40

Bios 6301: Assignment 9

Lan Shi

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:

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

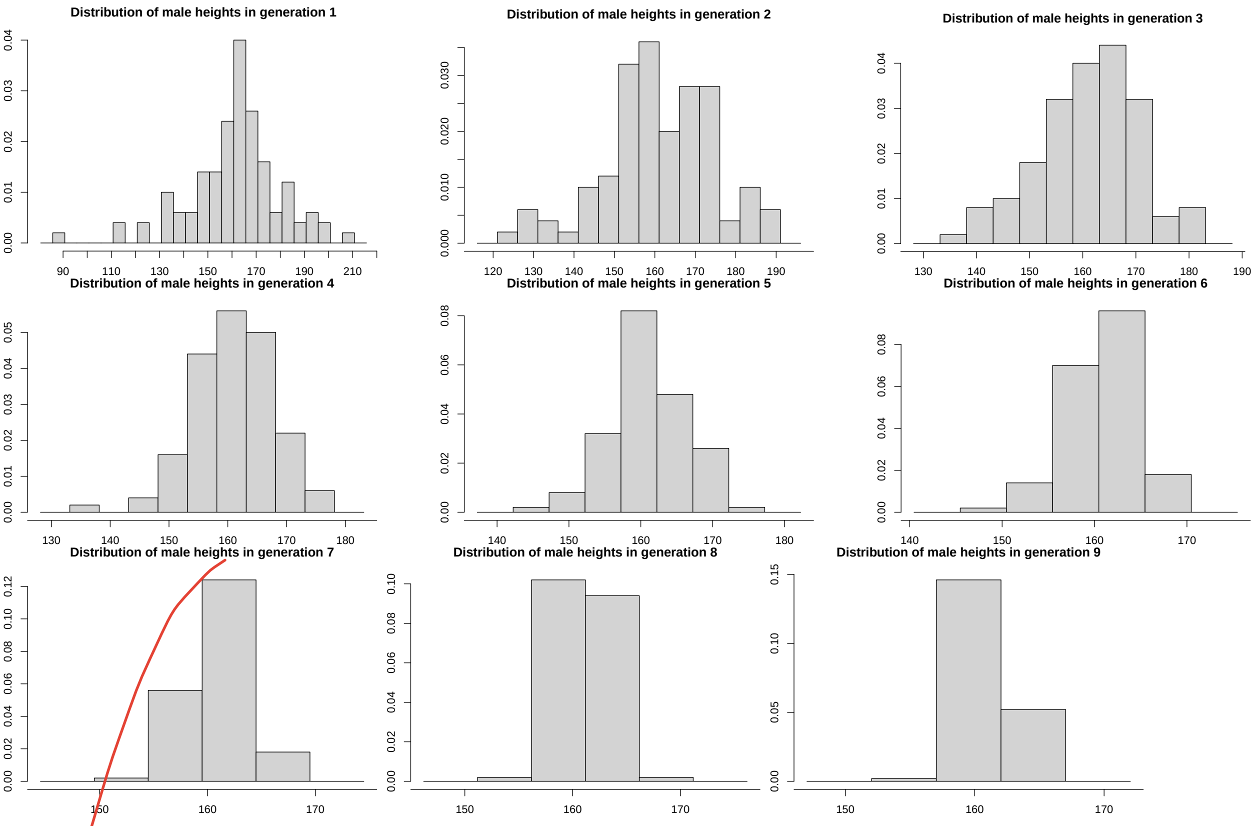
The following function takes the data frame `pop` and randomly permutes the ordering of the men. Men and women are then paired according to rows, and heights for the next generation are calculated by taking the mean of each row. The function returns a data frame with the same structure, giving the heights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

```r
next_gen <- function(pop) {
    pop$m <- sample(pop$m) # reorder men
    pop$m <- rowMeans(pop) # get mean of male and female of 1st gen.
    pop$f <- pop$m # next gen female is the same as next gen male
    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
pop$gen = 1
for (i in 2:9){
  parent = pop[pop$gen==(i-1),c('m','f')]
  new_gen = next_gen(parent)
  new_gen$gen = i
  pop = rbind(pop,new_gen)
}

# plot
par(mar = c(3, 3, 3, 3))
for (j in 1:9){
  each_gen = pop[pop$gen==j,]
  hist(each_gen$m, xlab='Male Height', freq=F,
       main=paste("Distribution of male heights in generation",j),
       breaks=seq(min(each_gen$m)-10,max(each_gen$m)+10,5), xaxt='n')
  axis(side=1, at=seq(90,220,10), labels=seq(90,220,10))
}
```



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(pop) +
  geom_point(aes(x=m, y=f),alpha = 1/4,shape=16) +
  scale_x_continuous(limits=c(100,220),
                     breaks=seq(100, 220, 20),
                     labels = seq(100, 220, 20)) +
  ylim(c(120,200)) +
  theme(axis.title=element_text(size=14)) +
  facet_wrap(~ gen,nrow=3)
```

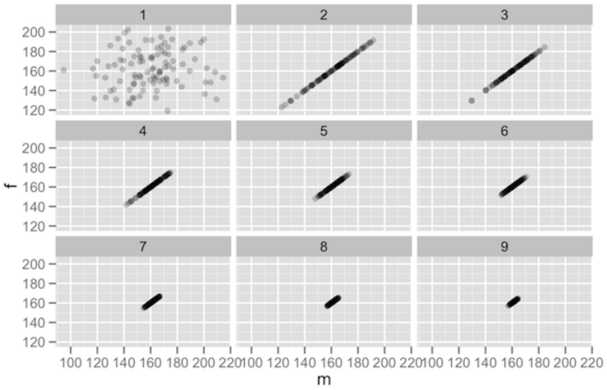
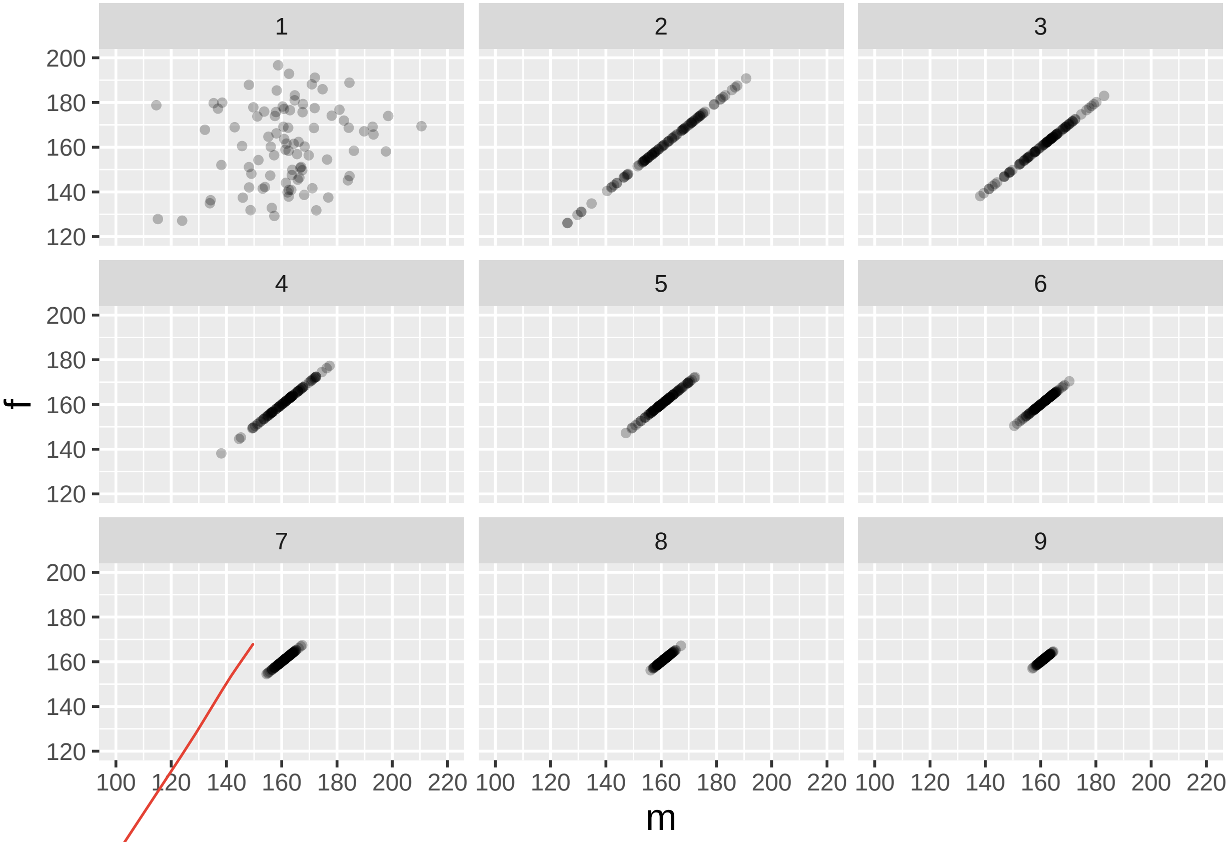


Figure 1: generations plot

```
## Warning: Removed 10 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)

```
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(tidyr)
set.seed(123)
ns = seq(250,2500,250)
num = length(ns)
CI0 = matrix(rep(0,num*4),ncol=4)
CI1 = matrix(rep(1,num*4),ncol=4)
colnames(CI0) = c('lb','ub','mean','treat')
colnames(CI1) = c('lb','ub','mean','treat')

for (i in 1:num){
  n = ns[i]
  treat_grp = rbinom(n, 1, 0.5)
  outcome = rnorm(n, mean=60, sd=20)
  outcome[treat_grp==1] = outcome[treat_grp==1] + 5
  dt0 = data.frame(treat_grp,outcome)
  x = replicate(1e3, {
    # bootstrap
    dt = as_tibble(dt0[sample(n,n,replace = T),])
    #mean
    mean_all = dt %>% group_by(treat_grp) %>%
      summarise_at(vars(outcome),mean) %>% pull})
  mean0 = x[1,]  # mean of control group
  mean1 = x[2,]  # mean of treatment group
  CI0[i,1:3] = c(quantile(mean0,c(.025,.975)),mean(mean0))
  CI1[i,1:3] = c(quantile(mean1,c(.025,.975)),mean(mean1))
}
CI0
```

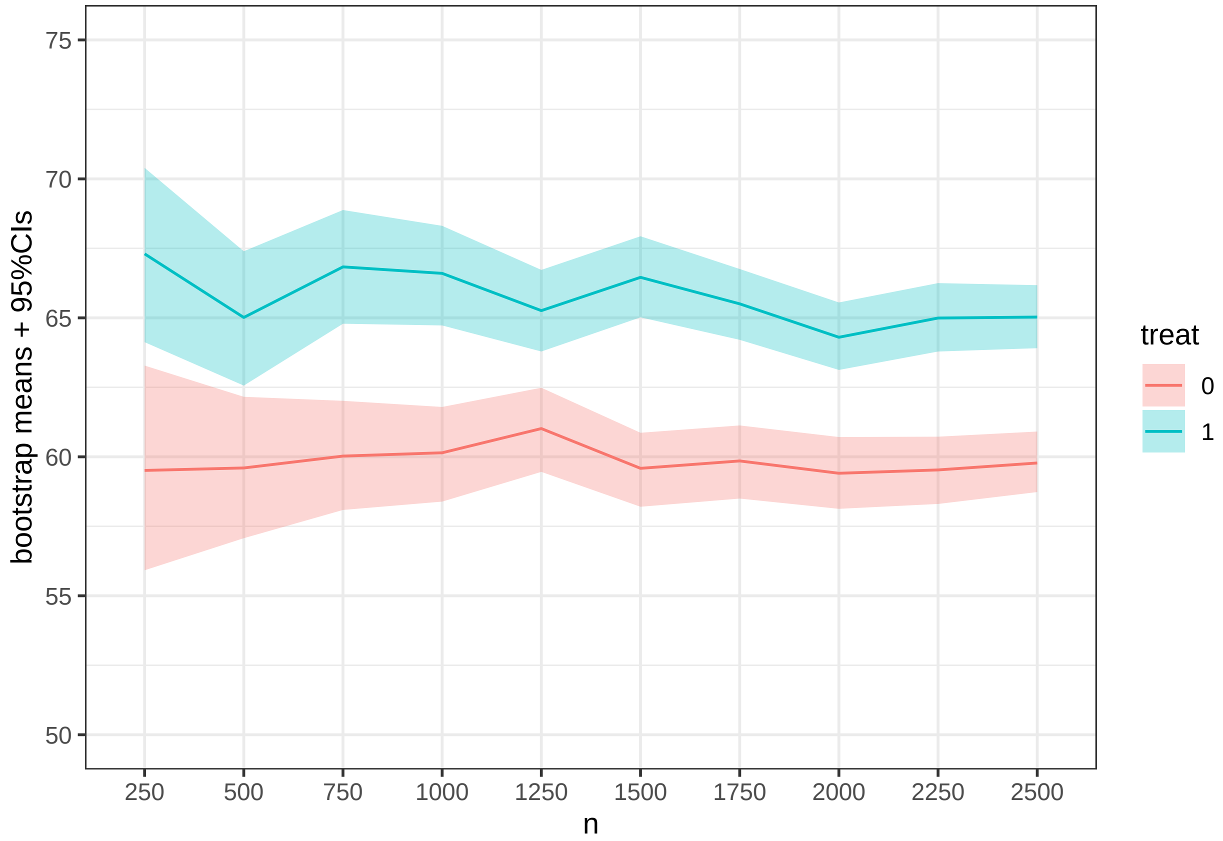
```
##             lb       ub     mean treat
##  [1,] 55.91750 63.28304 59.50949     0
##  [2,] 57.07174 62.15940 59.59880     0
##  [3,] 58.08739 62.01461 60.02586     0
##  [4,] 58.38772 61.79463 60.14449     0
##  [5,] 59.45831 62.48583 61.01573     0
##  [6,] 58.20300 60.86584 59.58514     0
##  [7,] 58.49761 61.12971 59.85104     0
##  [8,] 58.12759 60.71228 59.40797     0
##  [9,] 58.30381 60.72481 59.52818     0
## [10,] 58.73188 60.90966 59.77703     0
```

```
CI1
```

```
##              lb       ub     mean treat
##  [1,] 64.12448 70.39759 67.30290     1
##  [2,] 62.56188 67.39596 65.01327     1
##  [3,] 64.78761 68.87986 66.83372     1
##  [4,] 64.72681 68.31105 66.60038     1
##  [5,] 63.78877 66.72483 65.25995     1
##  [6,] 65.01845 67.93617 66.45780     1
##  [7,] 64.20826 66.75761 65.50336     1
##  [8,] 63.12464 65.55366 64.30165     1
##  [9,] 63.78783 66.25004 64.99365     1
## [10,] 63.90776 66.17580 65.02784     1
```

```
CIs = data.frame(idx=rep(1:10,2),rbind(CI0,CI1))
CIs$treat = as.factor(CIs$treat)

ggplot(data=CIs,aes(y=mean,x=factor(idx),group = treat,
                    color=treat,fill=treat)) +
  geom_line() + xlab("n") + ylab("bootstrap means + 95%CIs") +
  scale_x_discrete(breaks=1:10,labels=seq(250,2500,250)) +
  scale_y_continuous(limits = c(50,75)) +
  geom_ribbon(aes(ymin=lb,ymax=ub),alpha=0.3, linetype=0)+
  theme_bw()
```



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).

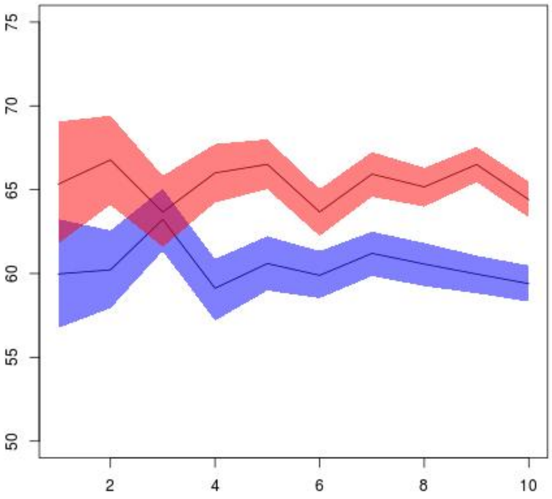


Figure 2: bp interval plot

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)
}

CI0 = as.data.frame(CI0)
CI1 = as.data.frame(CI1)
par(new=FALSE)
plot(x=1:10,CI0$mean, type="l",
  xlim=c(0,10),
  ylim=c(50,75),
  xlab="",ylab="")
lines(x=1:10,CI1$mean)
polygon(x=c(seq(1, 10), seq(10, 1)),
        y=c(CI0$lb, CI0$ub), border=NA,
        col=makeTransparent('blue',alpha=0.5))
polygon(x=c(seq(1, 10), seq(10, 1)),
        y=c(CI1$lb, CI1$ub), border=NA,
        col=makeTransparent('red',alpha=0.5))
```