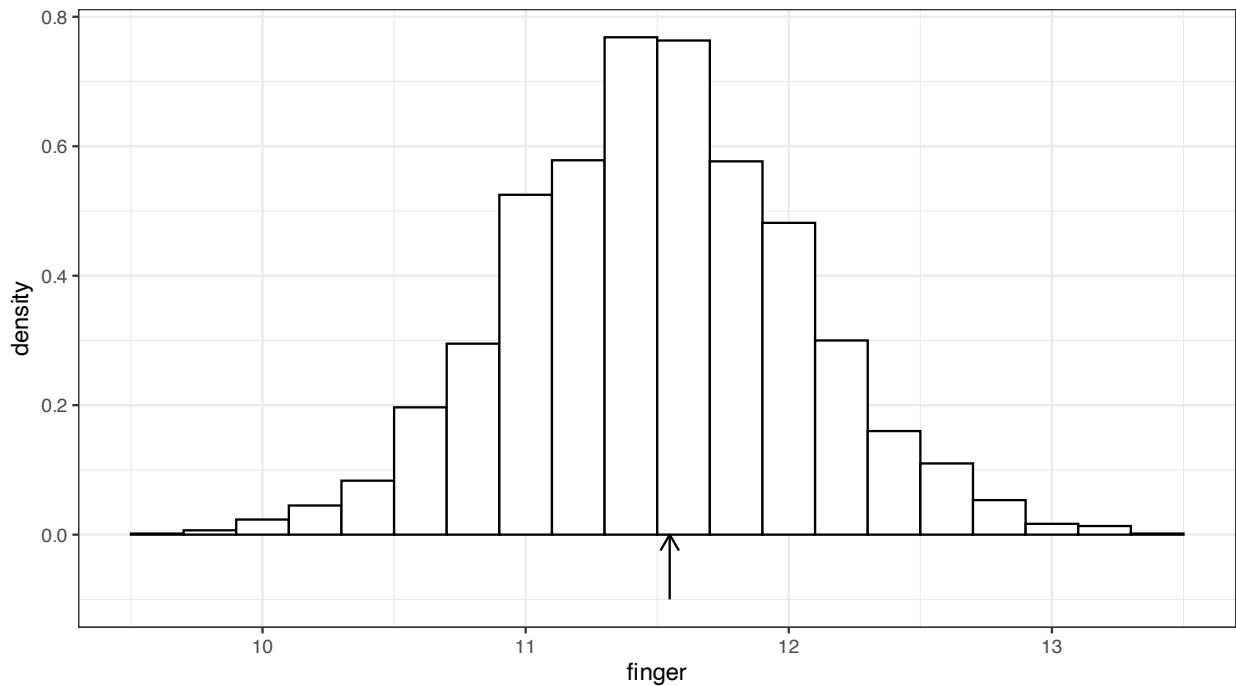


```
(g_f_2 <- g_f_1 +
  annotate("segment",
    x = mean_finger,
    xend = mean_finger,
```

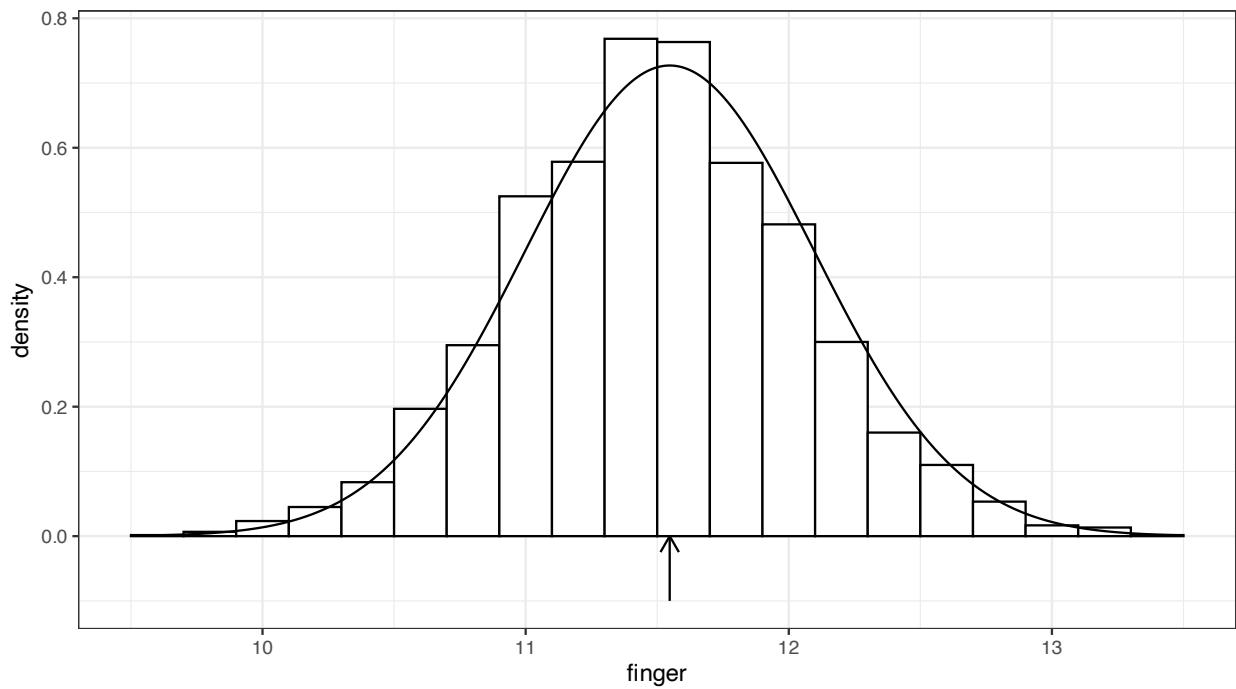
```
y = -0.1,  
yend = 0,  
arrow = arrow(length = unit(0.3, "cm"))))
```



```
(g_f_3 <- g_f_2 +  
  theme_bw())
```



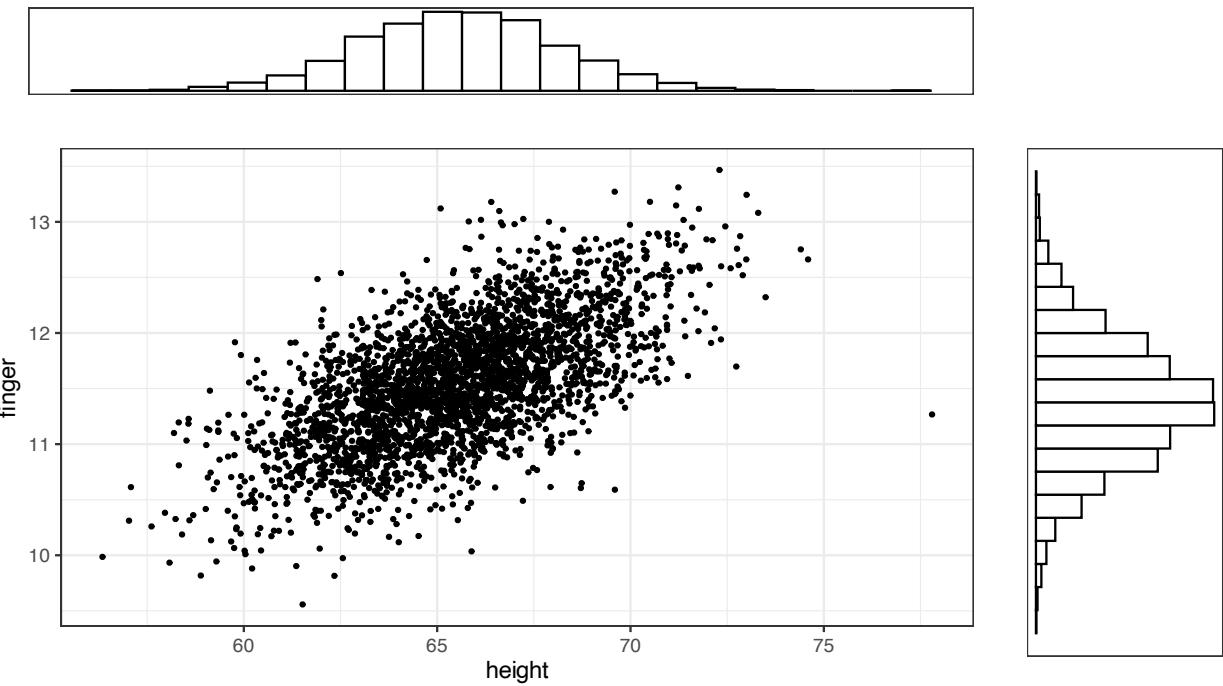
```
(g_f_4 <- g_f_3 +
  geom_line(aes(x = x_finger, y = y_finger)))
```



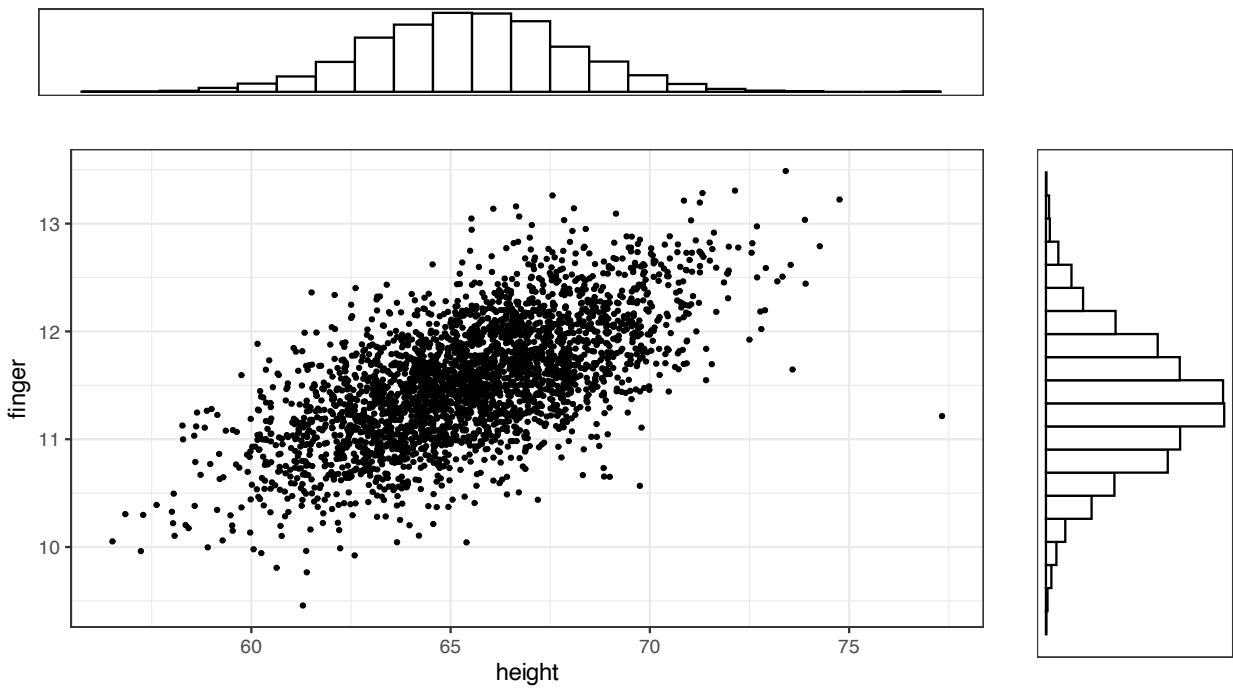
산점도와 히스토그램 함께 배열하기

grid 및 gridExtra 패키지와 함께 blank Grob 설정이 핵심. grid.arrange 사용법에 유의.

```
library(gridExtra)
grid.rect(gp = gpar(col = "white"), draw = FALSE) %>%
grid.arrange(g_h, ., g3, g_f,
            ncol = 2,
            widths = c(4, 1),
            heights = c(1, 4))
```



```
blank <- grid.rect(gp = gpar(col = "white"), draw = FALSE)
grid.arrange(g_h, blank, g3, g_f,
             ncol = 2,
             widths = c(4, 1),
             heights = c(1, 4))
```



```
blank <- nullGrob()
grid.arrange(g_h, blank, g3, g_f,
             ncol = 2,
             widths = c(4, 1),
             heights = c(1, 4))
```

