Final Project Writeup
STAT 580
Yeng Miller-Chang

Project Information

I am the only one who worked on this project. The GitLab link to the repository is `https://git.linux.iastate.edu/ymchang/stat580-finalproject`. You should have access to this repository.

Mathematical Exposition

Mathematically, my project is as follows:

Given a 2×2 contingency table of counts with $\alpha \in (0,1)$ fixed, generate a $100(1-\alpha)\%$ exact confidence interval for the odds ratio for the table.

This problem, although seemingly simple at its surface, requires a substantial amount of computational work.

Suppose we have a 2×2 contingency table of proportions which constitute a probability distribution:

π_{11}	π_{12}
π_{21}	π_{22}

The odds ratio is defined by $\theta = \dfrac{\pi_{11}\pi_{22}}{\pi_{12}\pi_{21}}$.[1]

Consider a 2×2 contingency table of counts derived from a sample:

n_{11}	n_{12}
n_{21}	n_{22}

Let $+$ denote summation over indices; for example, $n_{+1} = \sum_{i=1}^{2} n_{i1}$, and $n_{++} = \sum_{i=1}^{2}\sum_{j=1}^{2} n_{ij}$. The sample odds ratio is defined by[2]

$$\hat{\theta} = \frac{n_{11}n_{22}}{n_{21}n_{22}}.$$

It can be shown that, using a multinomial assumption, the Central Limit Theorem, and the Delta Method, that an approximate standard error for $\log(\hat{\theta})$ is [3]

$$\hat{\sigma}_{\hat{\theta}} = \sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}}.$$

One could easily use the approximation above to generate $100(1-\alpha)\%$ confidence intervals for $\log(\theta)$ based on a normal approximation; however, this is only appropriate for large samples. Suppose that the above table has been stratified by a variable (say, for example, age), for which

[1] Agresti, A. (2013), *Categorical Data Analysis* (3rd ed.), Hoboken, NJ: John Wiley & Sons, p. 606.
[2] Agresti, A. (2013), *Categorical Data Analysis* (3rd ed.), Hoboken, NJ: John Wiley & Sons, p. 69.
[3] Agresti, A. (2013), *Categorical Data Analysis* (3rd ed.), Hoboken, NJ: John Wiley & Sons, pp. 70-75.

each value of said variable has its own 2×2 contingency table. As an example, suppose we have the following 2×2 contingency table:

30	20
40	30

and that we decide to stratify these data based on another factor:

Group A

20	15
10	15

Group B

10	5
30	15

In cases such as the one above, it does not seem reasonable to use a normal approximation. Set $n = n_{++}$. It turns out that with the marginal totals given, it can be shown that n_{11} has probability mass function

$$f(t \mid n_{1+}, n_{+1}, n) = \frac{\binom{n_{1+}}{t}\binom{n - n_{1+}}{n_{+1} - t}\theta^t}{\sum_{u=m_-}^{m_+}\binom{n_{1+}}{u}\binom{n - n_{1+}}{n_{+1} - u}\theta^u} \tag{1}$$

for $m_- \leq t \leq m_+$ with $m_- = \max(0, n_{1+} + n_{+1} - n)$ and $m_+ = \min(n_{1+}, n_{+1})$, which is of the "noncentral hypergeometric distribution." Cornfield (1956)[4] provides one method to find a $100(1-\alpha)\%$ confidence interval for θ: one could solve for θ_0 and θ_1 in the equations

$$\frac{\alpha}{2} = \sum_{t \geq n_{11}} f(t \mid n_{1+}, n_{+1}, n, \theta_1) = \sum_{t \leq n_{11}} f(t \mid n_{1+}, n_{+1}, n, \theta_0). \tag{2}$$

Computational Exposition

The crux of this problem is to find the roots of the following functions of θ:

$$\sum_{t \geq n_{11}} f(t \mid n_{1+}, n_{+1}, n, \theta) - \frac{\alpha}{2} = \frac{\sum_{t \geq n_{11}}\binom{n_{1+}}{t}\binom{n - n_{1+}}{n_{+1} - t}\theta^t}{\sum_{u=m_-}^{m_+}\binom{n_{1+}}{u}\binom{n - n_{1+}}{n_{+1} - u}\theta^u} - \frac{\alpha}{2} \tag{3}$$

$$\sum_{t \leq n_{11}} f(t \mid n_{1+}, n_{+1}, n, \theta) - \frac{\alpha}{2} = \frac{\sum_{t \leq n_{11}}\binom{n_{1+}}{t}\binom{n - n_{1+}}{n_{+1} - t}\theta^t}{\sum_{u=m_-}^{m_+}\binom{n_{1+}}{u}\binom{n - n_{1+}}{n_{+1} - u}\theta^u} - \frac{\alpha}{2}. \tag{4}$$

These two problems can be described more succinctly as follows: let $\{a_t\}$ and $\{b_u\}$ be non-negative finite (with starting and stopping points) sequences. Then we wish to find the root of

$$\frac{\sum_t a_t\theta^t}{\sum_u b_u\theta^u} - \frac{\alpha}{2}. \tag{5}$$

[4]Cornfield, J. (1956), "A Statistical Problem Arising from Retrospective Studies," *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, 4, 135–148, Berkeley, CA: University of California Press. Available at https://projecteuclid.org/euclid.bsmsp/1200502552.

The function of θ above (5) is problematic. First of all, it is not continuous at $\theta = 0$, and who knows where else it could be discontinuous? Note the denominator of the first fraction implies that the function above is discontinuous for all θ satisfying $\sum_u b_u \theta^u = 0$. Therefore, using Newton-Raphson on this function of θ above is quite risky. Furthermore, since we do not know anything about the powers of θ beforehand, we can't determine where the function of θ above will be positive or negative, so the bisection method is off limits.

Instead, let's choose to ignore the problem of continuity. Then what happens is as follows: set (5) equal to 0 to obtain

$$\frac{\sum_t a_t \theta^t}{\sum_u b_u \theta^u} - \frac{\alpha}{2} = 0 \implies \sum_t a_t \theta^t = \frac{\alpha}{2} \sum_u b_u \theta^u \implies \sum_t a_t \theta^t - \frac{\alpha}{2} \sum_u b_u \theta^u = 0.$$

We define

$$\mathrm{poly}(\theta) = \sum_t a_t \theta^t - \frac{\alpha}{2} \sum_u b_u \theta^u. \tag{6}$$

This is what is `poly()` in `final_project.c`, set equal to `generate_poly()` in `functions.c`. Note that poly is a difference of two polynomials, so that it is not only continuous in $\mathbb{R}$, but differentiable in $\mathbb{R}$ as well. poly also has the same roots as our problematic function (5), as well as a few extras.

But this is a chance we are willing to take. Why? First of all, note that in the original equations (3) and (4) by the fact that $m_- \leq t \leq m_+$ that - if we ignore the $\alpha/2$ term for the moment - the remaining fraction has a denominator which must contain each term of the numerator. Therefore, in our more general formulation of the problem, it follows that $\sum_u b_u \theta^u$ must contain each term of $\sum_t a_t \theta^t$.

Looking back at $\mathrm{poly}(\theta)$, what does this imply? Since $\{a_t\}$ and $\{b_u\}$ are non-negative, this implies that where $t = u$, the coefficient of θ^t must be $a_t \left(1 - \frac{\alpha}{2}\right)$ in $\mathrm{poly}(\theta)$, which is a non-negative coefficient. Every other case where there is a coefficient of θ^u which is not in the θ^t summation yields a negative coefficient.

In other words, we may express $\mathrm{poly}(\theta)$ as follows:

$$\mathrm{poly}(\theta) = \sum_t a_t \left(1 - \frac{\alpha}{2}\right) \theta^t - \frac{\alpha}{2} \sum_v c_v \theta^v \tag{7}$$

where $\{c_v\}$ is a non-negative finite sequence. We also know that the t-index set is either the set such that $t \geq n_{11}$ or $t \leq n_{11}$. Therefore, the v-index must be the complement of the t-index set, and we have our main result:

Theorem. There is only one pair of sign changes in the coefficients of $\mathrm{poly}(\theta)$ when the terms are arranged in ascending power of θ.

From Descartes' Rule of Signs,[5] the previous theorem immediately implies that

Corollary. $\mathrm{poly}(\theta)$ has at most one positive root.

Therefore, we are safe in using Newton-Raphson on poly to find its positive roots as poly has at most one positive root (the odds ratio we desire), and as long as we choose a reasonable guess to

[5] Weisstein, Eric W. "Descartes' Sign Rule." *Mathworld* [online]. Available at `http://mathworld.wolfram.com/DescartesSignRule.html`.

begin with. The derivative of poly, based on (6), is

$$\text{poly\_deriv}(\theta) = \sum_t t a_t \theta^{t-1} - \frac{\alpha}{2} \sum_u u b_u \theta^{u-1}. \tag{8}$$

This is what is `poly_deriv()` in `final_project.c`, set equal to `generate_poly_deriv()` in `functions.c`. Therefore, the Newton-Raphson procedure for finding the root of poly is as follows:

1. Choose an initial $\theta^{(0)}$.
2. Until convergence is met, or the number of iterations exceeds the maximum number of iterations desired, set for $k = 1, 2, \ldots$:

$$\theta^{(k)} \leftarrow \theta^{(k-1)} - \frac{\text{poly}\left(\theta^{(k-1)}\right)}{\text{poly\_deriv}\left(\theta^{(k-1)}\right)}.$$

This computation is performed by using two of three arrays as inputs:

- the array of coefficients and powers of the "upper polynomial" $\displaystyle\sum_{t \geq n_{11}} \binom{n_{1+}}{t} \binom{n - n_{1+}}{n_{+1} - t} \theta^t$,
- the array of coefficients and powers of the "lower polynomial" $\displaystyle\sum_{t \leq n_{11}} \binom{n_{1+}}{t} \binom{n - n_{1+}}{n_{+1} - t} \theta^t$, and
- the array of coefficients and powers of the "denominator polynomial" $\displaystyle\sum_{u=m_-}^{m_+} \binom{n_{1+}}{u} \binom{n - n_{1+}}{n_{+1} - u} \theta^u$.

Two of these three arrays (either the "upper" and "denominator" **or** the "lower" and "denominator") are then used as inputs for `poly()` and `poly_deriv()` in `final_project.c`.

To compile and execute, use the following commands in the docker command line:

```
gcc -o final_project final_project.c functions.c -lm -pedantic -Wall
./final_project 1 2 3 4 -alpha 0.05 -theta 20
Input matrix:
1 2
3 4

Assuming fixed row and column totals...
Row totals: 3, 7
Column totals: 4, 6

The 95.0% confidence interval for the odds ratio is [0.009, 20.296].
```

Documentation is available when you put in incorrect input:

```
[root@badab0901fb9 stat580-finalproject]# ./final_project 1 2 3 4
ERROR: The input cannot be processed. Use the template below.
REQUIRED:
```

```
 a b c d  ------- four numeric inputs in row-major order
                  These must precede all other inputs.
-theta e  ------- e is an initial guess for the odds ratio

OPTIONAL:
-alpha f  ------- f is a number to the nearest thousandth
                  with 0 < f < 1, with 1 - alpha
                  being the level of the confidence interval.
                  f is set to 0.05 by default.
-v g      ------- g is either 0 or 1. Setting g to 1 shows
                  all Newton-Raphson output, including function
                  outputs, derivative outputs, and updated theta
                  values.
                  g is set to 0 by default.
```

Future Directions

I hope to make this eventually into an R package. The main question I have is that given the derivation based on Descartes' Rule of Signs, there is at most one root of $\text{poly}(\theta)$. Is there a way to get this value without having to rely on an initial guess from the user for what the root is?