Homework9

Ashley Spirrison

Due Tuesday, 28 November, 1:00 PM 33/40

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

40 points total.

Question 1

15 points

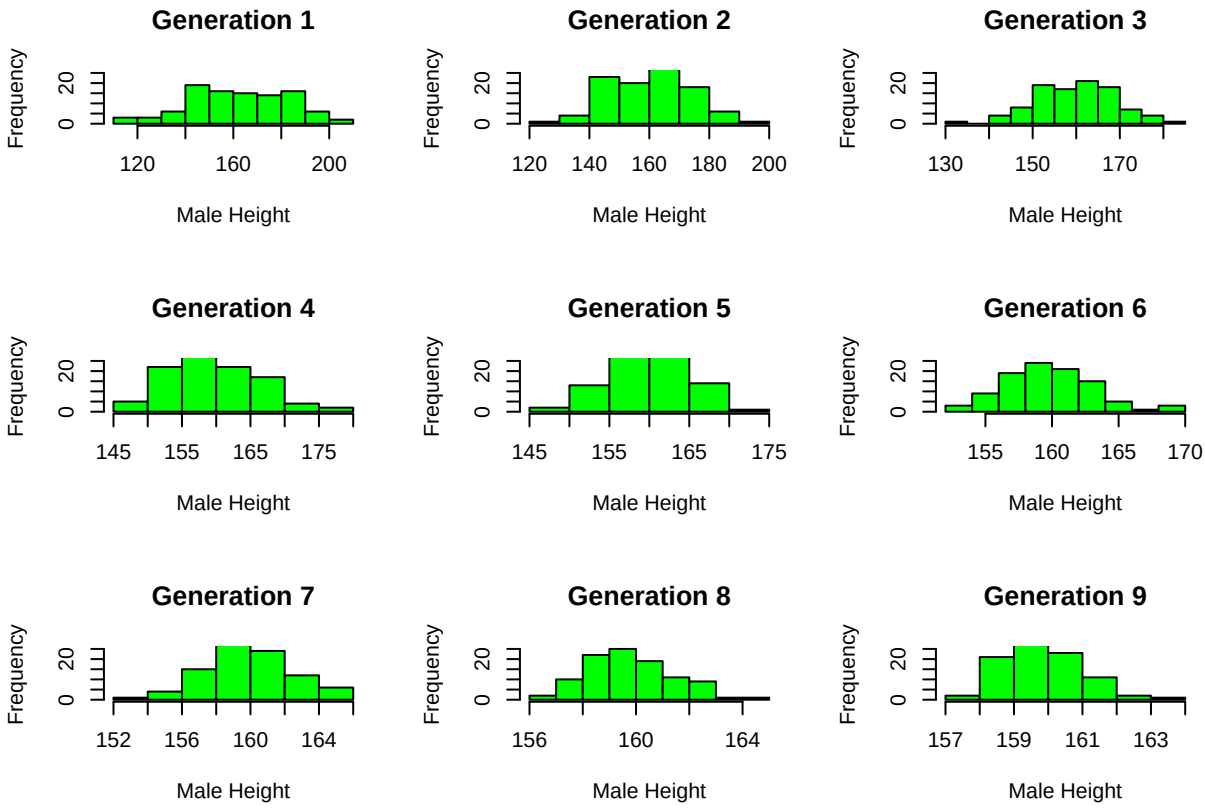
Representing the heights of the current generation as a data frame with two variables, m and f, for the two sexes. We can use `rnorm` to randomly generate the population at generation 1:

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

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
}
```

```
# Plotting the distribution of male heights for each generation.
par(mfrow=c(3, 3))

for (i in 1:9) {
    hist(pop$m, main = paste("Generation", i), xlab = "Male Height", col = "green", ylim = c(0, 25))
    pop <- next_gen(pop)
}
```



Question 2

10 points

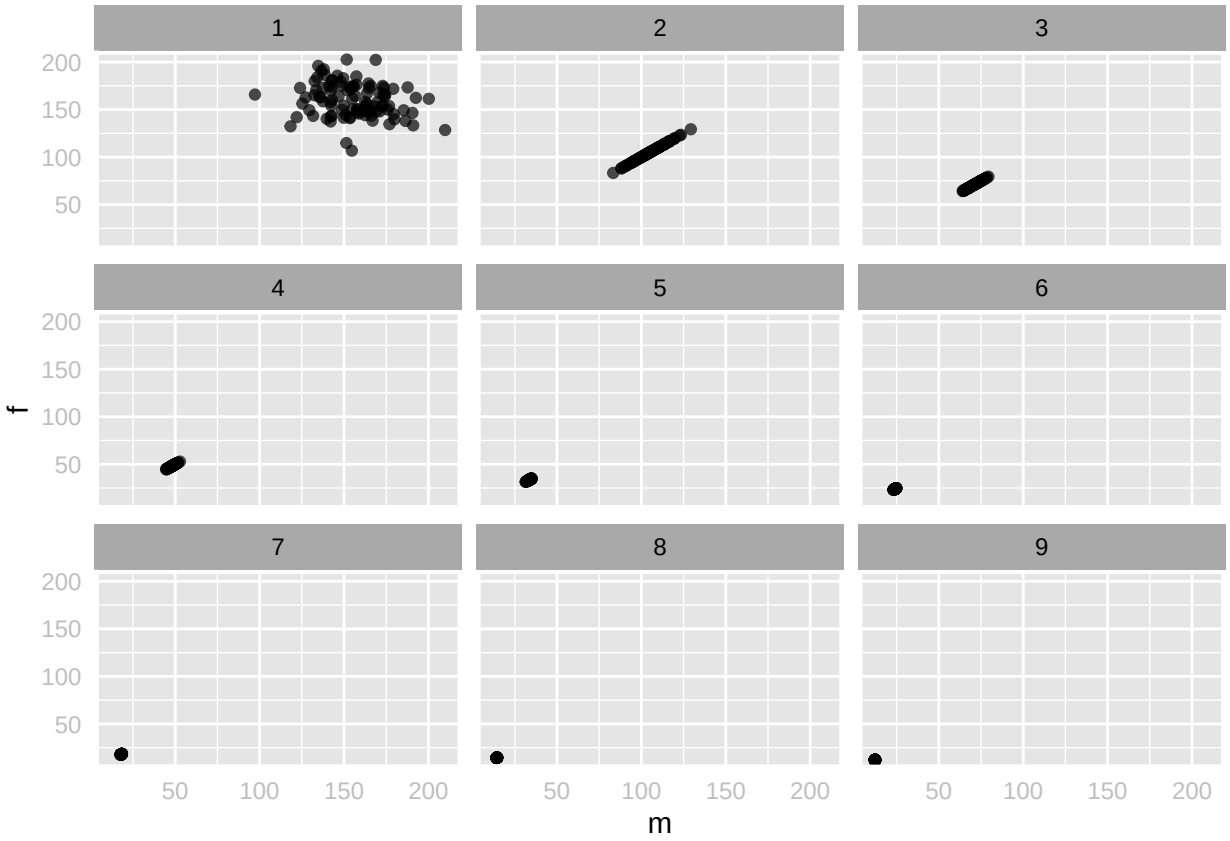
```
#Loading ggplot2
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
#Creating function to generate the next generation.
next_gen <- function(pop) {
  pop$m <- sample(pop$m)
  pop$m <- rowMeans(pop)
  pop$f <- pop$m
  pop
}

#Generating data for nine generations.
populations <- list()
populations[[1]] <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20), generation = 1)
for (i in 2:9) {
  populations[[i]] <- next_gen(populations[[i - 1]])
  populations[[i]]$generation <- i
}
```

```r
#Combining populations into a single data frame.
combined_data <- do.call(rbind, populations)

#Plotting using ggplot2.
ggplot(combined_data, aes(x = m, y = f)) +
  geom_point(alpha = 0.7, na.rm = TRUE) +
  labs(x = "m", y = "f") +
  facet_wrap(~generation, ncol = 3) +
  theme_minimal() +
  theme(
    panel.background = element_rect(fill = "grey90", color = "white", size = 1.5),
    plot.background = element_rect(color = "white", size = 1.5),
    axis.line = element_line(color = "white", size = 1.5),
    axis.text = element_text(color = "grey"),
    axis.ticks = element_line(color = "white", size = 1.5),
    panel.grid = element_line(color = "white", size = 0.5),
    strip.background = element_rect(fill = "darkgrey", color = NA),
    strip.text = element_text(color = "black")
  )
```

```
## Warning: The ‘size‘ argument of ‘element_line()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.

## Warning: The ‘size‘ argument of ‘element_rect()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



something is off here, they should be headed towards the mean rather than zero

Question 3

15 points

```
# Loading ggplot2 and boot.
library(ggplot2)
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.3.2
```

```
# Creating function to generate data for the study
generate_data <- function(sample_size) {
  set.seed(42)
  treatment_groups <- sample(c(0, 1), size = sample_size, replace = TRUE, prob = c(0.5, 0.5))
  outcome <- rnorm(n = sample_size, mean = 60, sd = 20)
  outcome[treatment_groups == 1] <- outcome[treatment_groups == 1] + 5
  data <- data.frame(Treatment_Group = as.factor(treatment_groups), Outcome = outcome)
  return(data)
}

#Creating function to calculate the mean and percentile intervals for each group
calculate_bootstrap_intervals <- function(data, num_bootstrap_samples = 1000) {
  bootstrap_results <- boot(data, function(x, indices) tapply(x[indices, "Outcome"], x[indices, "Treatm
```

```r
#Reshaping the bootstrap results into a 2x4 data frame
  results_df <- as.data.frame(matrix(bootstrap_results$t, nrow = ncol(bootstrap_results$t), byrow = TRU
  colnames(results_df) <- c("Group_0_Mean", "Group_1_Mean", "Group_0_LB", "Group_1_LB", "Group_0_UB", "

  return(results_df)
}

#Creating function to perform bootstrap for different sample sizes.
perform_bootstrap <- function(initial_sample_size, num_intervals = 10, sample_increase = 250) {
  intervals_list <- vector("list", length = num_intervals)

  for (i in 1:num_intervals) {
    data <- generate_data(initial_sample_size + (i - 1) * sample_increase)
    intervals_list[[i]] <- calculate_bootstrap_intervals(data)
  }

  return(do.call(rbind, intervals_list))
}

#Setting parameters.
num_intervals <- 10
sample_increase <- 250
initial_sample_size <- 250

#Performing bootstrap for different sample sizes.
bootstrap_intervals <- perform_bootstrap(initial_sample_size, num_intervals, sample_increase)

#Creating a data frame for plotting.
plot_data <- data.frame(
  Sample_Size = rep(seq(initial_sample_size, initial_sample_size + (num_intervals - 1) * sample_increas
#Fixing the number of rows.
  Group = rep(c("Group 0", "Group 1"), times = num_intervals),
  Mean = c(bootstrap_intervals$Group_0_Mean, bootstrap_intervals$Group_1_Mean),
  Lower_Bound = c(bootstrap_intervals$Group_0_LB, bootstrap_intervals$Group_1_LB),
  Upper_Bound = c(bootstrap_intervals$Group_0_UB, bootstrap_intervals$Group_1_UB)
)
print(plot_data)
```

```
##    Sample_Size   Group     Mean Lower_Bound Upper_Bound
## 1          250 Group 0 58.22314    57.28229    62.12898
## 2          250 Group 1 66.14335    65.01588    64.29437
## 3          500 Group 0 59.42604    59.24636    58.04178
## 4          500 Group 1 63.69800    64.98571    63.92798
## 5          750 Group 0 58.01048    59.74731    58.69005
## 6          750 Group 1 63.68894    63.57108    63.99989
## 7         1000 Group 0 59.77444    58.94598    60.12428
## 8         1000 Group 1 64.29640    63.98590    64.32902
## 9         1250 Group 0 59.19556    58.35885    60.16518
## 10        1250 Group 1 64.77620    65.48962    65.43167
## 11        1500 Group 0 59.51963    59.69378    58.94308
## 12        1500 Group 1 64.49533    64.78088    63.62688
## 13        1750 Group 0 61.95703    59.73356    59.38194
## 14        1750 Group 1 65.77275    64.91038    64.51818
```

```
## 15        2000 Group 0 60.36317    60.81239    59.48553
## 16        2000 Group 1 64.42294    64.15383    64.78068
## 17        2250 Group 0 59.87026    60.76247    60.81067
## 18        2250 Group 1 64.56383    64.70367    64.82456
## 19        2500 Group 0 60.09352    59.78705    59.83702
## 20        2500 Group 1 64.18404    63.35483    63.68575
## 21         250 Group 0 58.82106    57.81741    59.88847
## 22         250 Group 1 64.56337    65.35685    66.23775
## 23         500 Group 0 59.70721    60.96441    59.74208
## 24         500 Group 1 66.21416    65.97849    63.53738
## 25         750 Group 0 59.68251    60.00629    58.26717
## 26         750 Group 1 65.44460    65.49058    63.14033
## 27        1000 Group 0 59.93616    59.02471    61.48413
## 28        1000 Group 1 63.29411    63.63961    65.10843
## 29        1250 Group 0 60.14159    60.47993    58.73783
## 30        1250 Group 1 65.06859    64.86818    65.97288
## 31        1500 Group 0 59.33377    60.54770    59.74943
## 32        1500 Group 1 64.44960    64.23121    65.30639
## 33        1750 Group 0 60.20636    61.54347    61.35781
## 34        1750 Group 1 65.47390    66.34740    66.94533
## 35        2000 Group 0 60.44163    60.27428    60.01258
## 36        2000 Group 1 65.61326    64.24732    65.31570
## 37        2250 Group 0 60.12693    60.81300    61.57917
## 38        2250 Group 1 63.97268    65.25300    64.78865
## 39        2500 Group 0 60.13671    60.26631    59.20211
## 40        2500 Group 1 64.46172    63.99115    63.84663
```

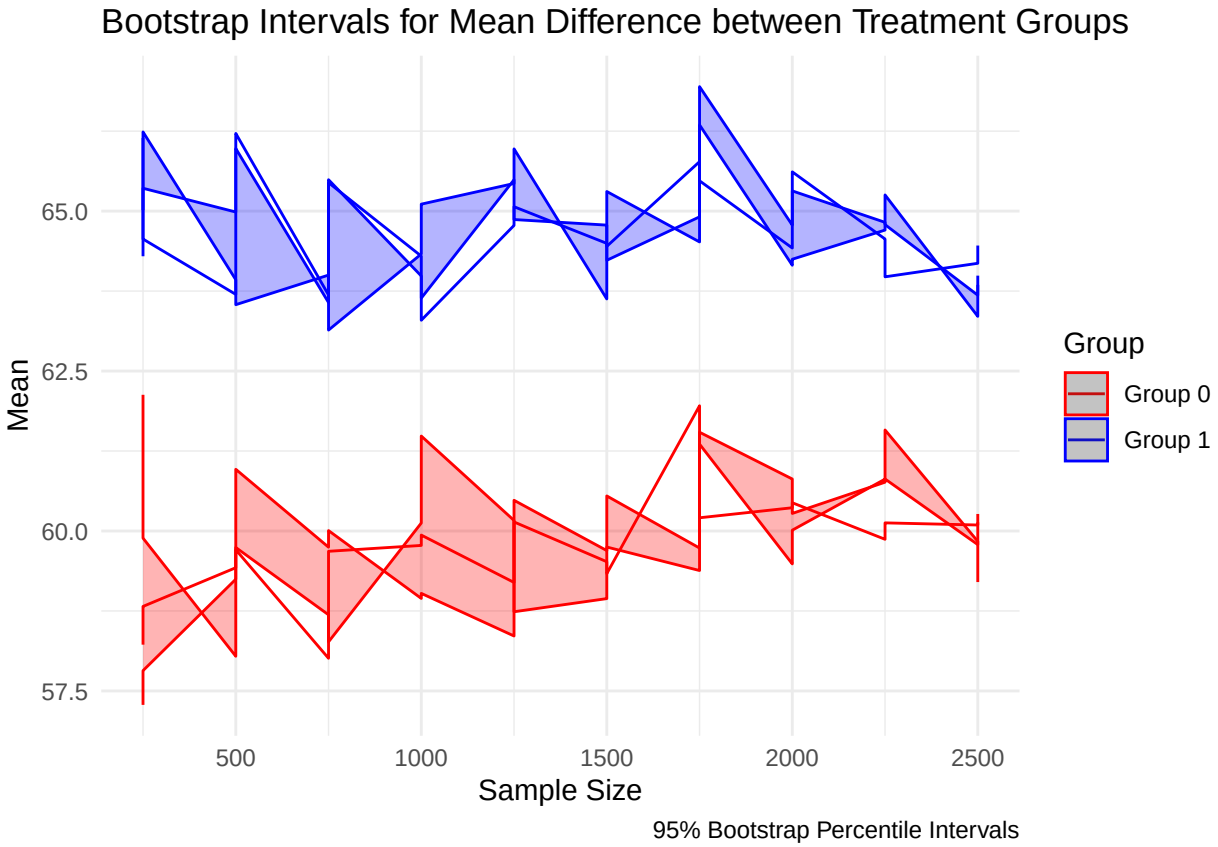
```
#Plotting the line chart.
ggplot(plot_data, aes(x = Sample_Size, y = Mean, color = Group, group = Group)) +
  geom_line() +
  geom_ribbon(aes(ymin = Lower_Bound, ymax = Upper_Bound, fill = Group), alpha = 0.3) +
  labs(
    title = "Bootstrap Intervals for Mean Difference between Treatment Groups",
    x = "Sample Size",
    y = "Mean",
    caption = "95% Bootstrap Percentile Intervals"
  ) +
  scale_color_manual(values = c("red", "blue")) +
  scale_fill_manual(values = c("red", "blue"), guide = FALSE) +
  theme_minimal()
```

```
## Warning: The ‘guide‘ argument in ‘scale_*()‘ cannot be ‘FALSE‘. This was deprecated in
## ggplot2 3.3.4.
## i Please use "none" instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

i recommend specifying x and y inside of each geom_xx() command, for example

ggplot(out, mapping=aes(sample_size))+
geom_line(aes(y=treat_mean, colour = "var1"))+
geom_line(aes(y=cont_mean, colour = "var0"))+
geom_ribbon(aes(ymin = treat_ub, ymax = treat_lb, fill = "grey70"), alpha=0.5)+ geom_ribbon(aes(ymin = cont_ub, ymax = cont_lb, fill = "green"), alpha=0.5)

because it looks like the plot was confused about which values to use in the geom_line function


Bootstrap Intervals for Mean Difference between Treatment Groups
65.0
62.5
60.0
57.5
Mean
500
1000
1500
2000
2500
Sample Size
95% Bootstrap Percentile Intervals
Group
Group 0
Group 1