SEB QBIO310 HW7

Code

1. The Poisson distribution has one parameter lambda. The expectation and the variance of a Poisson random variable are both lambda. We are going to consider three different method of moments estimators to estimate lambda: the sample mean (use the R command mean(x)),the “biased” variance (use the R command mean(x^2) – mean(x)^2), and the sample variance (use the R command var(x)).

Hide

```
set.seed(123)
library(ggplot2)

# Fxn to calculate all three estimates
calculate_estimates <- function(x) {
  mean_est <- mean(x)
  biased_var <- mean(x^2) - mean(x)^2
  sample_var <- var(x)
  return(c(mean_est, biased_var, sample_var))
}
```

a. Simulate 10 independent Poisson random variables each with lambda = 3. Compute the three different estimates for lambda. Independently repeat the previous simulation and computations 1,000 times. For each of the three estimators make a histogram of the 1,000 estimates and compute the expectation, the variance, and the mean squared error.

Hide

```
# Simulation with n = 10
cat("Part a: n = 10\n")
```

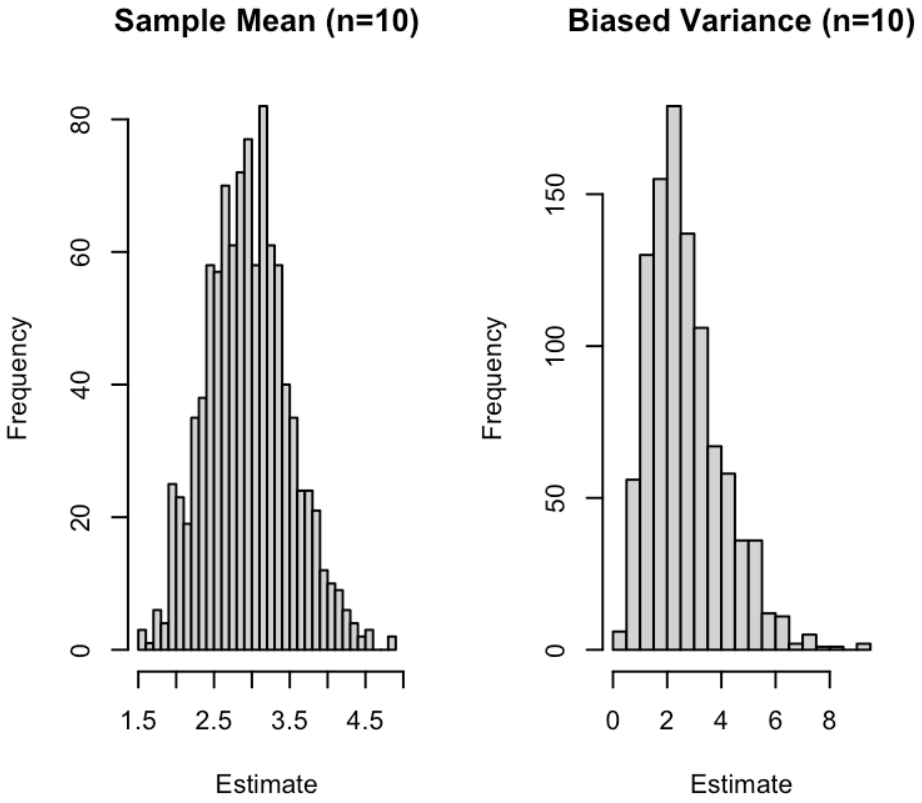
```
Part a: n = 10
```

Hide

```
results_10 <- matrix(nrow = 1000, ncol = 3)

# Running sims
for (i in 1:1000) {
  x <- rpois(10, lambda = 3)
  results_10[i,] <- calculate_estimates(x)
}

colnames(results_10) <- c("Mean", "Biased_Var", "Sample_Var")

#Histograms
par(mfrow = c(1, 3))
hist(results_10[,1], main = "Sample Mean (n=10)", xlab = "Estimate", breaks = 30)
hist(results_10[,2], main = "Biased Variance (n=10)", xlab = "Estimate", breaks = 30)
```

Hide

```
hist(results_10[,3], main = "Sample Variance (n=10)", xlab = "Estimate", breaks = 30)
```



Hide

```
# Calculate statistics
expectations_10 <- colMeans(results_10)
variances_10 <- apply(results_10, 2, var)
mse_10 <- colMeans((results_10 - 3)^2)

# Print results
cat("Expectations:\n")
```

```
Expectations:
```

Hide

```
print(expectations_10)
```

```
      Mean Biased_Var Sample_Var
  3.000800   2.690760   2.989733
```

Hide

```
cat("Variances:\n")
```

```
Variances:
```

Hide

```
print(variances_10)
```

```
      Mean Biased_Var Sample_Var
 0.3011405  1.8369275  2.2678118
```

Hide

```
cat("Mean Squared Errors:\n")
```

```
Mean Squared Errors:
```

Hide

```
print(mse_10)
```

```
      Mean Biased_Var Sample_Var
  0.300840   1.930720   2.265649
```

b. Repeat part (a) but now each simulation has 100 independent Poisson random variables with lambda = 3

Hide

```
# Part b: Simulation with n = 100
cat("\nPart b: n = 100\n")
```

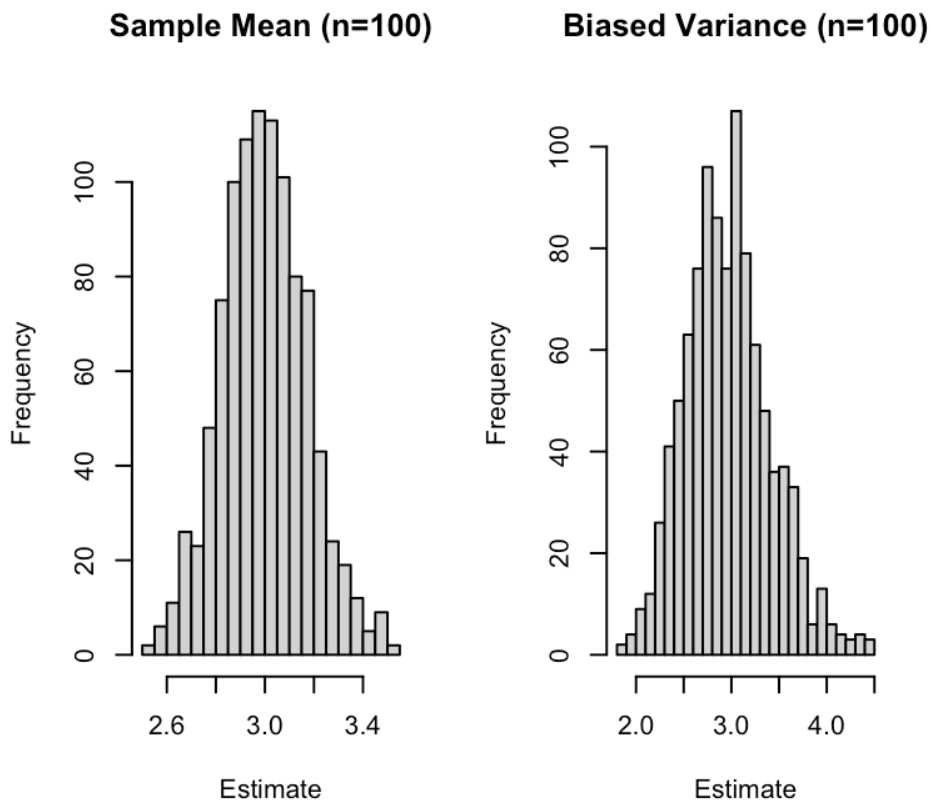
```

Part b: n = 100
```

Hide

```
# Matrix to store results
results_100 <- matrix(nrow = 1000, ncol = 3)

# Run simulations
for(i in 1:1000) {
  x <- rpois(100, lambda = 3)
  results_100[i,] <- calculate_estimates(x)
}

# Name columns
colnames(results_100) <- c("Mean", "Biased_Var", "Sample_Var")

# Histograms
par(mfrow = c(1, 3))
hist(results_100[,1], main = "Sample Mean (n=100)", xlab = "Estimate", breaks = 30)
hist(results_100[,2], main = "Biased Variance (n=100)", xlab = "Estimate", breaks = 30)
```

Hide

```
hist(results_100[,3], main = "Sample Variance (n=100)", xlab = "Estimate", breaks = 30)
```



Hide

```
# Calculating stats
expectations_100 <- colMeans(results_100)
variances_100 <- apply(results_100, 2, var)
mse_100 <- colMeans((results_100 - 3)^2)

# Print results
cat("Expectations:\n")
```

```
Expectations:
```

Hide

```
print(expectations_100)
```

```
     Mean Biased_Var Sample_Var
  3.003290   2.968791   2.998779
```

Hide

```
cat("Variances:\n")
```

```
Variances:
```

Hide

```
print(variances_100)
```

```
     Mean Biased_Var Sample_Var
0.02959767 0.20052274 0.20459416
```

Hide

```
cat("Mean Squared Errors:\n")
```

```
Mean Squared Errors:
```

Hide

```
print(mse_100)
```

```
     Mean Biased_Var Sample_Var
 0.0295789  0.2012962  0.2043911
```

c. Repeat part (a) but now each simulation has 1,000 independent Poisson random variables with lambda = 3.

Hide

```
# Part c: Simulation with n = 1000
cat("\nPart c: n = 1000\n")
```

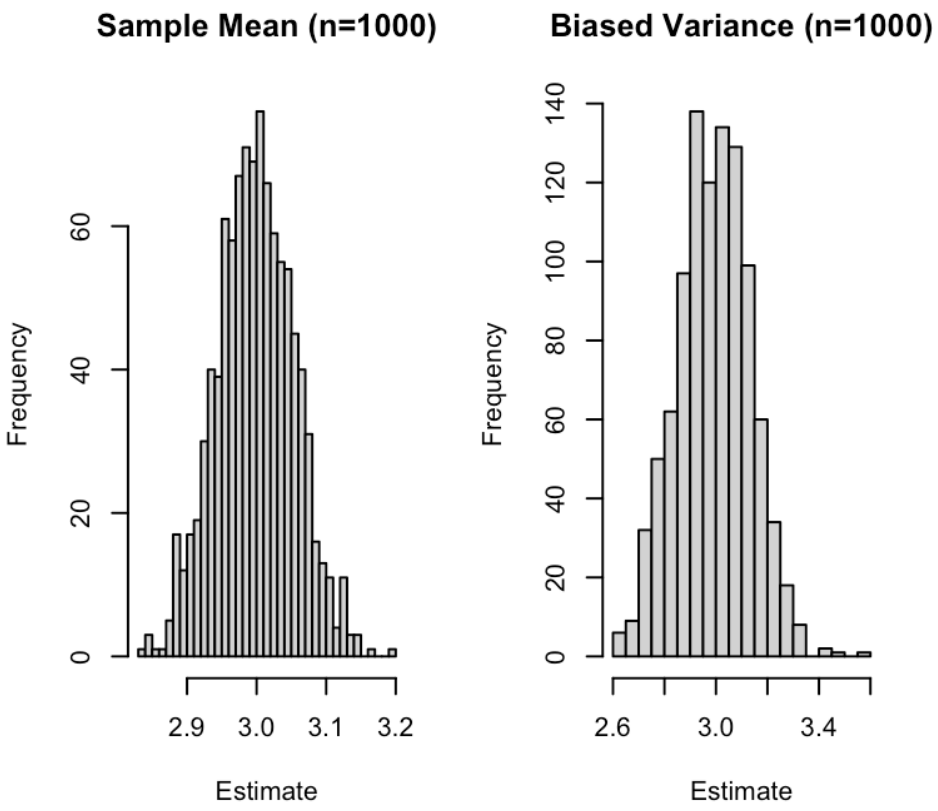
```
Part c: n = 1000
```

Hide

```
# Matrix to store results
results_1000 <- matrix(nrow = 1000, ncol = 3)

# Run simulations
for(i in 1:1000) {
  x <- rpois(1000, lambda = 3)
  results_1000[i,] <- calculate_estimates(x)
}

# Name columns
colnames(results_1000) <- c("Mean", "Biased_Var", "Sample_Var")

# Create histograms
par(mfrow = c(1, 3))
hist(results_1000[,1], main = "Sample Mean (n=1000)", xlab = "Estimate", breaks = 30)
hist(results_1000[,2], main = "Biased Variance (n=1000)", xlab = "Estimate", breaks = 3
0)
```

Hide

```
hist(results_1000[,3], main = "Sample Variance (n=1000)", xlab = "Estimate", breaks = 3
0)
```



Hide

```
# Calculate statistics
expectations_1000 <- colMeans(results_1000)
variances_1000 <- apply(results_1000, 2, var)
mse_1000 <- colMeans((results_1000 - 3)^2)

# Print results
cat("Expectations:\n")
```

```
Expectations:
```

Hide

```
print(expectations_1000)
```

```
     Mean Biased_Var Sample_Var
  2.998892   2.991333   2.994327
```

Hide

```
cat("Variances:\n")
```

```
Variances:
```

Hide

```
print(variances_1000)
```

```
      Mean  Biased_Var  Sample_Var
0.003035224 0.019867555 0.019907349
```

Hide

```
cat("Mean Squared Errors:\n")
```

```
Mean Squared Errors:
```

Hide

```
print(mse_1000)
```

```
      Mean  Biased_Var  Sample_Var
0.003033416 0.019922811 0.019919626
```

d. Which (if any) of the estimators appear consistent? Which estimator appears to be the most efficient? Which (if any) of the estimators appear unbiased? All three estimators are consistent as the sample size increases from n = 10 to n = 100 to n = 1000, their expectations approach the true value, lambda = 3, and the histograms become more concentrated around 3. The sample mean is the most efficient because it has the smallest variance of all the sample sizes based off of their R.console numbers. This is also evident in the histograms where the sample mean has the tightest distribution. The sample mean and sample variance are unbiased, with expectations very close to 3 at all samples sizes. The biased variance is biased at small sample sizes, but the bias decreases as n increases.

2. In this problem we are going to again consider the Cleveland Heart data from last week.

a. Compute the difference of the mean MaxHR for men and the mean MaxHR for women.

Hide

```
# Load the dataset
clevheart = read.csv("ClevelandHeart.csv")

# Check the column names
colnames(clevheart)
```

```
 [1] "Age"         "Sex"         "ChestPain"   "RestBP"      "Chol"
 [6] "Fbs"         "RestECG"     "MaxHR"       "ExAng"       "Fluoroscopy"
[11] "Thal"        "AHD"
```

Hide

```
# Check the first few rows
head(clevheart)
```

	Age <int>	Sex <chr>	ChestPain <chr>	RestBP <int>	Chol <int>	Fbs <lgl>	RestECG <int>	MaxHR <int>	ExAng <chr>
1	63	male	typical	145	233	TRUE	2	150	no
2	67	male	asymptomatic	160	286	FALSE	2	108	yes
3	67	male	asymptomatic	120	229	FALSE	2	129	yes
4	37	male	nonanginal	130	250	FALSE	0	187	no
5	41	female	nontypical	130	204	FALSE	2	172	no
6	56	male	nontypical	120	236	FALSE	0	178	no

6 rows | 1-10 of 12 columns

Hide

```
# Check the unique values in the Sex column
unique(clevheart$Sex)
```

```
[1] "male"   "female"
```

Hide

```
# Check for missing values in the Sex and MaxHR columns
sum(is.na(clevheart$Sex))
```

```
[1] 0
```

Hide

```
sum(is.na(clevheart$MaxHR))
```

```
[1] 0
```

Hide

```
# Compute the mean MaxHR for men (Sex = 1) and women (Sex = 0)
mean_maxhr_men = mean(clevheart$MaxHR[clevheart$Sex == "male"], na.rm = TRUE)
mean_maxhr_women = mean(clevheart$MaxHR[clevheart$Sex == "female"], na.rm = TRUE)

# Compute the difference: mean(MaxHR for men) - mean(MaxHR for women)
diff_maxhr = mean_maxhr_men - mean_maxhr_women

# Print the results
cat("Mean MaxHR for men:", mean_maxhr_men, "\n")
```

```
Mean MaxHR for men: 148.8447
```

Hide

```
cat("Mean MaxHR for women:", mean_maxhr_women, "\n")
```

```
Mean MaxHR for women: 151.2268
```

Hide

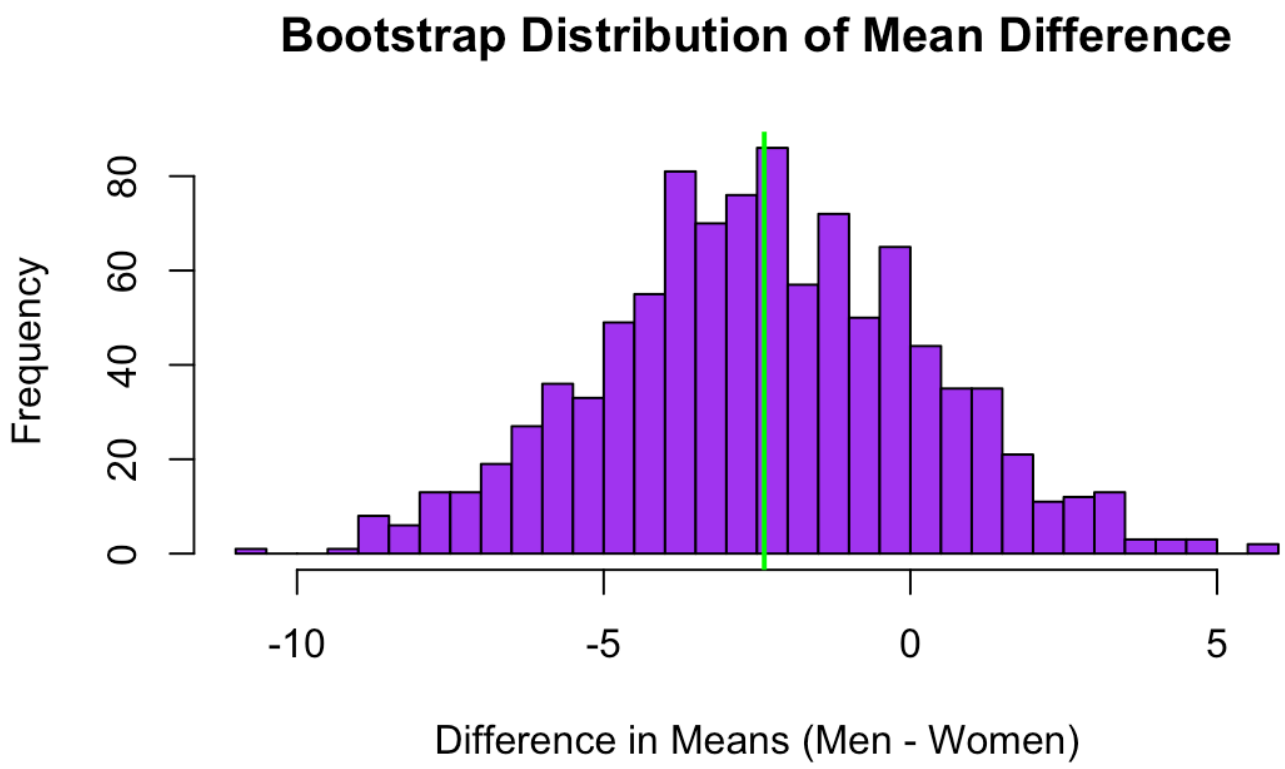
```
cat("Difference (men - women):", diff_maxhr, "\n")
```

```
Difference (men - women): -2.382144
```

b. Write a function to compute the 95% bootstrap confidence interval for the difference of the means from part (a). Compute three different bootstrap confidence intervals: the normal, the percentile, and the pivotal (also called the basic). Comment on any similarities or differences between these three confidence intervals. Also, make a histogram of the difference of the means from the different bootstrap samples.

Hide

```
# Function to compute bootstrap CIs for difference of means
bootstrap_diff_ci <- function(data, B = 1000, alpha = 0.05) {
  men_data = data$MaxHR[data$Sex == "male"]
  women_data = data$MaxHR[data$Sex == "female"]
  n_men = length(men_data)
  n_women = length(women_data)
  boot_diff = numeric(B)

  # Generate bootstrap samples
  for(i in 1:B) {
    boot_men = sample(men_data, n_men, replace = TRUE)
    boot_women = sample(women_data, n_women, replace = TRUE)
    boot_diff[i] = mean(boot_men, na.rm = TRUE) - mean(boot_women, na.rm = TRUE)
  }

  # Observed difference
  obs_diff = mean(men_data, na.rm = TRUE) - mean(women_data, na.rm = TRUE)

  # Normal CI
  se = sd(boot_diff)
  normal_ci = c(obs_diff - qnorm(1-alpha/2)*se, obs_diff + qnorm(1-alpha/2)*se)

  # Percentile CI
  percentile_ci = quantile(boot_diff, c(alpha/2, 1-alpha/2))

  # Pivotal (Basic) CI
  pivotal_ci = 2*obs_diff - quantile(boot_diff, c(1-alpha/2, alpha/2))

  # Histogram
  hist(boot_diff, breaks = 30, main = "Bootstrap Distribution of Mean Difference",
       xlab = "Difference in Means (Men - Women)", col = "purple")
  abline(v = obs_diff, col = "green", lwd = 2)

  return(list(normal = normal_ci, percentile = percentile_ci, pivotal = pivotal_ci))
}

# Compute CIs
set.seed(123)  # for reproducibility
boot_results = bootstrap_diff_ci(clevheart)
```



Hide

```
cat("Normal CI:", round(boot_results$normal, 3), "\n")
```

```
Normal CI: -7.557 2.792
```

Hide

```
cat("Percentile CI:", round(boot_results$percentile, 3), "\n")
```

```
Percentile CI: -7.646 2.939
```

Hide

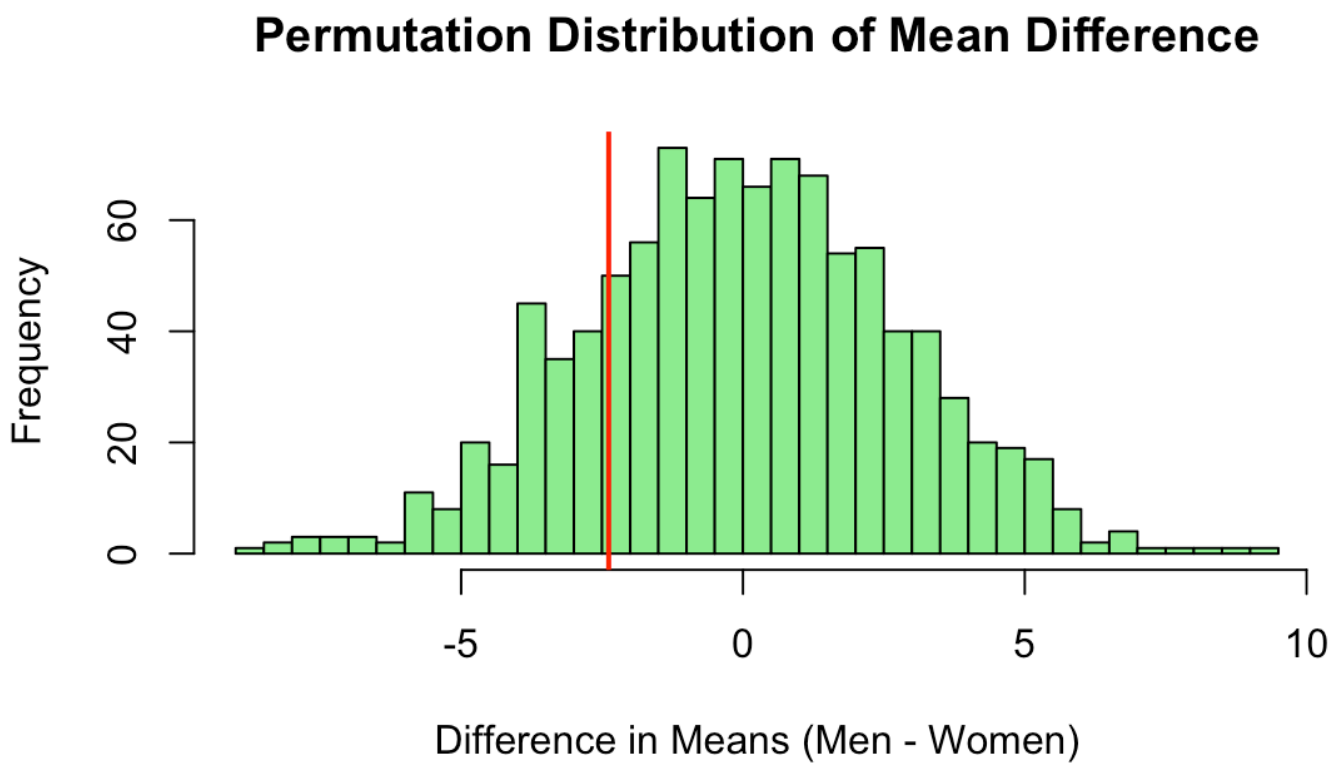
```
cat("Pivotal CI:", round(boot_results$pivotal, 3), "\n")
```

```
Pivotal CI: -7.704 2.881
```

c. Write a function that uses the permutation test to test the null hypothesis that the two means are equal (use significance level lambda = 0.05). Compute the p-value and clearly state your conclusion. Also, make a histogram of the difference of the means (or, instead, the t-statistics) from the different permutations.

Hide

```
# Function for permutation test
perm_test_means <- function(data, B = 1000, alpha = 0.05) {
  men_data = data$MaxHR[data$Sex == "male"]
  women_data = data$MaxHR[data$Sex == "female"]
  obs_diff = mean(men_data, na.rm = TRUE) - mean(women_data, na.rm = TRUE)
  combined = c(men_data, women_data)
  n_men = length(men_data)
  n_women = length(women_data)
  perm_diff = numeric(B)

  # Generate permutations
  for(i in 1:B) {
    perm_sample = sample(combined)  # shuffle all data
    perm_men = perm_sample[1:n_men]
    perm_women = perm_sample[(n_men+1):(n_men+n_women)]
    perm_diff[i] = mean(perm_men, na.rm = TRUE) - mean(perm_women, na.rm = TRUE)
  }

  # Two-sided p-value
  p_value = mean(abs(perm_diff) >= abs(obs_diff))

  # Histogram
  hist(perm_diff, breaks = 30, main = "Permutation Distribution of Mean Difference",
       xlab = "Difference in Means (Men - Women)", col = "lightgreen")
  abline(v = obs_diff, col = "red", lwd = 2)

  return(p_value)
}

# Run test
set.seed(123)
p_val_perm = perm_test_means(clevheart)
```



Hide

```
cat("Permutation test p-value:", round(p_val_perm, 3), "\n")
```

```
Permutation test p-value: 0.391
```

Hide

```
cat("Conclusion: ", ifelse(p_val_perm < 0.05,
                        "Reject H0: Evidence of difference in means",
                        "Fail to reject H0: No evidence of difference in means"), "\n")
```

```
Conclusion:  Fail to reject H0: No evidence of difference in means
```

d. The R command t.test uses the Central Limit Theorem to compute the confidence interval and the p-value. Run this command and comment on any similarities or differences compared to the bootstrap confidence intervals and the permutation test p-value.

Hide

```
# Run t.test
t_test_result = t.test(MaxHR ~ Sex, data = clevheart, var.equal = FALSE)  # Welch's t-te
st
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  MaxHR by Sex
t = 0.90442, df = 223.85, p-value = 0.3667
alternative hypothesis: true difference in means between group female and group male is
not equal to 0
95 percent confidence interval:
 -2.808276  7.572564
sample estimates:
mean in group female   mean in group male
            151.2268             148.8447
```

Hide

```
# Extract CI and p-value
cat("t.test CI:", round(t_test_result$conf.int, 3), "\n")
```

```
t.test CI: -2.808 7.573
```

Hide

```
cat("t.test p-value:", round(t_test_result$p.value, 3), "\n")
```

```
t.test p-value: 0.367
```

3. The attached data set “lawschools.csv” has data on average LSAT and average GPA for incoming law school students at 15 law schools (the two numbers on each row are for the same school).

a. Make a scatter plot with LSAT on the x-axis and GPA on the y-axis and label the axes. Use the R command cor to compute the correlation between LSAT and GPA.

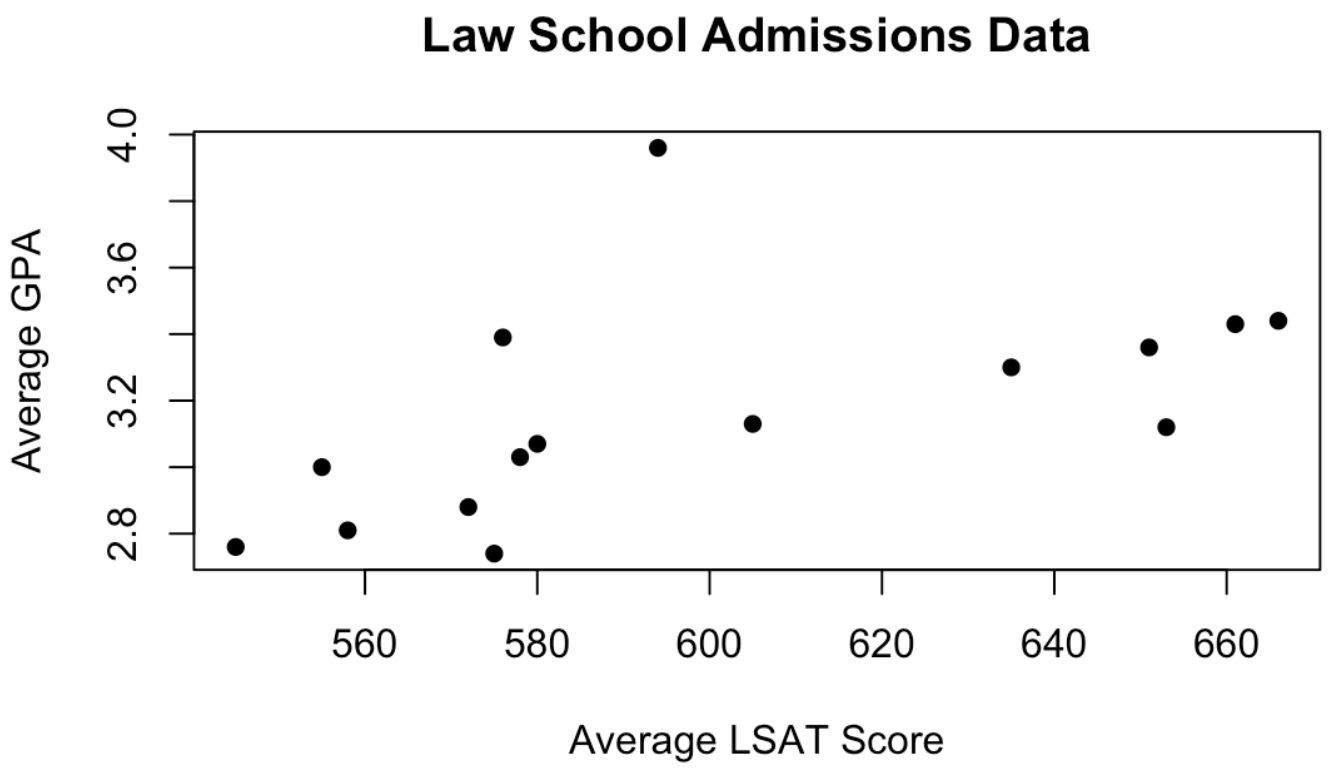
Hide

```
# Assuming lawschool data is loaded with columns LSAT and GPA
lawschool = read.csv("lawschools.csv")
dim(lawschool)  # Should show 15 rows and 2 columns
```

```
[1] 15  2
```

Hide

```
# Create scatter plot
plot(lawschool$LSAT, lawschool$GPA,
     xlab = "Average LSAT Score",
     ylab = "Average GPA",
     main = "Law School Admissions Data",
     pch = 16)  # solid dots
```



Hide

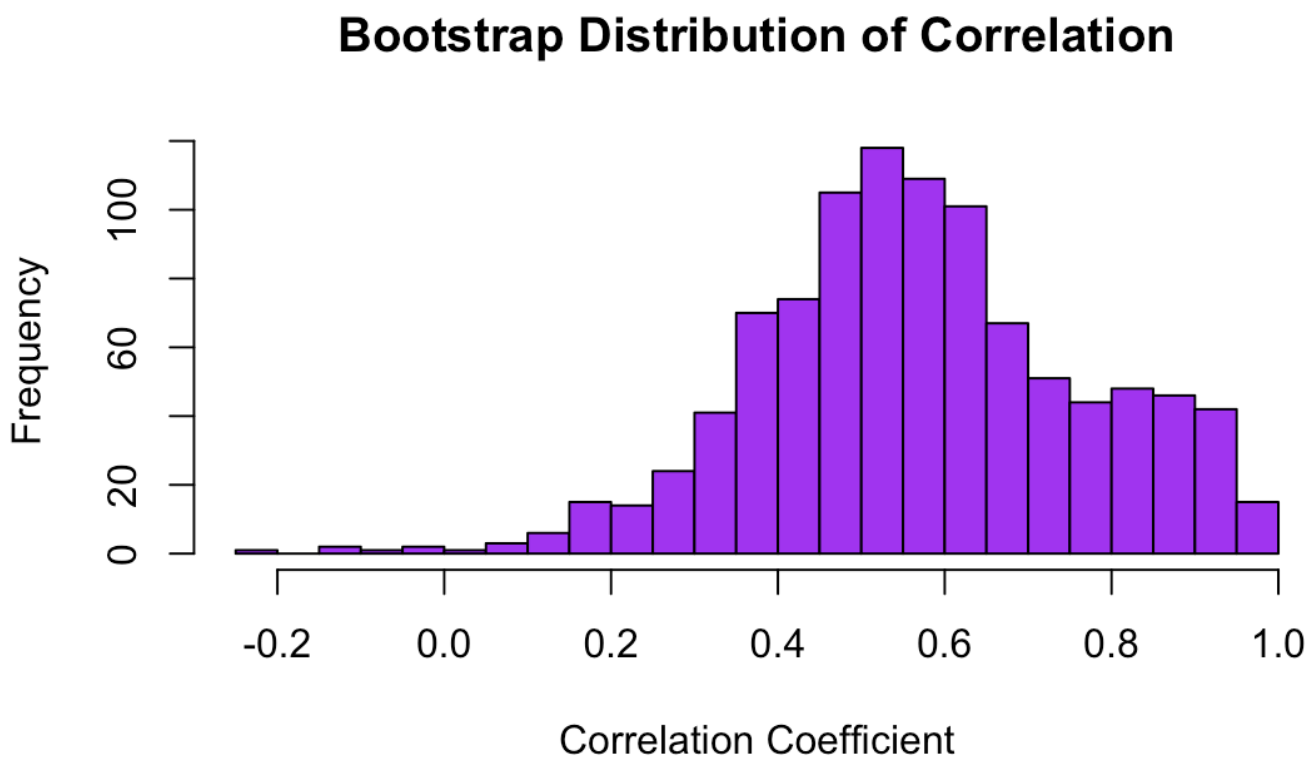
```
# Compute correlation
cor_value = cor(lawschool$LSAT, lawschool$GPA)
print(paste("Correlation coefficient:", round(cor_value, 3)))
```

```
[1] "Correlation coefficient: 0.546"
```

b. Write a function to compute the 95% confidence interval for the correlation between the LSAT and GPA numbers. Note: this data is paired and the Cleveland Heart data was unpaired, so this function should be different than your function from problem #2. Compute three different bootstrap confidence intervals: the normal, the percentile, and the pivotal (also called the basic). Comment on any similarities or differences between these three confidence intervals. Also, make a histogram of the correlation coefficients for the different bootstrap samples.

Hide

```
# Function to compute bootstrap CIs for correlation
bootstrap_cor_ci <- function(x, y, B = 1000, alpha = 0.05) {
  n <- length(x)
  boot_cor <- numeric(B)

  # Generate bootstrap samples
  for(i in 1:B) {
    indices <- sample(1:n, n, replace = TRUE)
    boot_cor[i] <- cor(x[indices], y[indices])
  }

  # Normal CI
  se <- sd(boot_cor)
  cor_est <- cor(x, y)
  normal_ci <- c(cor_est - qnorm(1-alpha/2)*se,
                 cor_est + qnorm(1-alpha/2)*se)

  # Percentile CI
  percentile_ci <- quantile(boot_cor, c(alpha/2, 1-alpha/2))

  # Pivotal (Basic) CI
  pivotal_ci <- 2*cor_est - quantile(boot_cor, c(1-alpha/2, alpha/2))

  # Histogram
  hist(boot_cor, breaks = 30, main = "Bootstrap Distribution of Correlation",
       xlab = "Correlation Coefficient", col = "purple")

  return(list(normal = normal_ci,
              percentile = percentile_ci,
              pivotal = pivotal_ci,
              boot_samples = boot_cor))
}

# Compute CIs
set.seed(123)  # for reproducibility
results <- bootstrap_cor_ci(lawschool$LSAT, lawschool$GPA)
```



Hide

```
cat("Normal CI:", round(results$normal, 3), "\n")
```

```
Normal CI: 0.164 0.928
```

Hide

```
cat("Percentile CI:", round(results$percentile, 3), "\n")
```

```
Percentile CI: 0.185 0.935
```

Hide

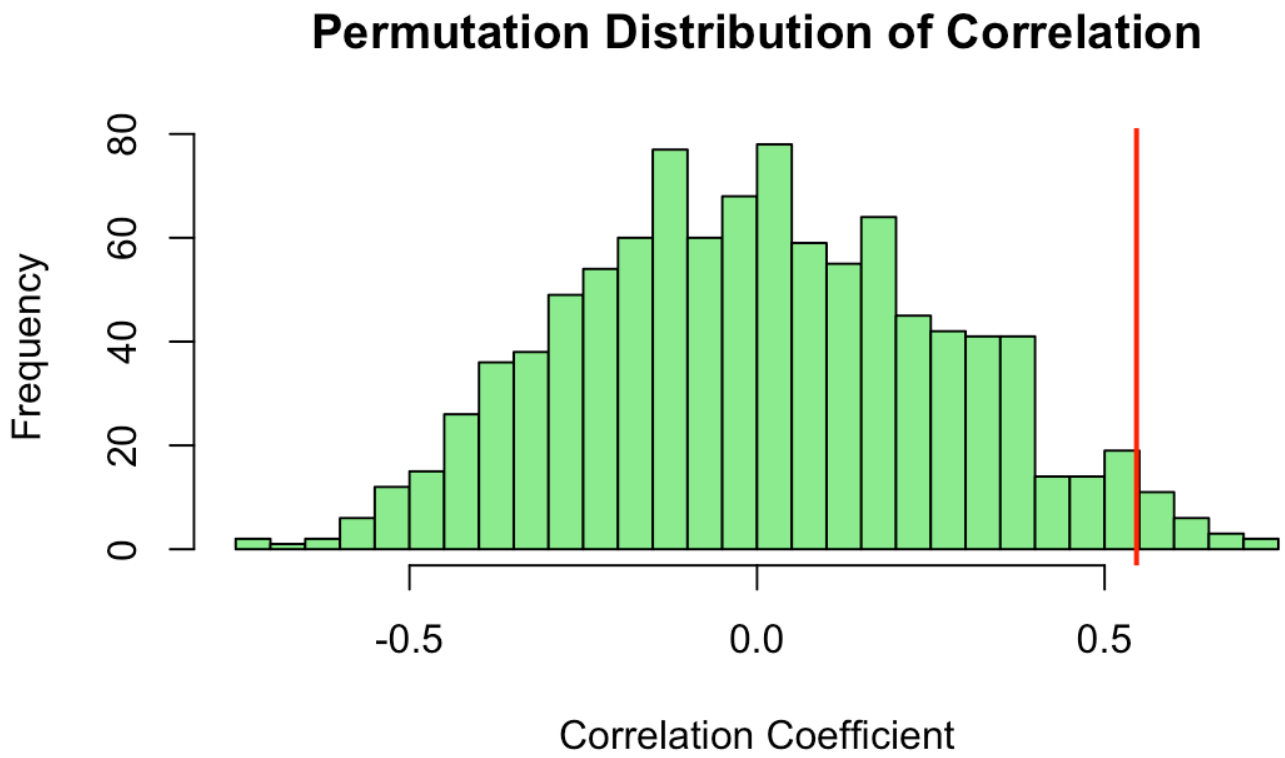
```
cat("Pivotal CI:", round(results$pivotal, 3), "\n")
```

```
Pivotal CI: 0.157 0.907
```

c. Write a function that uses the permutation test to test the null hypothesis that the LSAT and GPA numbers are uncorrelated (use significance level alpha = 0.05). Again, because this data is paired, this function should be different than your function from problem #2. Compute the p-value and clearly state your conclusion. Also, make a histogram of the correlation coefficients for the different permutations.

Hide

```
# Function for permutation test
perm_test_cor <- function(x, y, B = 1000, alpha = 0.05) {
  n <- length(x)
  obs_cor <- cor(x, y)
  perm_cor <- numeric(B)

  # Generate permutations
  for(i in 1:B) {
    perm_y <- sample(y)  # shuffle y values
    perm_cor[i] <- cor(x, perm_y)
  }

  # Two-sided p-value
  p_value <- mean(abs(perm_cor) >= abs(obs_cor))

  # Histogram
  hist(perm_cor, breaks = 30, main = "Permutation Distribution of Correlation",
       xlab = "Correlation Coefficient", col = "lightgreen")
  abline(v = obs_cor, col = "red", lwd = 2)

  return(p_value)
}

# Run test
set.seed(123)
p_val <- perm_test_cor(lawschool$LSAT, lawschool$GPA)
```



Hide

```
cat("Permutation test p-value:", round(p_val, 3), "\n")
```

```
Permutation test p-value: 0.034
```

Hide

```
cat("Conclusion: ", ifelse(p_val < 0.05,
                        "Reject H0: Evidence of correlation",
                        "Fail to reject H0: No evidence of correlation"), "\n")
```

```
Conclusion:  Reject H0: Evidence of correlation
```

d. The R command cor.test uses the Central Limit Theorem to compute the confidence interval and the p-value. Run this command and comment on any similarities or differences compared to the bootstrap confidence intervals and the permutation test p-value.

Hide

```
# Run cor.test
test_result <- cor.test(lawschool$LSAT, lawschool$GPA)
print(test_result)
```

```

    Pearson's product-moment correlation

data:  lawschool$LSAT and lawschool$GPA
t = 2.3493, df = 13, p-value = 0.03527
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.04672209 0.82692804
sample estimates:
      cor
0.5459189
```

Hide

```
# Extract CI and p-value
cat("cor.test CI:", round(test_result$conf.int, 3), "\n")
```

```
cor.test CI: 0.047 0.827
```

Hide

```
cat("cor.test p-value:", round(test_result$p.value, 3), "\n")
```

```
cor.test p-value: 0.035
```