Bios 6301: Assignment 9

Yan Yan

Due Thursday, 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:

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

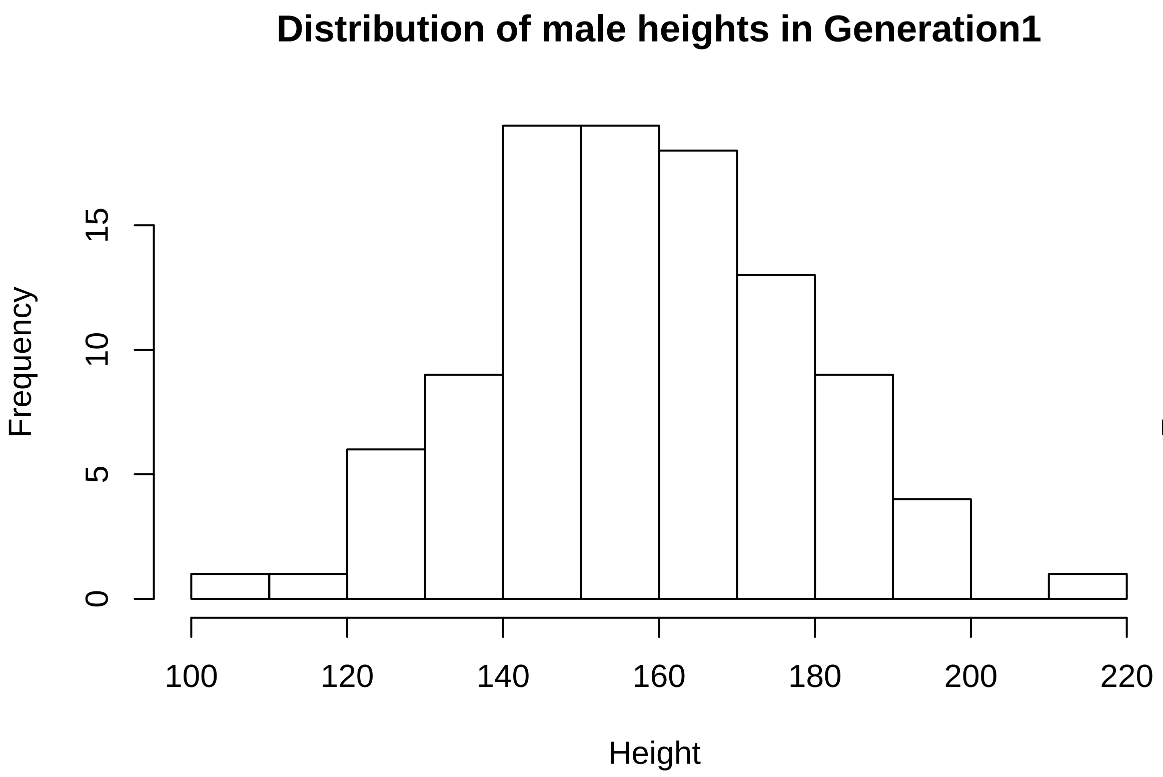
The following function takes the data frame `pop` and randomly permutes the ordering of the men. Men and women are then paired according to rows, and heights for the next generation are calculated by taking the mean of each row. The function returns a data frame with the same structure, giving the heights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

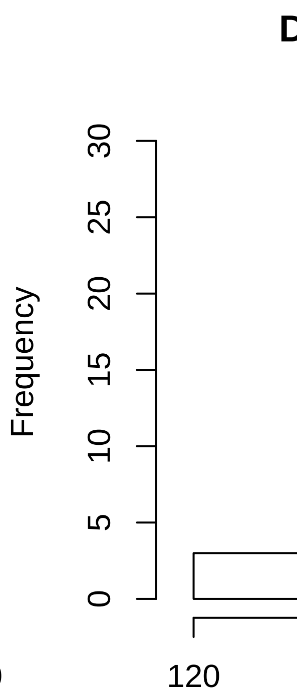
```r
next_gen <- function(pop) {
    pop$m <- sample(pop$m)
    pop$m <- rowMeans(pop)
    pop$f <- pop$m
    pop
}
```

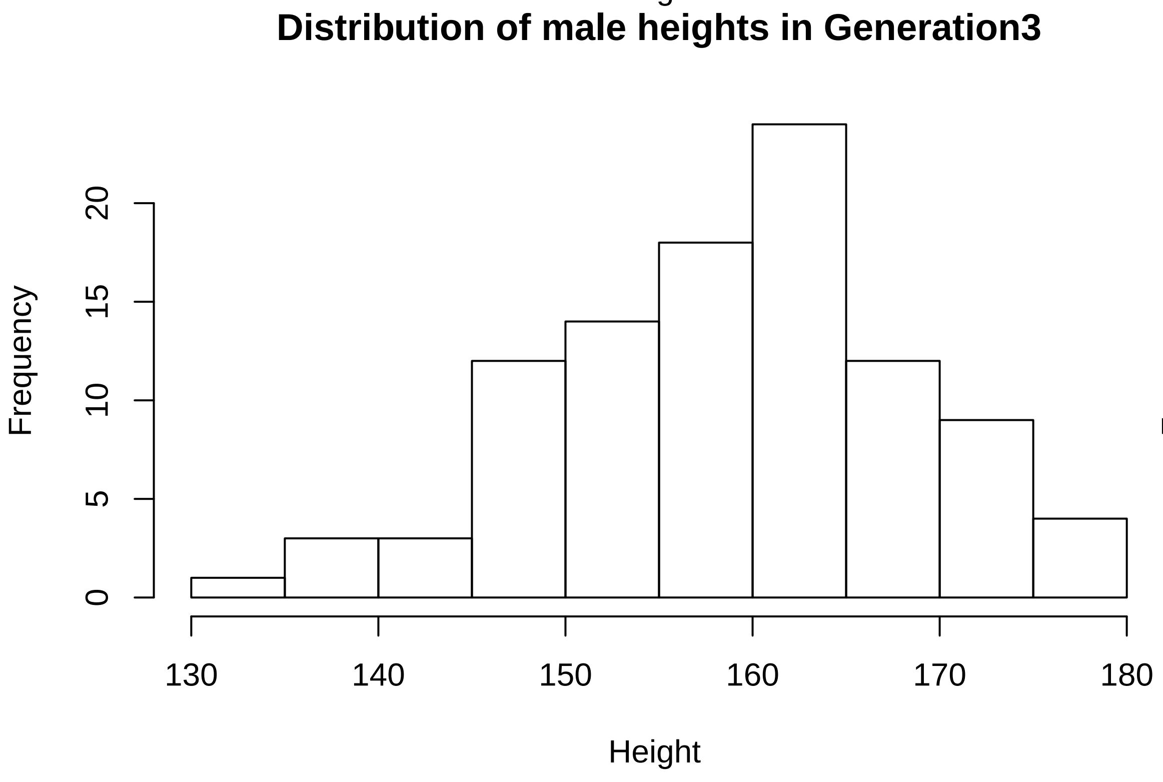
Use the function `next_gen` to generate nine generations (you already have the first), then use the function `hist` to plot the distribution of male heights in each generation (this will require multiple calls to `hist`). The phenomenon you see is called regression to the mean. Provide (at least) minimal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

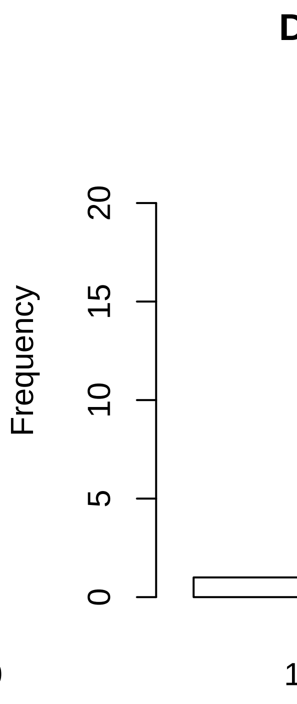
```r
pop2 <- list()
pop2[[1]] <- pop
for(i in 1:8){
  gnext <- next_gen(pop2[[i]])
  pop2[[i+1]] <- gnext
}
lapply(1:length(pop2),function(x){
```

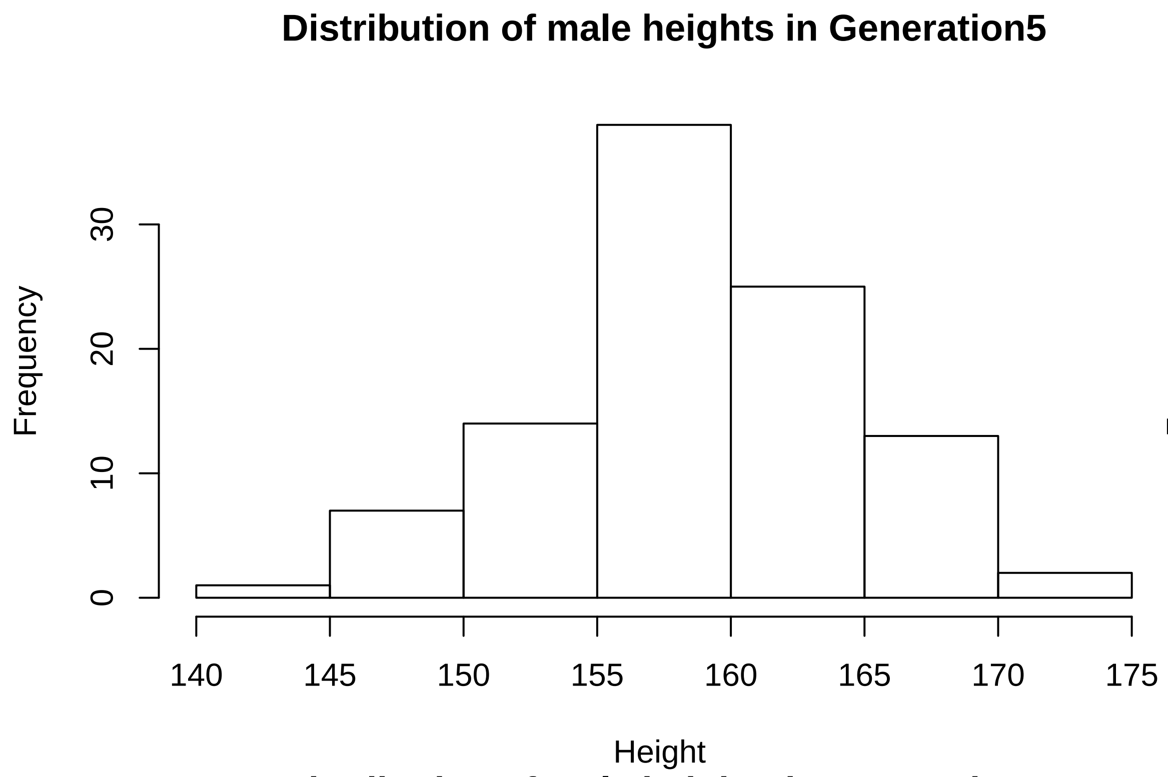
```
hist(pop2[[x]]$m, main = paste("Distribution of male heights in Generation",x, sep = ''), xlab = "Hei
})
```

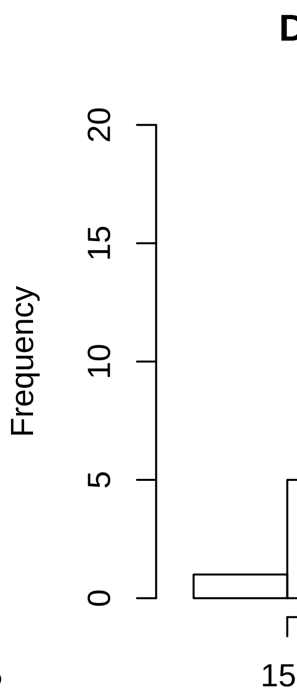


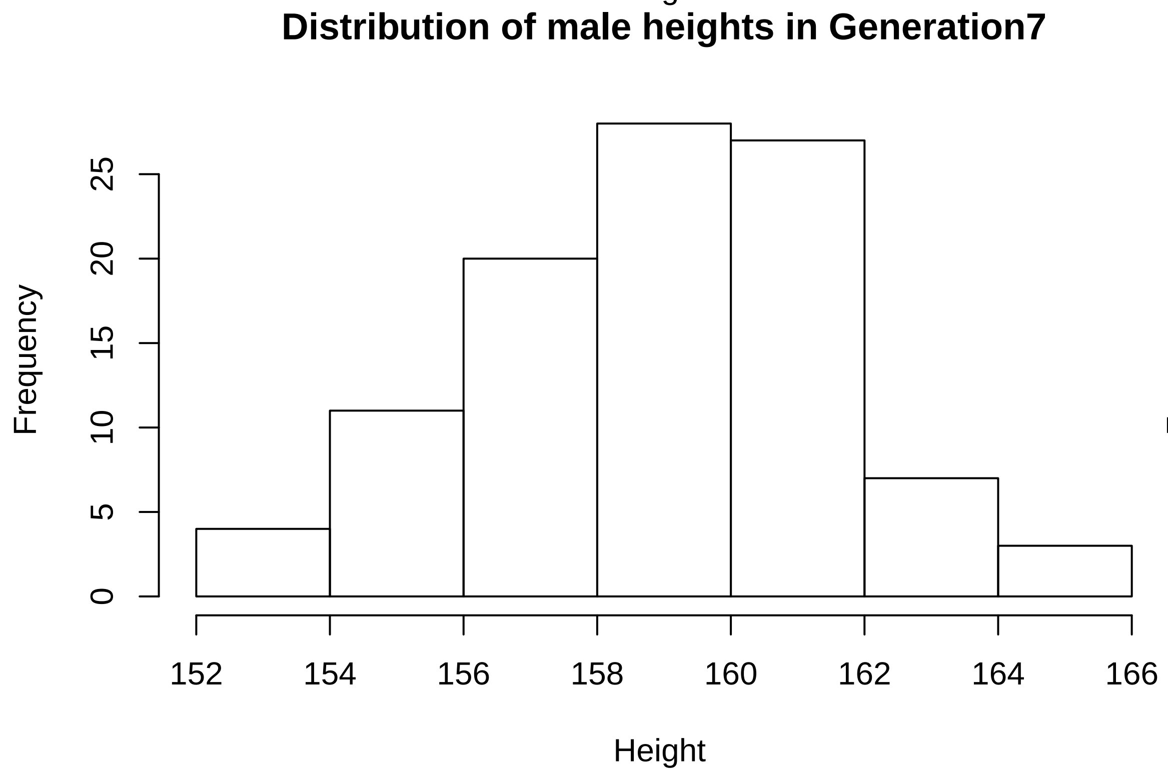


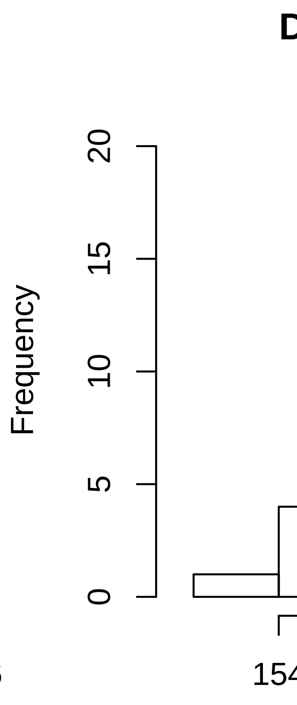


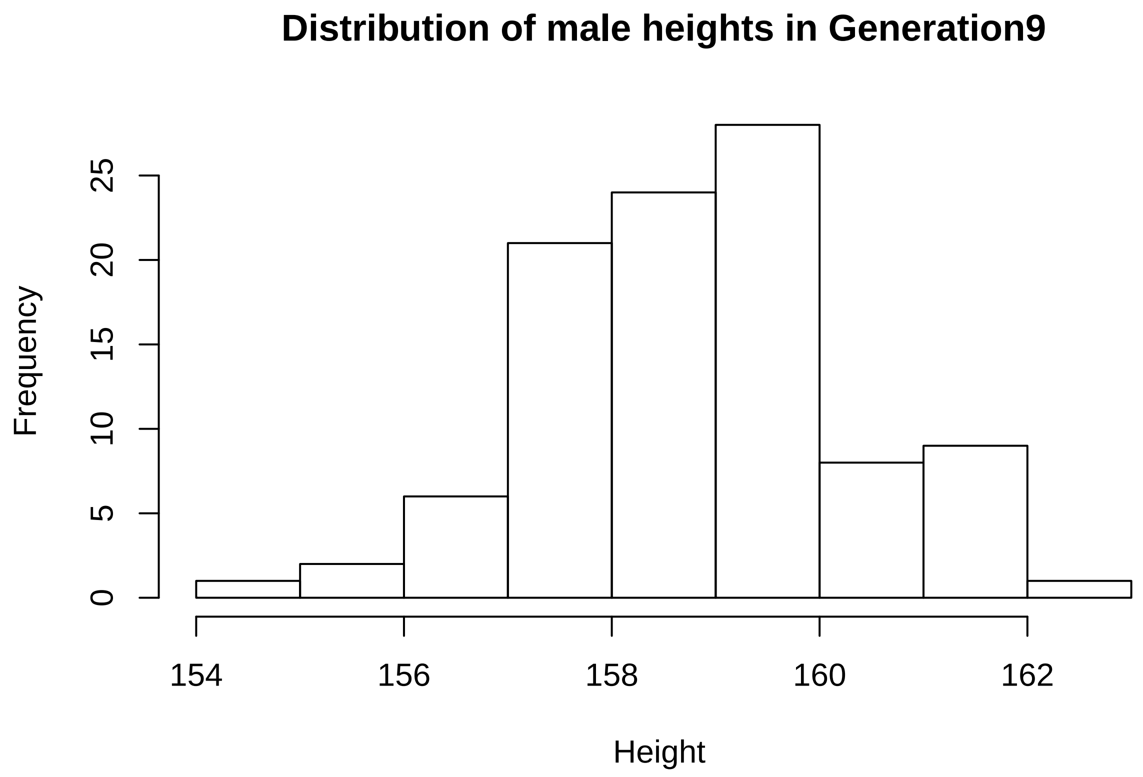



Distribution of male heights in Generation5
Frequency
0
10
20
30
140
145
150
155
160
165
170
175
Height


Frequency
0
5
10
15
20


Distribution of male heights in Generation7
Frequency
0
5
10
15
20
25
152
154
156
158
160
162
164
166
Height


Frequency
0
5
10
15
20



```
## [[1]]
## $breaks
##  [1] 100 110 120 130 140 150 160 170 180 190 200 210 220
##
## $counts
##  [1]  1  1  6  9 19 19 18 13  9  4  0  1
##
## $density
##  [1] 0.001 0.001 0.006 0.009 0.019 0.019 0.018 0.013 0.009 0.004 0.000
## [12] 0.001
##
## $mids
##  [1] 105 115 125 135 145 155 165 175 185 195 205 215
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[2]]
## $breaks
## [1] 120 130 140 150 160 170 180 190
##
## $counts
## [1]  3  8 16 23 31 12  7
```

```
##
## $density
## [1] 0.003 0.008 0.016 0.023 0.031 0.012 0.007
##
## $mids
## [1] 125 135 145 155 165 175 185
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[3]]
## $breaks
##  [1] 130 135 140 145 150 155 160 165 170 175 180
##
## $counts
##  [1]  1  3  3 12 14 18 24 12  9  4
##
## $density
##  [1] 0.002 0.006 0.006 0.024 0.028 0.036 0.048 0.024 0.018 0.008
##
## $mids
##  [1] 132.5 137.5 142.5 147.5 152.5 157.5 162.5 167.5 172.5 177.5
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[4]]
## $breaks
## [1] 135 140 145 150 155 160 165 170 175
##
## $counts
## [1]  1  1 14 16 22 24 16  6
##
## $density
## [1] 0.002 0.002 0.028 0.032 0.044 0.048 0.032 0.012
##
## $mids
## [1] 137.5 142.5 147.5 152.5 157.5 162.5 167.5 172.5
##
## $xname
## [1] "pop2[[x]]$m"
##
```

```
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[5]]
## $breaks
## [1] 140 145 150 155 160 165 170 175
##
## $counts
## [1]  1  7 14 38 25 13  2
##
## $density
## [1] 0.002 0.014 0.028 0.076 0.050 0.026 0.004
##
## $mids
## [1] 142.5 147.5 152.5 157.5 162.5 167.5 172.5
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[6]]
## $breaks
##  [1] 148 150 152 154 156 158 160 162 164 166 168
##
## $counts
##  [1]  1  5  6 13 17 16 20 12  7  3
##
## $density
##  [1] 0.005 0.025 0.030 0.065 0.085 0.080 0.100 0.060 0.035 0.015
##
## $mids
##  [1] 149 151 153 155 157 159 161 163 165 167
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[7]]
## $breaks
## [1] 152 154 156 158 160 162 164 166
##
```

```
## $counts
## [1]  4 11 20 28 27  7  3
##
## $density
## [1] 0.020 0.055 0.100 0.140 0.135 0.035 0.015
##
## $mids
## [1] 153 155 157 159 161 163 165
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[8]]
## $breaks
##  [1] 153 154 155 156 157 158 159 160 161 162 163 164
##
## $counts
##  [1]  1  4  6  7 12 18 21 16 11  3  1
##
## $density
##  [1] 0.01 0.04 0.06 0.07 0.12 0.18 0.21 0.16 0.11 0.03 0.01
##
## $mids
##  [1] 153.5 154.5 155.5 156.5 157.5 158.5 159.5 160.5 161.5 162.5 163.5
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## [[9]]
## $breaks
##  [1] 154 155 156 157 158 159 160 161 162 163
##
## $counts
## [1]  1  2  6 21 24 28  8  9  1
##
## $density
## [1] 0.01 0.02 0.06 0.21 0.24 0.28 0.08 0.09 0.01
##
## $mids
## [1] 154.5 155.5 156.5 157.5 158.5 159.5 160.5 161.5 162.5
##
## $xname
## [1] "pop2[[x]]$m"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

Question 2

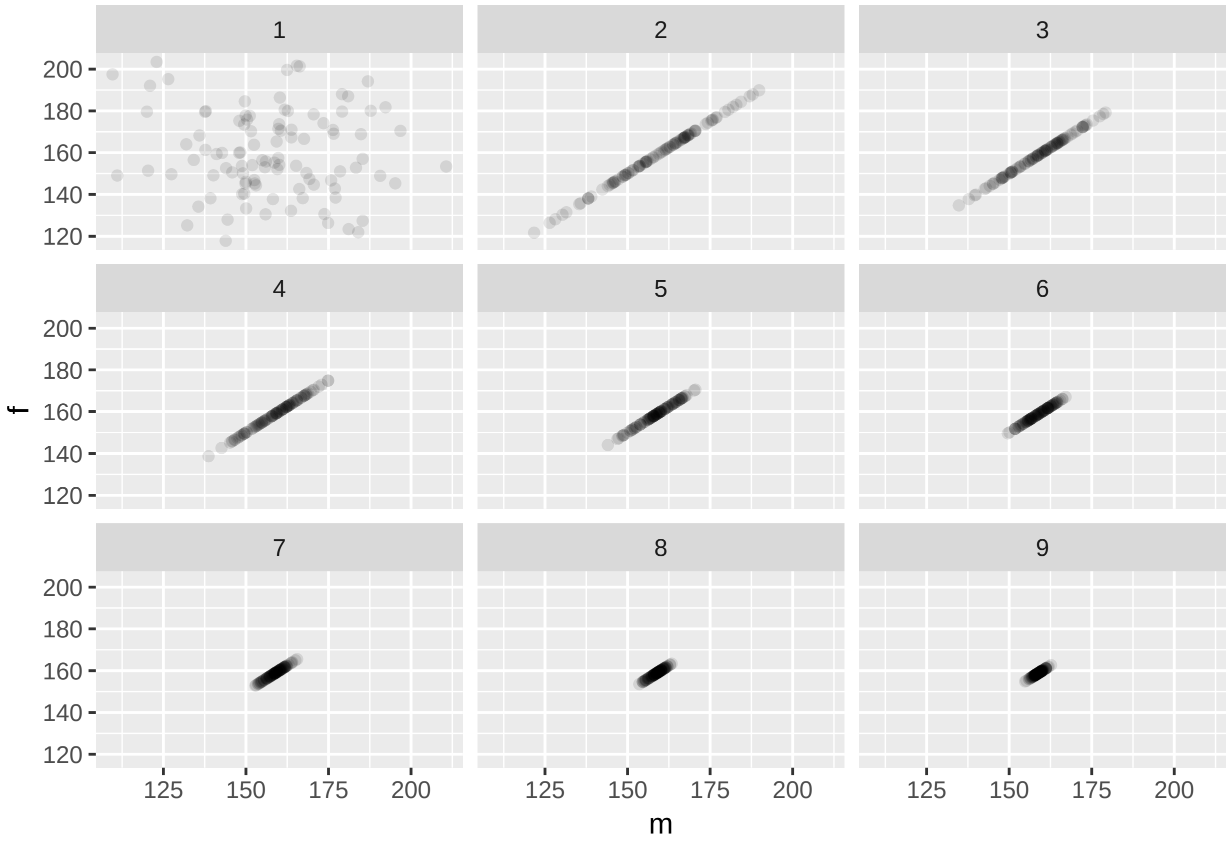
10 points

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

```
h <- do.call(rbind, pop2)
h[,"generation"] <- rep(1:9,each = 100)

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
p <- ggplot(data=h) + geom_point(mapping = aes(x=m, y=f), alpha = 1/10)
p + facet_wrap(~ generation)
```



Question 3

10 points

You calculated the power of a study design in question #2 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. (4 points)

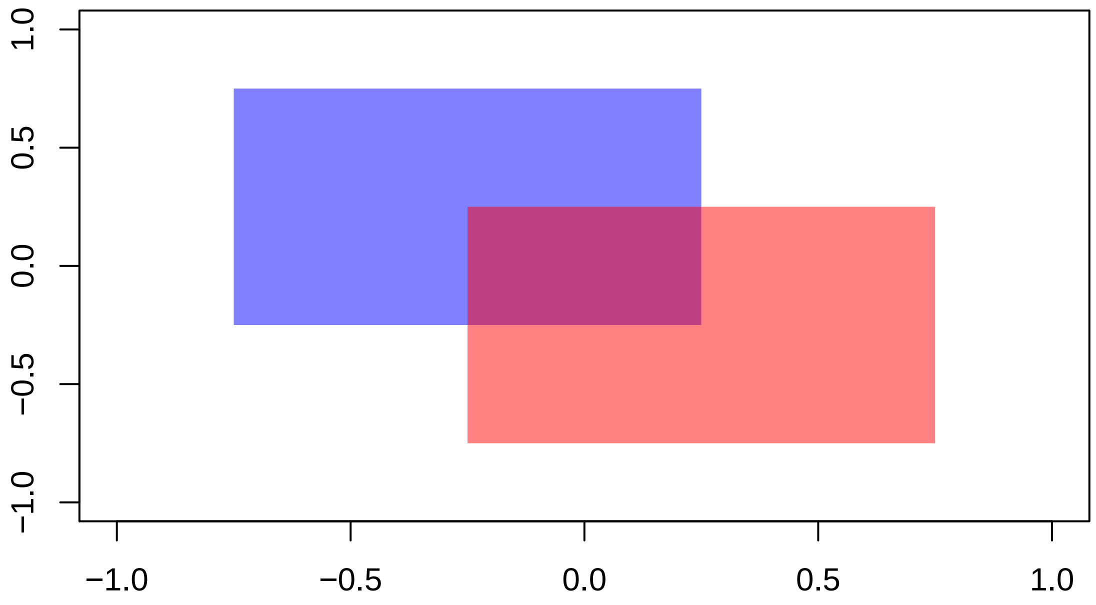
```r
set.seed(20)
# function for bootstrap percentile interval for the mean
boot.mean  <-  function(x,B = 1000) {
  n = length(x)
  #B = 1000
  boot.samples = matrix( sample(x,size=n*B,replace=TRUE), B, n)
  boot.statistics = apply(boot.samples,1,mean)
  btmean <- mean(boot.statistics)
  CI <- quantile(boot.statistics, c(0.025,0.975))
  return( data.frame(mean = btmean, CI.lower=CI[1], CI.upper=CI[2]) )
}

treMean <- list()
crlMean <- list()

for (j in 1:10) {
  n <- 250*j                       # sample size
  treat <- numeric(n)
  outcome <- numeric(n)
    for (i in seq(n)) {
      treat[i] <- sample(c(0,1),1)   # assign treatment to each patient
      outcome[i] <- rnorm(1,mean = 60,sd = 20)  # assign treatment to each patient
      if (treat[i] == 1) outcome[i] = outcome[i] + 5
    }
  tre <- (outcome[which(treat==1)])
  crl <- (outcome[which(treat==0)])

  treMean[[j]] <- boot.mean(tre,1000)
  crlMean[[j]] <- boot.mean(crl,1000)
}
# add indicator variable and sample size to each group
tre.data <- do.call(rbind, treMean)
tre.data[,'Tre'] <- 1
tre.data[,"samplesize"] <- seq(250, 2500, by = 250)

crl.data <- do.call(rbind, crlMean)
crl.data[,'Tre'] <- 0
crl.data[,"samplesize"] <- seq(250, 2500, by = 250)

#combine together
result <- rbind(tre.data,crl.data)
```

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)

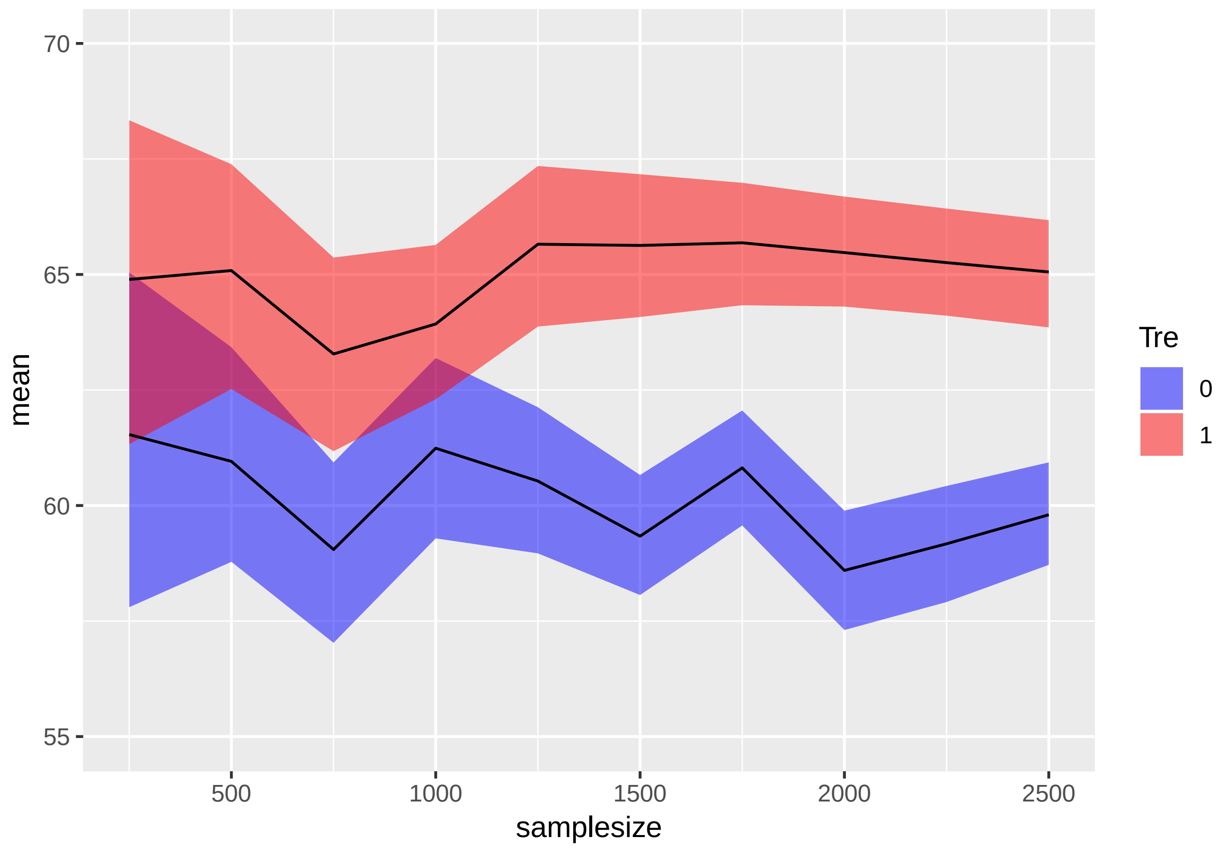
You may use base graphics or ggplot2. It should look similar to this (in base).

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

```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
result$Tre <- factor(result$Tre)
#result$Tre

h <- ggplot(data=result, mapping = aes(x=samplesize, y=mean,group = Tre))
h +
  geom_ribbon(aes(ymin = CI.lower, ymax = CI.upper, fill = Tre)) + ylim(55,70) +
  scale_fill_manual(values=c(makeTransparent('blue',alpha=0.5),makeTransparent('red',alpha=0.5)))+
  geom_line() + theme()
```



70
65
60
55
mean
500
1000
1500
2000
2500
samplesize
Tre
0
1