Homework Assignment #2

Math 437 - Modern Data Analysis

Due February 24, 2023

Instructions

```
library(ISLR2)
library(ggplot2)
library(dplyr)
library(car)
library(boot)
```

You should submit either two or three files:

1. You should write your solutions to the Simulation and Applied Problems in this R Markdown file and submit the (.Rmd) file.
2. You should knit the final solution file to pdf and submit the pdf. If you are having trouble getting code chunks to run, add `eval = FALSE` to the chunks that do not run. If you are having trouble getting R Studio to play nice with your LaTeX distribution, I will begrudgingly accept an HTML file instead.
3. Solutions to the Key Terms and Conceptual Problems can be submitted in a separate Word or pdf file or included in the same files as your solutions to the Simulation and Applied Problems.

This homework assignment is worth a total of **40 points**.

Key Terms (5 pts)

Read Chapter 13 of Introduction to Statistical Learning, Second Edition. Based on your reading, answer the following questions.

1. What is a *p-value*? What is the difference between a one-sided and a two-sided p-value? A `p-value` is defined as the probability of observing a test statistic equal to or more extreme than the observed statistic, under the assumption that H_0 is true.

2. In traditional NHST-style significance testing, what are the two possible decisions? When do we make each decision? The two possible decisions are to reject the null hypothesis or fail to reject it. We make each decision based on what our p-value is. A smaller p-value indicates that there is evidence against H_0, thus provides evidence against it.

3. What is the difference between a *Type I Error* and a *Type II Error*? A `Type I Error` is when we reject H_0 when it is actually true. A `Type II Error` is when we do not reject H_0 even though it is false.

4. Briefly explain why it is necessary to adjust the significance level (or equivalently, the p-values) when testing a large number of null hypotheses. It is necessary to adjust the significance level when testing a large number of null hypotheses because there is a possibility of rejecting a lot of true null hypotheses, leading to a great number of Type I Errors.

5. Compare and contrast the *Family-Wise Error Rate* (FWER) and the *False Discovery Rate* (FDR). `FWER` is the probability of making at least one Type I Error while `FDR` is a ratio of false positives to total positives to ensure that we are rejecting as many null hypotheses as possible while guaranteeing

that the a certain percentage of those rejected null hypotheses are false positives. It is impossible to control the `FDR` but we can control the `FWER`. ##################

6. Compare and contrast the *Bonferroni Method* and *Holm's Step-Down Method* for controlling the FWER. The `Bonferroni method` refers to adjusting the alpha in FWER to control the probability of performing a Type I Error. `Holm's method` also controls the FWER, but it is adjusted to result in fewer Type II Errors. Holm's method is less conservative to Bonferroni's method, and uniformly more powerful. In Holm's method, the threshold depends on the values of all m p-values, while Bonferri's method depends on the p-value lower than the ratio of alpha over m p-values.

7. Why do we prefer to use *Tukey's Method* or *Scheffe's Method* to control the FWER? In what conditions is it appropriate to use those methods instead of the Bonferroni or Holm methods? There are certain very specific settings that we prefer to use `Tukey's Method` or `Scheffe's Method`. `Tukey's Method` allows us to control the FWER at some level alpha while rejecting all null hypotheses where the p-value falls below the alpha. ##################

8. Briefly describe the *Benjamini-Hochberg* procedure for controlling the FDR. The `Benjamini-Hochberg` procedures ensures that, on average, no more than a fraction q of the rejected null hypotheses are false positives. First, we determine what level we want to control the FDR, known otherwise as q. Then we compute the p-values #####

9. What is/are the major assumption(s) of a *permutation test*? What is the general procedure for obtaining the null distribution of a test statistic using a permutation test? The major assumption of a `permutation test` is that the data was generated according to some kind of random assignment under H_0 where the result is no effect. For two sample t-tests, we assume that the data in both groups come from the same population distribution under H_0. The general procudre to obtain the null distribution is to create a sampling distribution of a t-statistic by "mimicking" the random assignment under H_0 using the observed values.

10. When is it useful/recommended to use a permutation testing approach as opposed to a traditional theory-based approach? A permutation testing approach is useful when we are dealing with a smaller sample size or when parametric are not met with traditional theory-based approach. There are only a few assumptions so it is easier to use.

Conceptual Problems

Conceptual Problem 1 (5 pts)

Textbook Exercise 13.7.6 ######## (a) How many false positives, false negatives, true positives, true negatives, Type I errors, and Type II errors result from applying the Bonferroni procedure to control the FWER at level $\alpha = 0.05$? There are no Type I Errors, as mentioned by the book, which implies that there are no false positives and 2 true positives when applying the Bonferroni method. Based on the graphs, the first two panels show that there are 7 true negatives and 1 false negative. The last panel show 3 true negatives and 5 false negatives. This implies there were 5 Type II Errors from applying the Bonferroni method.

(b) How many false positives, false negatives, true positives, true negatives, Type I errors, and Type II errors result from applying the Holm procedure to control the FWER at level $\alpha = 0.05$? There are no Type I errors, which also means no false positives from applying the Holm procedure. From the graph, the first panel shows that there are 7 true negatives and 1 false positive. In the second and third panel, there are 8 true negatives and no false positives. This implies there were no Type II Errors when applying the Holm procedure.

(c) What is the false discovery rate associated with using the Bonferroni procedure to control the FWER at level $\alpha = 0.05$? The false discovery rate is 0 since there were no Type I Errors when using the Bonferroni procedure.

(d) What is the false discovery rate associated with using the Holm procedure to control the FWER at level $\alpha = 0.05$? The false discovery rate is 0 since there were no Type I Errors when using the Holm's

procedure.

(e) How would the answers to (a) and (c) change if we instead used the Bonferroni procedure to control the FWER at level $\alpha = 0.001$? I believe I do not have the right answers.

Conceptual Problem 2 (2 pts)

Suppose that we test $m = 1000$ independent null hypotheses, of which 10% are true, at significance level $\alpha = 0.05$ and achieve a false discovery rate of $q = 0.20$. Construct a table following Table 13.2 in the textbook, identifying the appropriate values of V, S, U, W, and R in this situation.

```
table <- matrix(c(45, 955, 1000, 45, 55, 100, 90, 10101, 1100), ncol = 3, byrow = TRUE)
colnames(table) <- c("H0 is true","H0 is false", "Total")
rownames(table) <- c("Reject H0", "Do not reject H0","Total")
table
```

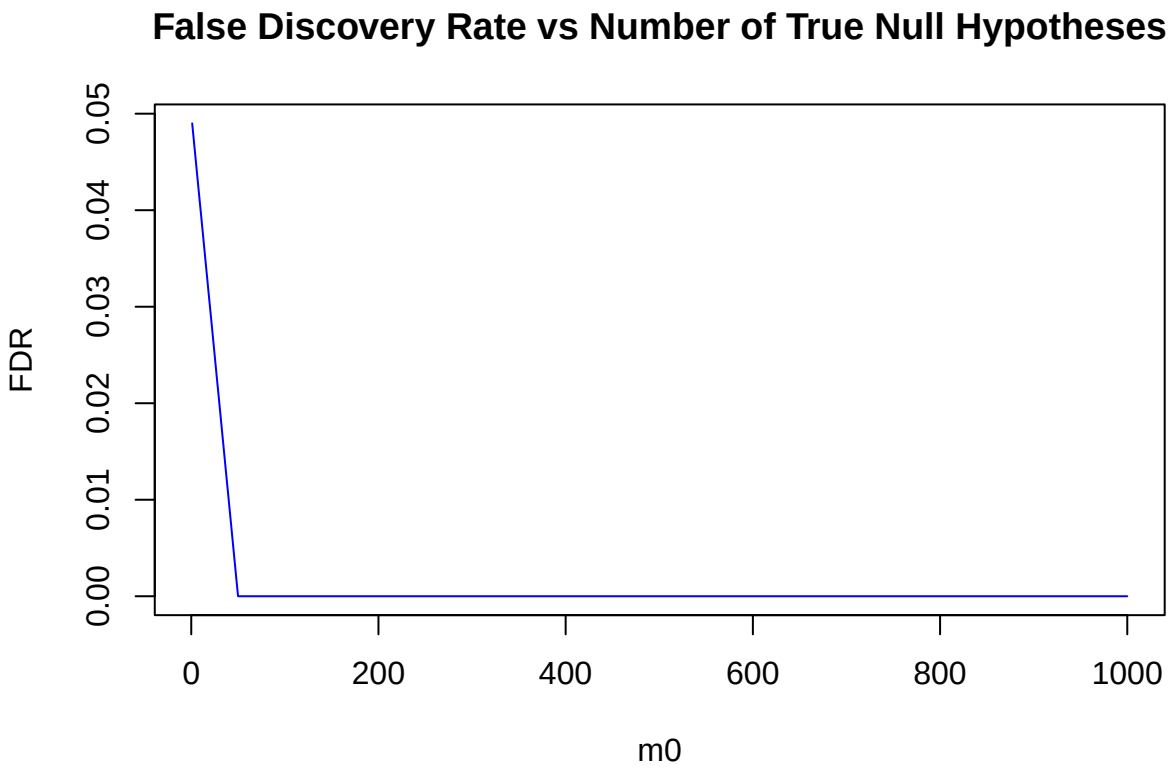
```
##                  H0 is true H0 is false Total
## Reject H0                45         955  1000
## Do not reject H0         45          55   100
## Total                    90       10101  1100
```

Conceptual Problem 3 (3 pts)

Suppose that we test $m = 1000$ independent null hypotheses, of which an unknown number m_0 are true, at significance level $\alpha = 0.05$. Suppose that each test also has a power of 0.80. Find and plot the false discovery rate as a function of m_0.

```
m <- 1000
alpha <- 0.05
power <- 0.80

m0 <- rep(0, m)
for (i in 1:m){
  m0[i] <- i
}

S <- m0
V <- m - m0
R <- m0 +  (m - m0) * (1 - power)
U <- pmax(R * (alpha - S/m), 0)

FDR <- U / pmax(R, 1)
```

```
plot(m0, FDR, type = "l", col = "blue", xlab = "m0", ylab = "FDR", main = "False Discovery Rate vs Numb
```



Conceptual Problem 4 (2.5 pts)

Textbook Exercise 5.4.2 parts (a), (b), and (c).

We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer. The probability that the first boothstrap observation is NOT the jth observation is 1 - 1/n because 1/n is if the jth observation is selected as the first bootstrap observation.

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample? Since bootstrapping is done with replacement, the probability that the second boostrap observation is not the jth observation is also 1 - 1/n.

(c) Argue that the probability that the jth observation is not in the bootstrap sample is (1 - 1/n)ˆn. The probability that the jth observation not in the boostrap sample is not (1-1/n)ˆn because we are boostrapping without replacement which means the probability would alwayas be 1-1/n.

Simulation Problems

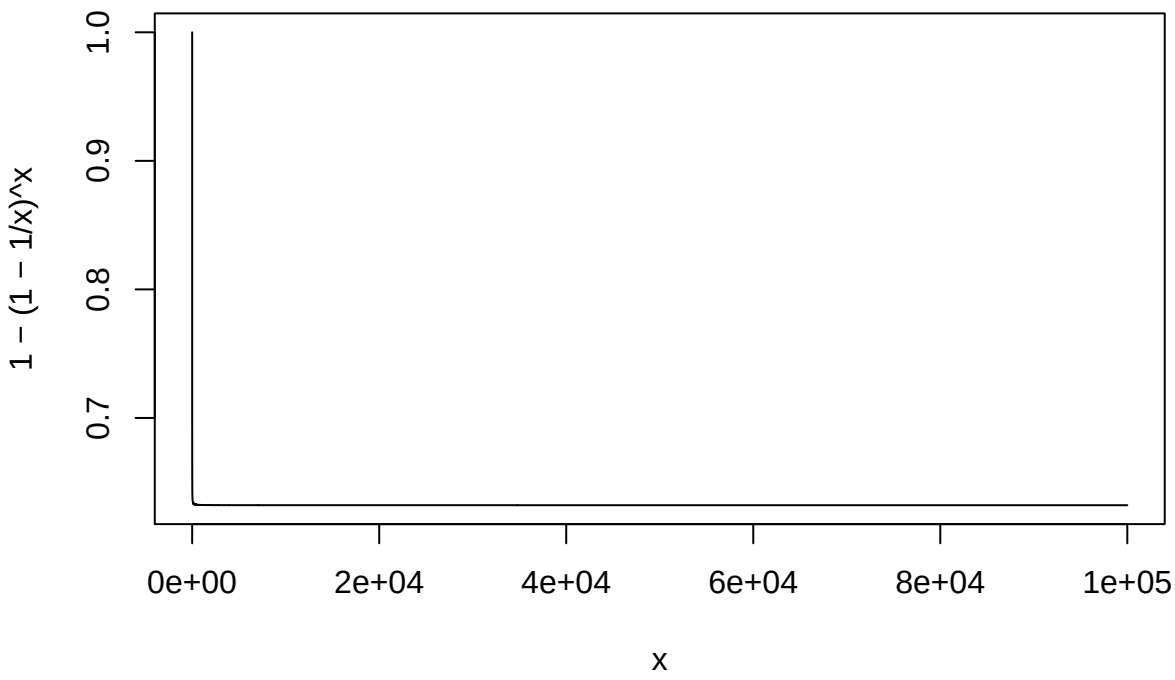
Simulation Problem 1 (Code: 1 pt; Explanation: 0.5 pts)

Textbook Exercise 5.4.2 parts (e), (g), and (h). For part (g), you should create a line plot (using either `plot` with argument `type = "l"` or `geom_line`). Then, to make clearer what you should be commenting on, find the limit as $n \to \infty$ of the probability that the j^{th} observation is in your bootstrap sample and add a horizontal red line (using `abline` or `geom_hline`) at that value. (Hint: the limit as $n \to \infty$ of the expression in part (c) is well-known and easily found on the Internet.)

(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample? The probability when n = 100 is (1 - (1/100)ˆ100 = 0.634

(f) Create a plot that displays, for each integer value of n from 1 to 100, 000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

```
x = 1:100000
plot(x, 1 -  (1 - 1/x)^x, type = "l")
```




```
# I compared with other groups and they got a different answer
```

```
store = rep(NA, 100000)
for(i in 1:100000){
  store[i] = sum(sample(1:100, rep = TRUE) == 4)>0
}

mean(store)
```

```
## [1] 0.63414
```

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

Simulation Problem 2 (Code: 1.5 pts; Explanation: 3.5 pts)

Copy the *functions* you created in the Bootstrap Confidence Intervals class activity as well as Simulation Parts 3, 4, and 5.

Write a brief summary of what you learned from the activity. Make sure to address the following questions:

functions:

```
bootstrap_resample <- function(data_vector, B, summary_fn = mean, seed = 100,...){
  # data_vector: a vector of data
  # B: the number of bootstrap resamples
  # summary_fn: the name of a function to apply to the resampled data
  # ...: any additional arguments to summary_fn
```

```
  boot_samples <- matrix(0, nrow = length(data_vector), ncol = B)

  n <- length(data_vector)
  # We need to add some code here!
  set.seed(seed)
  for(i in 1:B){
      boot_samples[,i] <- sample(data_vector, size = n, replace = TRUE)
  }


  boot_stat <- apply(boot_samples, 2, summary_fn, ...)
  # Seriously, do not name any arguments x or FUN when using apply within a function
  # Advice from someone who spent 30 minutes debugging a sample solution

  return(boot_stat)
}
```

```
bootstrap_ci <- function(data_vector, method, B = 1000, seed = 100, C = 0.95, summary_fn = mean, ...){
  # data_vector: a vector of data
  # method: the CI method
  # B: the number of bootstrap resamples
  # seed: the seed to use
  # C: the confidence level as a decimal
  # summary_fn: the name of a function to apply to the resampled data
  # ...: any additional arguments to summary_fn



  obs_stat <- do.call(summary_fn, args = list(data_vector, ...))
  # do.call allows you to call a function without having to hard-code what that function is
  # the args argument is a list of arguments to the function
  # so this will find the observed value of the statistic given the original data vector

  bootstrap_values <- bootstrap_resample(data_vector, B, summary_fn, ...)

  alpha = 1 - C
  if (method == "percentile"){
      # write code to get the percentile confidence level out of the returned
      # bootstrap_values and store in a length 2 vector boot_ci
    boot_ci = quantile(bootstrap_values, probs = c(alpha/2, 1 - alpha/2))
  } else if (method == "basic"){
    # write code to get the "basic" confidence level and store in a length 2 vector boot_ci
    boot_ci = 2 * obs_stat - quantile(bootstrap_values, probs = c(1 - alpha/2, alpha/2))
  } else if (method == "normal"){
    # write code to get the normal-theory confidence interval and store in a length 2 vector boot_ci
    center = 2 * obs_stat - mean(bootstrap_values)
    crit_value = qnorm(alpha/2)
    se_boot = sd(bootstrap_values)
    boot_ci = center + c(1,-1)*crit_value*se_boot
    # make sure to use the adjustments in the course notes, e.g., don't use 1/sqrt(n) as your standard
  }

  return(boot_ci)
}
```

simulation part 3

```
set.seed(437)
# need to do the simulation now!
sim_data3 = matrix(rexp(1000*100), nrow = 1000, ncol = 100)
```

```
pop_mean3 <- 1
ci_t3 <- apply(sim_data3, 1, function(x) t.test(x)$conf.int)
ci_t_df3 <- as.data.frame(t(ci_t3))
names(ci_t_df3) <- c("lower", "upper")
ci_t_df3 %>% mutate(covered = lower <= pop_mean3 & upper >= pop_mean3) %>%
  summarize(coverage_probability = mean(covered))
```

```
##   coverage_probability
## 1                0.937
```

```
# You should be able to just run this code chunk without any fixes
ci_perc3 <- apply(sim_data3, 1, bootstrap_ci, method = "percentile", B = 1000, summary_fn = mean, na.rm
# don't need any ... arguments here, but illustrating the idea of the ...
# notice that bootstrap_ci has default seed = 100 and C = 0.95 arguments
ci_perc_df3 <- as.data.frame(t(ci_perc3))
names(ci_perc_df3) <- c("lower", "upper")
ci_perc_df3 %>% mutate(covered = lower <= pop_mean3 & upper >= pop_mean3) %>%
  summarize(coverage_probability = mean(covered))
```

```
##   coverage_probability
## 1                0.936
```

```
ci_basic3 <- apply(sim_data3, 1, bootstrap_ci, method = "basic", B = 1000, summary_fn = mean, na.rm = TR
# don't need any ... arguments here, but illustrating the idea of the ...
# notice that bootstrap_ci has default seed = 100 and C = 0.95 arguments
ci_basic_df3 <- as.data.frame(t(ci_basic3))
names(ci_basic_df3) <- c("lower", "upper")
ci_basic_df3 %>% mutate(covered = lower <= pop_mean3 & upper >= pop_mean3) %>%
  summarize(coverage_probability = mean(covered))
```

```
##   coverage_probability
## 1                0.924
```

```
ci_normal3 <- apply(sim_data3, 1, bootstrap_ci, method = "normal", B = 1000, summary_fn = mean, na.rm =
# don't need any ... arguments here, but illustrating the idea of the ...
# notice that bootstrap_ci has default seed = 100 and C = 0.95 arguments
ci_normal_df3 <- as.data.frame(t(ci_normal3))
names(ci_normal_df3) <- c("lower", "upper")
ci_normal_df3 %>% mutate(covered = lower <= pop_mean3 & upper >= pop_mean3) %>%
  summarize(coverage_probability = mean(covered))
```

```
##   coverage_probability
## 1                0.929
```

simulation part 4

```
set.seed(437)
# need to do the simulation now!
sim_data4 = matrix(rexp(1000*10), nrow = 1000, ncol = 10)
```

```r
pop_mean4 <- 1
ci_t4 <- apply(sim_data4, 1, function(x) t.test(x)$conf.int)
ci_t_df4 <- as.data.frame(t(ci_t4))
names(ci_t_df4) <- c("lower", "upper")
ci_t_df4 %>% mutate(covered = lower <= pop_mean4 & upper >= pop_mean4) %>%
  summarize(coverage_probability = mean(covered))
```

```
##   coverage_probability
## 1                0.903
```

```r
# You should be able to just run this code chunk without any fixes
ci_perc4 <- apply(sim_data4, 1, bootstrap_ci, method = "percentile", B = 1000, summary_fn = mean, na.rm
# don't need any ... arguments here, but illustrating the idea of the ...
# notice that bootstrap_ci has default seed = 100 and C = 0.95 arguments
ci_perc_df4 <- as.data.frame(t(ci_perc4))
names(ci_perc_df4) <- c("lower", "upper")
ci_perc_df4 %>% mutate(covered = lower <= pop_mean4 & upper >= pop_mean4) %>%
  summarize(coverage_probability = mean(covered))
```

```
##   coverage_probability
## 1                0.857
```

```r
ci_basic4 <- apply(sim_data4, 1, bootstrap_ci, method = "basic", B = 1000, summary_fn = mean, na.rm = TR
# don't need any ... arguments here, but illustrating the idea of the ...
# notice that bootstrap_ci has default seed = 100 and C = 0.95 arguments
ci_basic_df4 <- as.data.frame(t(ci_basic4))
names(ci_basic_df4) <- c("lower", "upper")
ci_basic_df4 %>% mutate(covered = lower <= pop_mean4 & upper >= pop_mean4) %>%
  summarize(coverage_probability = mean(covered))
```

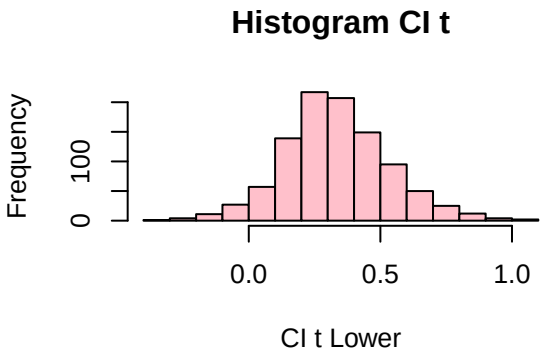
```
##   coverage_probability
## 1                0.845
```

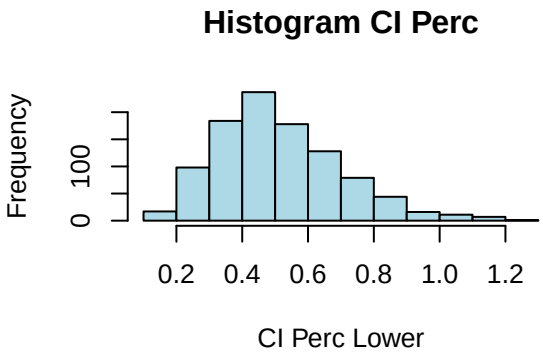
```r
ci_normal4 <- apply(sim_data4, 1, bootstrap_ci, method = "normal", B = 1000, summary_fn = mean, na.rm =
# don't need any ... arguments here, but illustrating the idea of the ...
# notice that bootstrap_ci has default seed = 100 and C = 0.95 arguments
ci_normal_df4 <- as.data.frame(t(ci_normal4))
names(ci_normal_df4) <- c("lower", "upper")
ci_normal_df4 %>% mutate(covered = lower <= pop_mean4 & upper >= pop_mean4) %>%
  summarize(coverage_probability = mean(covered))
```

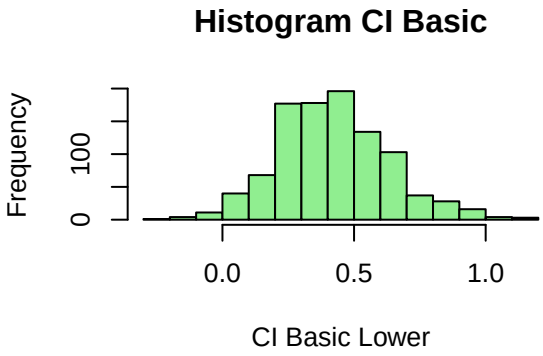
```
##   coverage_probability
## 1                0.856
```

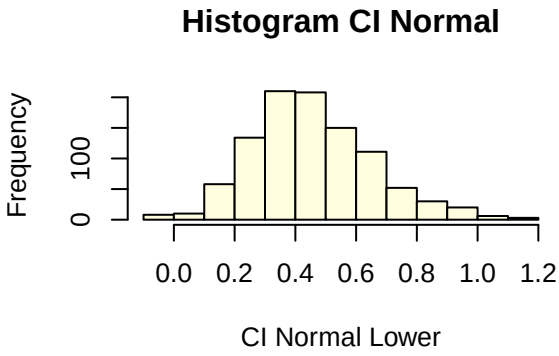
simulation part 5

```r
par(mfrow = c(2,2))

hist(ci_t_df4$lower, col = "pink", main = "Histogram CI t", xlab = "CI t Lower")

hist(ci_perc_df4$lower, col = "lightblue", main = "Histogram CI Perc", xlab = "CI Perc Lower")

hist(ci_basic_df4$lower, col = "lightgreen", main = "Histogram CI Basic", xlab = "CI Basic Lower")

hist(ci_normal_df4$lower, col = "lightyellow", main = "Histogram CI Normal", xlab = "CI Normal Lower")
```









- Are theory-based methods *guaranteed* to achieve the appropriate coverage? What about bootstrap-based methods?
- Which of the four methods appear to be range-preserving even in a "worst-case scenario"?
- When and why would a bootstrap method be useful to obtain a confidence interval even if it doesn't achieve the appropriate coverage?

summary: Both types of methods are not *guaranteed* to achieve the appropriate coverage. In terms of theory-based methods, the confidence intervals were closer to the 95% confidence interval with simulation 3 being 0.937 and simulation 4 being 0.903. Boostrap-based methods are guaranteed to achieve the appropriate coverage when we are looking at larger samples, but are less than 0.9 when we are looking at smaller samples.

Based on the histograms produced, I noticed that only the percentile confidence interval appears to be range-preserving. The other three histograms start before 0, but only percentile CI starts at 1.

A bootstrap method would be useful to obtain a confidence interval even if it doesn't achieve the appropriate coverage because the bootstrap can be used to estimate standard errors of the coefficients. Bootstrap methods can be applied to a wide range of statistical learning methods because there is a standard error formula, unlike t-test where we do not know how to obtaain the standard error.

Applied Problems

Applied Problem 1 (Code: 4 pts; Explanation: 2 pts)

Using the `dplyr` package, subset the `mpg` dataset from the `ggplot2` package to include only the cars from 2008 that are minivans, pickups, or SUVs (`%in%` is a useful replacement for `==` when trying to match to more than one possibility). Using this new dataset, determine which of the following statements is/are true, using an $\alpha = 0.10$ significance level/family-wise error rate or a $q = 0.10$ false discovery rate:

```
library(dplyr)
library(ggplot2)
library(ISLR2)
```

1. There is a significant difference in highway gas mileage between minivans and SUVs.
2. There is a significant difference in highway gas mileage between pickups and SUVs.
3. There is a significant difference in highway gas mileage between minivans and pickups.

Use the following methods.

```
new.mpg = data.frame(mpg %>% filter(year == "2008") %>% filter(class == c("minivan","suv","pickup")))

new.mpg1 = data.frame(mpg %>% filter(year == "2008") %>% filter(class == c("minivan","suv")))
```

```
## Warning in class == c("minivan", "suv"): longer object length is not a multiple
## of shorter object length
```

```
new.mpg2 = data.frame(mpg %>% filter(year == "2008") %>% filter(class == c("pickup","suv")))
```

```
## Warning in class == c("pickup", "suv"): longer object length is not a multiple
## of shorter object length
```

```
new.mpg3 = data.frame(mpg %>% filter(year == "2008") %>% filter(class == c("minivan","pickup")))
```

```
## Warning in class == c("minivan", "pickup"): longer object length is not a
## multiple of shorter object length
```

(a) Three two-sample t-tests with no adjustments for multiple testing. Store all three p-values in a single vector so that you can use the `p.adjust` function in later parts.

```
p1 <- t.test(hwy ~ class, data = new.mpg1,
             subset = which(new.mpg1$class %in% c("minivan","suv")))$p.value
p2 <- t.test(hwy ~ class, data = new.mpg2,
             subset = which(new.mpg2$class %in% c("pickup","suv")))$p.value
p3 <- t.test(hwy ~ class, data = new.mpg3,
             subset = which(new.mpg3$class %in% c("minivan","pickup")))$p.value
pvalue <- c(p1, p2, p3)
pvalue
```

```
## [1] 0.0006984001 0.0547861099 0.0008084105
```

true, true, true

(b) Three two-sample t-tests followed by Bonferroni's method.

```
B.values <- p.adjust(pvalue, method = "bonferroni")
B.values
```

```
## [1] 0.002095200 0.164358330 0.002425231
```

true, true, true

(c) Three two-sample t-tests followed by Holm's step-down method.

```
H.values <- p.adjust(pvalue, method = "holm")
H.values
```

```
## [1] 0.00209520 0.05478611 0.00209520
```

false, false, true

(d) A one-way ANOVA followed by Tukey's method.

```
anova1 = aov(formula = hwy ~ class, data = new.mpg1)
anova2 = aov(formula = hwy ~ class, data = new.mpg2)
anova3 = aov(formula = hwy ~ class, data = new.mpg3)
```

```
tukey1 = TukeyHSD(anova1)
tukey2 = TukeyHSD(anova2)
tukey3 = TukeyHSD(anova3)

T.values <- c(tukey1, tukey2, tukey3)
T.values
```

```
## $class
##              diff       lwr        upr      p adj
## suv-minivan -4.9 -9.363848 -0.4361524 0.03354218
##
## $class
##             diff       lwr      upr      p adj
## suv-pickup 2.85 0.1324145 5.567585 0.04070244
##
## $class
##                 diff       lwr       upr       p adj
## pickup-minivan -5.5 -8.860495 -2.139505 0.004902386
```

true, true, true

(e) Three two-sample t-tests followed by the Benjamini-Hochberg (BH) method.

```
BH.values <- p.adjust(pvalue, method = "BH")
BH.values
```

```
## [1] 0.001212616 0.054786110 0.001212616
```

true, true, true

Compare and contrast your results. For each part, there is a greater significant difference in highway gas mileage between pickup trucks and suvs. Part c and part e showed that the significant difference between minivans and suvs are the same as the significant difference between minivans and pickup trucks.

Applied Problem 2 (Code: 1 pt; Explanation: 1 pt)

Use a one-way ANOVA followed by Scheffe's method (`ScheffeTest` in the DescTools package) to determine whether the following statement is true at the $\alpha = 0.10$ significance level:

There is a significant difference in highway gas mileage between pickups and non-pickups (SUVs and minivans).

```
library(DescTools)
```

```
##
## Attaching package: 'DescTools'
```

```
## The following object is masked from 'package:car':
##
##     Recode
```

```
a.test2 = aov(formula = hwy ~ class, data = new.mpg2)
a.test3 = aov(formula = hwy ~ class, data = new.mpg3)

scheffe2 = ScheffeTest(a.test2)
scheffe3 = ScheffeTest(a.test3)

scheffe2
```

```
##
```

```
##   Posthoc multiple comparisons of means: Scheffe Test
##     95% family-wise confidence level
##
## $class
##            diff    lwr.ci   upr.ci   pval
## suv-pickup 2.85 0.1324143 5.567586 0.0407 *
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
scheffe3
```

```
##
##   Posthoc multiple comparisons of means: Scheffe Test
##     95% family-wise confidence level
##
## $class
##                 diff    lwr.ci    upr.ci   pval
## pickup-minivan -5.5 -8.860495 -2.139505 0.0049 **
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The statement is true. The p-value when comparing suvs and pickup trucks is 0.0407 and the p-value when comparing pickup trucks and minivans is 0.0049.

Applied Problem 3 (Code: 5 pts; Explanation: 3 pts)

Textbook Exercise 5.4.9.

We will now consider the Boston housing data set, from the ISLR2 library. (a) Based on this data set, provide an estimate for the population mean of medv. Call this estimate $\hat{\mu}$.

```
attach(Boston)
mu = mean(Boston$medv)
print(mu)
```

```
## [1] 22.53281
```

(b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result. Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.

```
sd(Boston$medv)/sqrt(nrow(Boston))
```

```
## [1] 0.4088611
```

The standard error is almost 50%, which means that

(c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

```
boot.fn = function(data, index) {
  return(mean(data[index]))
}

BS = boot(Boston$medv, boot.fn, 1000)
print(BS)
```

```
##
```

```
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original       bias    std. error
## t1* 22.53281 -0.005806719   0.4219161
```

The answer in part (b) is 0.40886 which is slightly less than the standard error from part (c) of 0.41318.

(d) Based on your bootstrap estimate from (c), provide a 95 % confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv). Hint: You can approximate a 95 % confidence interval using the formula [$\hat{\mu}$ - 2SE($\hat{\mu}$), $\hat{\mu}$ + 2SE($\hat{\mu}$)].

```
conf.int = c(BS$t0 - (2 * 0.3994119), BS$t0 + (2  * 0.3994119))

t.test(Boston$medv)
```

```
##
##  One Sample t-test
##
## data:  Boston$medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

(e) Based on this data set, provide an estimate, $\hat{\mu}$med, for the median value of medv in the population.

```
mu_med = median(Boston$medv)
print(mu_med)
```

```
## [1] 21.2
```

(f) We now would like to estimate the standard error of $\hat{\mu}$med. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```
boot.median.fn = function(data, index) {
  return(median(data[index]))
}

BS.median = boot(Boston$medv, boot.median.fn, 10000)
print(BS.median)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.median.fn, R = 10000)
##
```

```
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*     21.2 -0.01568   0.3792519
```

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston census tracts. Call this quantity ^μ0.1. (You can use the quantile() function.)

```
mu0.1 = quantile(Boston$medv, probs=c(.1))
print(mu0.1)
```

```
##   10%
## 12.75
```

(h) Use the bootstrap to estimate the standard error of ^μ0.1. Comment on your findings.

```
boot.quant.ten.fn = function(data, index) {
  return(quantile(data[index], probs=c(.1)))
}

boot.quant.ten = boot(Boston$medv, boot.quant.ten.fn, 10000)
print(boot.quant.ten)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.quant.ten.fn, R = 10000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.00237   0.5010943
```