FISHER'S EXACT TEST

WILLIAM H. CHUANG, LEA GUMARU, AND KIBEOM KIM

Abstract. Enumerating r-by-c (both r and c are integers greater than one) contingency tables is a serious computational task even today. In this paper, our goal is to build up a thorough understanding of the fundamentals. Hence, we restrained ourselves to only focus on two-by two contingency tables, and developed our own combinatorical argument of Fisher's Exact Test from scratch, and reconstructed the mainstream conditional probability approach. Within this restriction:

- we developed our own combinatorical argument proof of hypergeometric distribution in Fisher's Exact Test. (Compared to the mainstream textbooks, this combinatorical approach is a more natural, and original way to obtain the main result of Fisher's exact test, and compare it to the conventional approach, i.e. binomial distribution, and conditional probability)
- we compared the results of Fisher's Exact and asymptotic method, to understand why we can't use asymptotic method for small sample sizes sampling
- we implemented our own algorithm (developed in Part II, applied to real data in Part III)
- in Part IV, we derived a more accurate formula by using Saddle Point Approximation for future use (When $\epsilon > 0$ is given, this ϵ depends on the computational power, then we have control over how large the possible errors can be brought in by choosing an $L > 0$)

Contents

Date: May 11, 2018.

1. Introduction

In Fisher's 1935 book, "The Design of Experiments[1]," he introduced his points of views of Statistics and the designs of experiments. He started with a story that on a summer afternoon in Cambridge in the 1920's, a group of his friends discussed about a claim made by a lady (Muriel Bristol) about her superpower on distinguishing whether for a given cup of tea, milk was poured first or last. Given the situation and conditions, we have our main problem:

how to design an experiment to test whether this lady's claim is true?

A quick answer for the above problem is Fisher's Exact Test. This test is originally reported in Fisher's book[1]. In order to have a deep concise understanding, we only addressed the necessary notions. We used the lady's tasting tea example to introduce Fisher's Exact Test by justifying and defining the keywords.

The experiment is the first original exposition of his idea of a *null hypothesis*, which in his words is "never proved or established, but is possibly disproved, in the course of experimentation."

In addition to our knowledge of R to calculate the p-value for Fishers Exact Test, based on Fisher's Exact Test methodology, we can modify an exact test to our needs.[1] We want to learn more insights of Fisher's

[1]Suppose we define our 2 by 2 contingency data in a 2 by 2 matrix, called matrix_small_counts, then we can obtain the result of Fisher's Exact Test in R by typing this R code: fisher.test(matrix_small_counts) in R studio.

Exact Test so that in our future career, if we need to confront a new scenario, we can know how to redesign a customized exact test that is based on the essential idea of Fisher's Exact Test.

2. Part I: Developing Fisher's Exact Test

There are two different methods to derive Fisher's Exact Test:

- We used Chu-Vandermonde formula (in **Method I**, see below) to form a combinatorical argument to derive hypergeometric distribution. ***(Without this understanding/insight, it's unlikely that one can see the connection between a 2 by 2 table and using hypergeometric distribution to compute the p-value. Also, it's counter-intuitive for one to realize that this procedure actually enumerates all possible tables in the configuration space under given margins[2].)*** This has been completed by one of the authors of this paper which has not yet been seen in literature.
- **Method II** (see below) is also reconstructed in this section and it's the conventional way in Fisher's book. However, since this approach is based on conditional probability, it adds one more layer. In this approach, there is no way to know what role each term in the formula (8) plays in the final result. For the completeness of this paper, we still include this approach as our *Method II*, and that's the motivation we developed our own proof.

2.1. Method I (Our own proof).

Let's take the **Lady Tasting Tea problem**[1] as our main example.

The lady chooses 4 cups from 8 cups and she needs to guess which cup has the tea poured first or the milk poured first.

Table 1. Lady Tasting Tea

	True order is tea first (T)	True order is milk first (M)	Total (margin)
Lady Says Tea first (t)	$a = 3$	$b = 1$	$a + b = 4$
Lady Says Milk first (m)	$c = 1$	$d = 3$	$c + d = 4$
Total (margin)	$a + c = 4$	$b + d = 4$	$n = 8$

[2] $(a + c), (b + d), (a + b), (b + c)$ and n in Table 1.

$a = \#(\text{guess tea first}|\text{tea first is true}) = \#(t, T)$
$b = \#(\text{guess tea first}|\text{milk first is true}) = \#(t, M)$
$c = \#(\text{guess milk first}|\text{tea first is true}) = \#(m, T)$
$d = \#(\text{guess milk first}|\text{milk first is true}) = \#(m, M)$

Let the margins (i.e., fix $a + b, c + d, a + c, b + d$) be fixed.

TABLE 2. **Observed Data: One of the permutation of** $(a, b, c, d) = (3, 1, 1, 3)$

what lady says	t	t	t	t	m	m	m	m
the true order	T	T	T	M	T	M	M	M

Null Hypothesis

Let p_1 be the probability the lady says tea first and in reality the true order is also tea first.
Let p_2 be the probability the lady says milk first **but** in reality the true order is tea first.
Notation: let $p_1 = p_2 = p$, and this becomes our null hypothesis, H_0.

Why can we do permutations?

Answer: Under the null hypothesis distribution we count the number of tables which can be formed with the same marginal counts, but have a more extreme statistic.

TABLE 3. One possible scenario (permutation) of $(a, b, c, d) = (2, 2, 2, 2)$

what lady says	m	m	t	t	t	t	m	m
the true order	T	T	T	M	T	M	M	M

After the margins were fixed, we empty the cells of a, b, c and ,d. We focused on the column of the true order which was tea first. The question is: *How many combination of ways can we fill in the empty*

TABLE 4. One possible scenario (permutation) of $(a, b, c, d) = (1, 3, 3, 1)$

what lady says	t	m	m	t	m	t	m	t
the true order	T	T	T	M	T	M	M	M

TABLE 5. One possible scenario (permutation) of $(a, b, c, d) = (0, 4, 0, 4)$

what lady says	m	m	m	m	t	t	t	t
the true order	T	T	T	T	M	M	M	M

cells of $a, b, c,$ *and* d*?*

Imaging that each a, b, c, and d are fluctuating, we are sure that they meet the marginal constraints. We know that in our example, we have one more known condition for the both marginal constraints: $n = a + b + c + d$. We also know that there are two groups of samples in which each group has a certain amount of cups of tea. In one group, we have tea first (mark them with T), and it has $a + c$ cups of tea. In the other group we have $b + d$ cups of tea with milk first (mark them with M). We keep tracking of them with a hidden sticker (so that lady can't see it, but we won't forget which cup is tea first, and which is milk first). Under this setting, we know that there are n choose $a+c$ number of ways to make these eight cups of tea to be tea first. Likewise, there are n choose $b + d$ number of ways to make these eight cups of tea to be milk first.

On the other hand, suppose we have a friend that doesnt know which cup has the true result of having the tea or milk poured first. That means she is required to guess. After each guess is made, she labels the cup with her guess, e.g., guess tea first (t), or guess milk first (m). Before she makes these guesses, we don't know a, b, c, and d. But now we can actually break $(a+c)$ to be two integers, and it's similarly with $(b + d)$.

Now, if we have

$$a + b \geq a + c$$
$$c + d \geq a + c$$

then $b \geq c$ and $d \geq a$. Likewise, we have $b \leq c$, and $d \leq a$.

Suppose now the given case is $b \leq c$, and $d \leq a$. Then we can write down the following Chu-Vandermonde formula:

$$\binom{n}{a+c} = \sum_{k=0}^{a+c} \binom{a+b}{k}\binom{c+d}{a+c-k}$$

We can give a combinatorical argument that there are two ways to choose $a+c$ from n balls, or to assign $a+c$ guesses to the group of cups that are true with tea first (likewise, there are n choose $b+d$ ways to assign $b+d$ guesses to the group of cups that are true with milk first). The second way to do the same thing is we make k number of guesses from $a+b$ trials, and make another $(a+c)-k$ number of guesses from $c+d$ trials. Then the number of ways to do this is $\binom{a+b}{k}\binom{c+d}{a+c-k}$ In total there are $a+c$ branches, so we need to sum all of them (that is, k is running from 0 to $a+c$).

$$\binom{4}{0}\binom{4}{4}+\binom{4}{1}\binom{4}{3}+\binom{4}{2}\binom{4}{2}+\binom{4}{3}\binom{4}{1}+\binom{4}{4}\binom{4}{0} = \binom{8}{4} = 70$$

Each branch represents a certain scenario of the permutation that may be observed. Hence we can write them in Table 6. Let's denote S := success in guessing and F := failed in guessing. Since we already knew the whole configuration space and each branch of the possible outcome, we can calculate the probability of each possible outcome.

In general, we have

$$\text{Total} = \binom{n}{a+c} = \sum_{k=0}^{a+c} \binom{a+b}{k}\binom{c+d}{(a+c)-k} \tag{1}$$

It follows that we can derive the probability for each case in the above table as follows:

$$P(\text{success count } = 0) = \frac{\binom{4}{0}\binom{4}{4}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} \tag{2}$$

$$P(\text{success count } = 1) = \frac{\binom{4}{1}\binom{4}{3}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} \tag{3}$$

TABLE 6. Probability of Each Permutation.

Success count Only focus on a and c cells	Histogram (permutation of selection) For tea first is true column	$\#(permutations)$
0 $(a = 0, c = 4,\ k = 0)$	FFFF	$\binom{4}{0}\binom{4}{4} = 1 \times 1 = 1$
1 $(a = 1, c = 3,\ k = 1)$	FFFS, FFSF, FSFF, SFFF	$\binom{4}{1}\binom{4}{3} = 4 \times 4 = 16$
2 $(a = 2, c = 2,\ k = 2)$	FFSS, SFSF, SFFS, FSFS, SSFF, FSSF	$\binom{4}{2}\binom{4}{2} = 6 \times 6 = 36$
3 $(a = 1, c = 3,\ k = 1)$	FSSS, SFSS, SSFS, SSSF	$\binom{4}{3}\binom{4}{1} = 4 \times 4 = 16$
4 $(a = 4, c = 0,\ k = 0)$	SSSS	$\binom{4}{4}\binom{4}{0} = 1 \times 1 = 1$
	Total	$\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}$ $= \binom{8}{4} = 70$

$$P(\text{success count } = 2) = \frac{\binom{4}{2}\binom{4}{2}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} \tag{4}$$

$$P(\text{success count } = 3) = \frac{\binom{4}{3}\binom{4}{1}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} \tag{5}$$

$$P(\text{success count } = 4) = \frac{\binom{4}{4}\binom{4}{0}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} \tag{6}$$

It follows that the number of successes is distributed according to the **hypergeometric distribution**. Suppose we use the convention of significance level $\alpha = 0.05$, i.e., 5%.

Now, we have three cases, and the notion of p-values naturally emerges by considering these three cases:

Case 1: Success count = 4 (same as success count = 0)
The probability for the lady to correctly guess all 4 samples (choose from 8 cups of tea) is: P(all possible cases with 4 success)=

$$\frac{\binom{4}{0}\binom{4}{4}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} = \frac{1}{70} = 0.014 < 0.05$$

We can reject the null hypothesis (i.e., the hypothesis that the lady doesn't have the ability to distinguish can be rejected, if case 1 happens).

Case 2: Success count = 3 (same as success count = 1)
The probability for the lady to correctly guess all 4 samples (choose from 8 cups of tea) is: P(all possible cases with 4 success)=

$$\frac{\binom{4}{0}\binom{4}{4}+\binom{4}{1}\binom{4}{3}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} = \frac{1+16}{70} = 0.2429 > 0.05$$

We failed to reject the null hypothesis.

Case 3: Success count = 2
The probability for the lady to correctly guess all 4 samples (choose from 8 cups of tea) is: P(all possible cases with 4 success)=

$$\frac{\binom{4}{0}\binom{4}{4}+\binom{4}{1}\binom{4}{3}+\binom{4}{2}\binom{4}{2}}{\sum_{j=0}^{4}\binom{4}{j}\binom{8-4}{4-j}} = \frac{1+16+36}{70} = 0.6142 > 0.05 \tag{7}$$

We failed to reject the null hypothesis.

Summary I

Let X represent the "True order is Tea First" column in Table 1 and correspond to the marginal constraint: $a+c$ is fixed.
Let Y represent the "True order is Milk First" column in Table 1 and correspond to the marginal constraint: $b+d$ is fixed.
In general, for each specific observation, we have

$$P(X=a) = P(Y=d) = \frac{\binom{a+b}{a}\binom{c+d}{c}}{\binom{n}{a+c}} = \frac{\binom{d+c}{d}\binom{b+a}{b}}{\binom{n}{d+b}} \tag{8}$$
$$= \frac{(a+b)!\cdot(c+d)!\cdot(a+c)!\cdot(b+d)!}{a!\cdot b!\cdot c!\cdot d!\cdot n!}.$$

We can see that for two by two contingency tables, we have the following mirror symmetry: $a \leftrightarrow d$, $c \leftrightarrow b$, and $X \leftrightarrow Y$.

Notice that the probability in each of the above cases is calculated by using observed data and so far we only considered the one-tailed, i.e., we just derived the notion of p-value. Let's summarize the above argument in the following (more formal way to state the definition of p-value which also includes the definition of the two-tailed test).

Definition 1. ***The p-value*** *can be obtained in the following three cases: If X is a random variable,*

- $P(X \geq x|H)$*for right tail events*
- $P(X \leq x|H)$*for left tail events*
- $2\cdot\min\{P(X \geq x|H),\ P(X \leq x|H)\}$*for two tails events*

Summary II

In general, for computing p-value:
Suppose $a < b$ (if not, we can swap $a \leftrightarrow b$, $c \leftrightarrow d$, i.e., swap the two groups).

- **Case A:**
  If $\frac{a}{(a+c)} \geq 0.5$, then

$$p - \text{value} = P(X \geq a) = \sum_{k=a}^{a+c} \frac{\binom{a+b}{k}\binom{c+d}{a+c-k}}{\binom{n}{a+c}} \tag{9}$$

- **Case B:**
  If $\frac{a}{(a+c)} < 0.5$, then

$$p - \text{value} = P(X \leq a) = \sum_{k=0}^{a} \frac{\binom{a+b}{k}\binom{c+d}{a+c-k}}{\binom{n}{a+c}} \tag{10}$$

Let's summarize the meaning of p-value:

- The smaller the p-value, the less odds for the observed samples to occur, thus the premise (H_0 is true) is wrong.Therefore we reject the null hypothesis.
- The smaller p-value, it is less likely that one can make a mistake if the null hypothesis is abandoned.

2.2. **Method II: (The conventional approach).**

In this method we are going to use binomial distribution, and conditional probability to prove the same result.

Suppose $X = a =$ the number of true order is tea first and the lady also says (guesses) tea first (i.e., a correct guess).
Suppose $Y = c =$ the number of true order is tea first but the lady says (guesses) milk first (i.e., a wrong guess).
Again, $H_0 : p_1 = p_2 = p$.
Then under H_0, we have:

- $n_1 = (a + b)$, $n_2 = (c + d)$, $x = a$, $z = a + c$
- $X \sim BIN(n_1, p)$
- $Y \sim BIN(n_2, p)$
- $X + Y \sim BIN(n_1 + n_2, p)$

We can derive a hypergeometric pmf as follow:

Claim

$$P(X = x|X + Y = z) = \frac{\binom{n_1}{x}\binom{n_2}{z-x}}{\binom{n_1+n_2}{z}} \tag{11}$$

Proof. Now, we break this down to three pieces:

$$P(X = x) = \binom{n_1}{x} p^x (1-p)^{n_1-x} \tag{12}$$

$$P(Y = z - x) = \binom{n_2}{z-x} p^{z-x} (1-p)^{n_1+n_2-z} \tag{13}$$

$$P(X + Y = z) = \binom{n_1+n_2}{z} p^z (1-p)^{n_1+n_2-z} \tag{14}$$

On the other hand, we can rewrite 11 as follows:

$$\begin{aligned} P(X = x|X + Y = z) &= \frac{P(X = x \cap X + Y = z)}{P(X + Y = z)} \\ &= \frac{P(X = x \cap Y = z - x)}{P(X + Y = z)} = \frac{P(X = x)P(Y = z - x)}{P(X + Y = z)} \\ &= \frac{\binom{n_1}{x} p^x (1-p)^{n_1-x} \binom{n_2}{z-x} p^{z-x} (1-p)^{n_1+n_2-z}}{\binom{n_1+n_2}{z} p^z (1-p)^{n_1+n_2-z}}. \\ &= \frac{\binom{n_1}{x}\binom{n_2}{z-x}}{\binom{n_1+n_2}{z}} \end{aligned} \tag{15}$$

which is hypergeometric distribution, and this completes the proof. □

Therefore, according to the above derivation, what we have justified what we should do when the given data is a two by two contingency table is to use Fisher's Exact Test.

Question

What does it mean by "exact"?

Answer: In an exact test, we compute *p-value* to reject the null hypothesis. Next, we should give a formal definition.

Definition 2. *To call a test exact is to say that the test must allow an exact probability to be assigned to each of the possible outcomes.*

For instance, an exact test at significance level 1% will in the long run reject true null hypotheses exactly 1 % of the time.

Properties: (when should we use exact tests?) It is all about sample size. An exact test is especially used when sample sizes are small. Thus, the usage of exact tests is: it's used as an asymptotic test will not provide a good result.

Strictly speaking, in an exact test, all the assumptions of the distribution of the test statistics must be met and this opposes an approximate test because in an approximate test, we can increasing the sample size close as desired test statistics.

Question

Why is Fisher's test "exact"?

Answer: Fisher's test, we can safely say it is exact, because the sampling distribution (conditional on the marginals) is known exactly, hence no matter what size of the sample is given, the p-value is "exactly" computed.

3. Part II: Implementations of Algorithms

Our goal in Part II is to develop the best implementation of Fisher's Exact Test we can get within the time constraint of finishing this project, and with a large sample size, but without using Yates' approximation[5].

3.1. Problem: choose(n,k) in R doesn't work for large n.

Based on the above understanding, one should be able to implement this Fisher's Exact Test in R. More exactly, to implement a generalize formula of equation 7 to compute p-value. However, we confronted a computation problem.

Data source: CNN: exit polls–national president (viewed as tables)[4]. https://www.cnn.com/election/2016/results/exit-polls

We have 20 sets of data from 20 different questions, and in order to develop our own algorithm to derive a rather precise Fisher's Exact Test result, we use the following example (compare to Table 1, now the sample size is increased from 8 to 24558) as our starting point.

3.2. Our First Attempt.

Problem: without re-scaling approximation, and use choose(n,k) in R, we obtained NaN (not a number).

TABLE 7. A Division by Gender Behind 2016 President Exit Polls (24558 respondents)

	Doesn't Support Trump	Support Trump	Total (margin)
Men	$48\% := a$	$52\% := b$	100%
Women	$59\% := c$	$41\% := d$	100%
Total (margin)	$a + c = 107\%$	$b + d = 93\%$	$n = 200\%$

```
 #H_0: men and women are equally likely to "support Trump"
fisher_exact_p_value_fractional_input_no_approximation<-
    function(N, a, b, c, d){

  temp <- 0
  if (a>b)
  {
    temp<-b
    b<-a
    a<-temp
    temp<-c
    c<-d
    d<-temp
  }

  v<- matrix(ncol=3, nrow=3)
  colnames(v)<- c("Men","Women", "Column Total")
  rownames(v)<- c("Yes, Support Trump","No, Against Trump",
      "Row Total")

  A<-floor(N*a)
  B<-floor(N*b)
  C<-floor(N*c)
  D<-floor(N*d)

  #raw data
  v[1,1]<-floor(a*N)
  v[1,2]<-floor(b*N)
  v[2,1]<-floor(c*N)
  v[2,2]<-floor(d*N)
  v[1,3]<-floor((a+b)*N)
  v[2,3]<-floor((c+d)*N)
  v[3,1]<-floor((a+c)*N)
  v[3,2]<-floor((b+d)*N)
```

```
  v[3,3]<-floor((a+b+c+d)*N)
  show(v)

  p<-(((choose((A+B),A))*(choose((C+D),C)))/(choose(N,A+C)))
      # probability of the observed data

  N<-A+B+C+D

  #The results that are more extreme:
  # (1) fix the Row Total and Column Total
  # (2) if a<0.5, then sum all the cases from 0 to a*N
  #     if a>=0.5, then sum all the cases from a*N to N
  # (3) defrive the remaining three cells by using (1)
  p_sum<-p
  if(a>=0.5)
  {
    for(i in A+1:A+C)
    {
      new_A<-i
      new_B<-v[1,3]-new_A
      new_C<-v[3,1]-new_A
      new_D<-v[3,2]-new_B
      new_N<-new_A+new_B+new_C+new_D
      p_sum<-p_sum +
      (((choose((new_A+new_B),new_A))*(choose((new_C+new_D),new_C)))
      /(choose(new_N,new_A+new_C)))
    }
  }
  if(a<0.5)
  {
    for(i in 0:A-1)
    {
      new_A<-i
      new_B<-v[1,3]-new_A
      new_C<-v[3,1]-new_A
      new_D<-v[3,2]-new_B
      new_N<-new_A+new_B+new_C+new_D
      p_sum<-p_sum +
      (((choose((new_A+new_B),new_A))*(choose((new_C+new_D),new_C)))
      /(choose(new_N,new_A+new_C)))
    }
  }
  print(p_sum)
  return (p_sum)
}
```

```
p_approx<-fisher_exact_p_value_fractional_input_no_approximation(N=24558,
    a=0.52, b=0.41, c=0.48, d=0.59)
```

Output in the console of R Studio:

```
                      Men Women Column Total
Yes, Support Trump 10068 12770        22838
No, Against Trump  14489 11787        26277
Total              24558 24558        49116
[1] NaN
```

Result: Apparently, choose(n,k) function in R doesn't work for this large number. Hence, we tried to implement the same function but with a scaling to get an approximation result.

3.3. **Our Second Attempt.**

Scale n from 24558 to 100.

First of all, we can use fisher.test in R to compute the same scaled data. If one doesn't specify "less" or "greater", then a two-tailed test is considered. "greater" (or "less") refers to a one-sided test comparing a null hypothesis that $p_1 = p_2$ to the alternative $p_1 > p_2$ (or $p_1 < p_2$)

```
# Scale n from 24558 to 100
A<-41#floor(100*b)
B<-59#floor(100*d)
C<-52#floor(100*a)
D<-48#floor(100*c)

TeaTasting <-
  matrix(c(A, B, C, D),
         nrow = 2,
         dimnames = list(Support = c("Yes", "No"),
                         Gender = c("Men", "Women")))
fisher.test(TeaTasting, alternative = "less")
TeaTasting
```

Result:

```
> fisher.test(TeaTasting, alternative = "less")

        Fisher's Exact Test for Count Data

data:  TeaTasting
p-value = 0.07805
alternative hypothesis: true odds ratio is less than 1
95 percent confidence interval:
 0.000000 1.065993
sample estimates:
odds ratio
 0.6429149

> TeaTasting
       Gender
Support Men Women
    Yes  41    52
    No   59    48
```

Therefore, we should obtain p-value$=0.07$ in our implementation. Furthermore, if we set $\alpha = 0.05$, then the p-value is about $0.07 > 0.05$, so we failed to reject null-hypothesis under this scaled idea.

```
fisher_exact_p_value_fractional_input_approximation<-
   function(N, a, b, c, d){
  N<-100
  temp <- 0
  if (a>b)
  {
    temp<-b
    b<-a
    a<-temp
    temp<-c
    c<-d
    d<-temp
  }
  v<- matrix(ncol=3, nrow=3)
  colnames(v)<- c("Men","Women", "Column Total")
  rownames(v)<- c("Yes, Support Trump","No, Against Trump",
     "Row Total")
  A<-floor(N*a)
  B<-floor(N*b)
```

```
C<-floor(N*c)
D<-floor(N*d)
#raw data
v[1,1]<-floor(a*N)
v[1,2]<-floor(b*N)
v[2,1]<-floor(c*N)
v[2,2]<-floor(d*N)
v[1,3]<-floor((a+b)*N)
v[2,3]<-floor((c+d)*N)
v[3,1]<-floor((a+c)*N)
v[3,2]<-floor((b+d)*N)
v[3,3]<-floor((a+b+c+d)*N)
show(v)
n<-A+B+C+D
p<-(((choose((A+B),A))*(choose((C+D),C)))/(choose(n,A+C)))
p_sum<-p
if(a>=0.5)
{
  for(i in (a*(A+C))+1:(A+C))
  {
    new_A<-i
    new_B<-v[1,3]-new_A
    new_C<-v[3,1]-new_A
    new_D<-v[3,2]-new_B
    new_n<-new_A+new_B+new_C+new_D
    p_sum<-p_sum +
    (((choose((new_A+new_B),new_A))
    *(choose((new_C+new_D),new_C)))/(choose(new_n,new_A+new_C)))
  }
}
if(a<0.5)
{
  for(i in 0:(a*(A+C))-1)
  {
    new_A<-i
    new_B<-v[1,3]-new_A
    new_C<-v[3,1]-new_A
    new_D<-v[3,2]-new_B
    new_n<-new_A+new_B+new_C+new_D
    p_sum<-p_sum +
    (((choose((new_A+new_B),new_A))
    *(choose((new_C+new_D),new_C)))/(choose(new_n,new_A+new_C)))
  }
}
print(p_sum)
```

```
  return (p_sum)
}
p_approx<-fisher_exact_p_value_fractional_input_approximation(N=24558,
    a=0.52, b=0.41, c=0.48, d=0.59)
```

Output in the console of R Studio:

```
                     Men Women Row Total
Yes, Support Trump    41    52        93
No, Against Trump     59    48       106
Column Total         100   100       200
[1] 0.07805452
```

Except the scaled part, we can say this is computation is correct, because if one compares this result (one-tailed) to several online contingency table calculators, one could find that they're consistent. Furthermore, we will see that this Fisher's Exact Test is very sensitive in scaling the sample size, although we keep the ratios $(a/n, b/n, c/n, d/n)$ the same as the given data. But, in our following two attempts, we can see that if one can compute the data by using the original sample size, then we can derive a extremely small p-value, and with the same significance level $\alpha = 0.05$, we should actually reject the null-hypothesis.

3.4. **Our Third Attempt.**

This attempt is for paving the way for the next attempt, because we need a control group (benchmark) to compare our implementation. So, let see what result we can get by using the original sample size with fisher.test function in R.

```
# without scaling
n<-24558 #24558 respondents
a<-0.52
b<-0.41
c<-0.48
d<-0.59
A<-floor(24558*b)
B<-floor(24558*d)
C<-floor(24558*a)
D<-floor(24558*c)
TeaTasting <-
  matrix(c(A, B, C, D),
```

```
        nrow = 2,
        dimnames = list(Support = c("Yes", "No"),
                        Gender = c("Men", "Women")))
fisher.test(TeaTasting, alternative = "less")
TeaTasting
```

Output in the console of R Studio:

```
> fisher.test(TeaTasting, alternative = "less")

        Fisher's Exact Test for Count Data

data:  TeaTasting
p-value < 2.2e-16
alternative hypothesis: true odds ratio is less than 1
95 percent confidence interval:
 0.0000000 0.6609782
sample estimates:
odds ratio
 0.6414059

> TeaTasting
       Gender
Support   Men Women
    Yes 10068 12770
    No  14489 11787
```

Let's calculate the error between scaled and non-scaled results. By using fisher.test, we can only know an upper bound of the p-value: $2.2e-16 = 2.2 \times 10^{-16}$, so we can know the error rate is at least this large:

$$\text{Error}\% = \frac{|\text{Exact value} - \text{Approximate value}|}{\text{Exact value}} \times 100\% \tag{16}$$

$$= \frac{(0.07805452) - 2.2e-16}{2.2e-16} \times 100\% = 3.5477272727272627\overline{27} \times 10^{12}\% \tag{17}$$

Although we just scaled down the sample size with a fraction $\frac{1}{245.58}$, the error is amplified by this exact test, or more precisely speaking, the hypergeometric distribution.

3.5. **Our Fourth Attempt.**
In this attempt, we implemented our own code with **dhyper** function. Apparently, we beat the Fisher's Exact Test implementation in R, because we obtain a smaller p-value. **Our implementation:**

```
fisher_exact_p_value_fractional_input<- function(N, a, b, c,
    d){
  temp <- 0
  if (a>b)
  {
    temp<-b
    b<-a
    a<-temp
    temp<-c
    c<-d
    d<-temp
  }
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("Men","Women", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")

A<-floor(N*a)
B<-floor(N*b)
C<-floor(N*c)
D<-floor(N*d)

#raw data
v[1,1]<-floor(a*N)
v[1,2]<-floor(b*N)
v[2,1]<-floor(c*N)
v[2,2]<-floor(d*N)
v[1,3]<-floor((a+b)*N)
v[2,3]<-floor((c+d)*N)
v[3,1]<-floor((a+c)*N)
v[3,2]<-floor((b+d)*N)
v[3,3]<-floor((a+b+c+d)*N)
show(v)

# Method: Use dhyper(x, m, n, k, log = FALSE),
# x=A, n=C+D, k=A+C
p_Mathod_A<-dhyper(x=A, m=A+B, n=C+D, k=A+C, log = FALSE)

#H_0: men and women are equally likely to "support Trump"
```

```
N<-A+B+C+D

#The results that are more extreme:
# (1) fix the Row Total and Column Total
# (2) if a<0.5, then sum all the cases from 0 to a*100
#     if a>=0.5, then sum all the cases from a*100 to 100
# (3) defrive the remaining three cells by using (1)

p_sum<-p_Mathod_A
if(a>=0.5)
{
  for(i in A+1:A+C)
  {
    new_A<-i
    new_B<-v[1,3]-new_A
    new_C<-v[3,1]-new_A
    new_D<-v[3,2]-new_B
    new_N<-new_A+new_B+new_C+new_D
    p_sum<-p_sum+dhyper(x=new_A, m=new_A+new_B, n=new_C+new_D,
        k=new_A+new_C, log = FALSE)
  }
}
if(a<0.5)
{
  for(i in 0:A-1)
  {
    new_A<-i
    new_B<-v[1,3]-new_A
    new_C<-v[3,1]-new_A
    new_D<-v[3,2]-new_B
    new_N<-new_A+new_B+new_C+new_D
    p_sum<-p_sum+dhyper(x=new_A, m=new_A+new_B, n=new_C+new_D,
        k=new_A+new_C, log = FALSE)
  }
}
print(p_sum)
return (p_sum)
}
p_value_mA<-fisher_exact_p_value_fractional_input(N=24558,
    a=0.52, b=0.41, c=0.48, d=0.59)
```

Output in the console of R Studio:

```
                      Men Women Row Total
Yes, Support Trump 10068 12770     22838
No, Against Trump  14489 11787     26277
Column Total       24558 24558     49116
[1] 1.950922e-132
```

Discussion.
We can see that this p-value is way smaller than the result we derived from the fisher.test function, which is implemented in R packages. This is because the function didn't implement in the way we wanted (it uses a simpler formula without Saddle point correction which produces a larger error). Since the modern computer has a finite amount of data it can hold, to obtain a result in a reasonable time, each package has a certain approximation for a large input number in binomial coefficients. ***The approximation functions of Hypergeometric Distribution and Binomial Coefficients are all increasing functions therefore we don't want the probability to diverge. So the criterion we used to justify which implementation is better is to compare the output to see which one can have a smaller output.***

It is reasonable to believe the p-value we obtained in our own implementation is more accurate and much closer to the exact value. If one uses the default function fisher.test, one can only know an upper bound of the p-value: $2.2e-16$, but in our implementation, we know the value is $1.950922e-132 = 1.950922 \times 10^{-132}$ which is 116 order of magnitude smaller than the bound given by R Core Team's implementation.

To Unlock Next Level

It follows that our implementation can tolerate a larger number of sample sizes. But, we should not stop here. **Our next question is how can we beat our implementation?** If one really would love to do this, then one needs to dig out how this approximation formula of binomial coefficients was implemented by R Core Team[3].

After looking up the official documents of R[3] for **dhyper**, one will see the following:

Source

`dhyper` computes via binomial probabilities, using code contributed by Catherine Loader (see `dbinom`).

and so we go into the next level (looked up the **dbinom**), and found this

Source

For `dbinom` a saddle-point expansion is used: see

Catherine Loader (2000). *Fast and Accurate Computation of Binomial Probabilities*;

After reading Catherine Loader's manuscript[2], one could know that, **dhyper**, the function we used in our implementation is based on **Saddle Point Approximation**. *Hence, the dhyper function, we used is implemented by using Saddle point method. Furthermore, when the input is large enough this[3] implementation can fail at* 10^{17} *level. However, this number is totally reachable in real data. For instance, in our examples in the next section, we consider data with the sample size:* $n = 24558$. *Now,* $24558! \doteq 3.547 \times 10^{97151}$ *which is not a small number. Nevertheless, once the reader builds up a thorough understanding on how to derive the asymptotic formula like what Catherine Loader did, then reader should be able to derive as many higher order terms as one needs.*

4. Part III: Application of Part II Using Real Data

Sources.
After developing, and implementing our code in R, we now can use the algorithm we developed to analyze 20 questions about the President Donald Trump done by CNN. The 21th data were collected manually from debate.org. We traced back to each user that voted on that webpage[6].

Implementations.
First of all, since the code is almost the same, we just enclosed them into four functions. The first two functions are implemented using the algorithm we developed in Part II. The control group (benchmark, to compare the result obtained from our algorithm) is function 3 and 4. The reason why we implemented it in this way is because we have two different data types. For the CNN data, we only knew the total number of respondents and the fractions of each category. These fractional data are corresponding to function 1 and function 3. On the other hand, for the integral (non-fractional) data, we don't need to convert it. This is because the object-oriented coding style we wrote for the other two functions can handle this case which are function 2 and function 4.

Significance Level: we reject H_0 if p-value$< 1\%$ ($\alpha = 0.01$).

```
##### Data Analysis #####
#source https://www.cnn.com/election/2016/results/exit-polls
#Define Funciotn
##### function 1 with fractional input #####
fisher_exact_p_value_fractional_input<- function(N, a, b, c,
    d, Col, Row){
  temp <- 0
  if (a>b)
  {
    temp<-b
    b<-a
    a<-temp
    temp<-c
    c<-d
    d<-temp
  }
  v<- matrix(ncol=3, nrow=3)
  colnames(v)<-Col#<- c("Men","Women", "Column Total")
```

```
rownames(v)<-Row # c("Yes, Support Trump","No, Against
    Trump", "Row Total")

A<-floor(N*a)
B<-floor(N*b)
C<-floor(N*c)
D<-floor(N*d)
#raw data
v[1,1]<-floor(a*N)
v[1,2]<-floor(b*N)
v[2,1]<-floor(c*N)
v[2,2]<-floor(d*N)
v[1,3]<-floor((a+b)*N)
v[2,3]<-floor((c+d)*N)
v[3,1]<-floor((a+c)*N)
v[3,2]<-floor((b+d)*N)
v[3,3]<-floor((a+b+c+d)*N)
show(v)
p_Mathod_A<-dhyper(x=A, m=A+B, n=C+D, k=A+C, log = FALSE)
N<-A+B+C+D
p_sum<-0
p_sum<-p_Mathod_A
if(a>=0.5)
{
  for(i in A+1:A+C)
  {
    new_A<-i
    new_B<-v[1,3]-new_A
    new_C<-v[3,1]-new_A
    new_D<-v[3,2]-new_B
    new_N<-new_A+new_B+new_C+new_D
    p_sum<-p_sum+dhyper(x=new_A, m=new_A+new_B,
        n=new_C+new_D, k=new_A+new_C, log = FALSE)
  }
}
if(a<0.5)
{
  for(i in 0:A-1)
  {
    new_A<-i
    new_B<-v[1,3]-new_A
    new_C<-v[3,1]-new_A
    new_D<-v[3,2]-new_B
    new_N<-new_A+new_B+new_C+new_D
```

```
      p_sum<-p_sum+dhyper(x=new_A, m=new_A+new_B,
          n=new_C+new_D, k=new_A+new_C, log = FALSE)
    }
  }
  print(p_sum)
  return (p_sum)
}

#### function 2 non-fractional input ###############
fisher_exact_p_value_non_fractional_input<- function(N, a, b,
    c, d, Col, Row){
  temp <- 0
  if (a>b)
  {
    temp<-b
    b<-a
    a<-temp
    temp<-c
    c<-d
    d<-temp
  }
  v<- matrix(ncol=3, nrow=3)
  colnames(v)<-Col
  rownames(v)<-Row
  A<-a
  B<-b
  C<-c
  D<-d
  #raw data
  v[1,1]<-a
  v[1,2]<-b
  v[2,1]<-c
  v[2,2]<-d
  v[1,3]<-(a+b)
  v[2,3]<-(c+d)
  v[3,1]<-(a+c)
  v[3,2]<-(b+d)
  v[3,3]<-(a+b+c+d)
  show(v)
  p_Mathod_A<-dhyper(x=A, m=A+B, n=C+D, k=A+C, log = FALSE)
  N<-A+B+C+D
  p_sum<-0
  p_sum<-p_Mathod_A
  if((a/(A+C))>=0.5)
  {
```

```
    for(i in A+1:A+C)
    {
      new_A<-i
      new_B<-v[1,3]-new_A
      new_C<-v[3,1]-new_A
      new_D<-v[3,2]-new_B
      new_N<-new_A+new_B+new_C+new_D
      p_sum<-p_sum+dhyper(x=new_A, m=new_A+new_B,
          n=new_C+new_D, k=new_A+new_C, log = FALSE)
    }
  }
  if((a/(A+C))<0.5)
  {
    for(i in 0:A-1)
    {
      new_A<-i
      new_B<-v[1,3]-new_A
      new_C<-v[3,1]-new_A
      new_D<-v[3,2]-new_B
      new_N<-new_A+new_B+new_C+new_D
      p_sum<-p_sum+dhyper(x=new_A, m=new_A+new_B,
          n=new_C+new_D, k=new_A+new_C, log = FALSE)
    }
  }
  print(p_sum)
  return (p_sum)
}

#### function 3 Control Group (benchmark) #####
fisher_test_fractional_input<- function(N, a, b, c, d){
A<-floor(24558*b)
B<-floor(24558*d)
C<-floor(24558*a)
D<-floor(24558*c)
TeaTasting <-
  matrix(c(A, B, C, D),
         nrow = 2,
         dimnames = list(Guess = c("Milk", "Tea"),
                         Truth = c("Milk", "Tea")))
p_less<-(fisher.test(TeaTasting, alternative = "less"))
print(p_less)
p_greater<-(fisher.test(TeaTasting, alternative = "greater"))
print(p_greater)
}
#### function 4 Control Group (benchmark) #####
```

```
fisher_test_non_fractional_input<- function(N, a, b, c, d){
  TeaTasting <-
    matrix(c(a, b, c, d),
           nrow = 2,
           dimnames = list(Guess = c("Milk", "Tea"),
                           Truth = c("Milk", "Tea")))
  p_less<-(fisher.test(TeaTasting, alternative = "less"))
  print(p_less)
  p_greater<-(fisher.test(TeaTasting, alternative =
      "greater"))
  print(p_greater)
}
############################

# Data_1_age
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("18-44","45 and older", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.39
b<-0.52
c<-0.61
d<-0.48
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 8.777315e-185
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_2_race
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("white","non-white", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.57
b<-0.21
c<-0.43
d<-0.79
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
```

```
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_3_college graduate
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("college graduate","non-college graduate",
    "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.42
b<-0.51
c<-0.58
d<-0.49
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 2.87179e-89
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_4_income
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("under $100k","$100k or more", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.45
b<-0.47
c<-0.55
d<-0.53
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 4.575248e-06
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "less") gives p-value
    = 4.574e-06
# Result: reject H_0, our implementation works

# Data_5_marital status
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
```

```
colnames(v)<- c("married","unmarried", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.52
b<-0.37
c<-0.48
d<-0.63
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 1.88984e-246
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "less") gives p-value
    < 2.2e-16
# Result: reject H_0, our implementation works

# Data_6_how often do you attend religious services?
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("monthly or more","less often than that",
    "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.53
b<-0.39
c<-0.47
d<-0.61
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 1.177908e-213
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_7_served in the u.s. military
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("veterans","non-veterans", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.60
b<-0.44
c<-0.40
d<-0.56
```

```
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 3.133147e-277
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_8_were you born a u.s. citizen?
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("yes","no", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.49
b<-0.31
c<-0.51
d<-0.69
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_9_first-time voter?
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("yes","no", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.38
b<-0.47
c<-0.62
d<-0.53
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 7.583563e-91
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "less") gives p-value
    < 2.2e-16
# Result: reject H_0, our implementation works

# Data_10_when did you decide presidential vote?
```

```
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("in the last month","before that", "Column
    Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.48
b<-0.45
c<-0.52
d<-0.55
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 1.467703e-11
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value = 1.467e-11
# Result: reject H_0, our implementation works

# Data_11_illegal immigrants working in the u.s. should be:
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("offered legal status","deported to home
    country", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.33
b<-0.83
c<-0.67
d<-0.17
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_12_how is the fight against isis going?
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("well","badly", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.22
b<-0.68
```

```
c<-0.78
d<-0.32
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_13_in your vote, were supreme court appointments:
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("important","not important", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.49
b<-0.39
c<-0.51
d<-0.61
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 7.188362e-111
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "less") gives p-value
    < 2.2e-16
# Result: reject H_0, our implementation works

# Data_14_does the country's criminal justice system:
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("treat all fairly","treat blacks unfairly",
    "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.73
b<-0.22
c<-0.27
d<-0.78
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "less") gives p-value
    < 2.2e-16
```

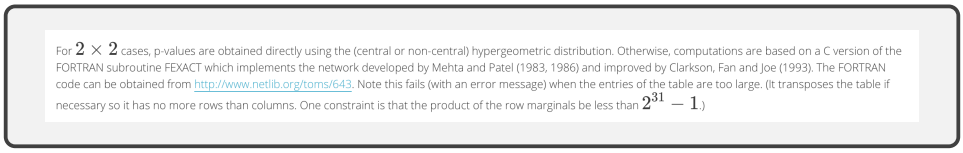
```
# Result: reject H_0, our implementation works

# Data_15_feelings about the federal government
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<-
    c("enthusiastic/satisfied","dissatisfied/angry", "Column
    Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.19
b<-0.57
c<-0.81
d<-0.43
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_16_opinion of government
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("government should do more","government doing
    too much", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.22
b<-0.72
c<-0.78
d<-0.28
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_17_opinion of barack obama as president
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("approve","disapprove", "Column Total")
```

```
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.10
b<-0.89
c<-0.90
d<-0.11
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_18_opinion of hillary clinton
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("favorable","unfavorable", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.03
b<-0.81
c<-0.97
d<-0.19
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "greater") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_19_opinion of donald trump
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("favorable","unfavorable", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.95
b<-0.15
c<-0.05
d<-0.85
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
```

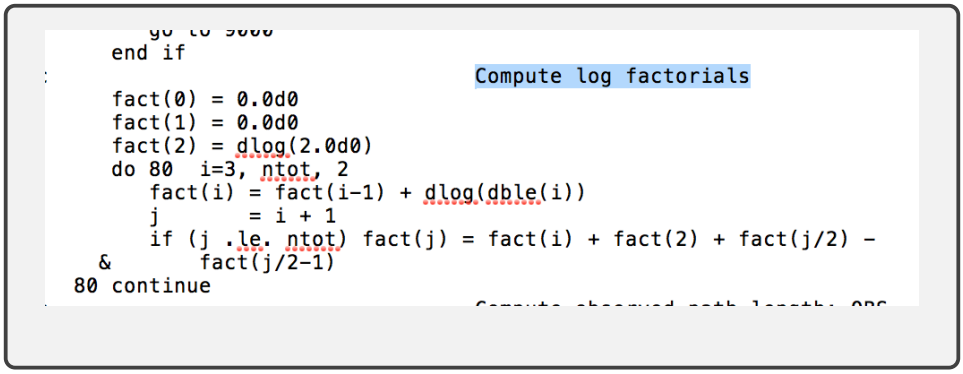
```
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "lessr") gives
    p-value < 2.2e-16
# Result: reject H_0, our implementation works

# Data_20_does clinton's use of private email bother you?
n<-24558 #24558 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("yes","no", "Column Total")
rownames(v)<- c("Yes, Support Trump","No, Against Trump",
    "Row Total")
a<-0.69
b<-0.06
c<-0.31
d<-0.94
p_value<-fisher_exact_p_value_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0
fisher_test_fractional_input(N, a, b, c, d)
# fisher.test(TeaTasting, alternative = "less") gives p-value
    < 2.2e-16
# Result: reject H_0, our implementation works

# Data_21
#source http://www.debate.org/polls/Trump-trade-war-with-china
n<-28 #28 respondents
v<- matrix(ncol=3, nrow=3)
colnames(v)<- c("< 20 yr of age","\u2265 20 yr of age",
    "Column Total")
rownames(v)<- c("Support Trump","Oppose Trump", "Row Total")
a<-7
b<-7
c<-8
d<-6
fisher_exact_p_value_non_fractional_input(N=n,
    a,b,c,d,colnames(v),rownames(v))
# p_value = 0.5
fisher_test_non_fractional_input(N=n, a, b, c, d)
# fisher.test(TeaTasting, alternative = "less") gives p-value
    = 0.5
# Result: Failed reject H_0, our implementation matchs the
    result from fisher.test
```

5. Part IV: Saddle Point Approximation

5.1. **Why Saddle Point Approximation?** Back to the last subsection of Part II, we already saw that the p-value we obtained is way smaller than the one we got from the fisher.test implementation in R. We have seen this result in Part III, for example, in Data 7, the fisher.test method can only give an upper bound, but our implementation can give an exact number (smaller than that bound, for sure). So, why does this happen? In that section, we have seen the R documents for **dbinom**, and **dhyper**. Hence, we should also take a look at **fisher.test**, since we already used it as the control group (benchmark).


For 2×2 cases, p-values are obtained directly using the (central or non-central) hypergeometric distribution. Otherwise, computations are based on a C version of the FORTRAN subroutine FEXACT which implements the network developed by Mehta and Patel (1983, 1986) and improved by Clarkson, Fan and Joe (1993). The FORTRAN code can be obtained from http://www.netlib.org/toms/643. Note this fails (with an error message) when the entries of the table are too large. (It transposes the table if necessary so it has no more rows than columns. One constraint is that the product of the row marginals be less than $2^{31} - 1$.)

If one clicks the URL (http://www.netlib.org/toms/643), one can download the FORTRAN code that was used by this fisher.test function in R. Hence, as long as one takes some time to read this code, one can find the the relevant part of our work in the following screenshot:



```
      go to 9000
   end if
                                        Compute log factorials
   fact(0) = 0.0d0
   fact(1) = 0.0d0
   fact(2) = dlog(2.0d0)
   do 80  i=3, ntot, 2
      fact(i) = fact(i-1) + dlog(dble(i))
      j       = i + 1
      if (j .le. ntot) fact(j) = fact(i) + fact(2) + fact(j/2) -
  &       fact(j/2-1)
80 continue
                                        Compute observed path length: OBS
```

From this source code, one can know that what's under the hood of fisher.test is the logarithms of factorial functions that is derived from Stirling formula. ***This tells us why the function we used (dhyper in function 1 and 2 in the previous section) can give us more precision. The function dbinom was used in dhyper and dbinom is implemented a formula derived by using Saddle Point Approximation. Since the Stirling formula is a result of the Saddle Point Method and the approximation formula in dbinom has a higher order correction, the*** $log(n!)$ ***doesn't have a high precision.***

The goal of this section is twofold: first, we want to understand how Saddle Point Approximation works in Catherine Loader's implementation of *dbinom*. Secondly, we will try to derive a more accurate binomial coefficient formula (a simpler form was implemented in R with less number of error correction terms), compared to the one in [2], and was implemented by R Core Team[3], and will implement this in our future work.

Although this method is usually covered in Complex Analysis, according to Wikipedia, it's definitely worth to learn for Statistics. Because "***It provides a highly accurate approximation formula for any PDF or probability mass function of a distribution, based on the moment generating function.***" Hence, this method is not only useful in improving the implementation of Fisher's Exact Test, but in general, is used in Mathematical Statistics.

5.2. **What Is Saddle Point Approximation?** The Laplace integral and stationary phase are in fact special cases of the general saddle point approximation. Hence, let's consider a more general setting beyond the Laplace's method and the notion of stationary phase (for example Airy function, and WKB method)[7]:

$$I(N) = \int_a^b dz g(z) e^{N f(z)}, N \gg 0, \tag{18}$$

where $f(z)$ is a complex analytic function. Setting $f(x, y) = u(x, y) + v(x, y)$. Also, for our purpose, let N be a positive integer.
Then, intuitively, one may expect $I(N)$ to be **dominated** by **maximum of** $u(x, y)$. Also, one needs the **stationary point**, z_0, of $v(x, y)$. That is, $f'(z) = 0$ is needed. Indeed, we only expect **Saddle Point** as extrema, reason being $u(x, y)$ and $v(x, y)$ satisfies $Laplace's$ equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

implies

$$\frac{\partial^2 u}{\partial x^2} < 0 \Rightarrow \text{ maximum in } x \text{ direction}$$

and

$$\frac{\partial^2 u}{\partial y^2} > 0 \Rightarrow \text{ maximum in } y \text{ direction.}$$

Hence we expect the integral to be dominated by the Highest Saddle Point (on the saddle), z_0. ***Then, we can deform the contour*** Γ

to include z_0***, near*** z_0***, by Cauchy Theorem***, and then we have[7]

$$f(z) \backsimeq f(z_0) + \frac{1}{2}f''(z_0)(z-z_0)^2, g(z) = g(z_0),$$

and then $I(N)$ becomes

$$I(N) \backsimeq g(z_0)e^{Nf(z_0)} \int_\Gamma dz \exp\left(\frac{1}{2}Nf''(z_0)(z-z_0)^2\right).$$

Our next goal is to determine the direction of steepest descent. Now, if we set

$$z - z_0 = re^{i\phi}, \text{ and } f''(z_0) = |f''(z_0)|e^{i\theta}.$$

The angles (ϕ, θ) **determines how we approach the Saddle Point.** We now have

$$I(N) \backsimeq g(z_0)\exp(Nf(z_0)) \int dre^{i\phi} \exp\left(\frac{1}{2}N|f''(z_0)|r^2e^{i\theta+2i\phi}\right).$$

We can make a convenient choice to render integral simple. Let $\theta+2\phi = \pi$. Hence,

$$\phi = \frac{\pi-\theta}{2},$$

and

$$I(N) \backsimeq g(z_0)\exp(Nf(z_0))e^{i\phi} \int dr \exp\left(\frac{-1}{2}N|f''(z_0)|\right).$$

By extending the integration range to $\pm\infty$, we can perform the Gaussian integral, then we can obtain an approximation of $I(N)$:

$$I(N) \backsimeq g(z_0)\exp(Nf(z_0))e^{i\phi}\left(\frac{2\pi}{N|f''(z_0)|}\right)^{\frac{1}{2}},$$

where $\phi = \frac{\pi-\theta}{2}$.

To recover stationary method: instead of having $\theta+2\phi = \pi$, we have $\theta + 2\phi = \frac{\pi}{2}$. Hence,

$$\int dx \exp\left\{\frac{N}{2}f''(z_0)(z-z_0)^2\right\} = e^{-\frac{i\theta}{2}+i\frac{\pi}{4}} \int dr \exp\left(\frac{iN}{2}|f''(z_0)|r^2\right).$$

Also, to recover to Laplace's method, we use saddle point on a real axis with $\frac{\partial^2 u}{\partial x^2} < 0$, then$\frac{\partial^2 u}{\partial y^2} > 0$, then we can go through the saddle point along real axis with $\phi = 0$, $\theta = \pi$, and $f'' < 0$.

5.3. **How To Derive Saddle Point Approximation of Binomial Coefficients?** Consider the following representation of Binomial Coefficients:

$$\binom{N}{M} = \oint_C dz \frac{1}{2\pi i} \frac{(1+z)^N}{z^{M+1}}$$

where C is a unit circle around origin.

Actually, it's quite straightforward to understand this formula because the numerator in the fraction $\frac{(1+z)^N}{z^{M+1}}$ can be easily written as binomial expansion, or Taylor expansion. Then after canceling the power by its denominator, it results in a Laurent series. By Residue Theorem, it follows that only z^M term makes a contribution. Hence, we obtained the $2\pi i\times$winding number, i.e., here it's the binomial coefficient. Therefore, to only get $\binom{N}{M}$, we must to divide this result with $2\pi i$.

Now, consider large N, M, we can set $M = Ny$, and turn the integral representation into

$$\binom{N}{M} = \oint_C dz \frac{1}{2\pi i} \frac{1}{z} \exp\left(N\left(\log(1+z) - y\log(z)\right)\right)$$

Identifying the **Saddle Point**:
Let $f(z) = \log(1+z) - y\log(z)$. Then

$$f'(z) = \frac{1}{1+z} - \frac{y}{z}, f''(z) = \frac{-1}{(1+z)^2} + \frac{y}{z}$$

Thus,

$$f'(z_0) = 0 \Rightarrow z_0 = \frac{y}{1-y}$$

$$f(z_0) = -y\log(y) - (1-y)\log(1-y)$$

$$f''(z_0) = \frac{(1-y)^3}{y}.$$

Please keep in mind that the integrand only has **Simple Pole** at $z = 0$. Therefore, we can deform the contour to include z_0. Also, since z_0 is on the real axis, $f''(z_0)$ is real value and $\theta = 0$. Hence $\phi = \frac{\pi}{2}$ (which is going through the contour from imaginary direction.) Therefore,

$$\binom{N}{M} \backsimeq \frac{1}{2\pi i}\frac{e^{i\phi}}{z_0}e^{Nf(z_0)}\left(\frac{2\pi}{Nf''(z_0)}\right)^{\frac{1}{2}} \tag{19}$$

$$= \left(\frac{1}{2\pi Ny(1-y)}\right)^{\frac{1}{2}} \exp\left(-N\left(y\log(y)+(1-y)\log(1-y)\right)\right). \tag{20}$$

where $M = Ny$.

The approximation formula (20) is a more accurate form that one should implement in R. This is the origin of the simplified form that was implemented in [3] and by using the same procedure of this section, one can derive Stirling's formula. By using Stirling's formula, one can derive the approximation of $log(n!)$ which is used in **fisher.test**. From here, we can understand why our implementations (function 1 and 2 in Part III) are more accurate than **fisher.test** when we take a large sample size n. Because under the hood, **fisher.test** didn't directly use Saddle Point Approximation. It uses the approximation based on Stirling's formula. Hence, after we have learned Saddle Point Approximation, we can think of it as an approximation of an approximation, i.e. the second layer of approximation brings unnecessary errors.

5.4. **Error Estimation and in Practice.** Once we derived the formula (20),

$$\binom{N}{M} \backsimeq \left(\frac{1}{2\pi Ny(1-y)}\right)^{\frac{1}{2}} \exp\left(-N\left(y\log(y)+(1-y)\log(1-y)\right)\right)$$

by using the power series of exponential function, let's define a partial sum:

$$S_n := \left(\frac{1}{2\pi Ny(1-y)}\right)^{\frac{1}{2}} \sum_{j=0}^{n} \frac{\left(-N\left(y\log(y)+(1-y)\log(1-y)\right)\right)^j}{j!}$$

we can rewrite (20) as follows:

$$\binom{N}{M} \backsimeq S_n + O(x^n) \tag{21}$$

where

$$x := \left(-N\left(y\log(y)+(1-y)\log(1-y)\right)\right)$$

Therefore, if one gives us an $\epsilon > 0$, where ϵ depends on the computation power of one's machine, we can have control over the approximation. That is, we can decide how many terms of Saddle Point Approximation

we want to implement in our customized R program. So, let $\epsilon > 0$ be given, we can choose $L > 0$, L is a positive integer, such that

$$\left| \binom{N}{M} - S_L \right| = \left| O(x^L) \right| < \epsilon \tag{22}$$

in other words, to implement this (20), we have two choices: one is to use the exact form (if we have a super computer), or we only implement a finite term (the finite partial sum, S_n, and choose $n = L$) to reach the precision we need on our local machine.

6. Part V: Conclusions

In the first part, we built up our own theory from scratch. The advantage of doing this kind of approach is that it can show us the essence of how hypergeometric distribution fits into Fisher's Exact Test. We can know that the essence is to enumerate how many extreme tables we could have. Hence, to go above 2 by 2 table, we can just hold this idea in mind and naturally figure out one's own way to enumerate an r by c table.

In the second part, we developed our own algorithm, and used fisher.test as our benchmark. We found that our implementation can have a better precision than the benchmark. We also read the R documents and found out in the source code that the *dhyper* function was implemented by using the Saddle Point Algorithm.

In Part III, in all 21 samples, the algorithm (implemented as function 1 and function 2) we developed worked really well. In some cases, our implementation could give a more meaningful value (rather than just an upper bound provided by fisher.test function).

On the other hand, we can see that in the first 20 samples that were collected from CNN.com, the null hypothesis were successfully rejected, so CNN did a good job on their data visualization. This also means the categories they chose to present are statistically meaningful, according to Fisher's Exact Test. Our algorithm and its implementations worked in all 21 examples. In the 21st data sample, the null hypothesis got rejected. This also shows us that when sample size is small, both methods (our implementations and R Core Team's[3] fisher.test) agree with each other.

In Part IV, we revisited Part III and investigated the possible sources of error of the fisher.test. We derived (20) and in future work, we can implement the Saddle Point Approximation (20) into our code. The result (20) and the procedure in deriving (22) allows us to reach any precision we want. Also, by learning this method, we can estimate the error of the p-value. To outlook the future work (that due to time constraint we couldn't cover them in this paper): a more detailed investigation and implementation to improve our function 1 and 2 can be done based on our work in this paper.

Since we started from scratch and derived this complete form of approximation of binomial coefficients, in the future we can have control over the precision of p-value using Fisher's Exact Test and other p-value tests. By using the moment generating function and our demo as a template, this provides a highly accurate approximation formula for any pdf or pmf of a distribution. Also, we believe this paper is especially useful for anyone who wants to implement their own algorithm and who wants to tackle the challenge to develop and implement a more efficient and precise software than the default function in R packages.

7. References

[1] Sir Ronald A. Fisher (1935). "Mathematics of a Lady Tasting Tea." In Fisher's "The Design of Experiments," Hafner Publishing Company Inc.

[2] Catherine Loader (2000). "Fast and Accurate Computation of Binomial Probabilities." Bell Lab's manuscripts.

[3] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

[4] "exit polls–national president." Viewed as Tables.
URL https://www.cnn.com/election/2016/results/exit-polls

[5] F. Yates (1984). "Tests of Significance for 2×2 Contingency Tables." Journal of the Royal Statistical Society. Series A (General), Vol. 147, No. 3 , pp. 426-463.

[6] Trumps trade war with China. Posted by: ladiesman. URL http://www.debate.org/polls/Trumps-trade-war-with-china

[7] Arfken, G. B., and Weber, H. J. (2005). "Mathematical methods for physicists," (6th ed.). Boston; New Delhi: Elsevier.