Bios 6301: Assignment 9

Yeji Ko

Due Tuesday, 30 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:

```
set.seed(111)
pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
head(pop)
```

```
##          m        f
## 1 164.7044 171.9924
## 2 153.3853 136.7934
## 3 153.7675 168.7819
## 4 113.9531 164.0971
## 5 156.5825 146.0164
## 6 162.8056 141.4675
```

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
}
```

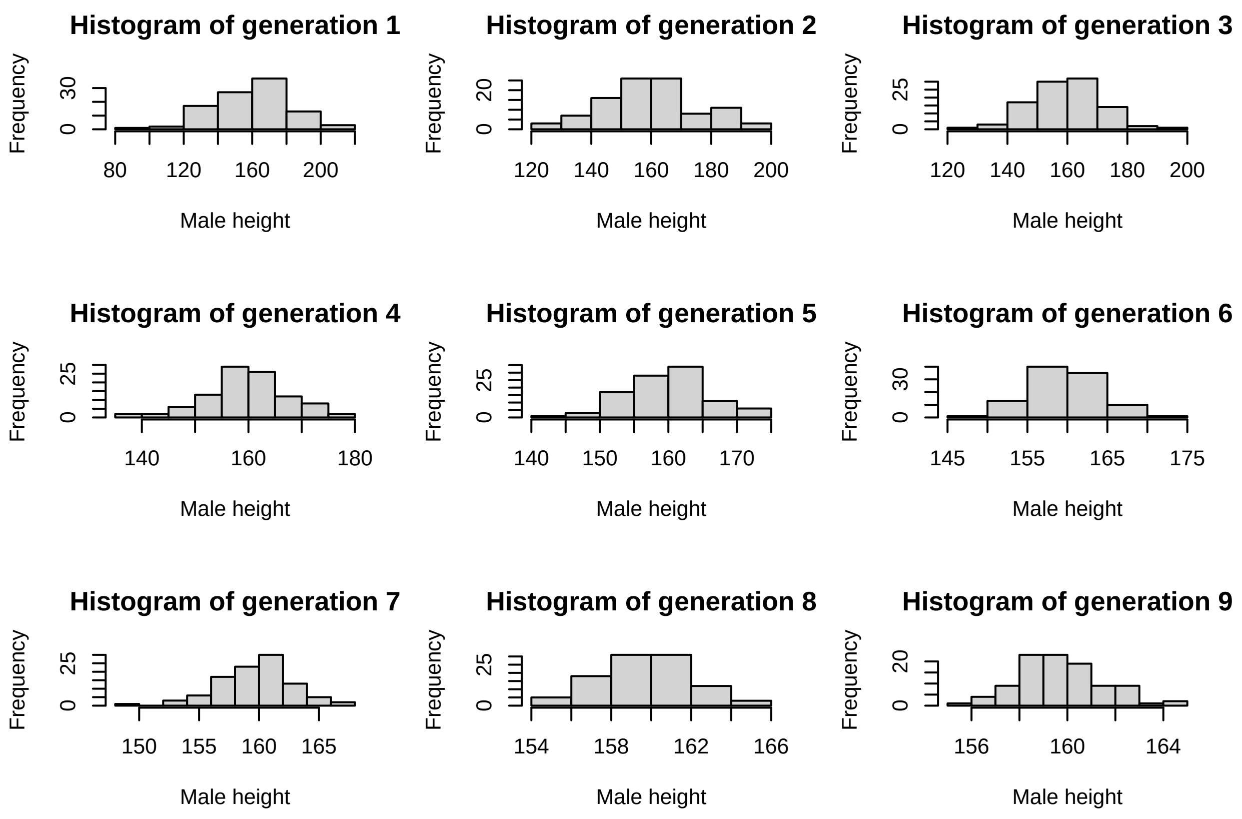
Use the function `next_gen` to generate nine generations (you already have the first), then use the function `hist` to plot the distribution of male heights in each generation (this will require multiple calls to `hist`). The phenomenon you see is called regression to the mean. Provide (at least) minimal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

```r
gen1 <- pop
gen2 <- next_gen(gen1)
gen3 <- next_gen(gen2)
gen4 <- next_gen(gen3)
gen5 <- next_gen(gen4)
gen6 <- next_gen(gen5)
gen7 <- next_gen(gen6)
gen8 <- next_gen(gen7)
gen9 <- next_gen(gen8)

gen1$gen <- '1'
gen2$gen <- '2'
gen3$gen <- '3'
gen4$gen <- '4'
gen5$gen <- '5'
gen6$gen <- '6'
gen7$gen <- '7'
gen8$gen <- '8'
gen9$gen <- '9'

ninegen <- rbind(gen1, gen2, gen3, gen4, gen5, gen6, gen7, gen8, gen9)

par(mfrow = c(3,3))
for(i in 1:9){
  hist(ninegen[ninegen$gen == paste(i),"m"], main = paste("Histogram of generation", i),
       xlab = "Male height")
}
```



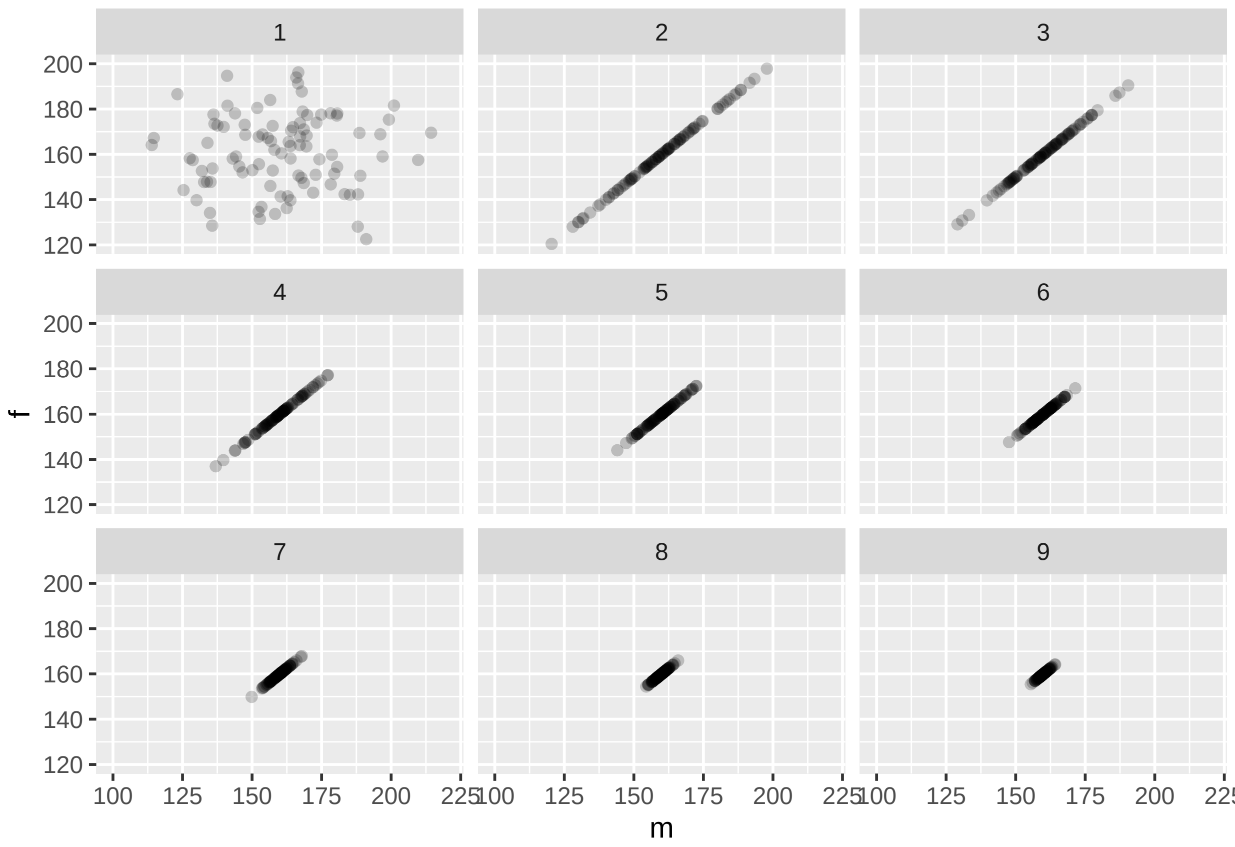
Question 2

10 points

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

```r
library(ggplot2)
ggplot(data=ninegen) + geom_point(mapping = aes(x=m, y=f), alpha = 0.2) +
  facet_wrap(~ gen) + xlim(c(100,220)) + ylim(c(120,200))
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



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)

```
# sample size = 250
set.seed(1234)
means1 <- c()
means0 <- c()
for(i in 1:1000) {
  treatment <- rbinom(250, 1, 0.5)
  outcome <- rnorm(250, 60, 20)
  x <- data.frame(treatment, outcome)
  x$outcome <- ifelse(x$treatment == 1, x$outcome + 5, x$outcome) # add 5 if treatment == 1
  means1 <- c(means1, mean(x[x$treatment ==1,'outcome']))
  means0 <- c(means0, mean(x[x$treatment ==0,'outcome']))
}

quantile(means1, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 61.21415 68.41831
```

```r
quantile(means0, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 56.68383 63.45234
```

```r
set.seed(1234)

# increasing sample size from 250 to 2500
boot.intervals <- data.frame(matrix(NA, 10,4))
colnames(boot.intervals) <- c('LB1', 'UB1', 'LB0','UB0')

means1 <- data.frame(matrix(NA, 10, 1000))
means0 <- data.frame(matrix(NA, 10, 1000))

for(i in 1:10){
  for(j in 1:1000) {
  treatment <- rbinom(250 * i, 1, 0.5)
  outcome <- rnorm(250 * i, 60, 20)
  x <- data.frame(treatment, outcome)
  x$outcome <- ifelse(x$treatment == 1, x$outcome + 5, x$outcome) # add 5 if treatment == 1
  means1[i,j] <- mean(x[x$treatment == 1,'outcome'])
  means0[i,j] <- mean(x[x$treatment == 0,'outcome'])
  }
  boot.intervals[i,1:2] <- quantile(means1[i,], c(0.025, 0.975))
  boot.intervals[i,3:4] <- quantile(means0[i,], c(0.025, 0.975))
}

boot.intervals
```

```
##         LB1      UB1      LB0      UB0
## 1  61.21415 68.41831 56.68383 63.45234
## 2  62.45556 67.68227 57.34872 62.47272
## 3  62.94190 66.80246 58.00507 61.94672
## 4  63.32775 66.72550 58.25907 61.79993
## 5  63.46290 66.52222 58.41390 61.52776
## 6  63.48907 66.46349 58.56842 61.42889
## 7  63.68488 66.31480 58.63777 61.35387
## 8  63.76870 66.23262 58.71989 61.25405
## 9  63.77528 66.16748 58.84702 61.24031
## 10 63.92204 66.02978 58.82478 60.98383
```

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)
  }
```

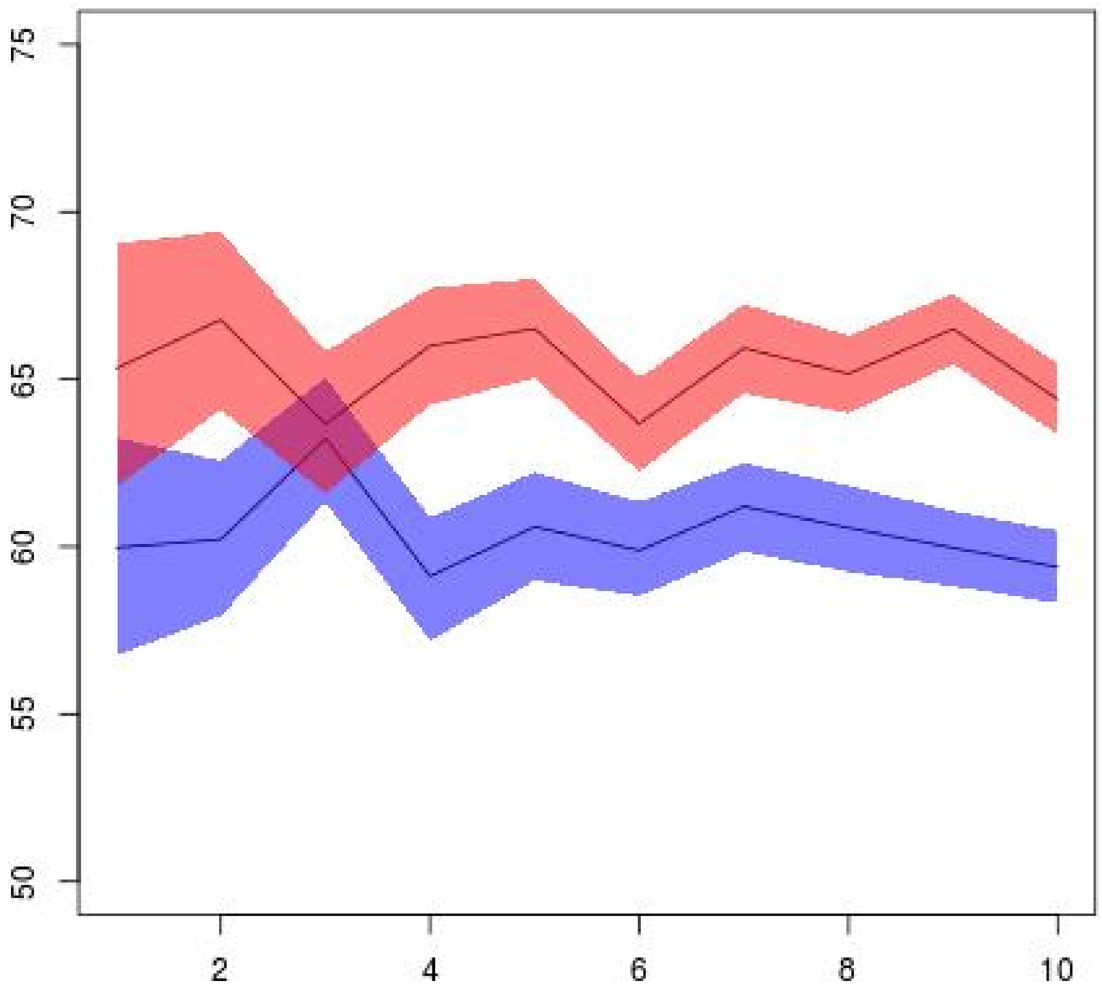
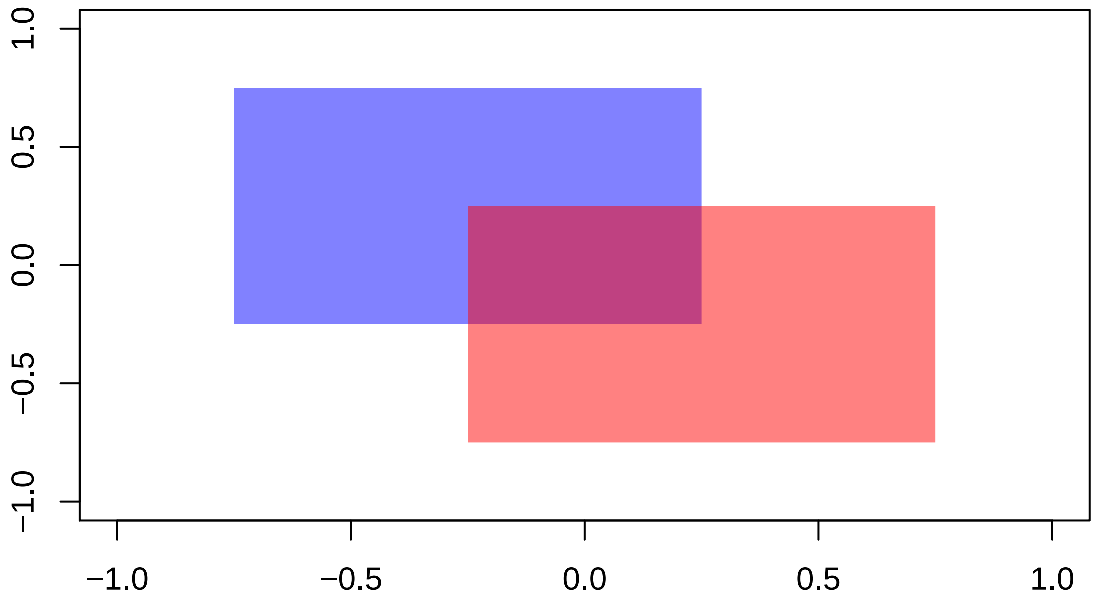


Figure 1: bp interval plot

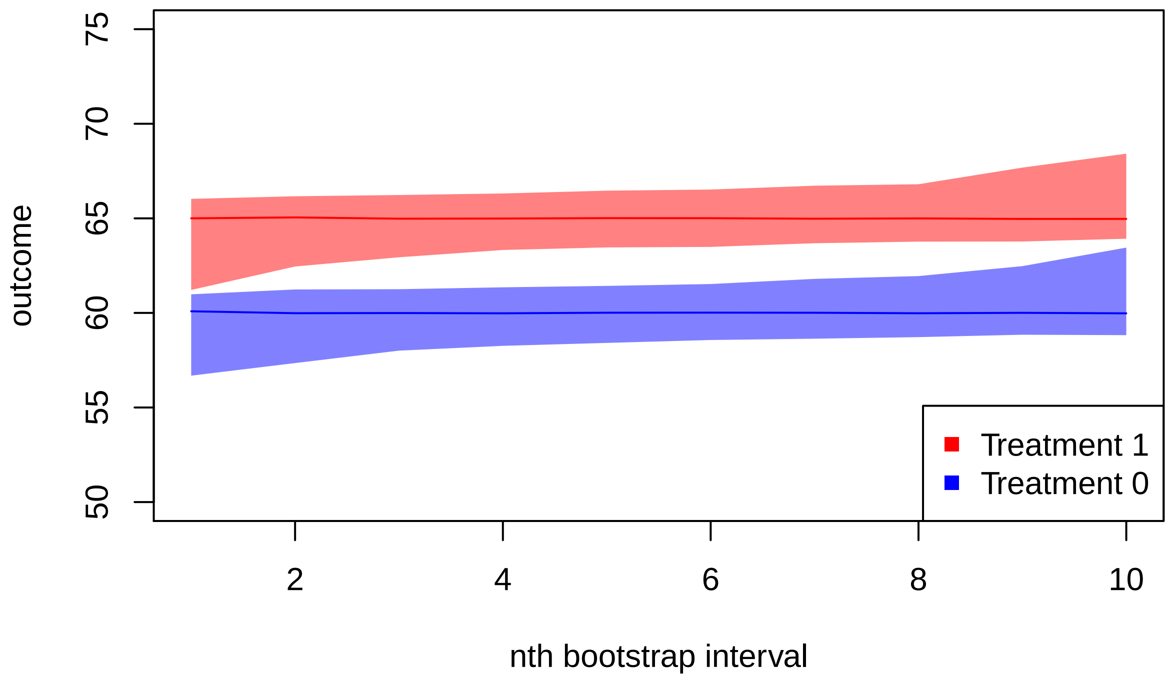
```
  newColor = apply(newColor, 2, .makeTransparent, alpha=alpha)
  return(newColor)
}

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))
```



```
plot(NULL,
  xlim=c(1,10),
  ylim=c(50,75),
  xlab="nth bootstrap interval",
  ylab="outcome"
)
lines(x=seq(1:10), y = rowMeans(means1), col = "red")
polygon(x=c(seq(1,10), seq(10,1)), y = c(boot.intervals$LB1, boot.intervals$UB1),
        border=NA, col=makeTransparent('red',alpha=0.5))
lines(x=seq(1:10), y = rowMeans(means0), col = "blue")
polygon(x=c(seq(1,10), seq(10,1)), y = c(boot.intervals$LB0, boot.intervals$UB0),
        border=NA, col=makeTransparent('blue',alpha=0.5))
lab <- c('Treatment 1', 'Treatment 0')
legend("bottomright", lab, pch=c(15,15), col=c('red','blue'))
```



outcome
75
70
65
60
55
50
2
4
6
8
10
nth bootstrap interval
Treatment 1
Treatment 0