Lab05

● Graded

Student

Sangwon Ji

Total Points

3 / 3 pts

Question 1

Exercise 1

1 / 1 pt

✔ **– 0 pts** Correct

Question 2

Exercise 2

1 / 1 pt

✔ **– 0 pts** Correct

Question 3

Exercise 3

1 / 1 pt

✔ **– 0 pts** Correct

No questions assigned to the following page.

Lab05

2023-02-27

Welcome to the Lab 5! In this lab will cover:

1) Illustration of how functions work in `R`
2) Learn about how to calculate confidence intervals (CIs) for t-tests.
3) Learn how to create confidence interval using bootstrap methods.

`apply`, `sapply`, and `function`

Previously, we introduced `replicate` to repeat a expression several times. Now what if we want to run a function several times, but change the argument every time we run it? For example, to calculate the square root of integers from 1 to 100. Though `for` loops would certainly work, there is a much faster and simpler way in `R`. Consider the two functions `apply` and `sapply`.

`apply` traverses row- or column-wise and applies a function to each row (or column). It is usually used on matrix and data frames. Depending on the function, it returns a vector, array, or list of values.

`sapply` traverses every element in an array or a list and applies a function on each element.

Let us look at the problem of calculating the square root of integers plus the integer value itself from 1 to 100. To implement with for loops:

```
sqroots = c()
for (i in 1:100) {
  sqroots = c(sqroots, i + sqrt(i))
}
```

And it is equivalent to the following with `sapply`:

```
sqroots <- sapply(1:100, function(x) {
  x + sqrt(x)
})
```

`sapply` usually results in a shorter run time and easier code implementation.

Now we create a matrix with 100 rows and 10 columns with each entries being a random number from $N(0, 1)$.

```
mat <- matrix(rnorm(1000), 100)
```

To obtain the maximum number of each row with `apply`:

```
row.max <- apply(mat, 1, max)
```

To obtain the sum of the third the the fifth element each row with `apply`:

```
row.sum35 <- apply(mat, 1, function(x) x[3] + x[5])
```

To obtain the maximum number of each column with `apply`:

```
col.max <- apply(mat, 2, max)
```

As you can see above, a function is different from an expression in that a function can take inputs and give an output specific to that input. Functions can also be given names, and saved in the environment. What a

Question assigned to the following page: 1

function executes can be wrapped in curly braces if there are multiple lines. In the function declaration, the parameters of the function must be named and listed. In the body of the function, those named parameters can be called on and operated upon. The value returned by a function does not have to be coded explicitly. They return the last output if `return()` is not invoked.

```
lab131 <- function(x, yy){
  x-yy
  yy-x}

lab131(5,3)
```

```
## [1] -2
```

```
lab131 <- function(x, yy){
  return(x-yy)
  yy-x}

lab131(5,3)
```

```
## [1] 2
```

```
lab131 <- function(x, yy){

  }

lab131(5,3)
```

```
## NULL
```

Exercise 1

Write a function called 'stat131' with parameters called `st` and `lb` (no quotes) that adds the square root of `st` to `lb` and returns the resulting sum.

```
# Insert answer here
stat131 <- function(st, lb) {
  sqrt (st) + lb
}
```

Question assigned to the following page: 1

In the following three exercises, we will look at a comic book data created by FiveThirtyEight for their story *Comic Books Are Still Made By Men, For Men And About Men*, where they claimed that Comic books vastly under-represent women. Specifically, they said that characters who are women have lower appearance counts after analyzing the Marvel and DC dataset. Well, is that true? Let's apply our testing methods to their analysis!

```
# Read in data.
marvel <- read.csv("marvel-wikia-data.csv")
```

The data `marvel-wikia-data.csv` comes from Marvel Wikia. It has the following variables:

- `page_id`: The unique identifier for that characters page within the wikia
- `name`: The name of the character
- `urlslug`: The unique url within the wikia that takes you to the character
- `ID`: The identity status of the character (Secret Identity, Public identity, [on marvel only: No Dual Identity])
- `ALIGN`: If the character is Good, Bad or Neutral
- `EYE`: Eye color of the character
- `HAIR`: Hair color of the character
- `SEX`: Sex of the character (e.g. Male, Female, etc.)
- `GSM`: If the character is a gender or sexual minority (e.g. Homosexual characters, bisexual characters)
- `ALIVE`: If the character is alive or deceased
- `APPEARANCES`: The number of appearances of the character in comic books (as of Sep. 2, 2014. Number will become increasingly out of date as time goes on.)
- `FIRST APPEARANCE`: The month and year of the character's first appearance in a comic book, if available
- `YEAR`: The year of the character's first appearance in a comic book, if available

We will focus on two columns `SEX` and `APPEARANCES`. To make thing simpler, we created two vectors `female.logappearances` and `male.logappearances` for you, which contains the log appearances counts for female and male characters separately. You will use these two vectors to do hypothesis testing in the rest of the lab.

```
female.logappearances <- log(marvel$APPEARANCES[marvel$SEX == "Female Characters"])
male.logappearances <- log(marvel$APPEARANCES[marvel$SEX == "Male Characters"])
```

Questions assigned to the following page: 2 and 3

T-test confidence intervals

Exercise 2.

(a) Get the confidence interval for the means of the female and male log appearance using the `t.test()` function.

```
# Insert your code here for calculating the CI, and save the CIs as
# `log.female.ci` and `log.male.ci`
log.female.ci <- t.test(female.logappearances, conf.level = 0.95)$conf.int
log.male.ci <- t.test(male.logappearances, conf.level =0.95)$conf.int
```

You may noticed that when you print the confidence interval, there is an additional line named `attr(,"conf.level")` with value 0.95. In R, all objects can have arbitrary additional attributes used to store metadata about the object. Attributes can be accessed using `attributes()` or `attr()`. For example, to get all the attributes of `log.male.ci`, run `attributes(log.male.ci)`. And to get the `conf.level` attribute of `log.male.ci`, run `attr(log.male.ci, "conf.level")`. Attributes do not influence the fact that `log.male.ci` is a two-dimensional vector.

(b) Get the confidence interval for the difference in the log appearance count for female and male using the `t.test()` function.

```
# Insert your code here for calculating the CI, and save the CI as
# `log.diff.ci`
log.diff.ci <- t.test(female.logappearances,male.logappearances, var.equal = TRUE) $conf.int
log.diff.ci
```

```
## [1] 0.2624728 0.3640706
## attr(,"conf.level")
## [1] 0.95
```

(c) Based on your calculation, which of the following statements do you support?

i. There is no significant difference in appearance counts for male and female in Marvel comics.
ii. The female character appearances count is significantly larger than the male in Marvel comics.
iii. The male character appearances count is significantly larger than the female in Marvel comics.

```
# Uncomment the line of your answer for this question:
# Ex1c.answer <- "i."
Ex1c.answer <- "ii."
# Ex1c.answer <- "iii."

Ex1c.answer
```

```
## [1] "ii."
```

Bootstrap confidence intervals

Exercise 3.

In this exercise, you will use the bootstrap to get the confidence interval for the difference in means between female and male log appearance count.

(a) Calculate the observed difference.

```
# Insert your code here for calculating the observed difference
# `obs.d`
obs.d <- mean(female.logappearances, na.rm=TRUE) - mean(male.logappearances, na.rm=TRUE)
obs.d
```

Question assigned to the following page: 3

```
## [1] 0.3132717
```

(b) Complete the following function to calculate the bootstrapped statistic. The function only needs to calculate one statistic; you will replicate it later.

```
# Complete the function for calculating the bootstrapped difference in means
bootOnce <- function() {
  boot.male <- mean(sample(male.logappearances, 1000),na.rm=TRUE)
  boot.female <- mean(sample(female.logappearances, 1000),na.rm=TRUE)
  diff <- boot.female - boot.male
  return(diff)
}
```

You can test whether your code works or not by running the following chunk:

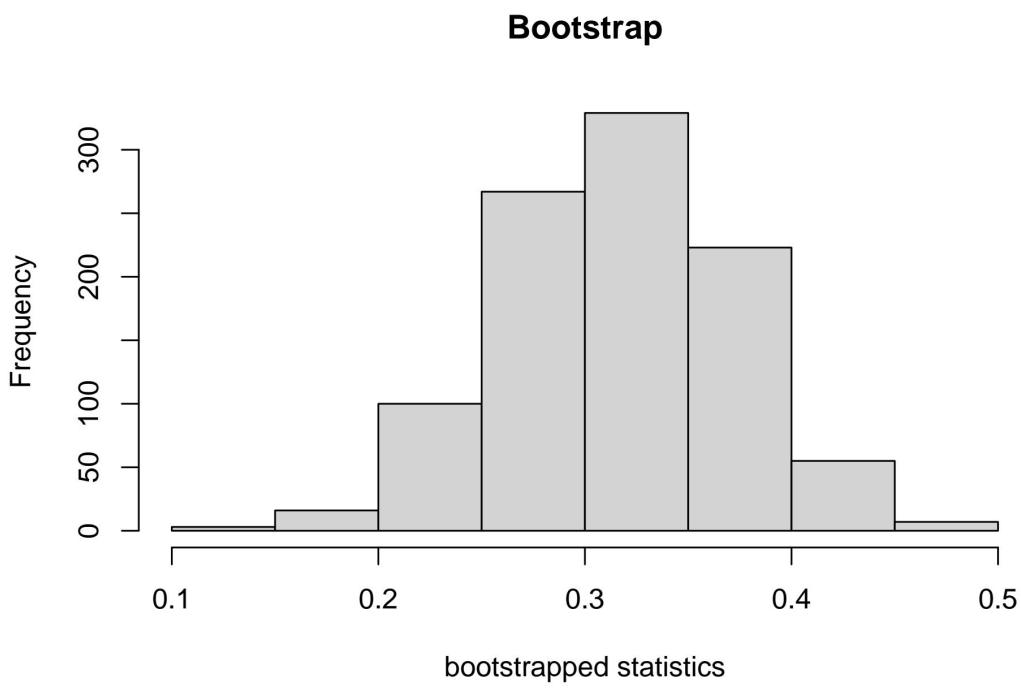
```
test <- bootOnce()
test  # This should give you a numeric value, which is a bootstrapped statistic
```

(c) Replicate the function `bootOnce` 1000 times to get 1000 bootstrapped statistics.

```
# Insert your code here and save the bootstrapped differences as
# `boot`
boot <- replicate(1000, bootOnce())
```

(d) Plot the histogram of your bootstrapped statistics.

```
# Insert your code for histogram here
hist(boot, xlab = "bootstrapped statistics", main = "Bootstrap")
```



(e) Calculate the bootstrapped CI at the 0.95 confidence level.___Hint:___ use the `quantile()` function to

Question assigned to the following page: 3

find the bounds; use lower bound = $(1 - 0.95)/2$th quantile, and upper bound = $1 - (1 - 0.95)/2$th quantile.

```
# Insert your code and save your bootstrapped CI as
# `boot.ci`
boot.ci <- quantile(boot, probs = c(0.025, 0.975))
boot.ci
```

```
##      2.5%     97.5%
## 0.2074532 0.4185848
```

(f) Based on your calculation, which of the following statements is supported?

i. I will fail to reject the hypothesis that there is no significant difference in appearance counts for male and female.

ii. I will reject the hypothesis that there is no significant difference in appearance counts for male and female.

```
# Uncomment the line of your answer for this question:
# Ex2f.answer <- "i."
Ex2f.answer <- "ii."
Ex2f.answer
```

```
## [1] "ii."
```