Simulation for Sample Size

Jose Antonio Aznar Roji (141964)

May 28, 2018

Abstract

Statistic courses in college often base their results in the potentially large size of the sample for the problem, nevertheless, in real life applications the resources are limited and we need to optimize their use. To that end, the study of relatively small sample sizes is very important to statisticians, also, it is more complex and could derive in distributions with no analytic answer. Simulation tools become critical to find an approximate result that satisfies the initial problem with some specified tolerance; more than that, these methods and algorithms are often preferred to the analytic solution, because of them being easy to generalize with some small changes to other specific problems.

1 Introduction

When facing a question that involves a non standard statistic problem, the main focus becomes the analytic solution or the simulation necessary to obtain an approximation; that puts the sample size as a given parameter, often established by a rule of thumb. That choice impacts the final result in various ways, for example, the size could be too small to use a certain distribution. That is why, here we show an alternative that emphasizes on a simple way to obtain the right sample size based on simulation tools; the algorithm used is called QuickSize[Ama99].

Making an algorithm for every possible problem is nearly impossible, in this case, it was designed to treat with the next simple versus compound hypothesis test:

$$H_0 : \theta = \theta_0 \quad \text{vs} \quad H_\alpha : \theta \neq \theta_0$$

The sample size must be determined so that the test has a stated amount of power $(1 - \beta)$ at H_α.

There are several parameters that affect the right sample size for a statistical experiment: the statistical test, the level of significance of the test α, the power of the test $1 - \beta$, and the values to be compared. Clearly, the most important parameter is the test to be employed, since all the others depend on it's results; in this way, different tests will have a different impact on the resulting sample size and will need different parameters to obtain the same number [MHK87].

In this text we will exemplify the algorithms functionality using a vector of two binomial random variables that should maintain some proportion between their results. Some parts of the code written in the open R language came from empirical data inferred from the problem statement for some parameter in Fisher's exact test.

2 Data

2.1 Problem statement [Ama99]

At the planning stage of a cancer clinical trial, it is known that roughly 10% of all patients respond to standard therapy and it is expected that over 50% of the patients would respond to a new experimental therapy. It is required to determine the number n of patients that should be assigned to each treatment for a 5% level one-sided Fisher's exact test to detect a difference at least as large as this with 80% or more power.

2.2 Analysis of the statement

Each essay of the trial X_i is a Bernoulli random variable, for each of the treatments we have a different parameter θ_j a priori, this gives us the well known following result:

$$\Sigma_{i=1}^{n} X_i = Y_j$$

where $Y_j \sim Binomial(n, \theta_j)$.

This way we have two Binomial random variables with $\theta_1 = 0.1$ and $\theta_2 = 0.5$ respectively, that vector will be the data that we have to simulate using the rbinom() function.

Other important parameters that come in the statement are $\alpha = 0.05$ and $\beta = 0.20$ that are the type I and type II errors cap. On the other hand, one parameter doesn't come explicitly in the problem, the odds ratio required for the Fisher's exact test, while experimenting with the code we found that an approximate for it's value is 1.25.

To use the fisher.test() function we need to construct the contingency matrix, which we do with the following code:

```
m.contingencia <- function(n,p1=0.1,p2=0.5){

    b1 <- rbinom(1,n,p1)
    b2 <- rbinom(1,n,p2)
    exitos <- c(b1,b2)
    rechazos <- c(n-b1,n-b2)
    datos <- c(exitos,rechazos)

    return(matrix(data=datos,nrow=2,ncol=2))
}
```

2.3 Analytic example

To show the complexity that we can get into, here we have an example that nearly fits our problem.

2.3.1 Conditional distribution analysis [MHK87]

Consider two independent samples of sizes n_1 and n_2 from two binomial distributions with parameters p_1 and p_2 respectively. Assume that it is desired to test the null hypothesis $H_0 : p_1 = p_2$ against the alternative hypothesis $H_1 : p_1 > p_2$ (one sided test). Let X and Y be random variables representing the number of successes in the two samples respectively, and let:

$$\psi = \frac{p_1(1-p_2)}{p_2(1-p_1)}$$

The conditional distribution of X given t and ψ, is given by the non-central hypergeometric distribution:

$$Pr(X = x|t, \psi) = \frac{\binom{n_1}{x}\binom{n_2}{t-x}\psi^x}{\Sigma_{k=L}^{U}\binom{n_1}{k}\binom{n_2}{t-k}\psi^k}$$

where t is the total number succeses in x and y, observations of X and Y; L = max(0,t-n_2) and U = min (n_1,t). Notice that under $H_0(\psi = 1)$, the above conditional simplifies to the central hypergeometric distribution:

$$Pr(X = s|t) = \frac{\binom{n_1}{x}\binom{n_2}{t-x}}{\binom{N}{t}}$$

With fixed marginal totals t, n_1 and n_2, the conditional probabilities in the above equation may be used to establish evidence for or against the null hypothesis.

2.3.2 Sample size approximation [JLF80]

Given a power for the test $(1-\beta)$ an approximate formula for the sample size in each of the two samples required to achieve the desired power is the following:

$$n' = \frac{(z_\alpha \sqrt[2]{(2pq)} + z_\beta \sqrt[2]{(p_1q_1 + p_2q_2)})^2}{\delta^2}$$

where $p = \frac{1}{2}(p_1 + p_2)$, q = 1 - p, z_u is the upper 100(1-u) percentile of the standard normal distribution, $\delta = p_2 - p_1$. Now, the above equation represents the formula for the sample size in each group.

Although, the next corrected formula for the sample size per group provides an excellent approximation to the sample size obtained by an exact analysis of power:

$$n = \frac{n'}{4}(1 + \sqrt[2]{1 + \frac{4}{n'\delta}})^2$$

This seems like an easy formula, but the analysis to get to this point is more complex than what it appears to be, also, as the presence of the standard normal distribution implies, it is an asymptotic result and has it's limitation when n is relatively small.

3 Methods

3.1 Fisher's exact test [Wei]

Fisher's exact test is a statistical test used to determine if there are nonrandom associations between two categorical variables.

Let there exist two such variables X and Y, with m and n observed states, respectively. Now form an m×n matrix in which the entries a_{ij} represent the number of observations in which x=i and y=j. Calculate the row and column sums R_i and C_j, respectively, and the total sum

$$N = \Sigma_i R_i = \Sigma_j C_j$$

of the matrix. Then calculate the conditional probability of getting the actual matrix given the particular row and column sums, given by

$$P_{cutoff} = \frac{(R_1!...R_m!)(C_1.!...C_n)}{N!\Pi_{i,j}a_{ij}!}$$

which is a multivariate generalization of the hypergeometric probability function. Now find all possible matrices of nonnegative integers consistent with the row and column sums R_i and C_j. For each one, calculate the associated conditional probability using the previous equation, where the sum of these probabilities must be 1.

To compute the P-value of the test, the tables must then be ordered by some criterion that measures dependence, and those tables that represent equal or greater deviation from independence than the observed table are the ones whose probabilities are added together. There are a variety of criteria that can be used to measure dependence. In the 2×2 case, which is the one Fisher looked at when he developed the exact test, either the Pearson chi-square or the difference in proportions (which are equivalent) is typically used. Other measures of association, such as the likelihood-ratio-test, G-squared, or any of the other measures typically used for association in contingency tables, can also be used.

The test is most commonly applied to 2×2 matrices, and is computationally unwieldy for large m or n. For tables larger than 2×2, the difference in proportion can no longer be used, but the other measures mentioned above remain applicable (and in practice, the Pearson statistic is most often used to order the tables). In the case of the 2×2 matrix, the P-value of the test can be simply computed by the sum of all P-values which are $\leq P_{cutoff}$.

For an example application of the 2×2 test, let X be a journal, say either Mathematics Magazine or Science, and let Y be the number of articles on the topics of mathematics and biology appearing in a given issue of one of these journals. If Mathematics Magazine has five articles on math and one on biology, and Science has none on math and four on biology, then the relevant matrix would be

-	Math. Mag.	Science
math	5	0
biology	1	4

Computing P_{cutoff} gives

$$P_{cutoff} = \frac{5!^2 6!4!}{10!(5!0!1!4!)} = 0.0238$$

and the other possible matrices and their P's are

$$\begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$$

P = 0.2381

$$\begin{bmatrix} 3 & 2 \\ 3 & 2 \end{bmatrix}$$

P = 0.4762

$$\begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix}$$

P = 0.2381

$$\begin{bmatrix} 1 & 4 \\ 5 & 0 \end{bmatrix}$$

P = 0.0238

which indeed sum to 1, as required. The sum of P-values less than or equal to P_{cutoff}=0.0238 is then 0.0476 which, because it is less than 0.05, is significant. Therefore, there would be a statistically significant association between the journal and type of article appearing.

3.2 QuickSize [Ama99]

This approach is based on the fact that the objective distribution is a bounded monotonically increasing function of n.

This algorithm works by searching through different values of n. To start the search, one specifies a starting value, n_0, for n; then at the ith $(1 \leq i < I)$ iteration:

1. Let n_i be the current value of n. Randomly generate B pseudosamples of size n_i and count how many of the B pseudosamples lead to rejection of H_0.

2. Let $B_i(n_i)$ be the total number of pseudosamples of the same size n_i drawn in all iterations up to and including the ith iteration. Suppose that $B_i^*(n_i)$ of them lead to rejection of H_0. At this stage, $P_i^*(n_i) = \frac{B_i^*(n_i)}{B_i(n_i)}$ serves as the current estimate of the objective distribution. If $P_i^*(n_i) < (1-\beta)$, set $n_{i+1} = n_i + m_i$; else if $P_i^*(n_i) > (1-\beta)$, set $n_{i+1} = n_i - m_i$; else if $P_i^*(n_i) = (1-\beta)$, $n_{i+1} = n_i$.

3. Proceed to the $(i+1)ith$ iteration.

At convergence, the iteration will tend to oscillate between two consecutive values of n, corresponding to powers just below and above 1 - β. The required sample size, N, is the larger of the two. The appropriateness of this sample size can be simply verified by means of a separate confirmatory simulation, in which two large independent sets of pseudosamples of sizes N - 1 and N respectively are generated, and it is checked that $P^*(N-1)$ and $P^*(N)$ bound $1-\beta$ as required.

Modifications to this basic procedure did not offer significant improvement but affected it's simplicity; all of this being said, this process makes no optimality claims, although it significantly accelerates the search for N and it is quite simple to program.

3.3 Code in R

```
sim.rechazo <- function(n){

  cont <- m.contingencia(n)
  or <- 1.25

  fish <- fisher.test(x=cont, or=or)

  if(fish$p.value<0.05){
    return(TRUE)
  }
  else{
    return(FALSE)
  }
}

sim.beta <- function(B,n){
  cont <- 0
  for(i in 1:B){
    if(sim.rechazo(n)){
      cont <- cont + 1
    }
  }
  return(cont)
}

quick.size <- function(n0,I,B,b,m){
  n <- n0
  N <- NULL
  n.aux <- NULL
  betas <- NULL
  llaves <- list()
  rechazos <- list()
  N <- list()
  num.elems <- 0

  for(i in 1:I){
    n.aux[i] <- n

    rech <- sim.beta(B,n)

    val <- match(n,llaves)
    if(is.na(val)){
      num.elems <- num.elems + 1

      llaves[num.elems] <- n
      rechazos[num.elems] <- rech
      N[num.elems] <- B
      beta.aprox <- rech/B
    }
    else{
      rechazos[val] <- as.numeric(rechazos[val][1]) + rech
      N[val] <- as.numeric(N[val]) + B

      beta.aprox <- as.numeric(rechazos[val])/as.numeric(N[val])
    }
    betas[i] <- beta.aprox
```

```
    if(beta.aprox < 1-b){
      n <- n + m
    }
    else{
      if(beta.aprox > 1-b){
        n <- n - m
      }
    }
  }
  return(list(n.aux, llaves, rechazos, N, betas))
}
```

4 Results

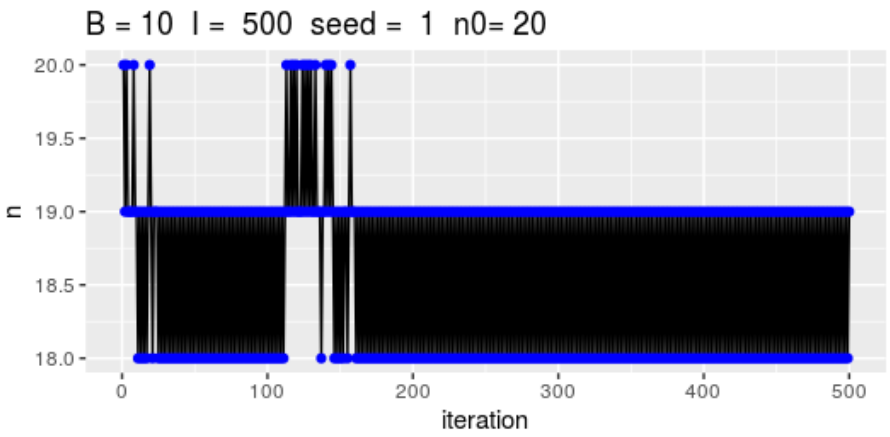
For the results to be easily replicated we set a seed for the random numbers generated.

The exact value found by Hammond for n in the previously described problem is 19, also, the following results were obtained by Amaratunga and his team after 500 iterations with B=10 and m=1 [Ama99]:

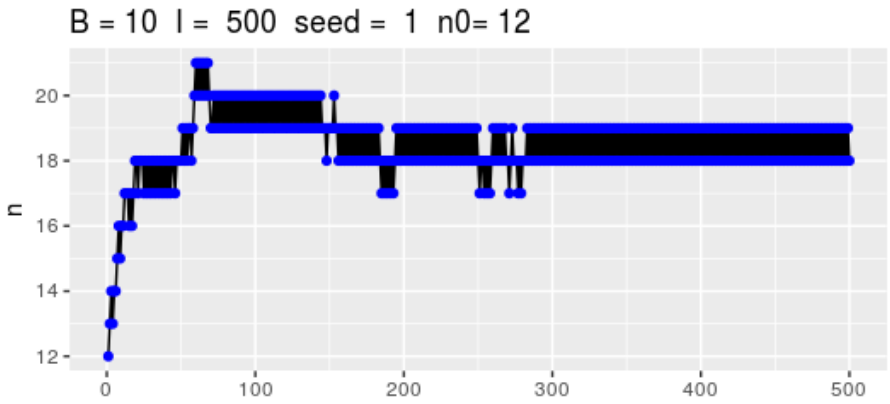
n	$B^*_+(n)$	$B_+(n)$	$P^*_+(n)$
17	370	500	.740
18	1959	2490	.787
19	1643	2000	.822
20	9	10	.900

We will be running trials with different seeds (s), iterations (I), number of pseudosamples per iterations (B) and initial value (n0). This way, we will be able to determine how each of this variables affect the result.

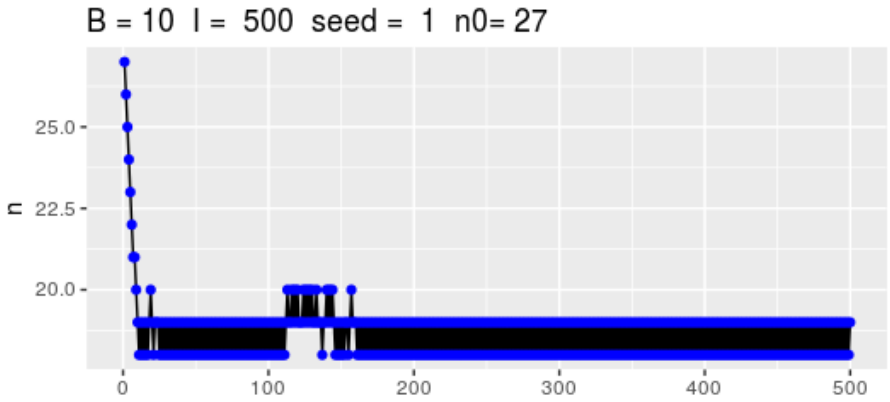
First we start with a certain seed and change the parameters:



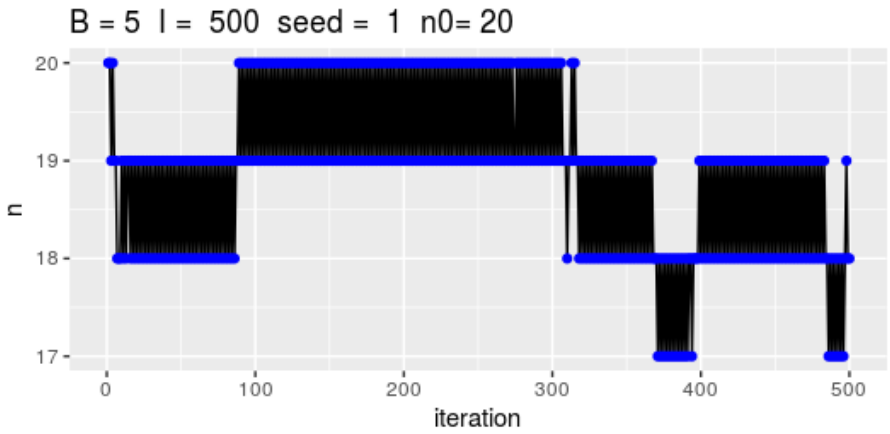
n	B*(n)	B(n)	P*(n)
18	1752	2250	0.778666666666667
19	2094	2580	0.811627906976744
20	144	170	0.847058823529412



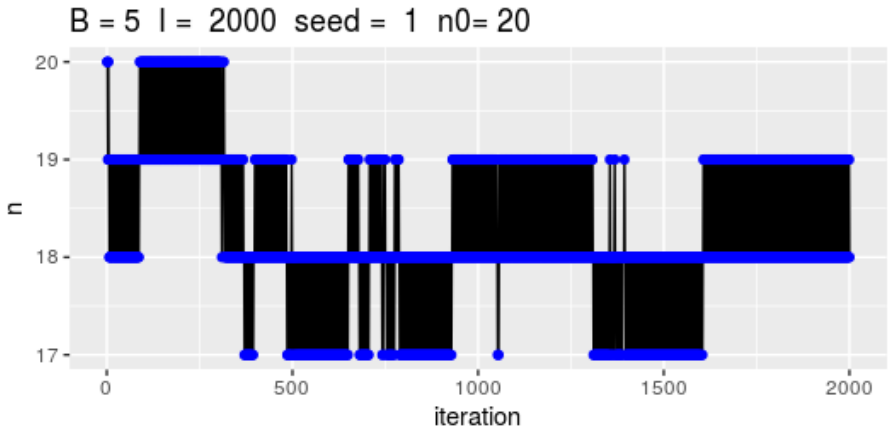
n	B*(n)	B(n)	P*(n)
12	6	10	0.6
13	11	20	0.55
14	20	30	0.666666666666667
15	14	20	0.7
16	39	50	0.78
17	230	290	0.793103448275862
18	1577	2000	0.7885
19	1701	2090	0.813875598086124
20	356	440	0.809090909090909



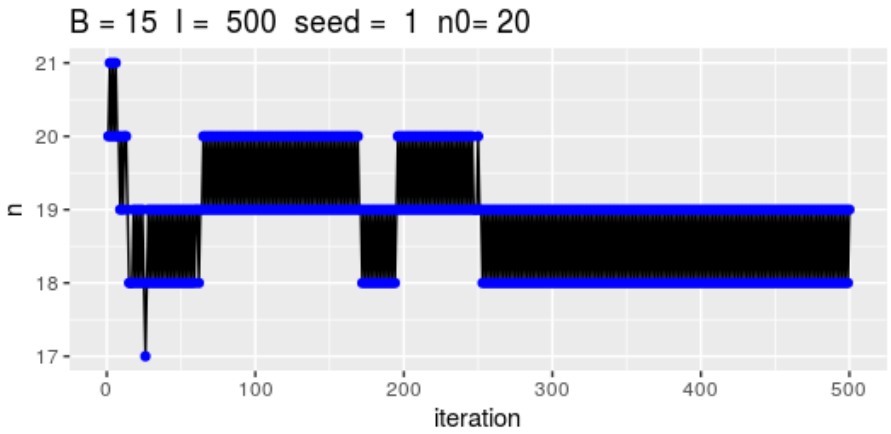
n	B*(n)	B(n)	P*(n)
18	1752	2250	0.778666666666667
19	2046	2520	0.811904761904762
20	126	150	0.84
21	17	20	0.85
22	9	10	0.9
23	10	10	1
24	10	10	1
25	10	10	1
26	10	10	1



n	B*(n)	B(n)	P*(n)
17	69	90	0.766666666666667
18	553	690	0.801449275362319
19	933	1150	0.811304347826087
20	469	570	0.82280701754386

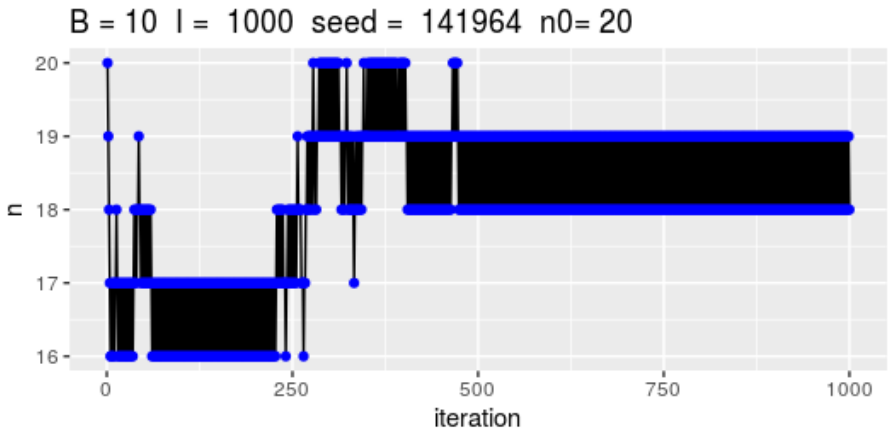


n	B*(n)	B(n)	P*(n)
17	1286	1620	0.793827160493827
18	3615	4545	0.795379537953795
19	2654	3265	0.812863705972435
20	469	570	0.82280701754386

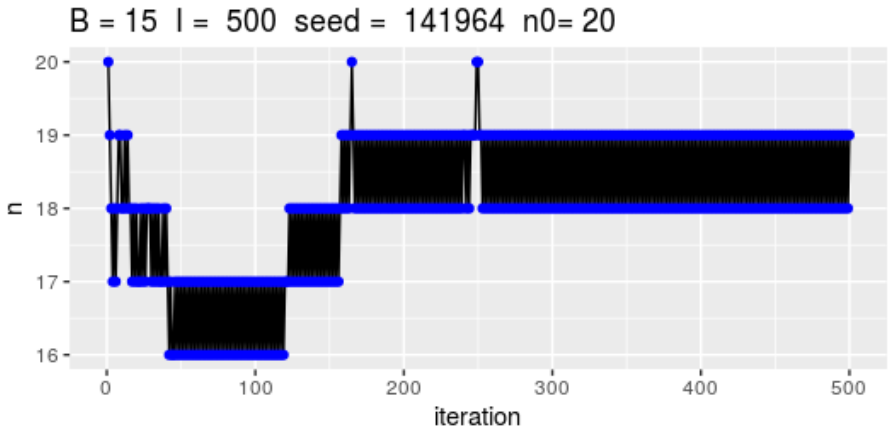


n	B*(n)	B(n)	P*(n)
17	11	15	0.733333333333333
18	1901	2415	0.787163561076605
19	2990	3705	0.807017543859649
20	1121	1320	0.849242424242424
21	41	45	0.911111111111111

Now we change the seed and see how many iterations it takes to converge making a variation on B and I:

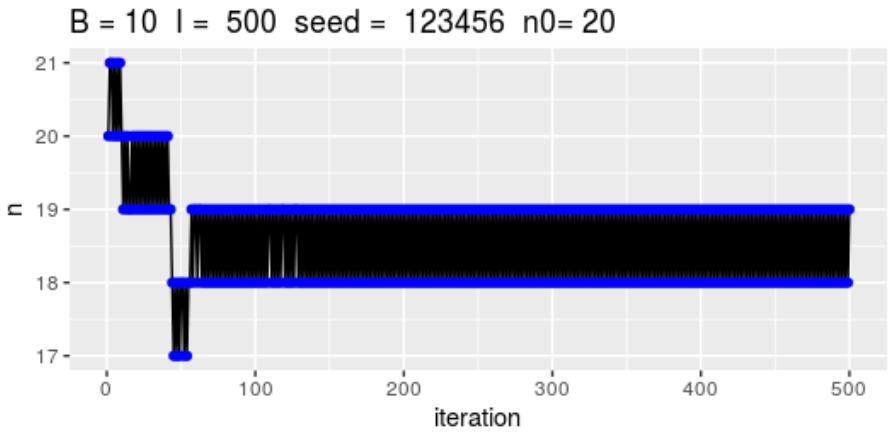


n	B*(n)	B(n)	P*(n)
16	736	1010	0.728712871287129
17	1054	1320	0.798484848484848
18	2729	3460	0.788728323699422
19	3062	3740	0.818716577540107
20	393	470	0.836170212765957

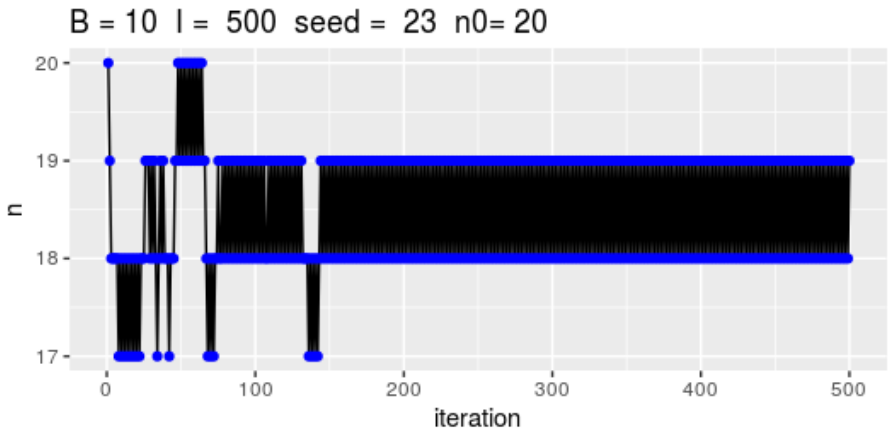


n	B*(n)	B(n)	P*(n)
16	443	600	0.738333333333333
17	871	1095	0.795433789954338
18	2400	3060	0.784313725490196
19	2190	2685	0.815642458100559
20	51	60	0.85

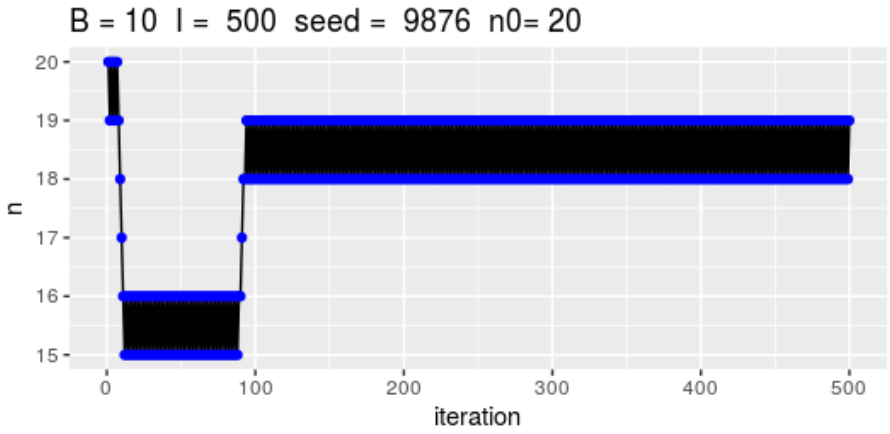
Finally, we change the seed to confirm it's effect:



n	B*(n)	B(n)	P*(n)
17	35	50	0.7
18	1767	2270	0.7784140969163
19	1952	2430	0.803292181069959
20	166	200	0.83
21	43	50	0.86



n	B*(n)	B(n)	P*(n)
17	123	170	0.723529411764706
18	1921	2460	0.780894308943089
19	1871	2270	0.824229074889868
20	89	100	0.89



n	B*(n)	B(n)	P*(n)
15	279	390	0.715384615384615
16	327	410	0.797560975609756
17	15	20	0.75
18	1604	2060	0.778640776699029
19	1715	2080	0.824519230769231
20	36	40	0.9

The code to run this results can be found at https://github.com/joseaznar/quicksize.git.

5 Conclusions

In the results section, different parameters where modified to evaluate their contribution to the sample size found after the corresponding number of iterations. The first graph gives us the reference with a certain seed and the same parameters used by Amaratunga, it confirms that the code matches what we expected; the second and third graphs, confirm that the choice of n_0 does not disturb the final n found by the algorithm; the fourth and fifth graphs tells us that lowering the number of samples per iteration makes the algorithm converge in more steps, and the sixth graph shows that increasing B does not necessarily increase the convergence speed.

On the other hand, when the seed is altered, the iteration needed to establish the result differ. Having 141964 as the seed gets us a slower convergence that we had with 1, although by increasing B we can reduce the number of iterations; other changes in the seed confirm the implications it has on the final n, this states the importance of randomness when we are using simulation tools.

A brute force approximation to this problem might give us the same result, nevertheless the time it could take can't be precisely bounded; in the same way, the initial value n_0 will have a big effect on the final result, having this amount of uncertainty is not something we want to add to the problem.

Because of the previously stated reasons to reject a brute force solution, the potential complexity of the target distribution and the results we got that confirm the convergence of our algorithm, *QuickSize* is a very suitable and simple solution to find the right n.

An extension to this study should be to test using other examples, like the other ones in Amaratunga's paper. In the same way, compared to a brute force algorithm, maybe combined with some heuristic or meta heuristic, it could give more light to the efficiency and optimality of this method.

It is important to remark that the odd ratio parameter for Fisher's exact test was found empirically because of the vague problem statement. Other than that, the study was a success and endorsed the use of this algorithm to find the sample to be used.

References

[Ama99] Dhammika Amaratunga. Searching for the right sample size. *The American Statistician*, 53(1):52–55, 1999.

[JLF80] Hans K. Ury Joseph L. Fleiss, Alex Tytun. A simple approximation for calculating sample sizes for comparing independent proportions. *Biometrics*, 36(1):342–346, 1980.

[MHK87] K.M. Magruder-Habib M.K. Habib and L.L. Kupper. A review of sample size determination in comparative studies. *Institute of Statistics Mimeo Series*, 1(1840):1–30, 1987.

[Wei] Eric W. Weisstein. *Fisher's Exact Test*. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/FishersExactTest.html.