Bios 6301: Assignment 9

Yiqing Pan

Due Thursday, 21 November, 1:00 PM

$5^{n=day}$ points taken off for each day late.

40 points total.

Submit a single quarto file (named `homework9.qmd`), along with a valid PDF output file. Inside the file, clearly indicate which parts of your responses go with which problems (you may use the original homework document as a template). Add your name as `author` to the file's metadata section. Raw R code/output or word processor files are not acceptable.

Failure to name file `homework9.qmd` or include author name may result in 5 points taken off.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggplot2)
library(patchwork)
```

Question 1

15 points

Consider the following very simple genetic model (*very* simple – don't worry if you're not a geneticist!). A population consists of equal numbers of two sexes: male and female. At each generation men and women are

paired at random, and each pair produces exactly two offspring, one male and one female. We are interested in the distribution of height from one generation to the next. Suppose that the height of both children is just the average of the height of their parents, how will the distribution of height change across generations?

Represent the heights of the current generation as a dataframe with two variables, m and f, for the two sexes. We can use `rnorm` to randomly generate the population at generation 1:

```
set.seed(123)

pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
```

The following function takes the data frame `pop` and randomly permutes the ordering of the men. Men and women are then paired according to rows, and heights for the next generation are calculated by taking the mean of each row. The function returns a data frame with the same structure, giving the heights of the next generation.

```
next_gen <- function(pop) {
    pop$m <- sample(pop$m)
    pop$m <- rowMeans(pop)
    pop$f <- pop$m
    pop
}
```

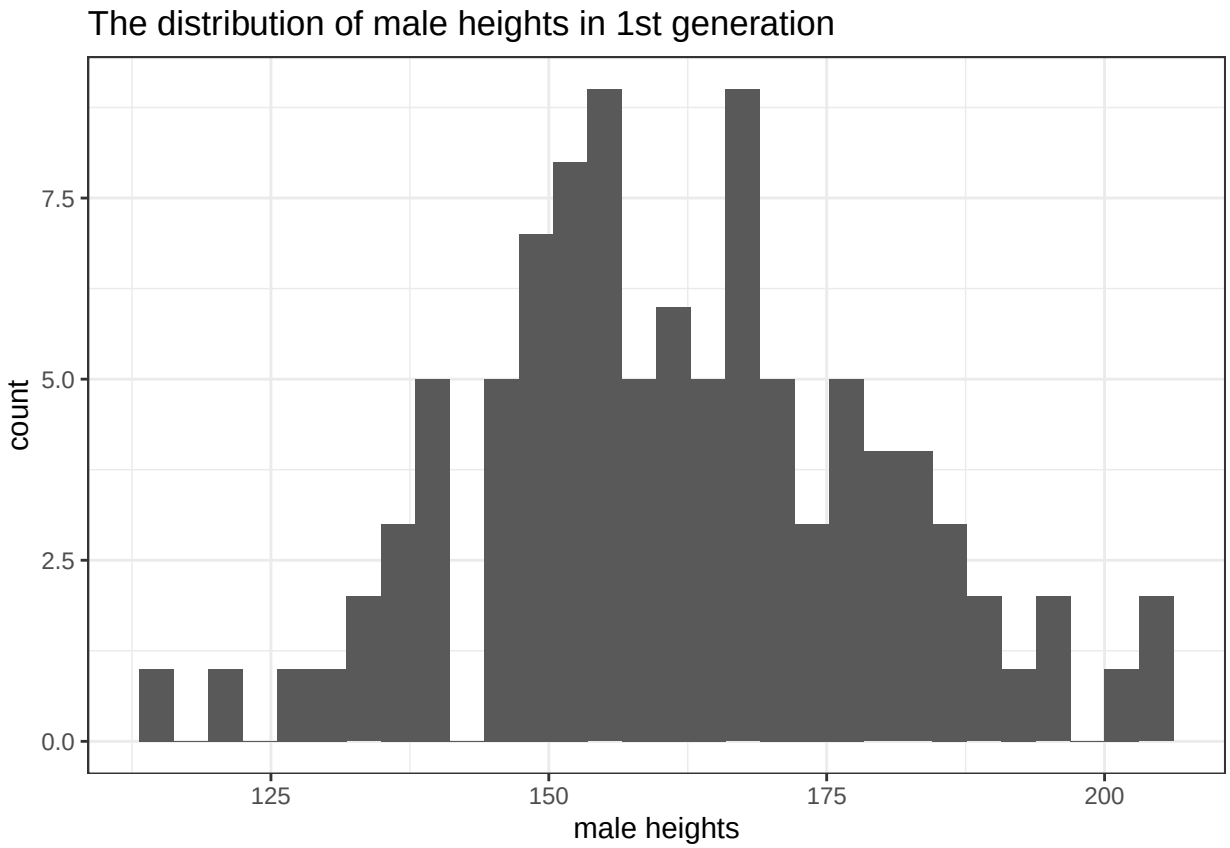
Use the function `next_gen` to generate nine generations (you already have the first), then use the function `hist` to plot the distribution of male heights in each generation (this will require multiple calls to `hist`). The phenomenon you see is called regression to the mean. Provide (at least) minimal decorations such as title and x-axis labels.

```
next_gen <- function(pop) {
    pop$m <- sample(pop$m)
    pop$m <- rowMeans(pop)
    pop$f <- pop$m
    return(pop)
}



generation <- c('1st', '2nd', '3th', '4th', '5th', '6th', '7th', '8th', '9th')
plot <- seq(1,9)

plot_var <- paste0("p", plot[1])
assign(plot_var,
  ggplot(data = pop, aes(x = m)) +
    geom_histogram() +
    ggtitle(paste("The distribution of male heights in", generation[1], "generation")) +
    xlab("male heights") +
    theme_bw())

for (i in 2:9) {
  pop <- next_gen(pop)
  plot_var <- paste0("p", plot[i])
  assign(plot_var,
  ggplot(data = pop, aes(x = m)) +
    geom_histogram() +
```

```
    ggtitle(paste("The distribution of male heights in", generation[i], "generation")) +
    xlab("male heights") +
    theme_bw())

}

mget(paste0("p", 1:9))
```

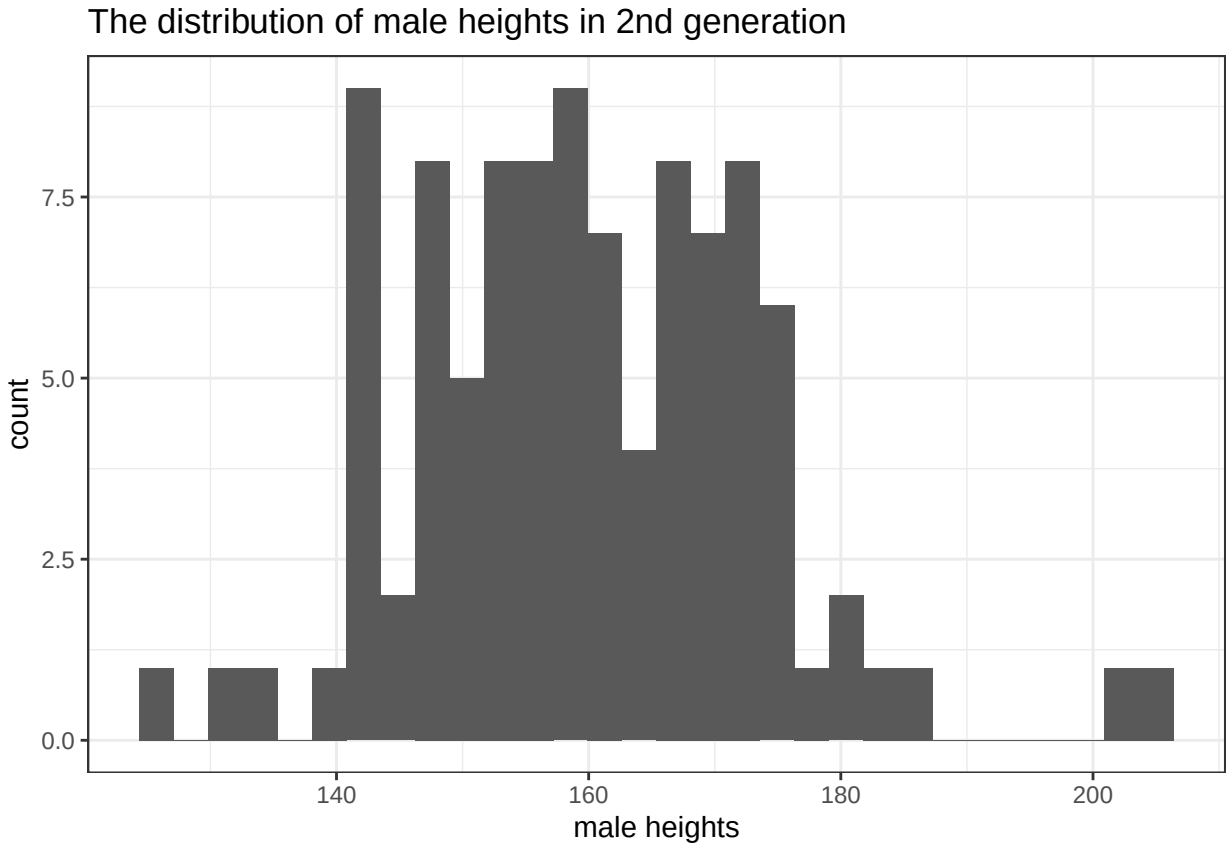
```
## $p1
```

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

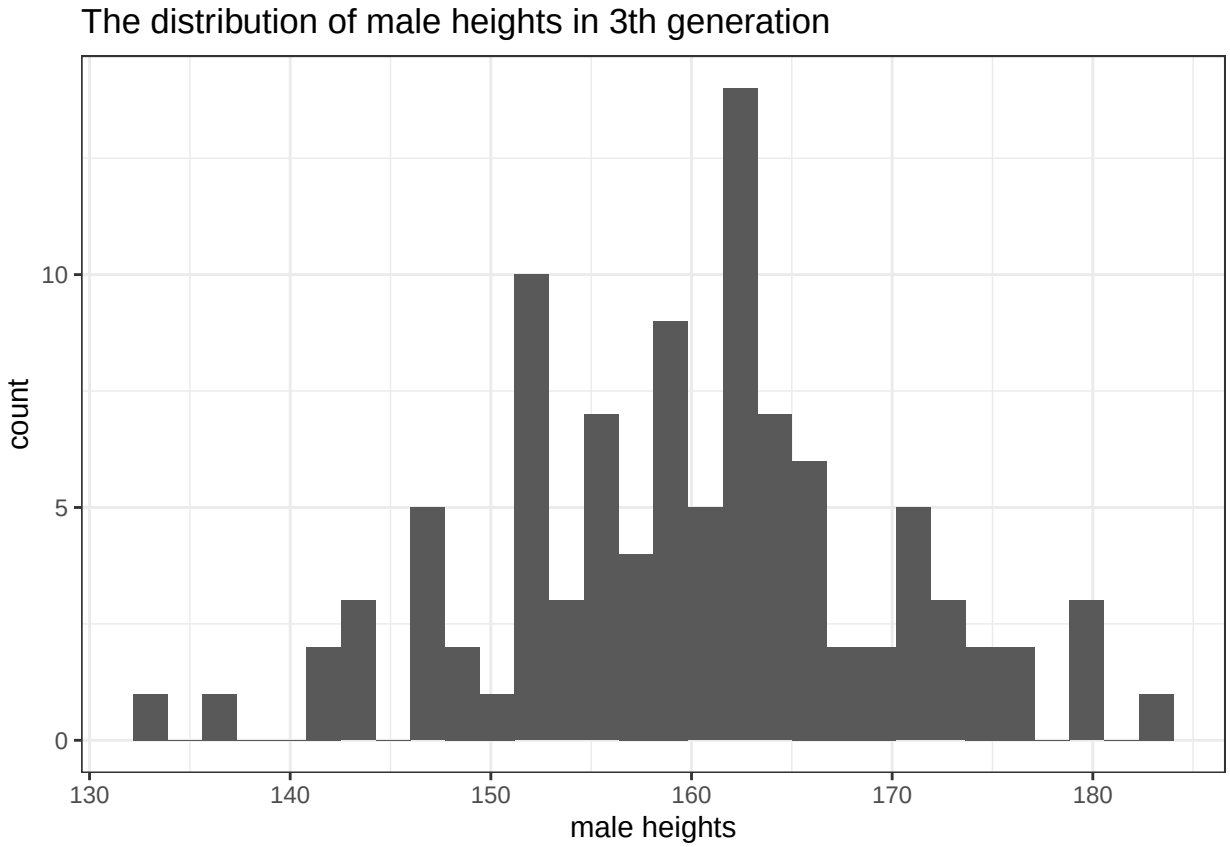



```
##
## $p2
```

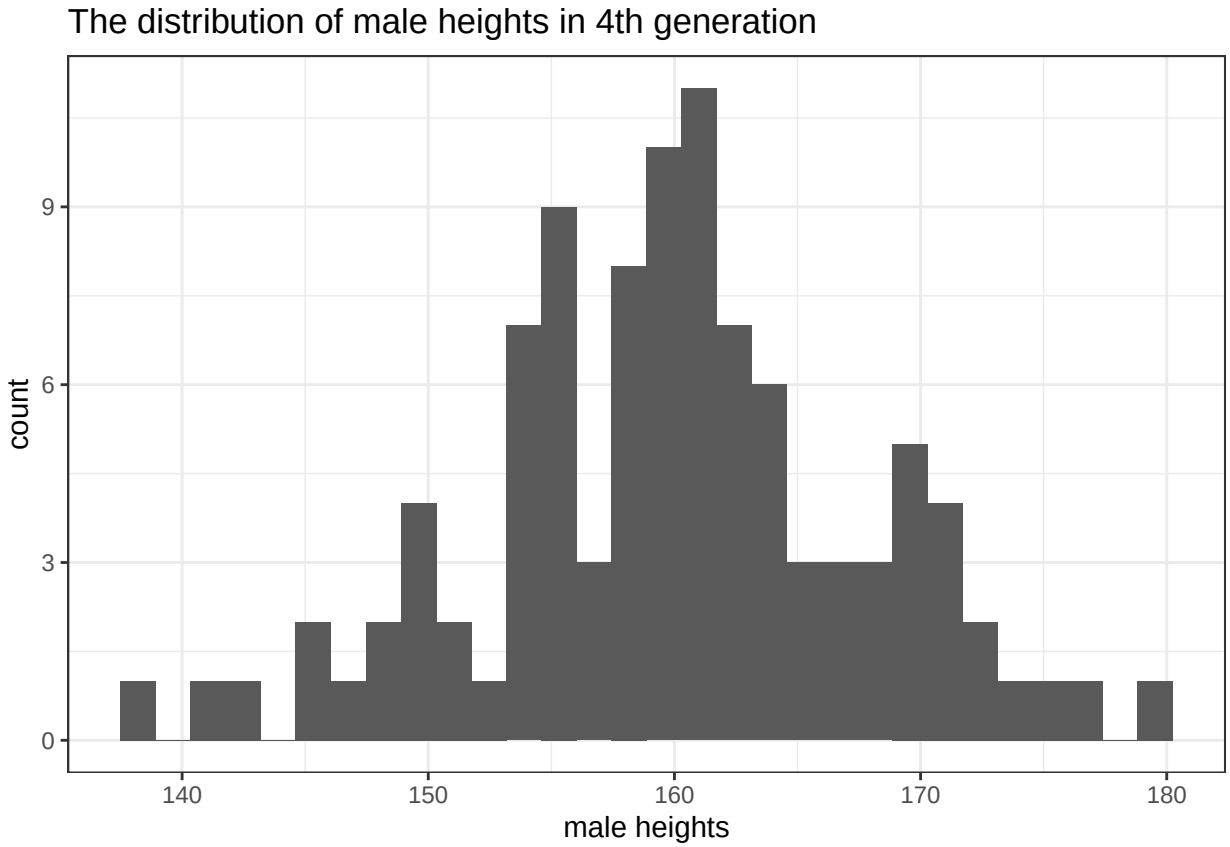
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



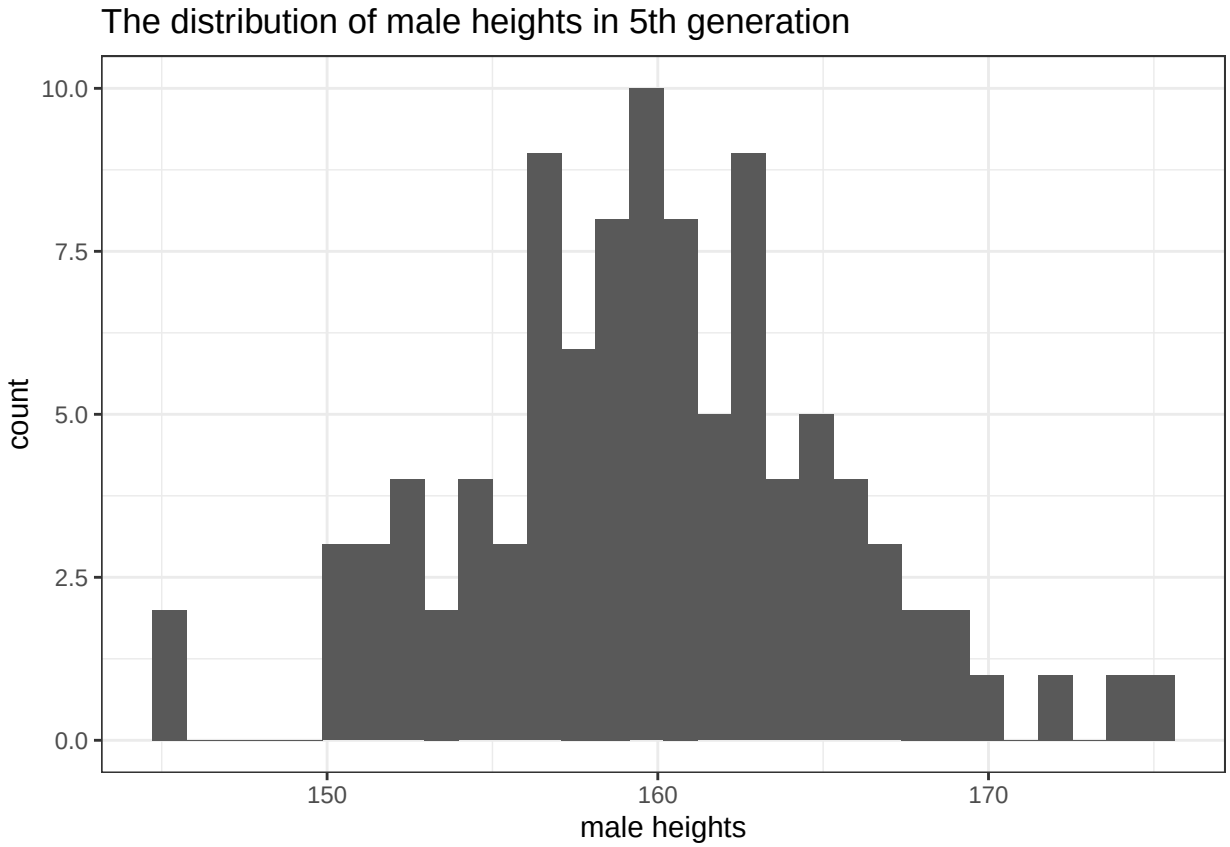

```
##
## $p3

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



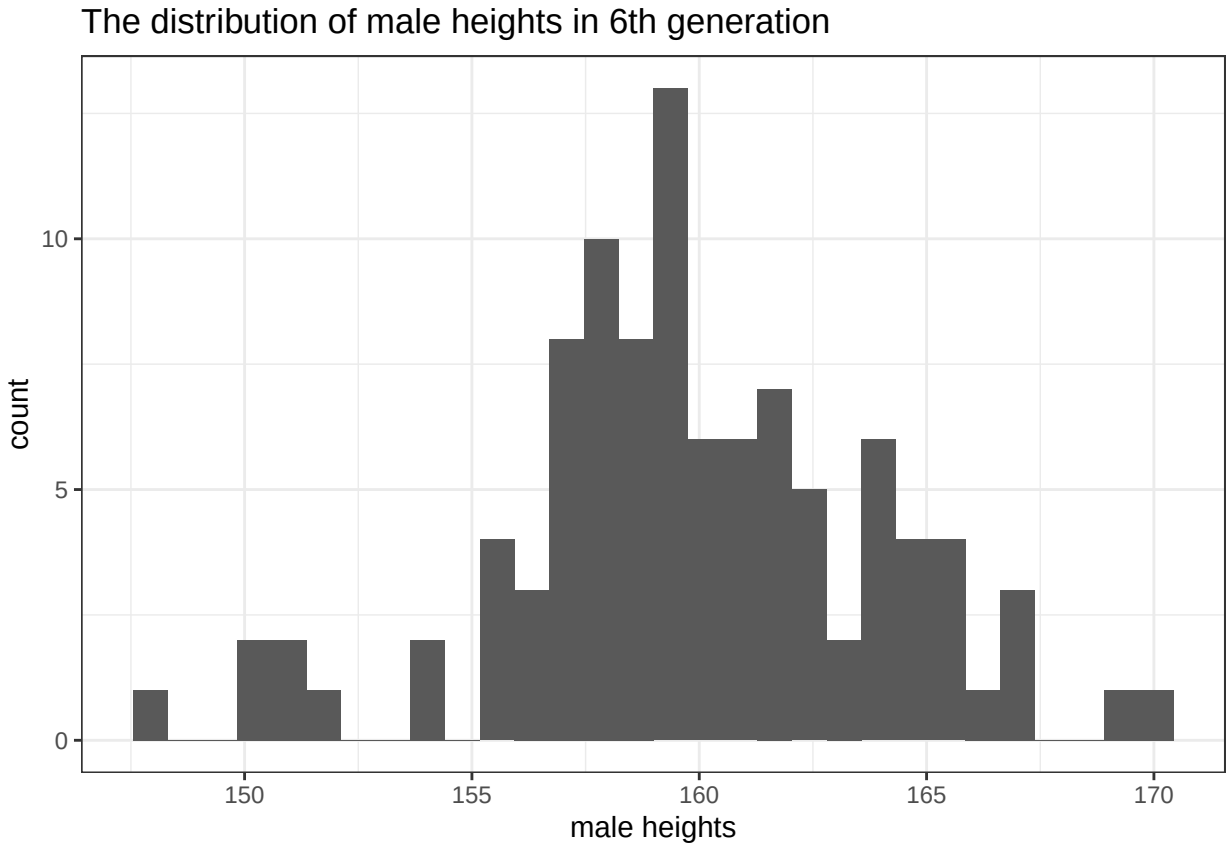

```
##
## $p4

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



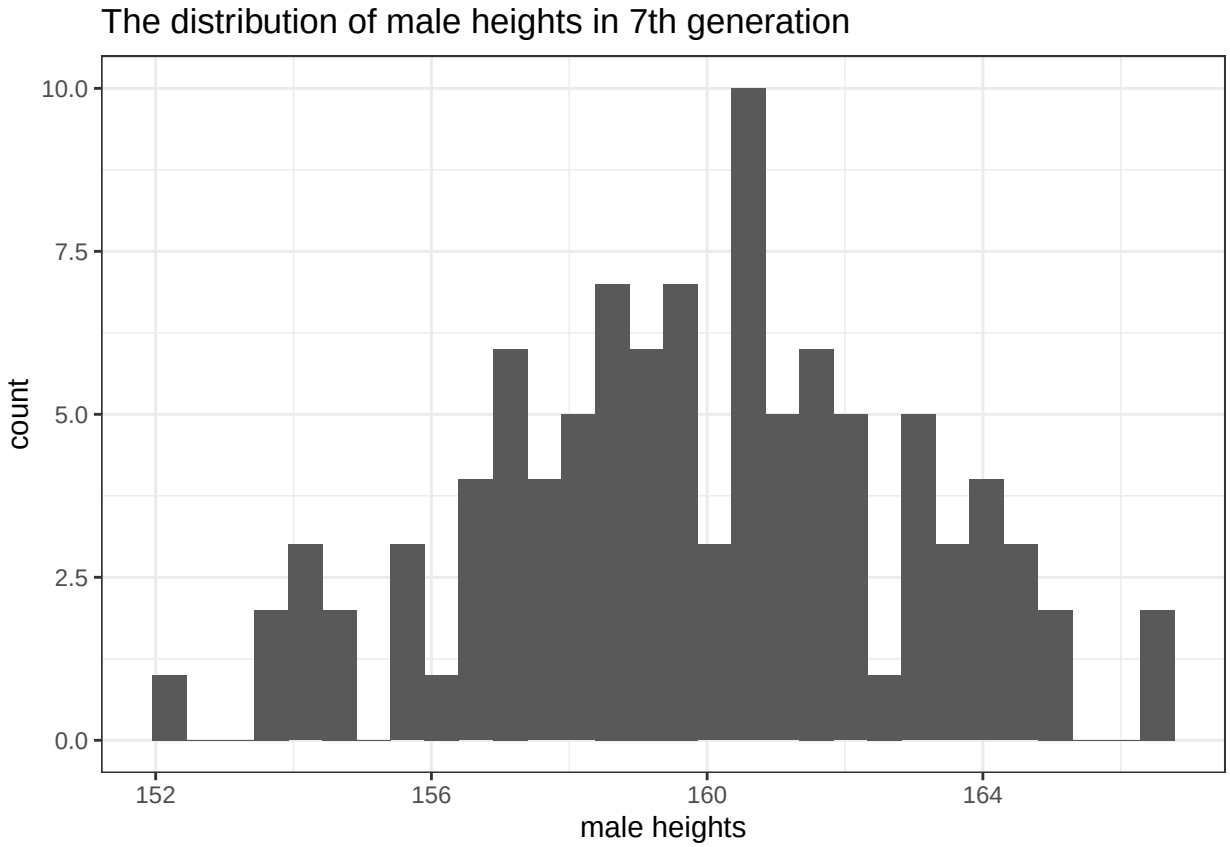

```
##
## $p5

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



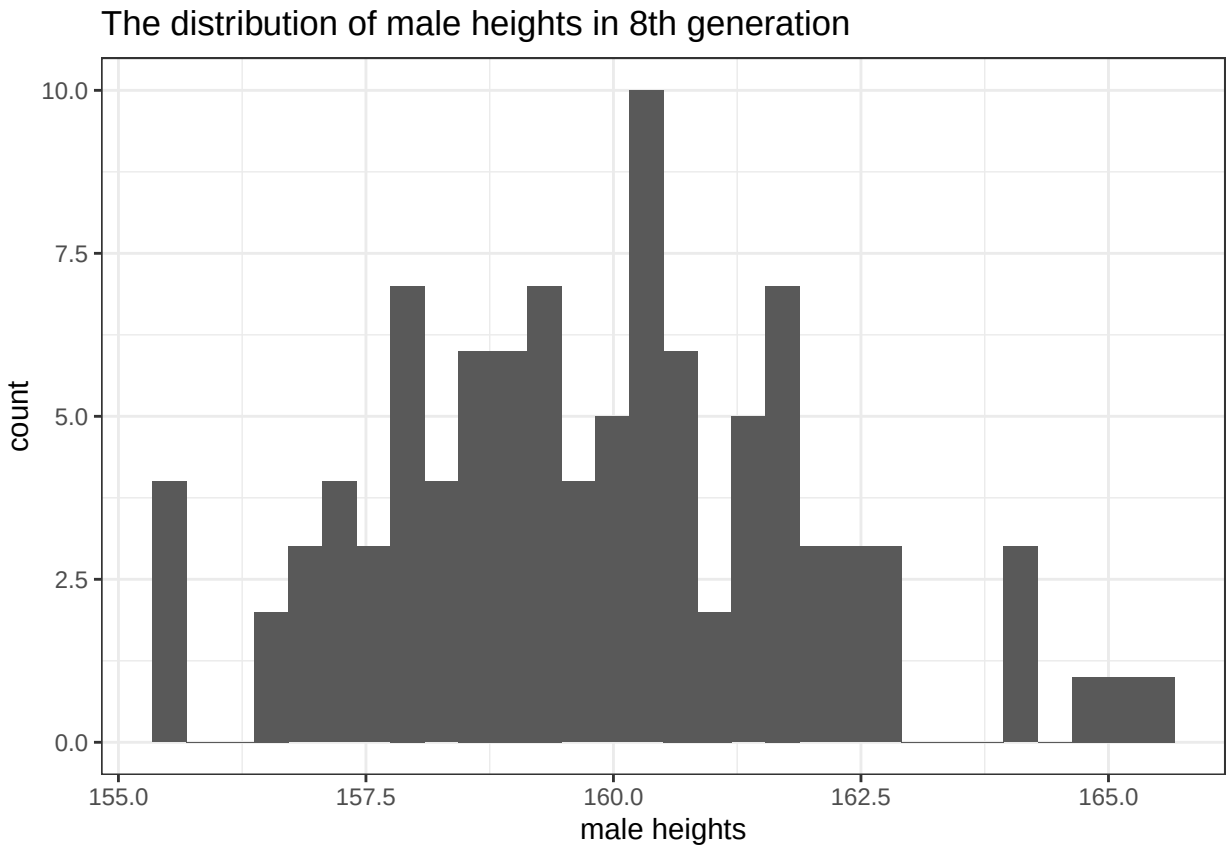

```
##
## $p6

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



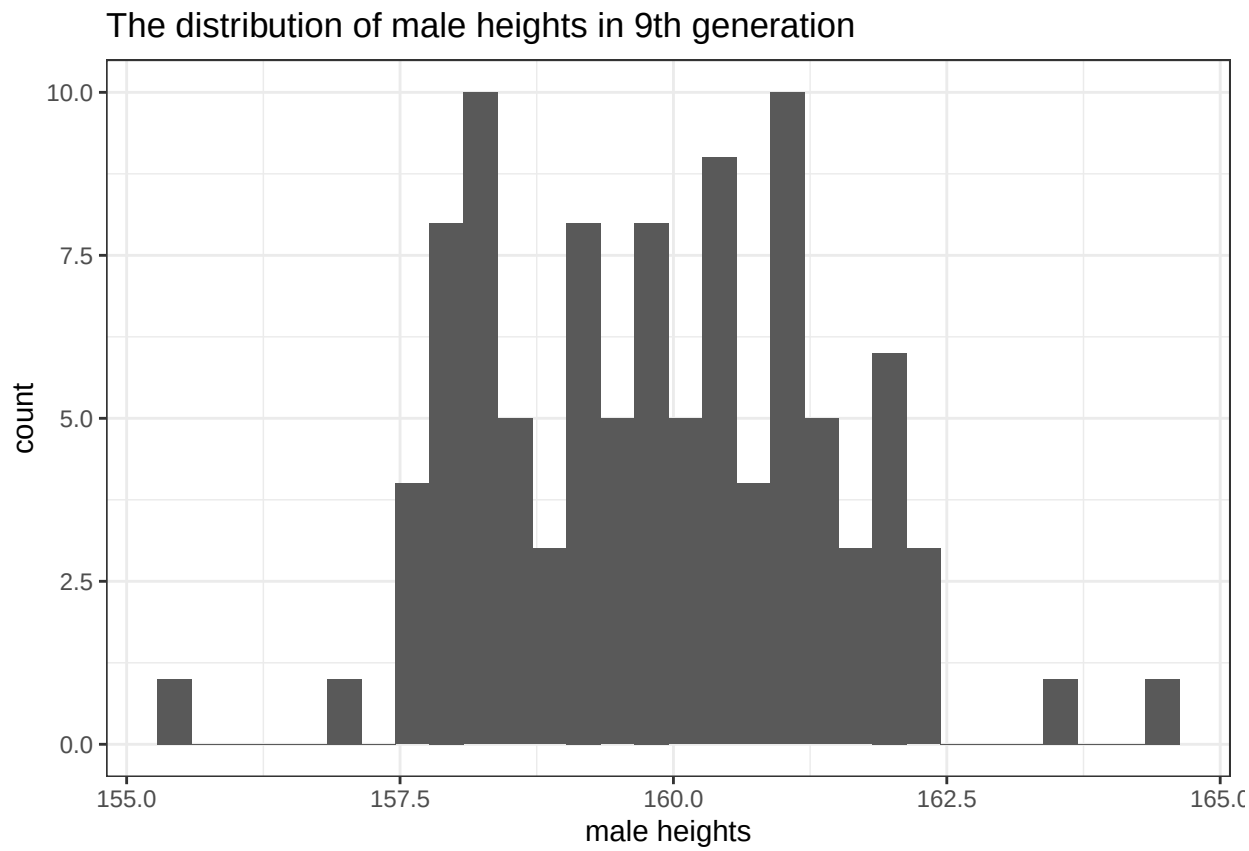

```
##
## $p7

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




```
##
## $p8

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

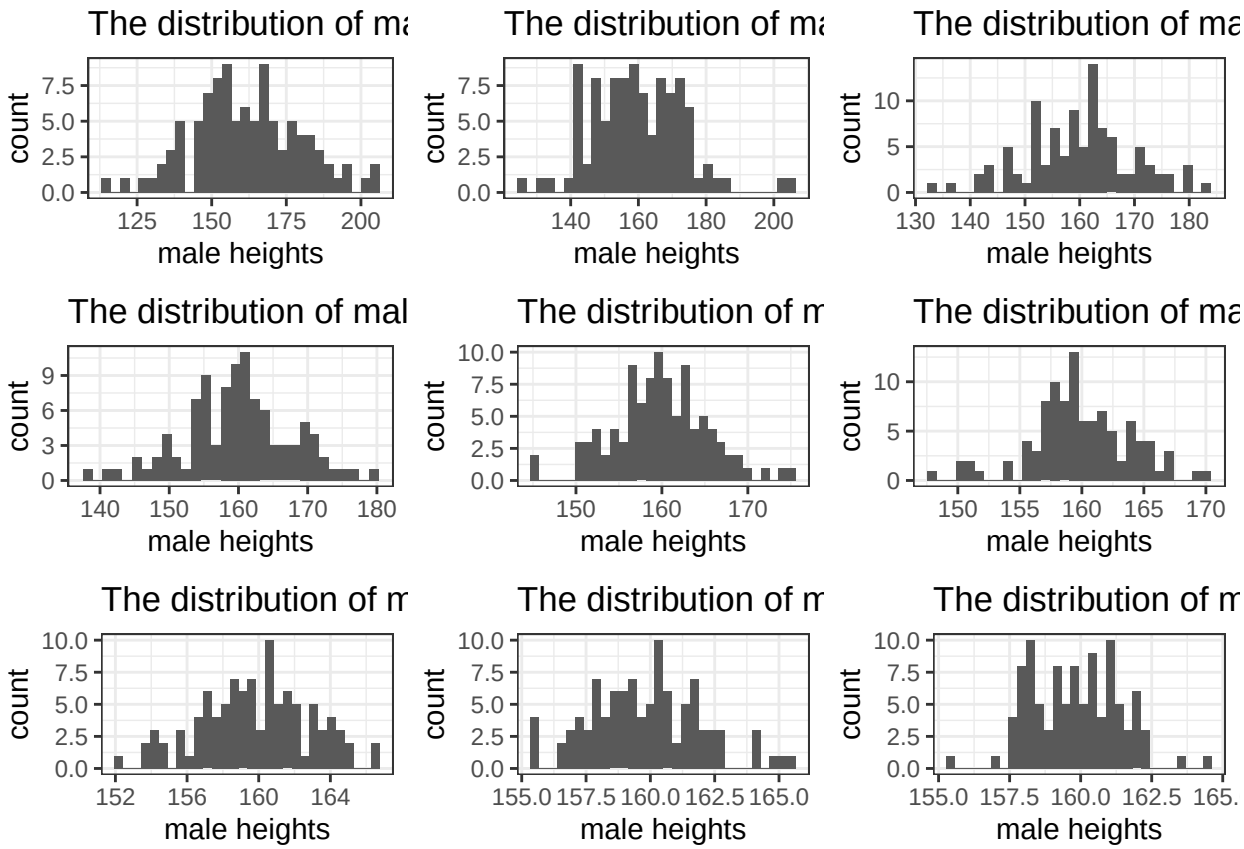



```
##
## $p9

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




```
grid.arrange(grobs = mget(paste0("p", 1:9)), nrow = 3, ncol = 3)
```

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



Question 2

10 points

Use the simulated results from question 1 to reproduce (as closely as possible) the following plot in ggplot2.

```
pop1 <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))

num <- seq(1,9)

for (i in 2:9) {
  pop_var <- paste0("pop", num[i])
  assign(pop_var, next_gen(pop1))
  pop1 <- get(pop_var)

}

pop1 <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))

for (i in 1:9) {
  pop_var <- paste0("pop", num[i])
  assign(pop_var,  get(pop_var) %>% mutate(generation = i))
}

merged_pop <- do.call(rbind, mget(paste0("pop", 1:9)))
```

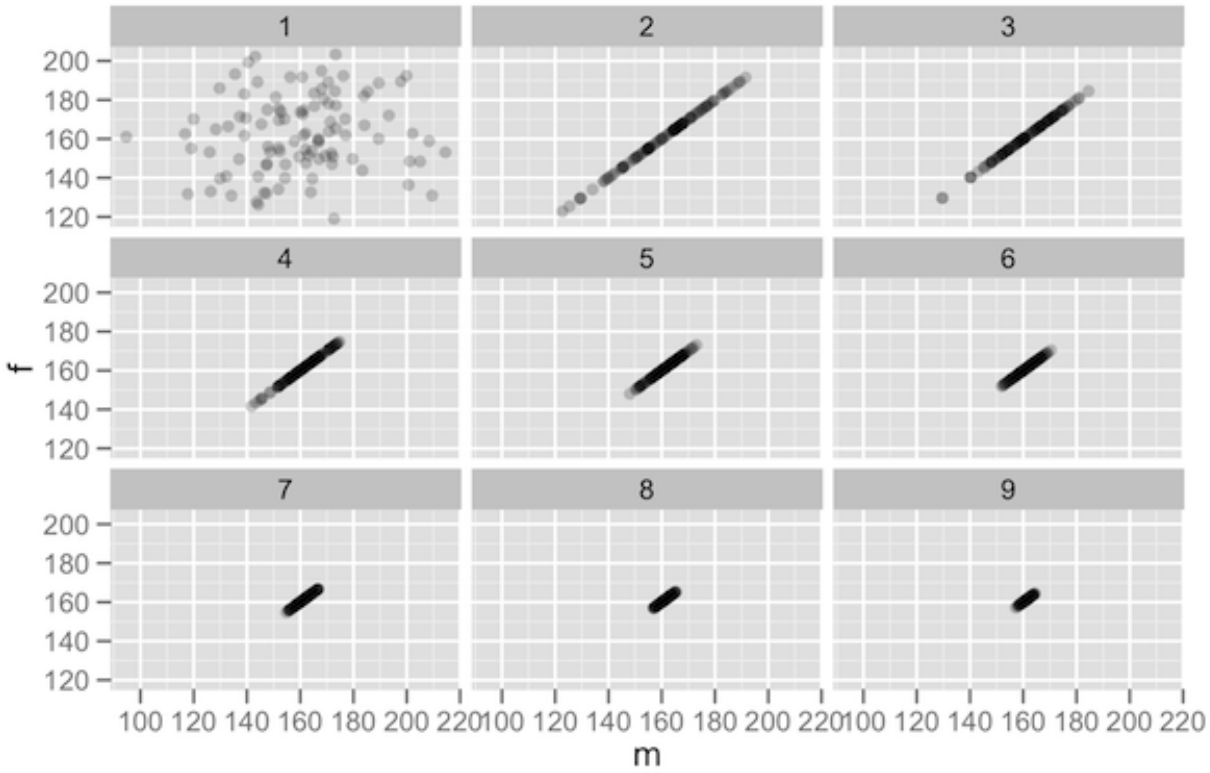
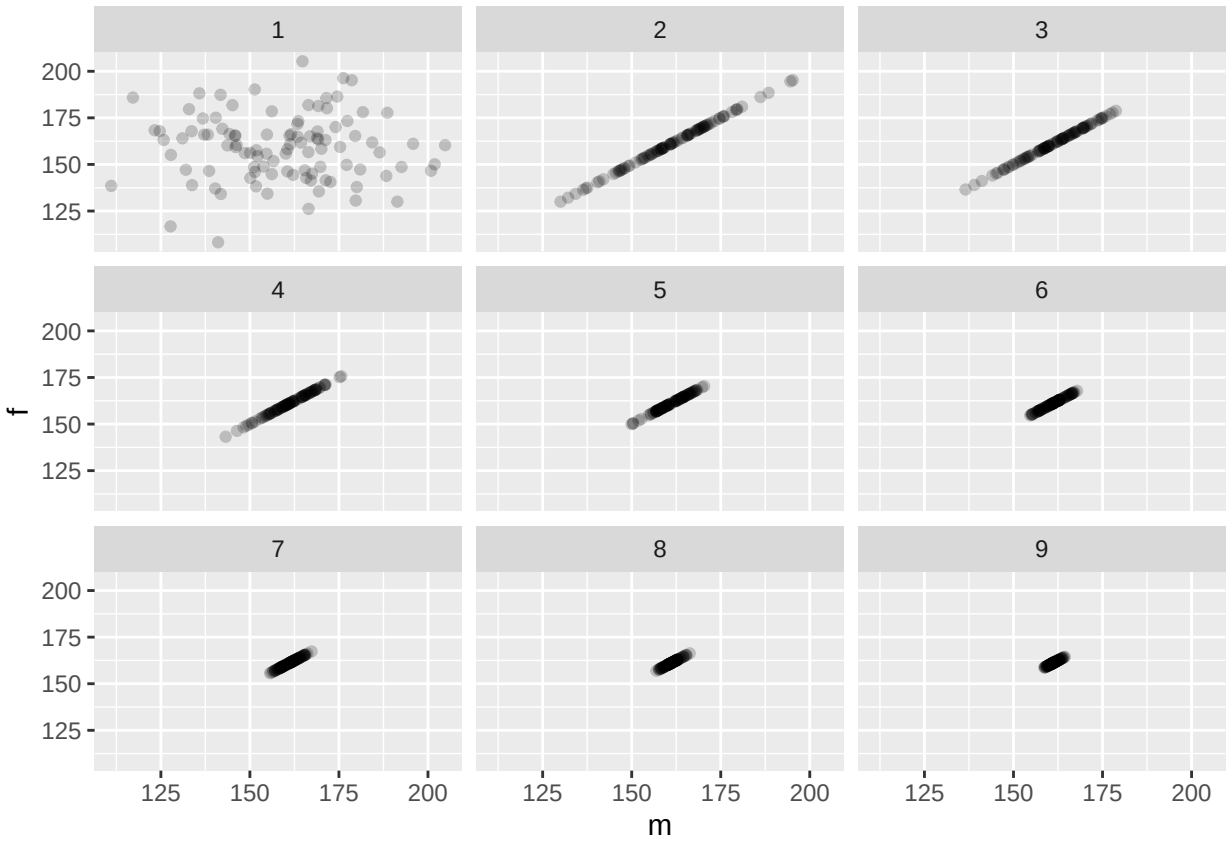


Figure 1: generations plot

```
ggplot(data = merged_pop, aes(x = m, y = f)) +
  geom_point(alpha = 0.2) +
  facet_wrap(~generation)
```



Question 3

15 points

You calculated the power of a study design in question #1 of assignment 3. The study has two variables, treatment group and outcome. There are two treatment groups (0, 1) and they should be assigned randomly with equal probability. The outcome should be a random normal variable with a mean of 60 and standard deviation of 20. If a patient is in the treatment group, add 5 to the outcome.

Starting with a sample size of 250, create a 95% bootstrap percentile interval for the mean of each group. Then create a new bootstrap interval by increasing the sample size by 250 until the sample is 2500. Thus you will create a total of 10 bootstrap intervals. Each bootstrap should create 1000 bootstrap samples. (9 points)

Produce a line chart that includes the bootstrapped mean and lower and upper percentile intervals for each group. Add appropriate labels and a legend. (6 points)

You may use base graphics or ggplot2. It should look similar to this (in base).

Here's an example of how you could create transparent shaded areas.

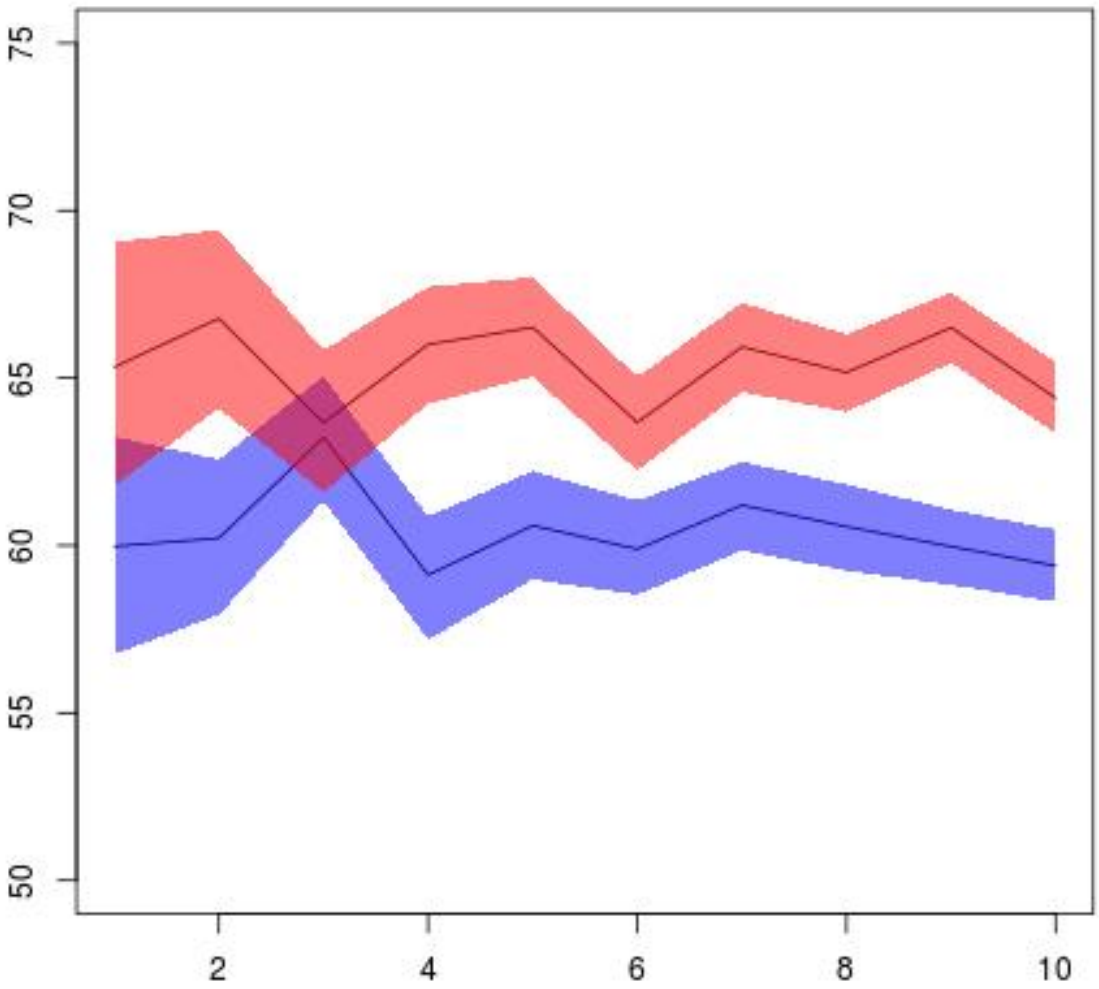
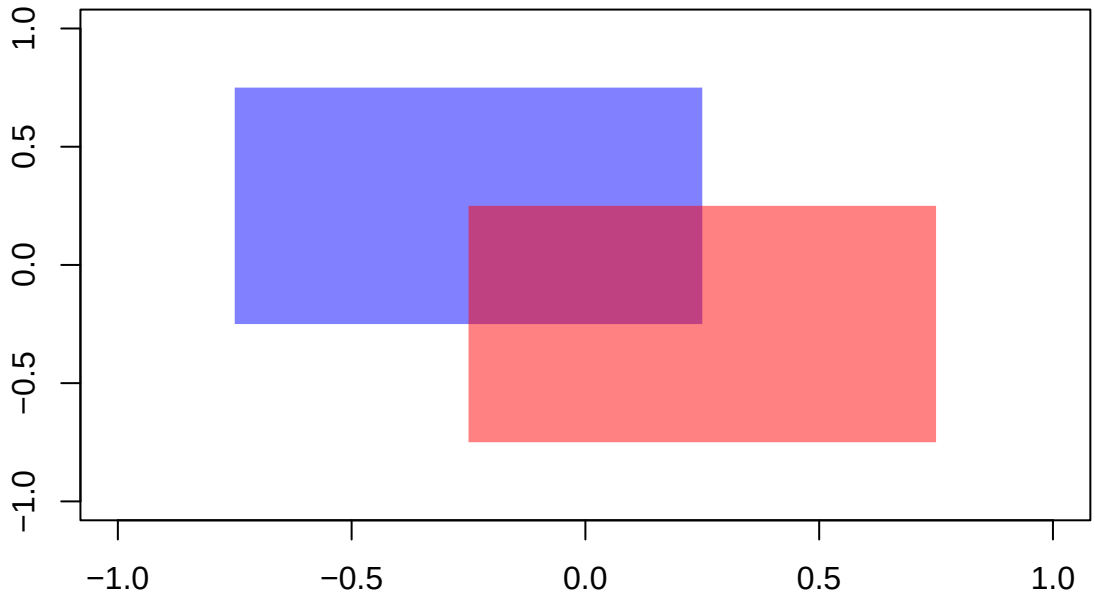
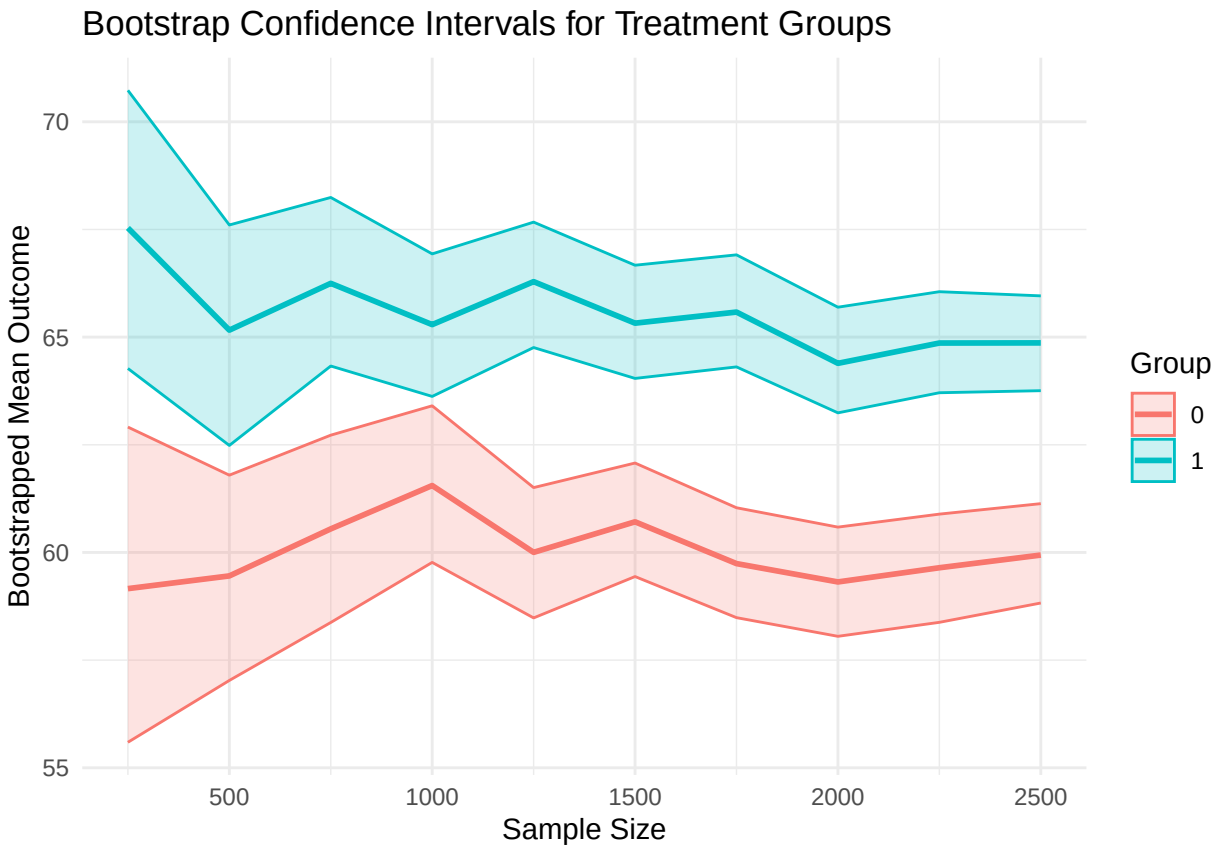


Figure 2: bp interval plot

```r
makeTransparent = function(..., alpha=0.5) {
  if(alpha<0 | alpha>1) stop("alpha must be between 0 and 1")
  alpha = floor(255*alpha)
  newColor = col2rgb(col=unlist(list(...)), alpha=FALSE)
  .makeTransparent = function(col, alpha) {
    rgb(red=col[1], green=col[2], blue=col[3], alpha=alpha, maxColorValue=255)
  }
  newColor = apply(newColor, 2, .makeTransparent, alpha=alpha)
  return(newColor)
}

par(new=FALSE)
plot(NULL,
  xlim=c(-1, 1),
  ylim=c(-1, 1),
  xlab="",
  ylab=""
)

polygon(x=c(seq(-0.75, 0.25, length.out=100), seq(0.25, -0.75, length.out=100)),
        y=c(rep(-0.25, 100), rep(0.75, 100)), border=NA, col=makeTransparent('blue',alpha=0.5))
polygon(x=c(seq(-0.25, 0.75, length.out=100), seq(0.75, -0.25, length.out=100)),
        y=c(rep(-0.75, 100), rep(0.25, 100)), border=NA, col=makeTransparent('red',alpha=0.5))
```



```r
bootstrap_ci <- function(data, group_col, value_col, n_boot = 1000) {
  boot_means <- replicate(n_boot, {
    resample <- data[sample(nrow(data), replace = TRUE), ]
    tapply(resample[[value_col]], resample[[group_col]], mean)
  })
  ci <- apply(boot_means, 1, quantile, probs = c(0.025, 0.975))
  boot_mean <- rowMeans(boot_means)
  list(mean = boot_mean, ci = ci)
}


set.seed(123)
n_sizes <- seq(250, 2500, by = 250)
n_boot <- 1000


results <- data.frame()

for (n in n_sizes) {

  group <- sample(c(0, 1), size = n, replace = TRUE)
  outcome <- rnorm(n, mean = 60, sd = 20) + group * 5
  data <- data.frame(group = factor(group), outcome = outcome)


  boot_result <- bootstrap_ci(data, group_col = "group", value_col = "outcome", n_boot = n_boot)



  for (g in 1:2) {
    results <- rbind(results, data.frame(
      group = g - 1,
      sample_size = n,
      mean = boot_result$mean[g],
      lower = boot_result$ci[1, g],
      upper = boot_result$ci[2, g]
    ))
  }
}


ggplot(results, aes(x = sample_size, y = mean, group = factor(group), color = factor(group))) +
  geom_line(linewidth = 1) +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = factor(group)), alpha = 0.2) +
  labs(
    title = "Bootstrap Confidence Intervals for Treatment Groups",
    x = "Sample Size",
    y = "Bootstrapped Mean Outcome",
    color = "Group",
    fill = "Group"
  ) +
  theme_minimal()
```


Bootstrap Confidence Intervals for Treatment Groups
Bootstrapped Mean Outcome
70
65
60
55
500
1000
1500
2000
2500
Sample Size
Group
0
1