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
}
```

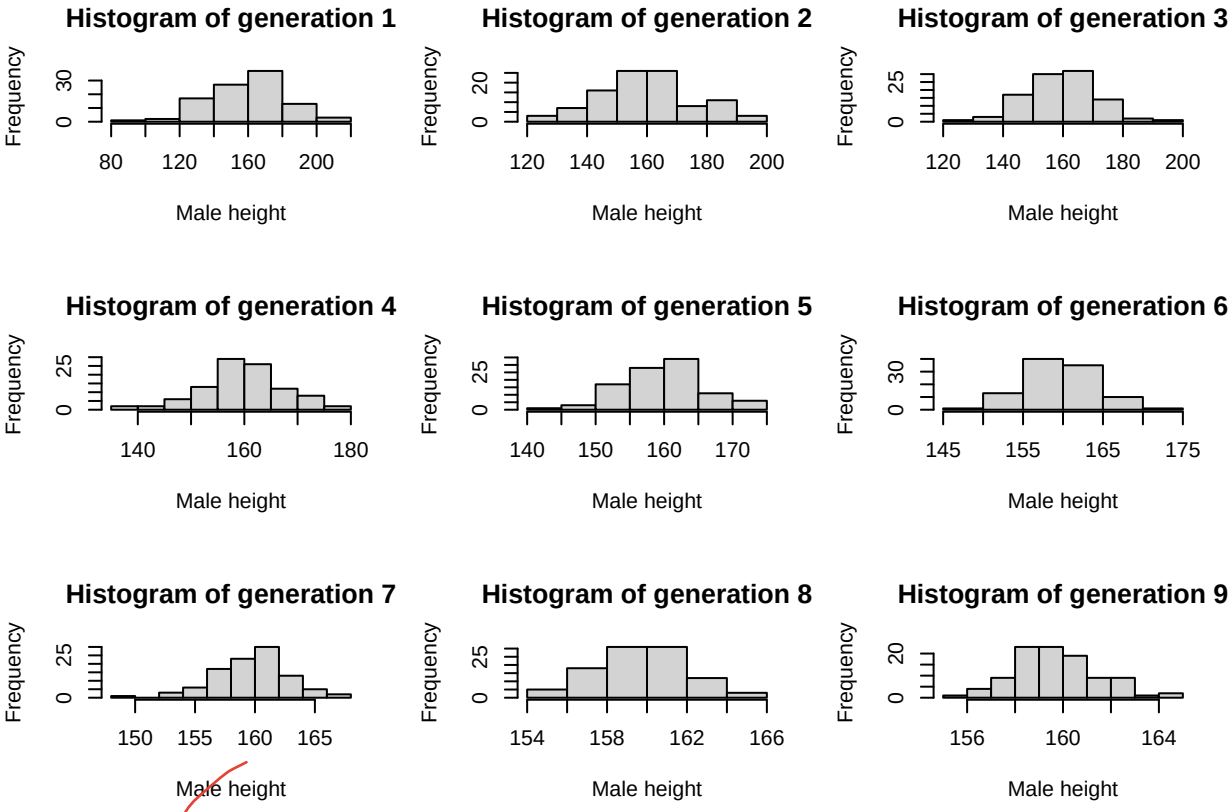
Use the function `next_gen` to generate nine generations (you already have the first), then use the function `hist` to plot the distribution of male heights in each generation (this will require multiple calls to `hist`). The phenomenon you see is called regression to the mean. Provide (at least) minimal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

```
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")
}
```



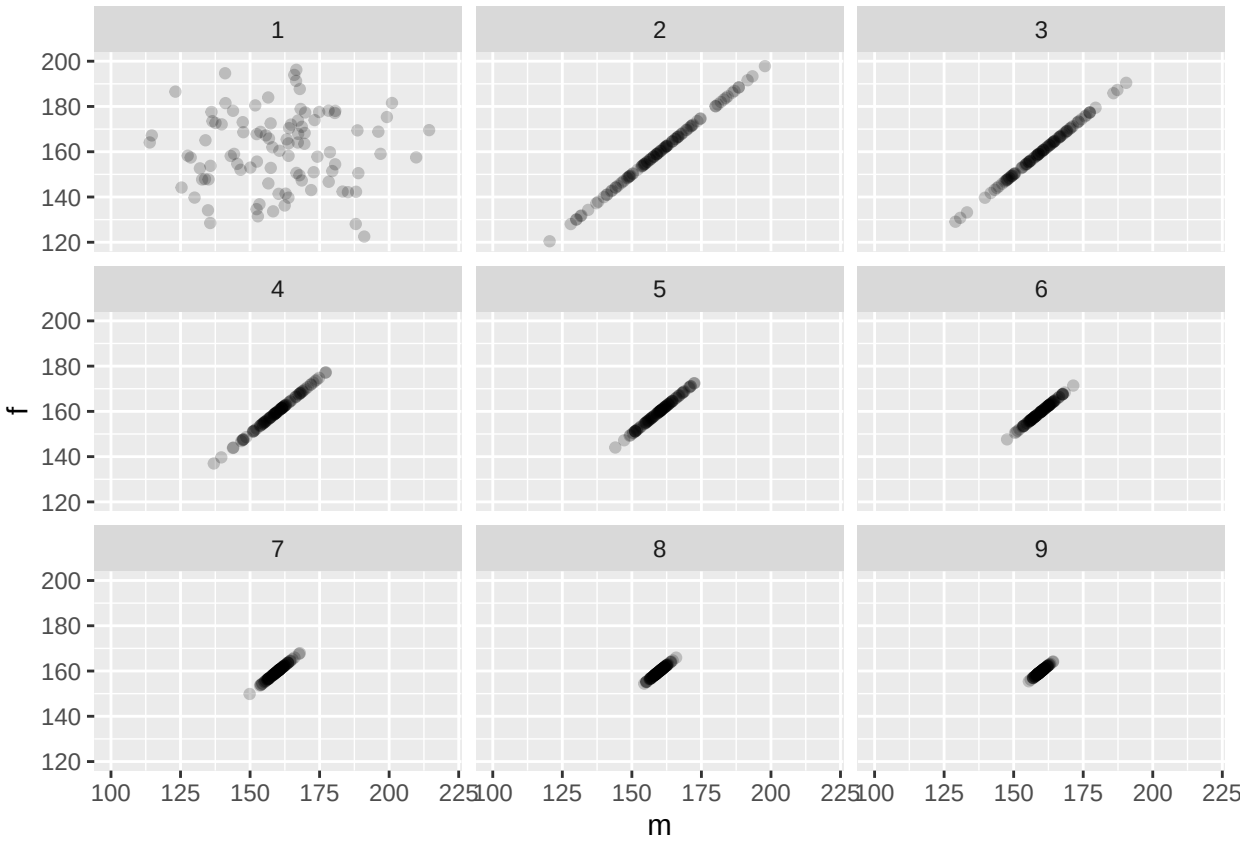
Question 2

10 points

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

```
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)

ss <- 250
trt <- rbinom(ss,1,0.5)
dat <- rnorm(ss, 60, 20) + trt *5
bs.res <- data.frame(matrix(NA, 1000, 2))
names(bs.res) <- c("treatment0", "treatment1")
for(i in seq(1000)){
  bs.rows <- sample(ss,ss, replace = TRUE)
  bs.res[i,] <- tapply(dat[bs.rows], trt[bs.rows], mean)
}

quantile(bs.res[[1]], c(0.025, 0.975)) # treatment 0
```

```
##     2.5%    97.5%
```

```
## 55.89838 62.96100
```

```
quantile(bs.res[[2]], c(0.025, 0.975)) # treatment 1
```

```
##     2.5%    97.5%
## 65.20988 72.36177
```

```
# increasing sample size from 250 to 2500
set.seed(36)
boot.intervals <- data.frame(matrix(NA, 10,5))
colnames(boot.intervals) <- c('LB1', 'UB1', 'LB0', 'UB0', 'size')
means0 <- c()
means1 <- c()
for(i in 1:10){
ss <- 250 * i
trt <- rbinom(ss,1,0.5)
dat <- rnorm(ss, 60, 20) + trt *5
bs.res <- data.frame(matrix(NA, 1000, 2))
names(bs.res) <- c("treatment0", "treatment1")
for(j in seq(1000)){
  bs.rows <- sample(ss,ss, replace = TRUE)
  bs.res[j,] <- tapply(dat[bs.rows], trt[bs.rows], mean)
}
means0 <- c(means0, mean(bs.res[[1]]))
means1 <- c(means1, mean(bs.res[[2]]))
boot.intervals[i,5] <- ss
boot.intervals[i,3:4] <- quantile(bs.res[[1]], c(0.025, 0.975)) # treatment 0
boot.intervals[i,1:2] <- quantile(bs.res[[2]], c(0.025, 0.975)) # treatment 1
}
boot.intervals
```

```
##         LB1      UB1      LB0      UB0 size
## 1  61.93121 69.23575 57.70934 64.53987  250
## 2  62.99479 67.84976 57.69321 62.83012  500
## 3  61.29202 65.37930 58.92458 63.16363  750
## 4  63.71710 67.13267 57.38326 60.90959 1000
## 5  63.48949 66.56869 59.04030 62.15686 1250
## 6  62.96047 65.79652 59.23418 61.99664 1500
## 7  62.99985 65.69651 58.11656 60.73649 1750
## 8  63.76893 66.22844 58.53836 61.14919 2000
## 9  64.28518 66.61612 59.85691 62.19627 2250
## 10 63.80390 66.27012 59.11748 61.39465 2500
```

```
means0
```

```
##  [1] 61.10306 60.26325 61.09595 59.12136 60.62638 60.59586 59.41967 59.81363
##  [9] 61.03114 60.24391
```

```
means1
```

```
##  [1] 65.69455 65.32952 63.31667 65.39258 65.04961 64.40310 64.32336 64.98808
##  [9] 65.42840 65.05436
```

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.

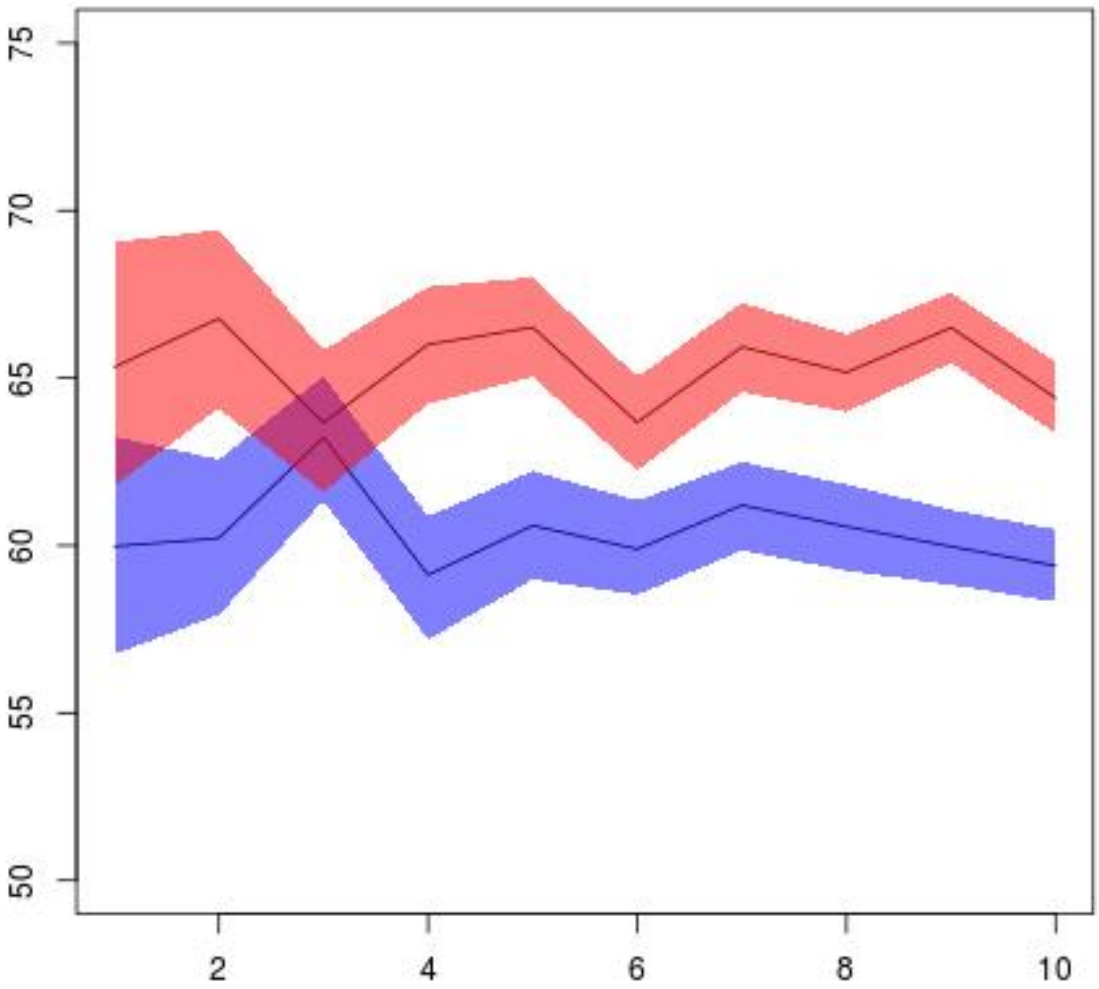
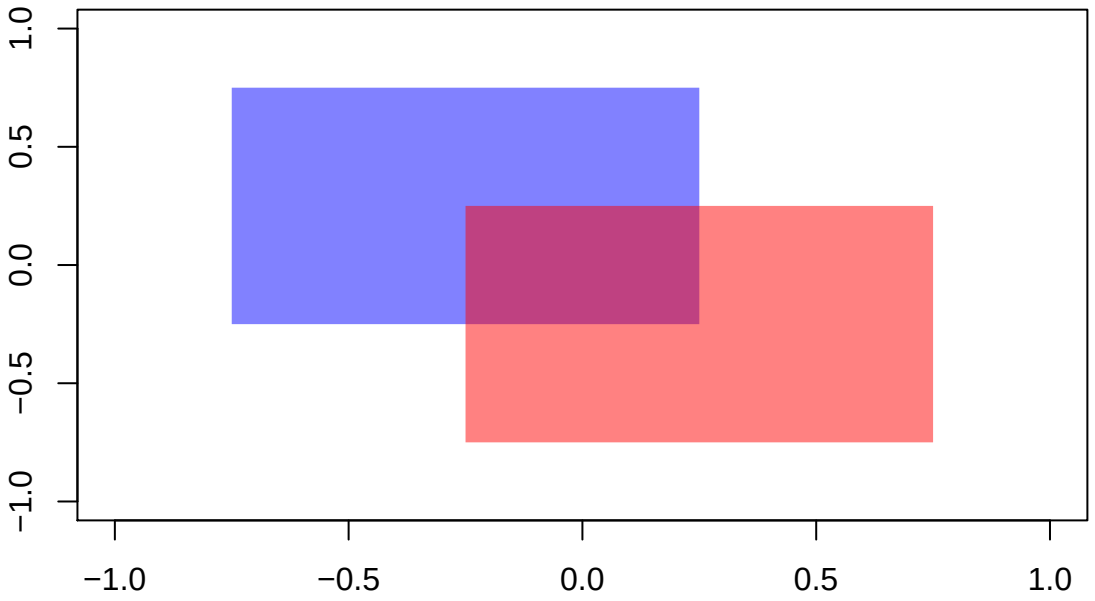


Figure 1: bp interval plot

```
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))
```




```
plot(NULL,
  xlim=c(1,10),
  ylim=c(50,75),
  xlab="nth bootstrap interval",
  ylab="outcome"
)
lines(x=seq(1:10), y = means1, col = "red")
polygon(x=c(seq(1,10), seq(10,1)), y = c(boot.intervals$LB1, rev(boot.intervals$UB1)),
        border=NA, col=makeTransparent('red',alpha=0.5))
lines(x=seq(1:10), y = means0, col = "blue")
polygon(x=c(seq(1,10), seq(10,1)), y = c(boot.intervals$LB0, rev(boot.intervals$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'))
```