Bios 6301: Assignment 9

Catherine Greene

Due Tuesday, 29 November, 1:00 PM

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

40 points total.

Submit a single knitr file (named `homework9.rmd`), 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.rmd` or include author name may result in 5 points taken off.

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:

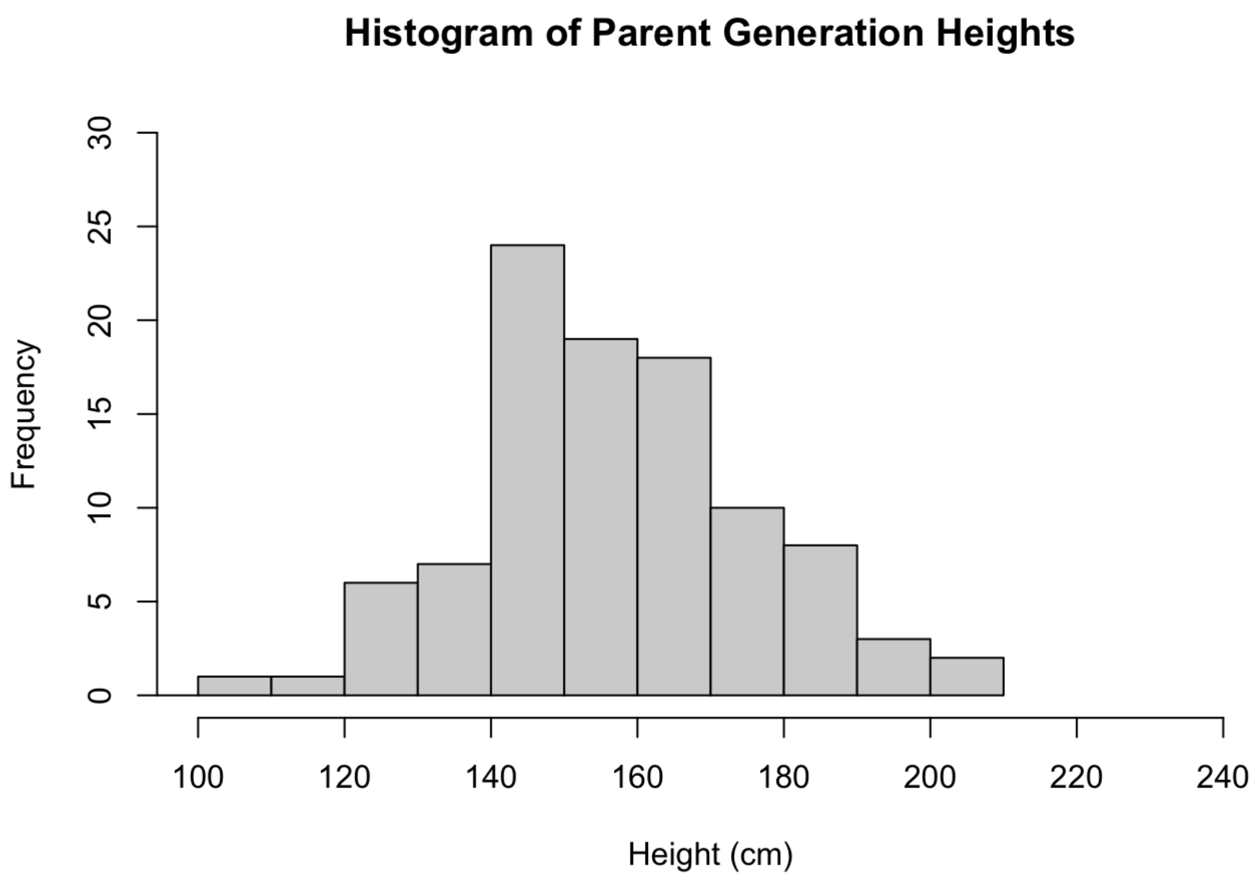
```
pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20)) # rnorm(n, mean, sd)

# m and f are paired at random
# each pair makes 2 offspring (1 m and 1 f)
# height of offspring = avg of parents
```

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.

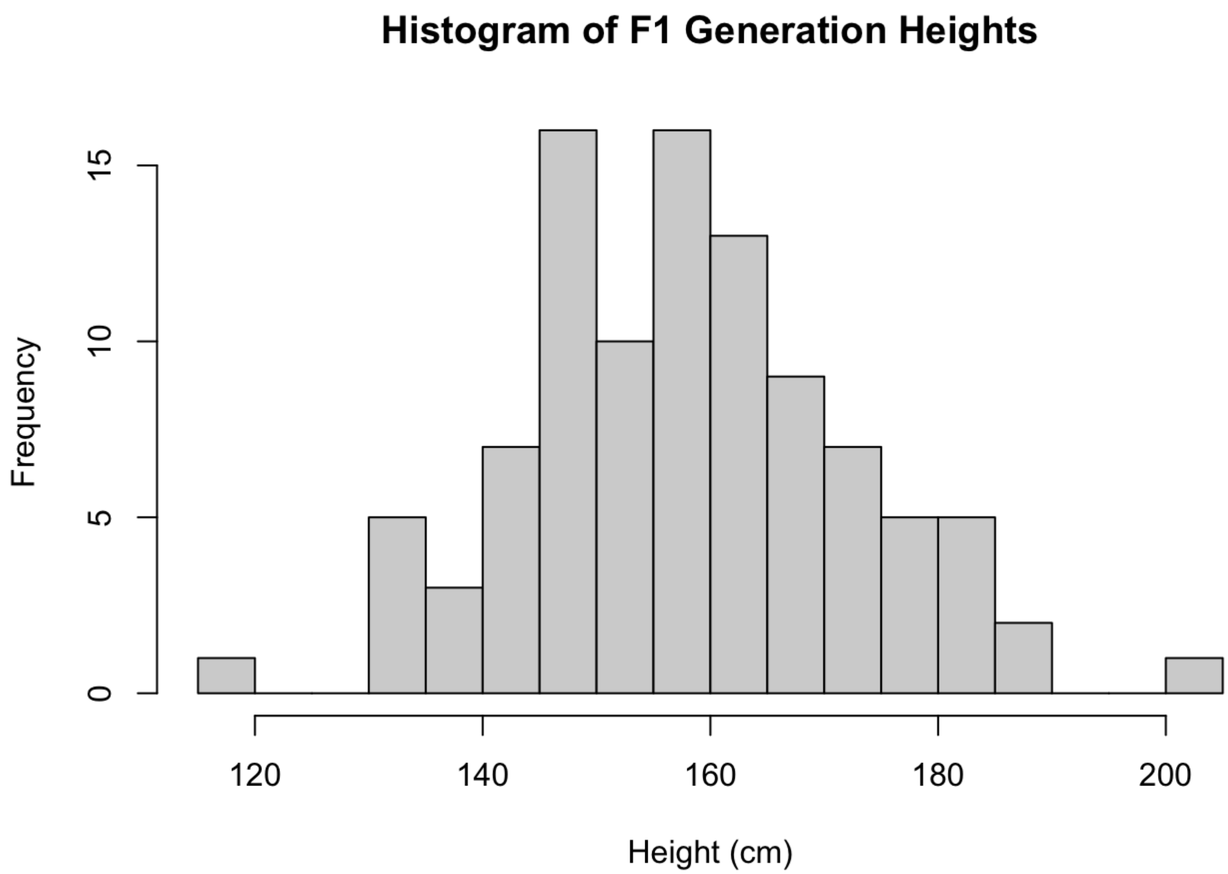
```
next_gen <- function(pop) {
    pop$m <- sample(pop$m) # randomly permutes order of the m
    pop$m <- rowMeans(pop) # finds mean of m & f (each row), defining it as the value fo
r the next generation
    pop$f <- pop$m # both m & f offspring have avg height of parents
    pop # returns a data frame with the same structure
}
```

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.

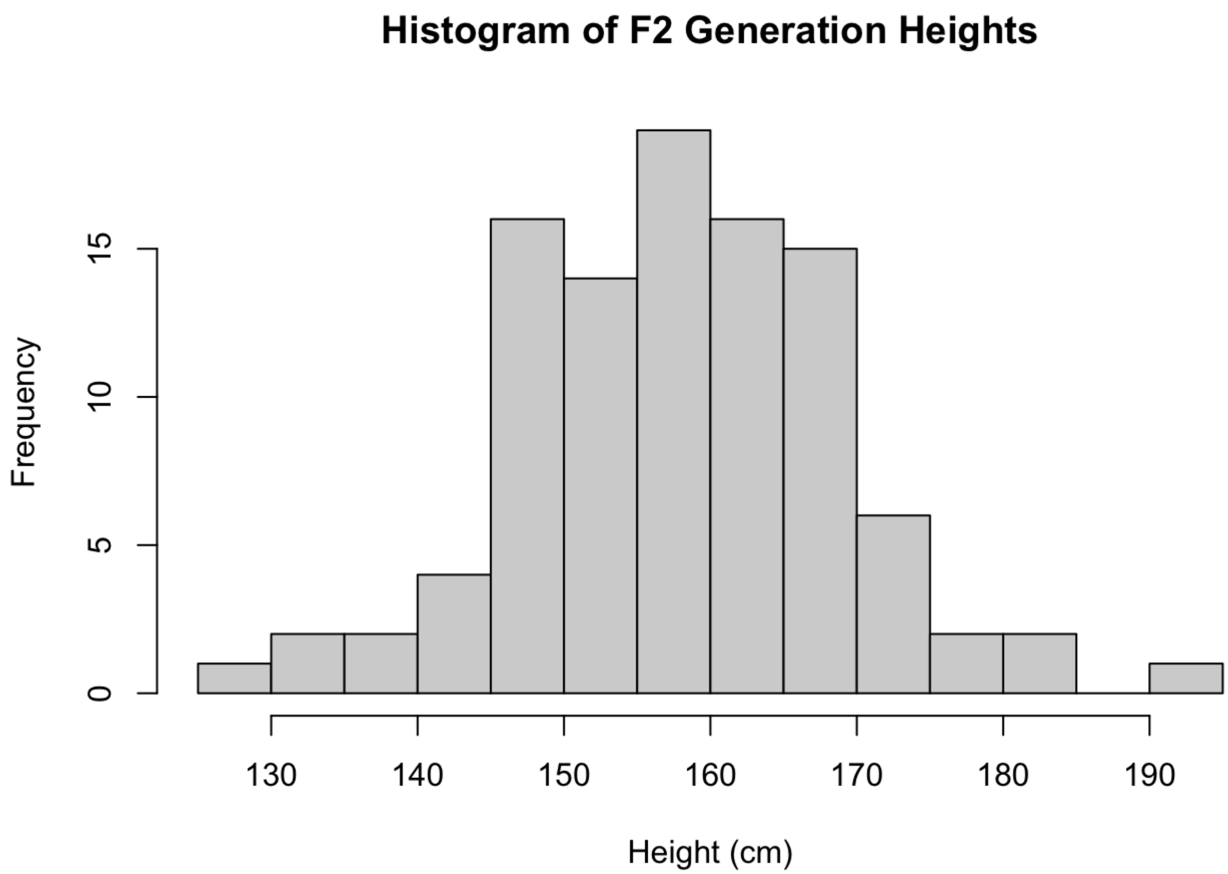
```
# make it a loop with a counter
# 9 generations, 9 - 1 = 8
# 1:8 to 2:9 so I can include the parent generation in the same list

f_gens <- list(pop) # list of data frames

# generate another 8 generations
for (i in 2:9) {
  f_gens[[i]] <- next_gen(f_gens[[(i-1)]]) # be careful to reference the previous genera
tion, not just the parent pop
}

# histograms
hist(f_gens[[1]]$m, breaks=15, xlim=range(100,240), ylim=range(0,30), main='Histogram of
Parent Generation Heights', xlab='Height (cm)')
```



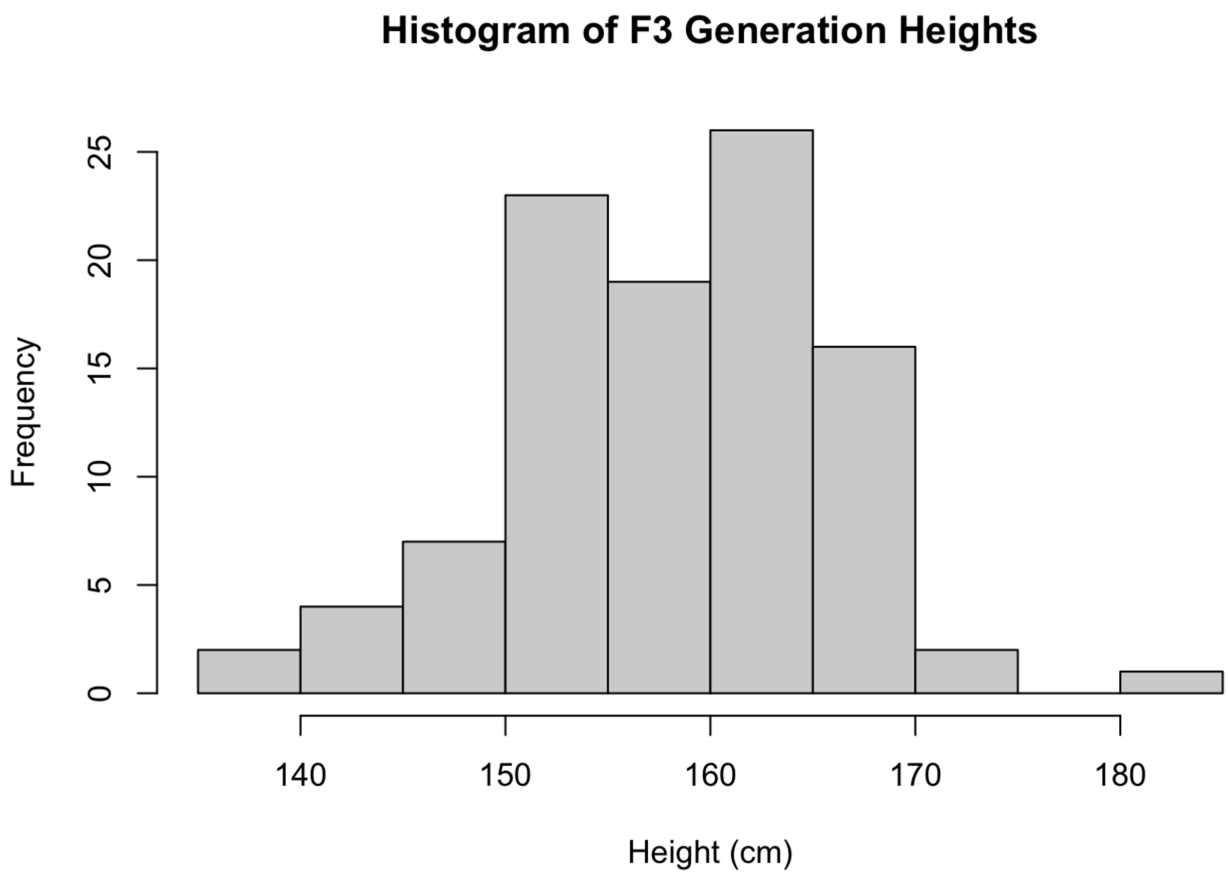

```
hist(f_gens[[2]]$m, breaks=15, main='Histogram of F1 Generation Heights', xlab='Height
 (cm)')
```



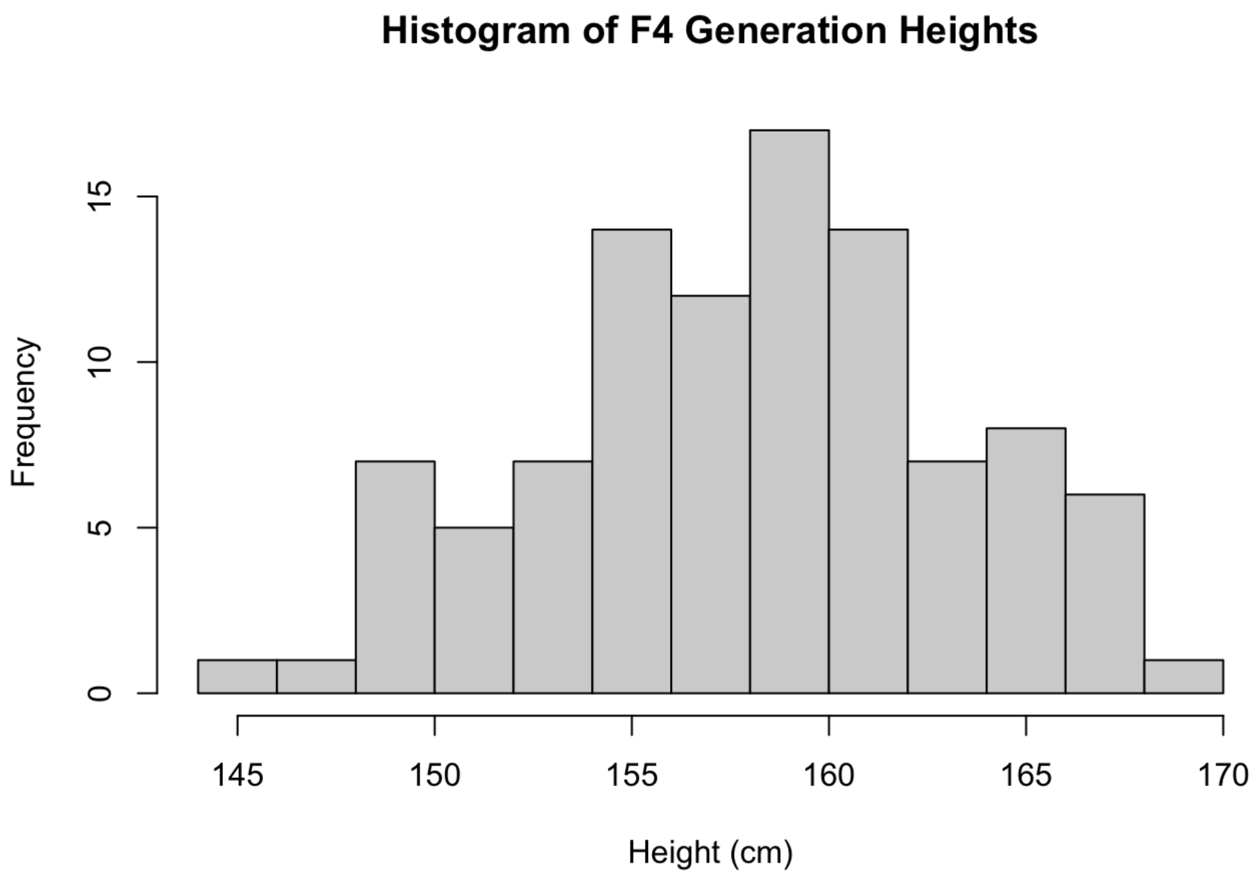

```
hist(f_gens[[3]]$m, breaks=15, main='Histogram of F2 Generation Heights', xlab='Height
 (cm)')
```



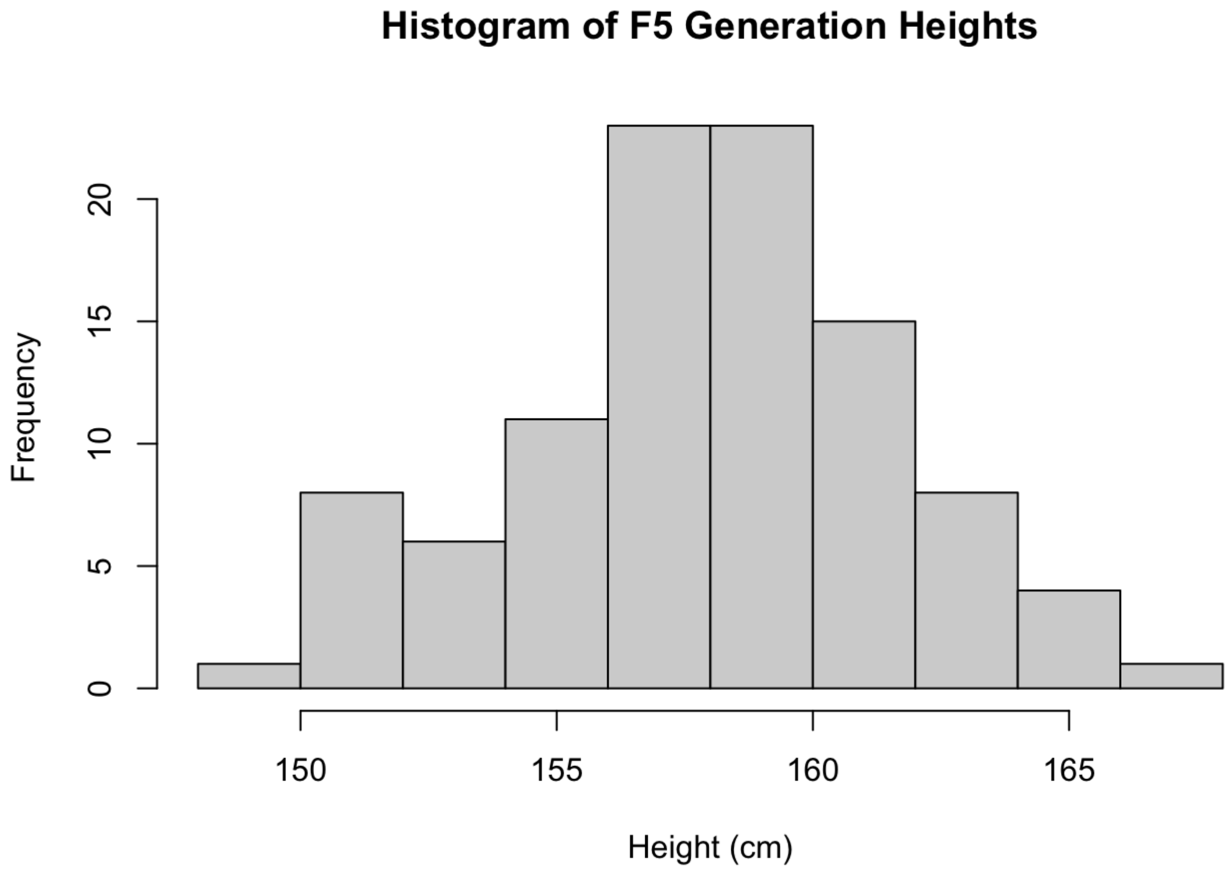

```
hist(f_gens[[4]]$m, breaks=13, main='Histogram of F3 Generation Heights', xlab='Height
 (cm)')
```



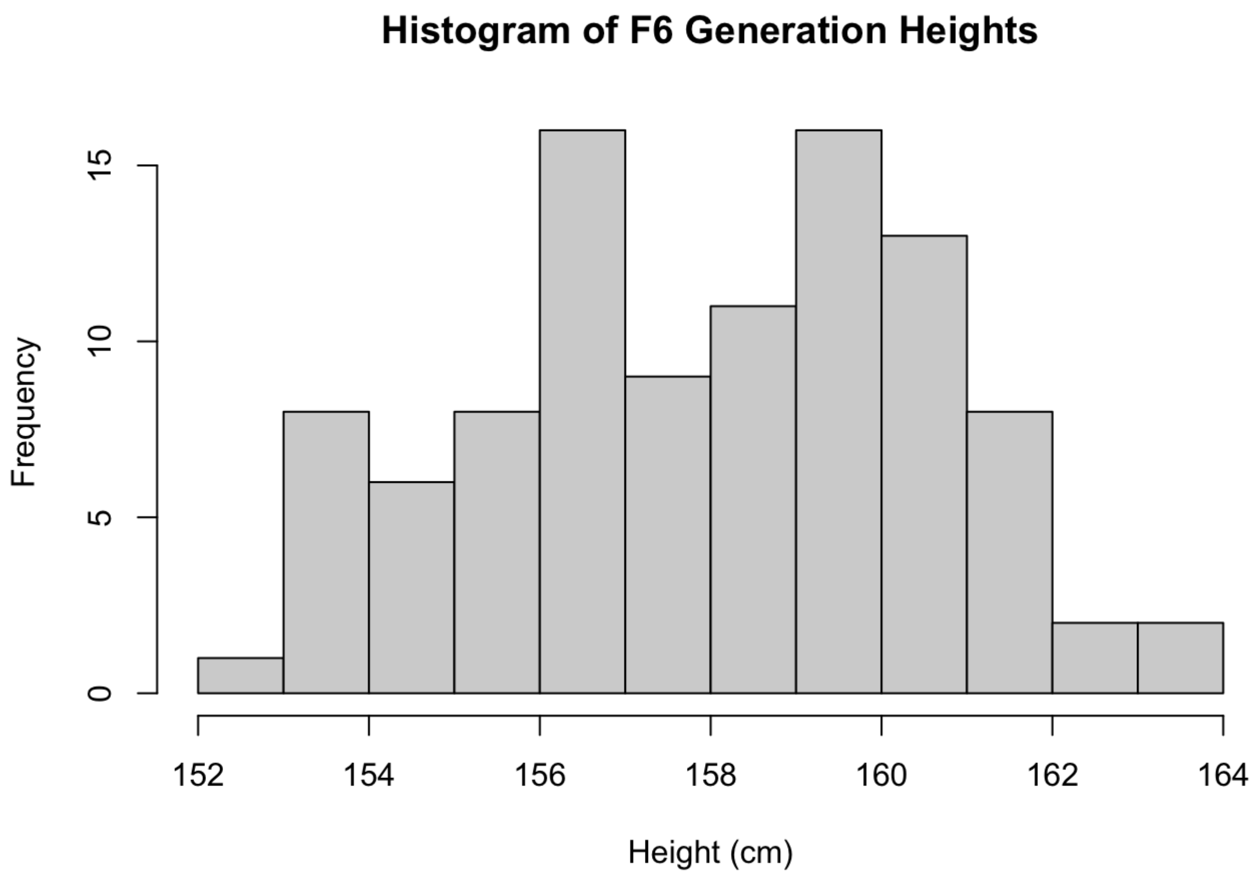

```
hist(f_gens[[5]]$m, breaks=10, main='Histogram of F4 Generation Heights', xlab='Height
 (cm)')
```



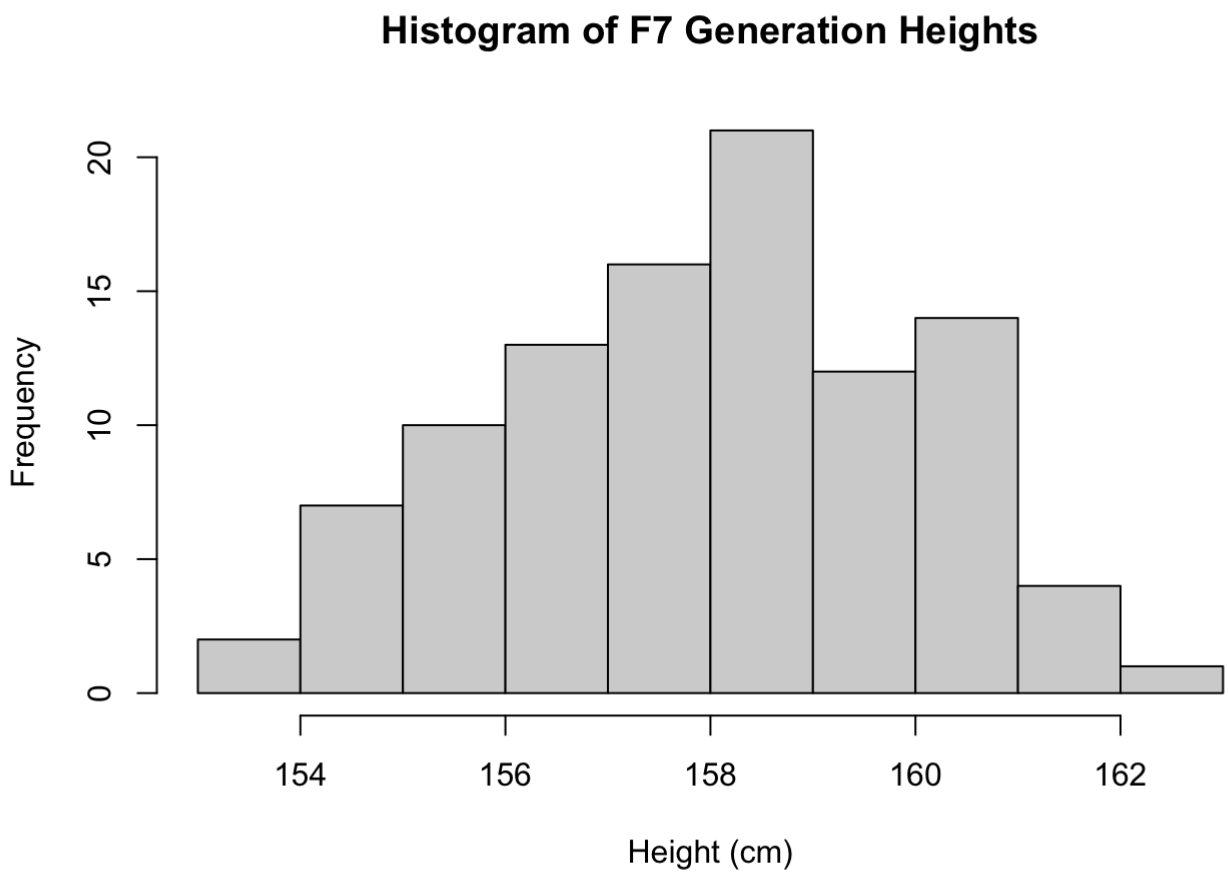

```
hist(f_gens[[6]]$m, breaks=10, main='Histogram of F5 Generation Heights', xlab='Height
 (cm)')
```



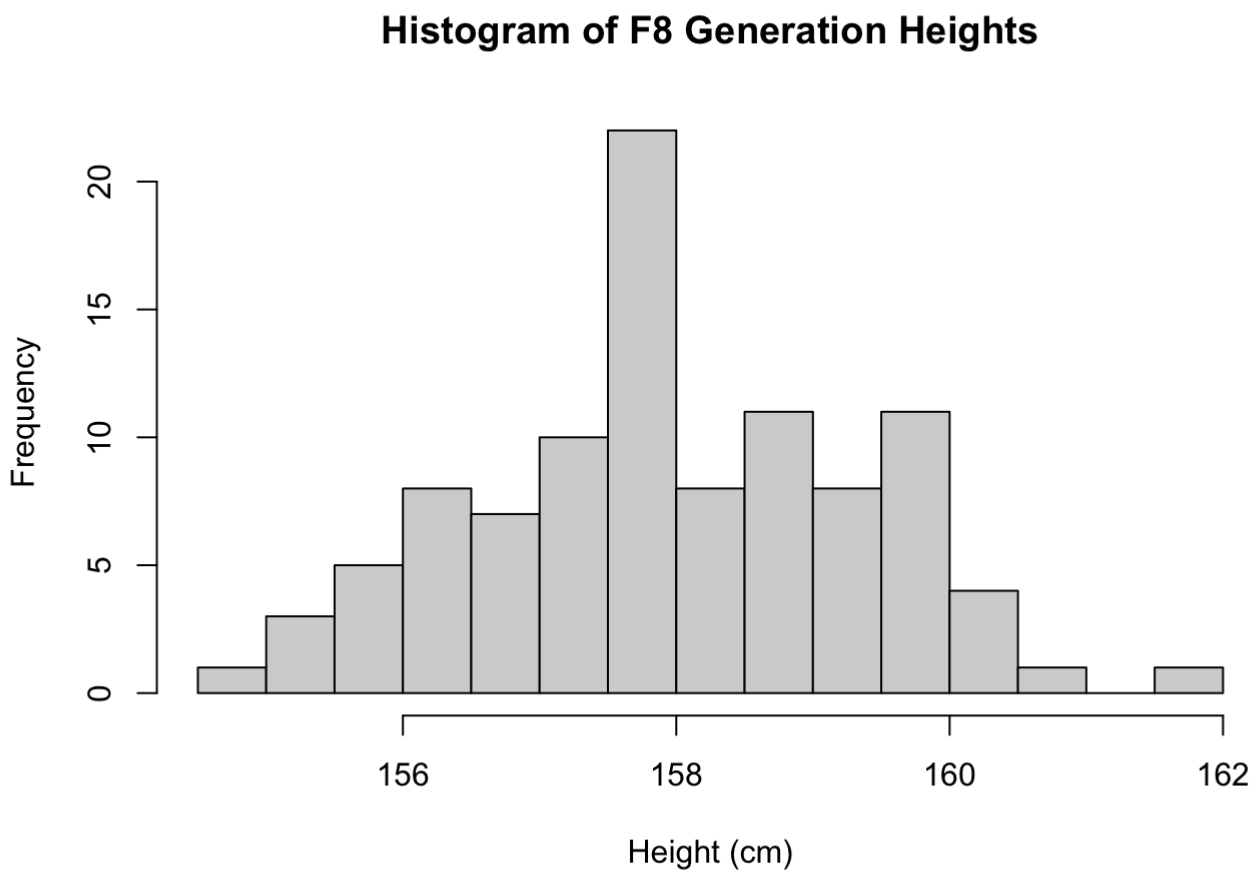

```
hist(f_gens[[7]]$m, breaks=15, main='Histogram of F6 Generation Heights', xlab='Height
 (cm)')
```




```
hist(f_gens[[8]]$m, breaks=13, main='Histogram of F7 Generation Heights', xlab='Height
 (cm)')
```



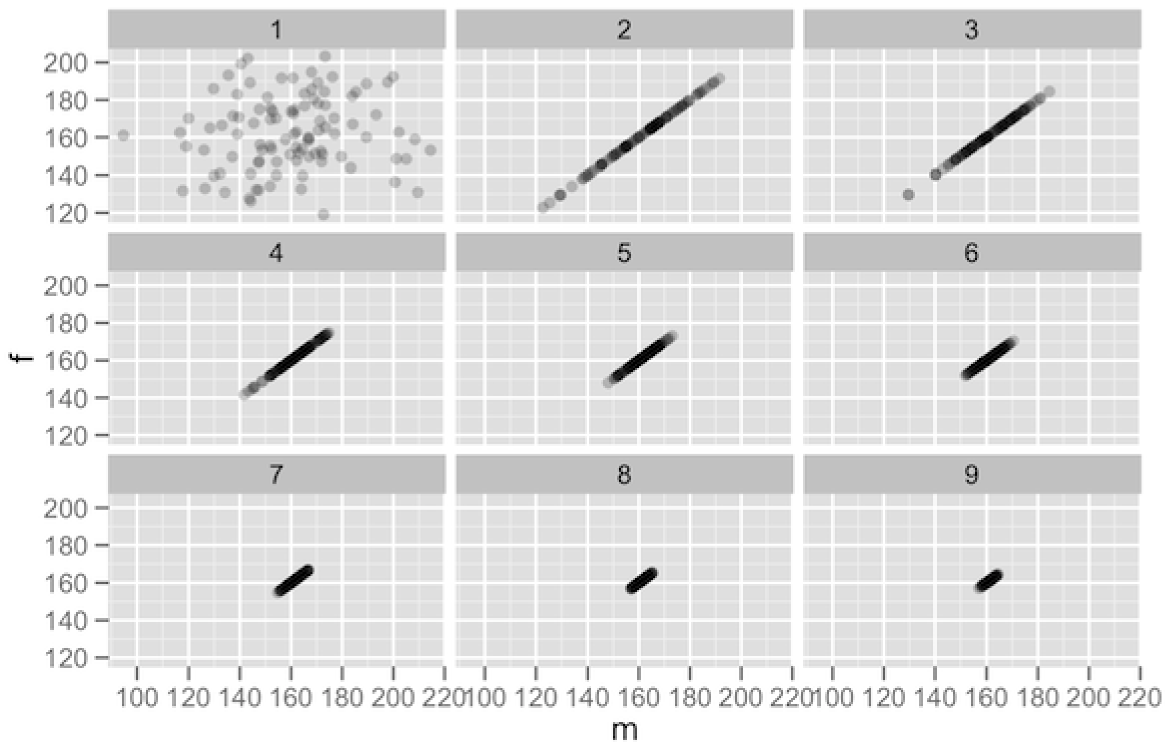

```
hist(f_gens[[9]]$m, breaks=10, main='Histogram of F8 Generation Heights', xlab='Height
 (cm)')
```



Question 2

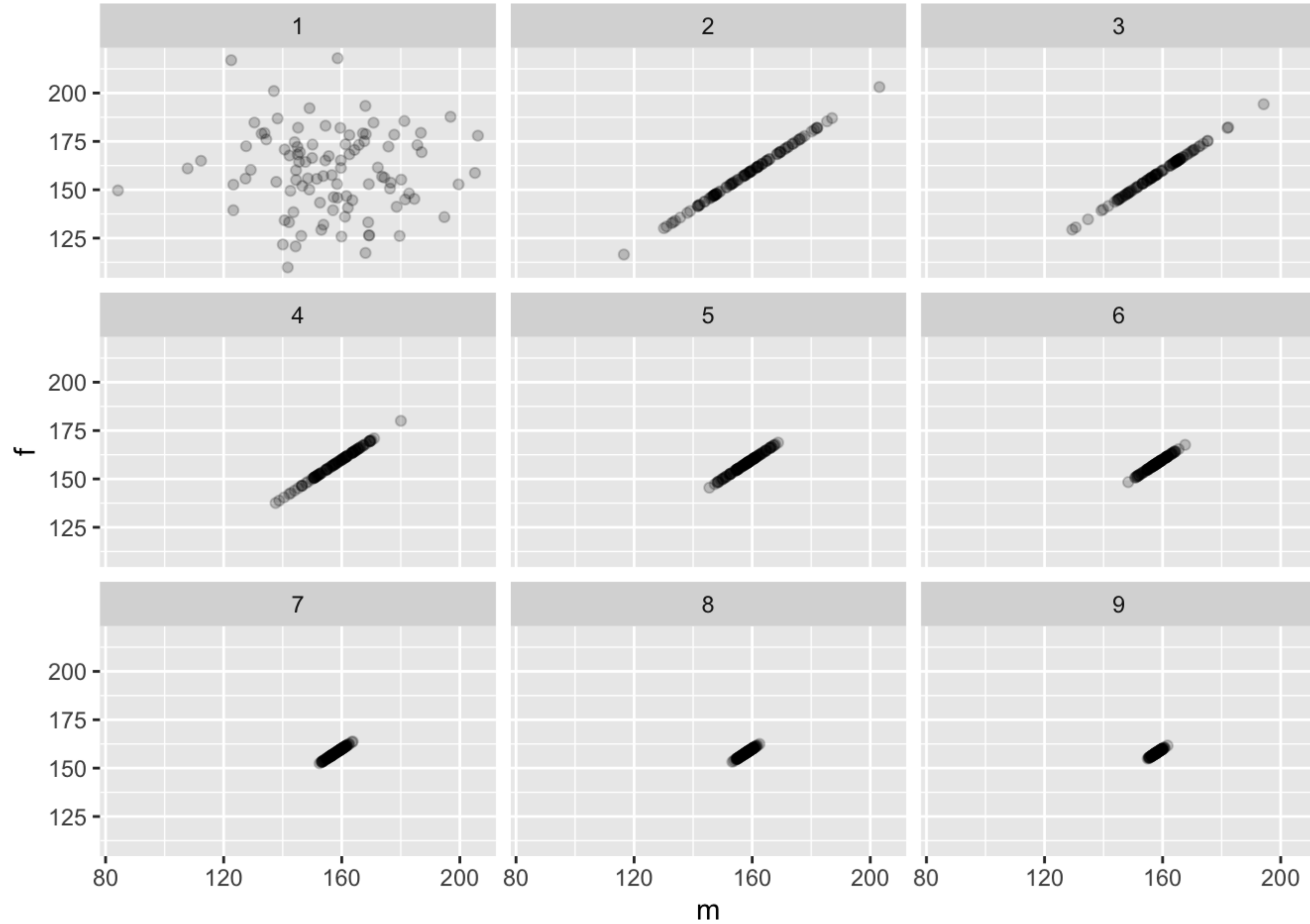
10 points

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



generations plot

```
library(ggplot2)
# make a generation variable, I need everything in one dataframe
gen_count <- rep(c(seq(1, 9)), each=100)

multi_gen <- rbind(f_gens[[1]], f_gens[[2]], f_gens[[3]], f_gens[[4]], f_gens[[5]], f_ge
ns[[6]], f_gens[[7]], f_gens[[8]], f_gens[[9]])

total <- cbind(multi_gen, gen_count)

# scatter plot
# all 9 are displayed on 1 device, 3x3
# m along x axis, f along y axis

p = ggplot(data=total) + geom_point(mapping = aes(x=m, y=f), alpha = 0.2)
p + facet_wrap(~ gen_count)
```



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)

```
# power function from assignment 3
power <- function(sample_size) {
  mean(replicate(1000, {
  treatment <- rbinom(sample_size, 1, 0.5)
  outcome <- rnorm(sample_size, mean=60, sd=20)
  for (i in 1:sample_size) {
    if (treatment[i]==1){
      outcome[i] <- outcome[i] + 5}
    }
  t.test(outcome ~ treatment, alternative='two.sided', mu=0)$p.value
  }) < 0.05)
}

# now working on the homework problem

# another attempt at bootstrapping
sim_sample <- function (sample_size) {
  treatment <- rbinom(sample_size, 1, 0.5)
  outcome <- rnorm(sample_size, mean=60, sd=20)
      for (i in 1:sample_size) {
        if (treatment[i]==1){
          outcome[i] <- outcome[i] + 5
        }
      }
  df <- data.frame(treatment, outcome)
  df
}

boot_interval <- function(sample_size) {
  x <- sim_sample(sample_size)
  # set up the empty matrix
  bootmean <- matrix(nrow=1000, ncol=2)

  # bootstrapping
  for(i in 1:nrow(bootmean)) {
    boot.x <- x[sample(nrow(x), nrow(x), replace=TRUE),] # bootstrap sample
    controls <- boot.x[which(boot.x[,1]==as.integer(0)),][,2]
    exp <- boot.x[which(boot.x[,1]==as.integer(1)),][,2]
    bootmean[i,] <- c(mean(controls), mean(exp))
  }

  # separate the control and treatment groups, and find the mean of each
  controls <- bootmean[,1]
  exp <- bootmean[,2]
  means <- c(mean(controls), mean(exp))
  # find the 5% and 95% bootstrap intervals for the mean of each group
  control_interval <- means[1] + qnorm(c(0.05, 0.95)) * sd(controls) #/ sqrt(length(cont
rols))
  exp_interval <- means[2] + qnorm(c(0.05, 0.95)) * sd(exp) #/ sqrt(length(exp))

  # get set up for the data frame - these will both be columns
  low_ci <- c(control_interval[1], exp_interval[1])
```

```
  high_ci <- c(control_interval[2], exp_interval[2])
  # data frame
  boot_data <- data.frame(
    means,
    low_ci, # low, rows still match up
    high_ci, # high
    rep(sample_size, 2),
    as.factor(c(0,1)) # controls first, then treatment group
  )
  names(boot_data) <- c('means', 'low_ci', 'high_ci', 'sample_size', 'treatment') # I wa
nt the names of columns to all make sense
  boot_data
}

boot_interval(250) # yay! It works
```

means <dbl>	low_ci <dbl>	high_ci <dbl>	sample_size <dbl>	treatment <fct>
60.71968	57.85433	63.58502	250	0
64.61281	61.84197	67.38364	250	1

2 rows

```
# on to running with different sample sizes!
boot_list <- seq(from=250, to=2500, by=250) # now to increase the sample size

# I originally wanted to combine these into one dataframe using a loop, but I was having
trouble getting it to work
final <- rbind(boot_interval(250), boot_interval(500), boot_interval(750), boot_interval
(1000), boot_interval(1250), boot_interval(1500), boot_interval(1750), boot_interval(200
0), boot_interval(2250), boot_interval(2500))

final # all info in one dataframe, with a column clarifying treatment vs control
```

means <dbl>	low_ci <dbl>	high_ci <dbl>	sample_size <dbl>	treatment <fct>
58.01292	54.91381	61.11203	250	0
65.25418	62.37914	68.12923	250	1
58.92446	56.79887	61.05006	500	0
66.96534	64.78807	69.14260	500	1
60.82545	59.06799	62.58290	750	0
65.52576	63.86911	67.18241	750	1
61.07669	59.44517	62.70820	1000	0
65.88155	64.42844	67.33465	1000	1

means	low_ci	high_ci	sample_size	treatment
<dbl>	<dbl>	<dbl>	<dbl>	<fct>
60.65178	59.29961	62.00395	1250	0
65.34606	64.13972	66.55240	1250	1

1-10 of 20 rows Previous **1** 2 Next

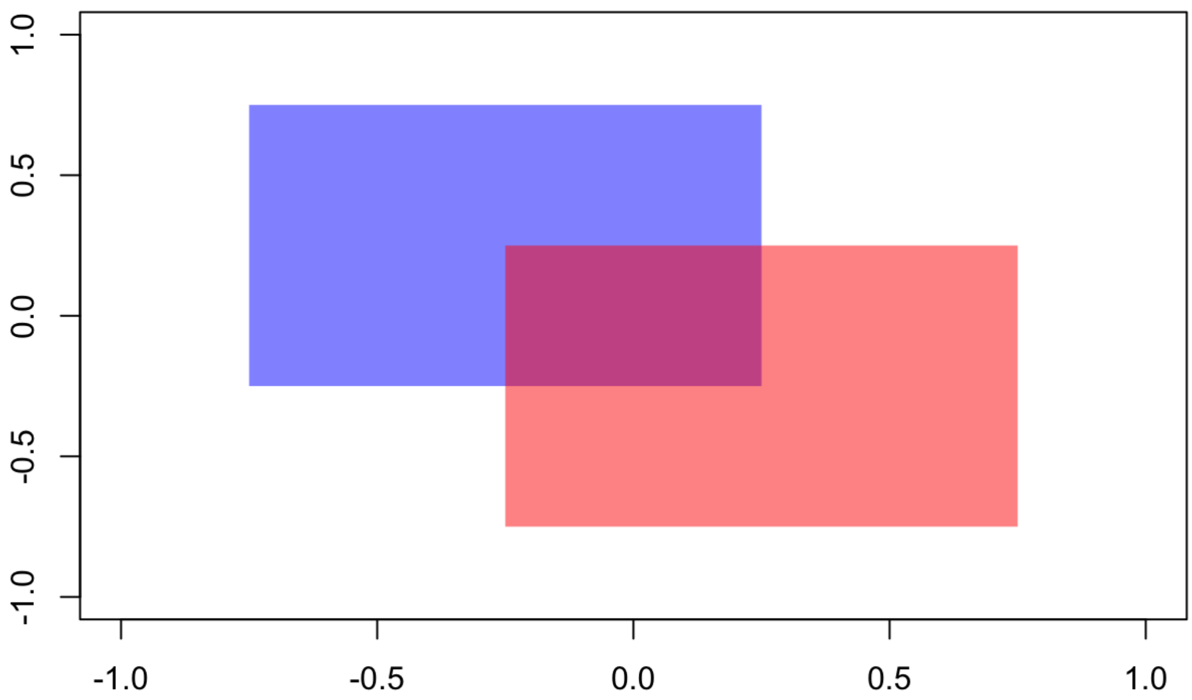
Produce a line chart that includes the bootstrapped mean and lower and upper percentile intervals for each group. Add appropriate labels and a legend. (6 points)

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

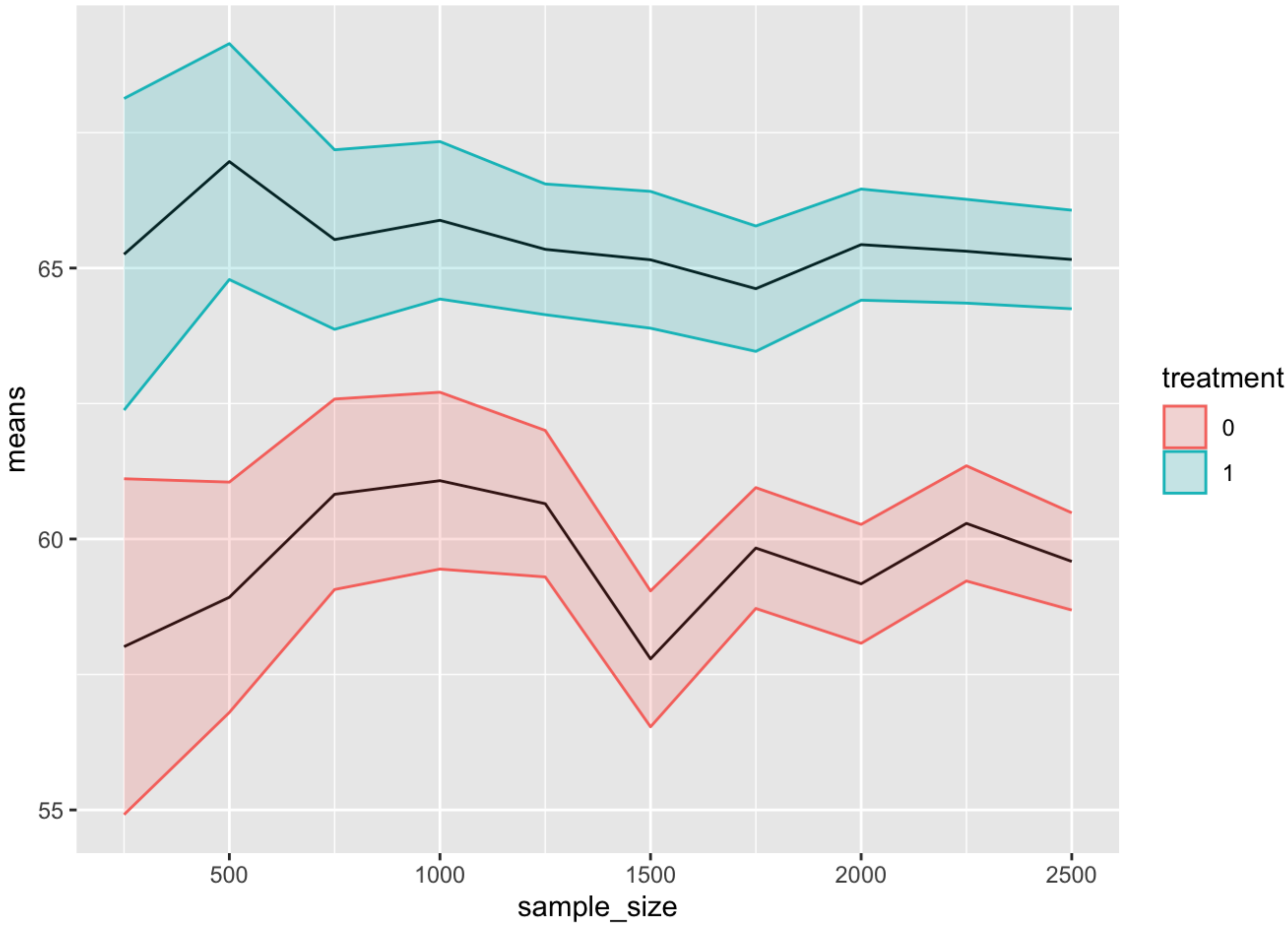
bp interval plot

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

```
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',alph
a=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
library(ggplot2)

ggplot(data=final, aes(x=sample_size, y=means, group=treatment, color=treatment, fill=tr
eatment)) +
  geom_line(col="black") +
  geom_ribbon(aes(ymin=low_ci, ymax=high_ci), alpha=0.2)
```



means
65
60
55
500
1000
1500
2000
2500
sample_size
treatment
0
1