Homework 3 [60 points] / Updates in magenta

Questions related to the homework can be posted on the Piazza forum site, but do not post answers to homework problems. We will try to respond promptly to posted questions. However, please do not send post questions after 9pm the day before the assignment is due.

You may discuss the homework problems and computing issues with other students in the class. However, you must write up your homework solution on your own. In particular, do not share your code or homework files with other students.

This homework will build upon the material presented in the labs and lectures, and will be focused on confidence intervals. `R` will be used extensively throughout the homework. We suggest you start early for this homework, especially if you don't have any previous experience with a programming language. If there are any difficulties starting the homework, please contact the TA immediately.

The homework is split into two parts:

Part 1: Confidence intervals in Applied Statistics

Part 2: Confidence intervals in Theoretical Statistics

For Part One, text in blue denotes what needs to be done.

Files to submit on CMS

Do check to see that you have submitted all these files. Each part will be graded as a whole. The writeups should either be TeXed up or done using R Markdown.

1. Part One [30 points]
    - `find_CI.r`
    - `writeup_one.html` or `writeup_one.pdf`
2. Part Two [30 points]
    - `Score_interval.r`
    - `writeup_two.html` or `writeup_two.pdf`

Part One: Confidence Intervals For Set Similarity Estimation (Applied Statistics)

In introductory statistics courses, the idea of a confidence interval is usually introduced with *sampling distributions*.

For example, suppose we are interested in p (our *parameter*), which is the proportion of students who wear glasses at Cornell.

Suppose you enter Willard Straight Hall, where you pick the first five students at the free popcorn stand, four of whom are wearing glasses. Then you might say: With a sample of five, my *estimate* (also called the *statistic*) is given by $\hat{p} := \frac{4}{5}$, which is the proportion of students in this sample who wear glasses.

Now, you walk downstairs to the Bear's Den where Casino Night is going on and a sizable majority of Cornell students are ~~gamb~~ donating money to charitable causes. There are 400 students there, and you spot 220 students wearing glasses. You might also then say: With a sample of 400, my *estimate* of $\hat{p}$, the proportion of students who wear glasses is $\frac{11}{20}$.

Both these estimates $\hat{p}$ are *point estimates*, but supposing you could only pick *one* of them, which would you pick to avoid making a spectacle of yourself?

One could argue thus: *In the second sample, we looked at more students. It's more probable that p is closer to $\frac{11}{20}$, rather than closer to $\frac{4}{5}$. Can we try to quantify this "closeness"?*

The idea of a confidence interval is then to create an interval $(a, b),\ a < b$ centered around $\hat{p}$ which tries to capture the true parameter p. More precisely, the intervals[1] (a, b) are usually of this form:

$$(a, b) := (\text{Statistic} - \text{Margin of Error}\ ,\ \text{Statistic} + \text{Margin of Error}) \tag{0.1}$$

and in the above case, you might have written something like:

$$(a, b) := \left(\hat{p} - z\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \hat{p} + z\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right)$$

These terms can be found in an introductory statistics textbook / lecture notes / etc, so do read up on how you would compute the respective margin of errors if you are unfamiliar with this material.

[1] Convention: We use $a < b$ for our intervals. Take a look at Zhenrui Liao's answer at `https://www.quora.com/Is-w20-an-acceptable-way-of-writing-20w-in-mathematics` for why we will use (a, b) here.

In this part, we will look at the coverage of confidence intervals for set similarity problems. By doing this part, we hope to show that:

1. Computing a confidence interval for "complicated" problems is simply getting an interval of the form Equation 0.1
2. We can usually simplify (but not always) a hard problem to one we already know how to solve
3. There are other ways to view estimates apart from the traditional "getting out and collecting samples"
4. Theory and practice can vary by a lot

We will also be using real data where we know the true value of our *parameters*, so we can see how well our confidence intervals work in practice[2].

Probability and Venn Diagrams (The Theory)

Suppose our universe consists of the following elements: $\Omega := \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, and we have $A := \{3, 6, 7, 8\}$ and $B := \{1, 2, 5, 6, 8\}$. Pictorially, we have:

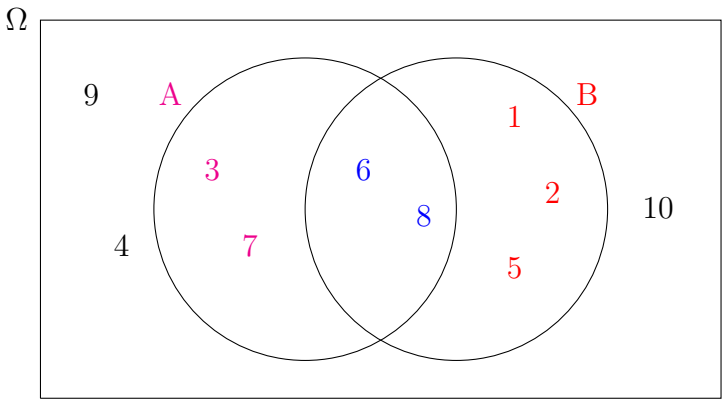

Figure 1: Original Universe

We denote $|A|$ to represent the *cardinality* of (the set) A, i.e. the number of elements in A. So for example $|\Omega| = 10$, $|A| = 4$.

We define the *set resemblance* to be $R(A, B)$, given by $\frac{|A \cap B|}{|A \cup B|}$. Think of this as "intersection" over the "union" of the sets. In this case, we have $R(A, B) = \frac{2}{7}$.

[2]Constructing a 95% CI theoretically is one thing, running simulations with artificial data to "see" a 95% CI is another, but using real data trumps all.

We introduce the idea of a random permutation[3] π.

We now consider a permutation on our universe, Ω. We call this permutation π_1

set.seed is the recommended way to specify seeds.

```
set.seed(1463)
perm = sample(1:10, replace = FALSE)
#  [1]  3  2 10  6  4  1  8  9  7  5
```

seed – A number.
Use the set.seed function when running simulations to ensure all results, figures, etc are reproducible

Note that we have 10! equally likely permutations. Under this permutation π_1, we see that 1 is mapped to 3, 2 is mapped to itself, 3 is mapped to 10, and so on. We can write:

$$\pi_1(1) = 3$$
$$\pi_1(5) = 4$$

With some abuse of notation[4] we can write:

$$\pi_1(\{3, 6, 7, 8\}) = \{10, 1, 8, 9\}$$
$$\pi_1(\{1, 2, 5, 6, 8\}) = \{3, 2, 4, 1, 9\}$$

and under π_1 in the universe, we would get:

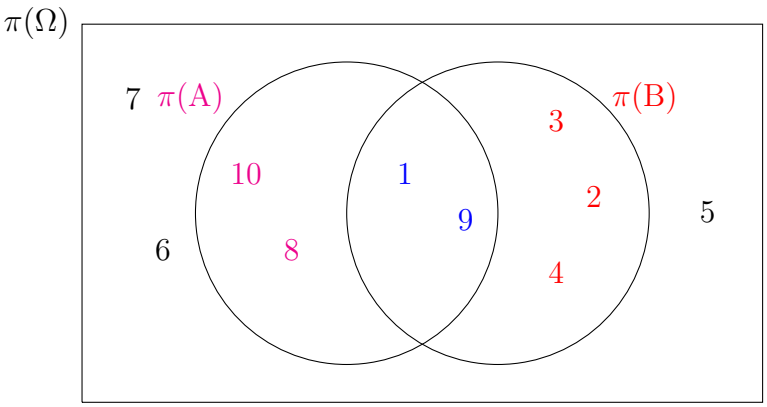

Figure 2: Universe under π_1

The question: What is the probability that the maximum number[5] in both sets $\pi(A)$ and $\pi(B)$ are equal, under a random permutation of Ω?

[3] Permutations are useful. In statistics, you'll think of bootstrapping. In industry, you'll use permutations in lots of different applications.

[4] Sets are meant to be unordered, i.e. $\{1, 2, 3, 4\} = \{2, 3, 1, 4\}$.

[5] Under π_1, the maximum number in A is 10, and the maximum number in B is 9. Sadly, they are not equal.

Similarly, what is the probability that the minimum number in both sets are equal, under a random permutation of Ω?

Suppose now we run this following algorithm (pseudo code given below):

```
Let prop := 0
Let A, B be any two sets with elements in Omega
for i in 1:K
  Construct permutation P_i over Omega
  Find "maximum number" under P_i in set A
  Find "maximum number" under P_i in set B
  If both "maximum numbers" are equal
    prop = prop + 1
  endif
endif
prop = prop/K
```

What should `prop` tend to in the long run as K increases? Use a theoretical result discussed in class, and the fact that the permutations are independent.

Since in our algorithm, we "only" increased `prop` in the loop when both "maximum numbers" are equal, and ended up dividing by N, we can see this as a Bernoulli distribution where success is when we see the event {both maximum numbers are equal}.

In the literature, a confidence interval for proportions based on the normal approximation to the binomial is usually given by:

$$\left(\hat{p} - z\sqrt{\frac{\hat{p}\hat{q}}{n}}, \hat{p} + z\sqrt{\frac{\hat{p}\hat{q}}{n}}\right) \tag{0.2}$$

Using the **results** of this algorithm, write down a similar expression for a 95% confidence interval for the *set resemblance, carefully defining all the terms you use.*

Does the size of Ω make a difference in deriving your answer in theory? How might the size of Ω affect computing your answer in practice?

Adam proposes a way to reduce the number of permutations to get an accurate estimation of the *resemblance*. He says, "Since we can equally look and see if the event {maximum number in both sets are the same}, or the event {minimum number in both sets are the same} under permutations, then let's just increase `prop` by 1 each time any of these events {both maximum numbers are equal}, { both minimum numbers are equal} occur."

Do you agree with Adam? Explain your answer - a theoretical proof suffices.

haven't done

Confidence Interval Coverage (The Computing)

We have covered an example (on webpages) during Lab. We now look at *word pairs* in documents, where the set up is a bit different.

On CMS, we have uploaded a `words.zip` file. This file consists of 2705 files (words[6]) from the MSN Word Crawl Data. We explain what each file consists of, by taking a look at the word `STATISTICS`.

We load in the file (assume it is in your current directory).

```
curr_word = read.table("STATISTICS")

dim(curr_word)
# [1] 1276    2

head(curr_word)
  V1  V2
1  2   1
2  1  11
3  1  60
4  1  67
5  1 127
6  2 236
```

The first column `V1` is the number of times the *word* (`statistics`) appears in *document number* in column `V2`. So reading the first six lines, we see that the word `statistics` appears twice in document 1, appears once in documents 11, 60, 67, 127, and appears twice in document 236.

The dimensions of the file is (1276,2) indicates that the word `statistics` only appears in 1276 documents. If a document number does not appear in column `V2`, then the word does not appear in that document.

Overall, there are $2^{16} = 65536$ documents.

In this part, we are interested in looking at *word pairs*. We explain what this means. Suppose we are interested[7] in two words: `monday` and `sucks`. Both of them may appear in different

[6] While we will not use most of these words, feel free to play about with them.

[7] Sadly, we do not have "*Donald*" or "*Trump*" as words here ...

documents, and we are interested in the parameter:

$$p := \frac{\#\text{ of documents that } \texttt{monday} \text{ and } \texttt{sucks} \text{ both appear in}}{\#\text{ of documents that either } \texttt{monday} \text{ or } \texttt{sucks} \text{ appear in}}$$

While we can easily compute p for a word pair from the data we have, we instead want to get an estimate for p by using the algorithm described in the previous section.

First, carefully define the following terms and/or explain what they would correspond to in our *word pairs* context:

- Ω corresponds to the set of all documents, from document # 1 to document # 65536.
- Sets $A, B, \ldots$ which correspond to the words we have
- For any two sets A, B:
  1. $A \cup B$
  2. $A \cap B$
  3. $A \setminus B$
  4. $\Omega \setminus \{A \cup B\}$

We have defined Ω and the sets $A, B, \ldots$ to start you off - but feel free to improve on our definition.

Now, write a function `findCI` that takes in *four* inputs, namely:

- `word1`, a string corresponding to a word
- `word2`, a string corresponding to another word
- `numP`, the number of permutations
- `alpha`, to create a $1 - \alpha$ confidence interval similar to Equation 0.2

The function should compute the true p, but also generate `numP` permutations to get an estimate of $\hat{p}$, and construct a $(1 - \alpha)$ confidence interval (a, b). If `word1` is equal to `word2`, the function tells the user this (and stops).

Concretely, the output of this function should be a *list*, containing four items:

- `res` - the actual p between two word pairs

- `phat` - the estimated $\hat{p}$ after `numP` permutations as in Equation 0.2
- `l_int` - a in our confidence interval (a, b)
- `r_int` - b in our confidence interval (a, b)

Use this template

```
findCI<-function(word1, word2, numP, alpha){
  ## Your code here
}
```

Some example inputs and outputs:

```
myresults = findCI("five", "guys", 1000, 0.05) # function should run

myresults$res     # output should show actual p
# [1] 0.06797497
myresults$phat    # output should show phat
# [1] 0.07
myresults$l_int   # output should show l_int
# [1] 0.05418611
myresults$r_int   # output should show r_int
# [1] 0.08581389

# One could imagine a function going
gradeStud<-function("wbn8", testfunction)
# or even a loop that goes through all names in a list
# This is also why we emphasize no extraneous code in your r. files
```

Submit `findCI.r` to CMS.

We'll look at the following word pairs.

- *happy* and *dave*
- *music* and *night*
- *united* and *states*
- *hong* and *kong*

We're interested in whether our confidence intervals actually provide coverage for our resemblance. In other words, if for a fixed number of permutations, we repeatedly generate confidence intervals, will about 95% of them capture our true resemblance?

Setting the number of permutations to be `100`, generate 10,000 confidence intervals for each word pair, and report the proportion of times that the confidence intervals capture the true resemblance. You are encouraged to modify your function `findCI.r` to do so efficiently, but do not submit this modified version on CMS. Warning: Unoptimized code can take up to three hours to run (or worse), optimized code would take about 10 minutes. This timing is for all parts of this question, and not just for one pair / one permutation.

take a subset of permutation 10 instead of 65534

The Interpretation

Usually, when we have constructed a confidence interval, we would say something of the following form: *I am 95% confident that the true resemblance lies between 0.2701104 and 0.2876896.* However, is this really the case?

By only looking at your results (four word pairs), explain why we might not get 95% coverage for our confidence intervals[8].

One solution would be to increase the number of permutations such that for all word pairs (i, j), we should get 95% coverage by our confidence intervals.

Let us re-look at the webpage / permutation set up as shown in Lab, and consider our solution. Suppose the number of words in the universe remains fixed at $|\Omega| = D$, but a new webpage springs up every second.

In the worst case scenario, would we need to generate more permutations every other second to ensure that all our confidence intervals for the resemblance of (i, j) webpage pairs have 95% coverage? Explain your answer.

This isn't the case in real life though. So another solution would be to generate a fixed number of permutations N, and stop, even though we might get "95% confidence intervals which don't have 95% coverage" - thus our estimate of the resemblance may be way off. Would it be prudent to do so in practice? Consider the following points:

1. Why would someone be interested in the resemblance?
2. Are *all* pairwise resemblances equally important?

If you are working in an area where finding the resemblance / similarity between two "objects" (genes, webpages, documents, books, etc) is of importance, you may adapt this question to suit your interests.

[8] Unhappy Dave...

Part Two: Construction Of Confidence Intervals (Theoretical Statistics)

Let $X_1, X_2, \ldots, X_n$ be a random sample from a Poisson distribution with mean μ.

1. Write down the likelihood function for μ and show that $\sum_{i=1}^{n} X_i$ is a sufficient statistic. (Hint: What is the distribution of the sum? Derive the conditional distribution of the sample given the sum.)
2. Show that the sample mean, $\bar{X}_n$, is the maximum likelihood estimator for μ.
3. Write down the score statistic for testing the hypothesis, $H_0 : \mu = \mu_0$. Explain why this statistic has a chi-squared distribution (approximately) if the hypothesis is true.
4. What are the corresponding Wald and likelihood ratio (LR) statistics?
5. Use a Taylor series argument to show that the LR statistic is asymptotically (large n) equivalent to the score statistic.
6. Show that the endpoints of a confidence interval obtained by inverting the score test are the solutions of a quadratic equation. Show that the endpoints are always positive unless the all the data values are zero.
7. Create a function to determine the endpoints of a confidence interval obtained by inverting the score test. This function should be of the form:

```
Score_interval(data,alpha)
{
  R code
  return(lower,upper)
}
```

   where "data" is the vector containing the sample values, and "alpha" is the desired confidence level.
8. Write R code to simulate N=10,000 confidence intervals for a given sample size and alpha level and known value of μ
9. Run your code with $n = 10$, $\alpha = 0.9$ and $\mu = 1.0$.
10. Determine the proportion of times that μ is below the lower confidence limit, and above the upper confidence limit in your simulations.
11. Repeat the simulation for μ values equal to 5 and 10, and report your results in a table, and plot the proportions in a graph with the value of μ on the x-axis, using different colors and symbols for the lower and upper values. Indicate (e.g. using `abline(v=)`) the target probability on the plot.

12. What value of N is required to estimate the proportion of times μ is below the lower confidence limit to within a tolerance of 0.001 with 95% confidence? Explain your answer.