Inverting binomial distributions

Ed Baranoski (ed.baranoski@gmail.com)

G4G15, February 2024

Abstract

The binomial distribution shows the likelihood of discrete success or failures for a given probability of success. This paper shows how to back out the probability density function of success probability from a binomial sample set. It compares this to real-world results, including Major League Baseball winning percentages.

1 Introduction

The binomial distribution is the well known way of showing the probability of a given outcome for things like number of heads when tossing a coin twenty times, or even the likelihood of a baseball team winning a best-of-seven series of games. It starts with an assumed success rate and then computes the likelihood of each possible outcome. The common test is to ask the likelihood of the next coin flip even after ten successive tosses yield heads. For something like a coin flip of a fair coin, it is reasonable to assume that the likelihood of a toss yielding either heads or tails is exactly 50% so it is still a 50/50 chance that tails will be the outcome of the next toss. However, what if you don't know whether the coin is fair or not? Perhaps it isn't a physical coin at all, just a poorly written simulation of one, or the process of flipping the coin isn't random. What is the likelihood of the actual success probability from a sample if you truly don't know the true success probability?

This paper shows how to use the Bayes Theorem to show the likely probability density function. The paper discusses two example use cases. The first is completely unknown success rates when you have no idea of what the likely success rate is. A coin toss obviously has physics behind the 50/50 heads-or-tails assumption, but other example like the likelihood of hitting a half court basketball shot are more instructive. For cases where the true success rate is close to 50%, the results aren't that surprising. For likelihoods closer to zero or 100%, the results are more interesting.

The second case uses historical records for baseball teams. Is the final record for a team really indicative of their *ideal* winning percentage, or were they lucky (or not) relative to their final record? This will also discuss the Pythagorean Expectation as a way of estimating a team's expected winning percentage based on runs scored and runs allowed.

2 Bayes Rule

The binomial distribution formula of $P(k|p) = \binom{n}{k} p^k (1-p)^{n-k}$ gives us the likelihood of getting k successes out of n trials when the probability of success is p for each individual trial. $P(k|p)$ is called a conditional probability of k given the parameter p. The term $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the how many combinations of k successes can be achieved out of n trials. For a given probability p, we can calculate the likelihood for all values of k. The question we address here is how do you solve the inverse problem; that is, for a given k successes in $n$trials, what is the likelihood of the probability being $p$or within any arbitrary range of values of p.

Bayes theorem is exactly what we need! Bayes rule states that you the conditional and relative probabilities are related–for a given n, $P(p|k) = P(k|p)P(p)/P(k)$. If we don't have any idea of the distribution of , we can assume that $P(p)$ is uniform between zero and one. Since we are starting with the value of k, that probability is one.

Ideally we want to compute the full distribution of $P(p|k)$ over the continuum of $p$from zero to one. The cumulative distribution function

$$C(p < P) = \int_0^P p(p|k)dp \tag{1}$$

would contain the likelihood of $p \leq P$, but a straight-forward implementation using the binomial distribution is very computationally expensive. Thankfully, you can use the ratio of binomial distributions so you only have to compute a single binomial probability. Appendix A shows a very computationally efficient way to compute this distribution. Appendix B includes code for computing the cumulative probability density function for the programming language R.

3 Use Case 1: Consecutive Successes/Failures

Suppose there is an series of event where you have no advance idea of what the success rate should be (unlike a coin flip, where the probability of heads or tails can be assumed to be 50%). If the first trial is a success, the likelihood that the actual probability of success is small (say under 10%) is much smaller than the likelihood the actual probability was high (over 90%, for example). The only information you have is that single success and are using the assumption that the actual distribution of the success probability is even from 0 to 1. Figure 1 shows this by looking at the cumulative density functions of the probability likelihoods for different numbers of consecutive successes (again assuming zero prior information on the actual probability density function). As the number of consecutive successes grows, the likelihood of the actual probability becomes closer to one.

You can then use these updated probability likelihoods to compute the likelihood that the next trial is a success. Table 1 shows the likelihood of the next success after a sequence of consecutive successes when you have no prior information on the actual probability function.

Table 1: Likelihood of the next success after a string of consecutive successes, assuming zero prior information on the actual probability density function

Consecutive Successes	1	2	3	4	5	6	7	8	9	10
Probability of Next Success	0.67	0.75	0.80	0.83	0.86	0.88	0.89	0.90	0.91	0.92

As we will see in the next section, large sample sizes (particularly with probabilities away from zero or one) do not need the inverse binomial distribution. You can approximate the continuous probability density function with the discrete points of the binomial distribution itself.

4 Use Case 2: Baseball Records and the Pythagorean Expectation

Baseball statistics are a great way to dive into large scale statistical analysis. Sources like www.retrosheet.com, baseballsavant.mlb.com, and www.baseball-reference.com offer treasure troves of quantifiable data for free. In this section, we will use look at probability distributions of winning percentages of teams over the years, the expected distribution around a "true" winning percentage, and comparison to a concept called Pythagorean Winning Percentage first introduced by Bill James [3].

A baseball season has been 162 games long since 1961 (excluding a few years due to strikes and the pandemic, which were shorter). A typical team wins between 60 and 100 of those games, with some exceptions on either end. Figure 2 shows the continuous inverse binomial distributions(continuous lines) and discrete binomial likelihoods for three different winning percentages, corresponding to an expected 60, 80, ands 100 wins. Note that the binomial distribution is a good discretized sampling of the actual continuous probability density function using the assumed winning percentages. Also note if a omniscient winning percentage was known in advance, the actual team's winning percentage can easily vary by a half dozen wins either way over the sample size of a 162 game season.[1]

[1]This is usually attributed to luck (either good or bad), but often correlates well with a team's performance in games decided by a single run as we'll see shortly.

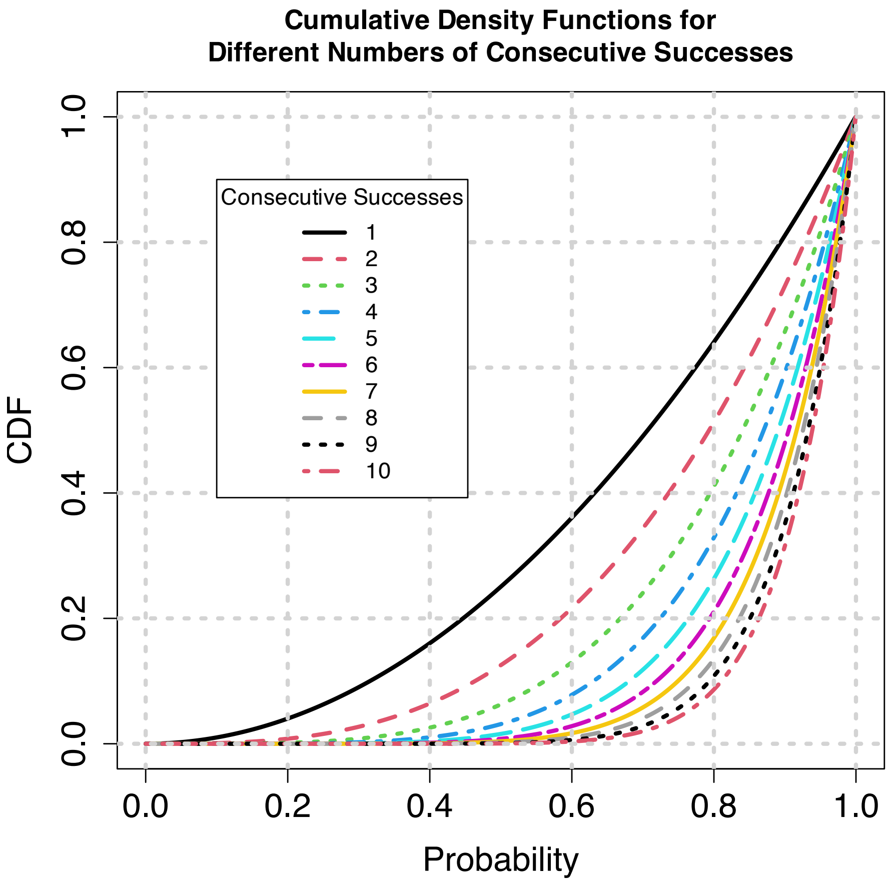

Figure 1: Cumulative density functions of the probability likelihoods for different numbers of consecutive successes, assuming zero prior information on the actual probability density function.

Bill James is largely credited with starting the SABRmetrics revolution in baseball analytics. (SABR is the Society for American Baseball Research.) His *Baseball Abstracts* introduced many analytical innovations, including what he called Pythagorean Expectation[3]. This was a heuristic approximation using number of runs a team scored (RS) and the number of runs they allowed (RA) to estimate a team's expected winning percentage using the formula

$$Pct = \frac{RS^2}{RS^2 + RA^2} \tag{2}$$

The name is due to the obvious similarity to the Pythagorean Theorem in geometry. Later analysis[4, 2] showed improvements,[2]including adjusting the exponent to 1.8 to better represent actual results.

This Pythagorean approximation holds amazingly well, and produces a tighter distribution of potential outcomes than the Bayesian distribution would indicate. To show this, we used Retrosheet[1] data for all 162 game seasons from 1961 through 2023. Over this time period, the number of teams varied from 24 to the current 30, giving us 1,329 end-of-year team records.[3] Figure 3 shows a histogram of the relative error between a team's actual record and what was predicted with the Pythagorean expectation. It also overlays the expected binomial distribution showing the expected variance assuming a team's actual record

[2]The exponent is generally low for low scoring games which have more variability in outcomes. For sports like basketball with much higher scores, there is less variability in outcomes and the exponent is closer to 15 instead of 2.

[3]A few seasons included one last tie-breaking game at the end of the season, or missed a game due to a rainout that was never rescheduled provided it had no impact on which teams advanced to the playoffs.

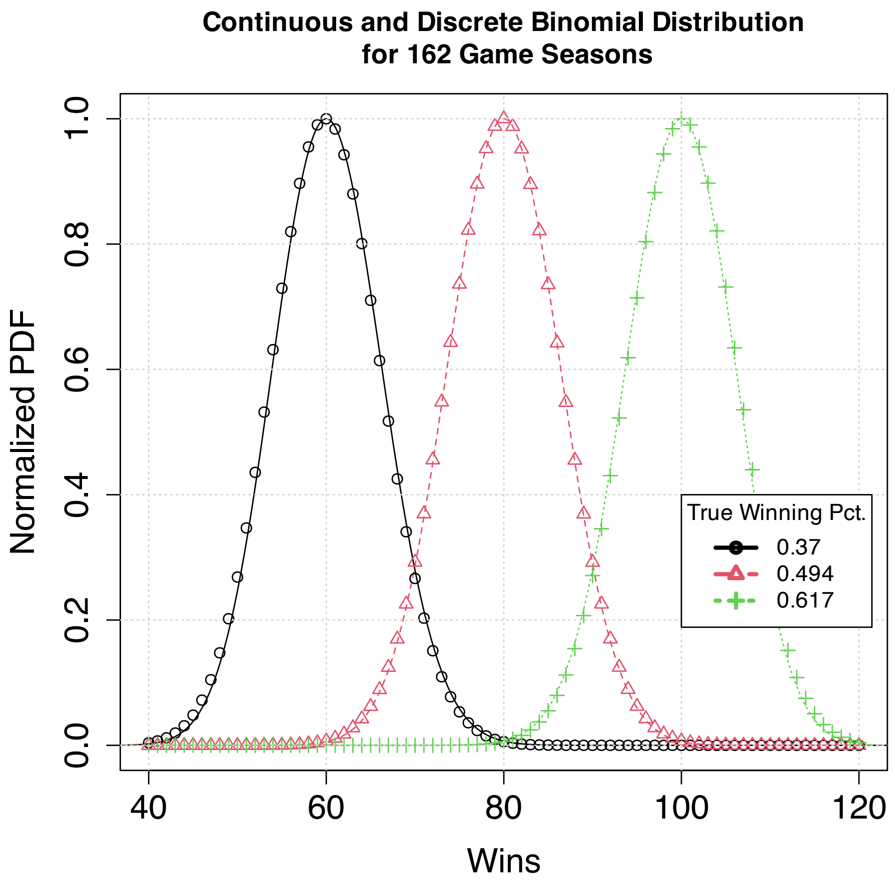

Figure 2: Continuous inverse binomial distributions(continuous lines) and discrete binomial likelihoods for three different winning percentages, corresponding to an expected 60, 80, ands 100 wins.

was indicative of the "true" winning percentage. This indicates that the variance around the predicted expectation is narrower for the Pythagorean Expectation than that predicted by the binomial distribution. This is due to the additional *a priori* information the Pythagorean Expectation has by using the actual runs scored and runs against. As expected, there is a correlation between a team's record in one-run games (games decided by a single run separating the winner and loser) and the offset between a team's predicted win using the Pythagorean Expectation and their actual record as shown in Figure 4.

5 Summary

This paper shows how to invert the binomial distribution to determine the density function for the true probability given a discrete outcome. This is most useful for small sample number, particularly for low probability (probability close to zero) or high probability (probability close to one) that are not represented well by a simple discrete binomial calculation around the success rate of the small sample size. We also evaluated the Pythagorean Expectation for baseball to show that it provides more information to predict a team's true winning record than a simple binomial sampling around a team's actual record would indicate.

References

[1] http://www.retrosheet.org.

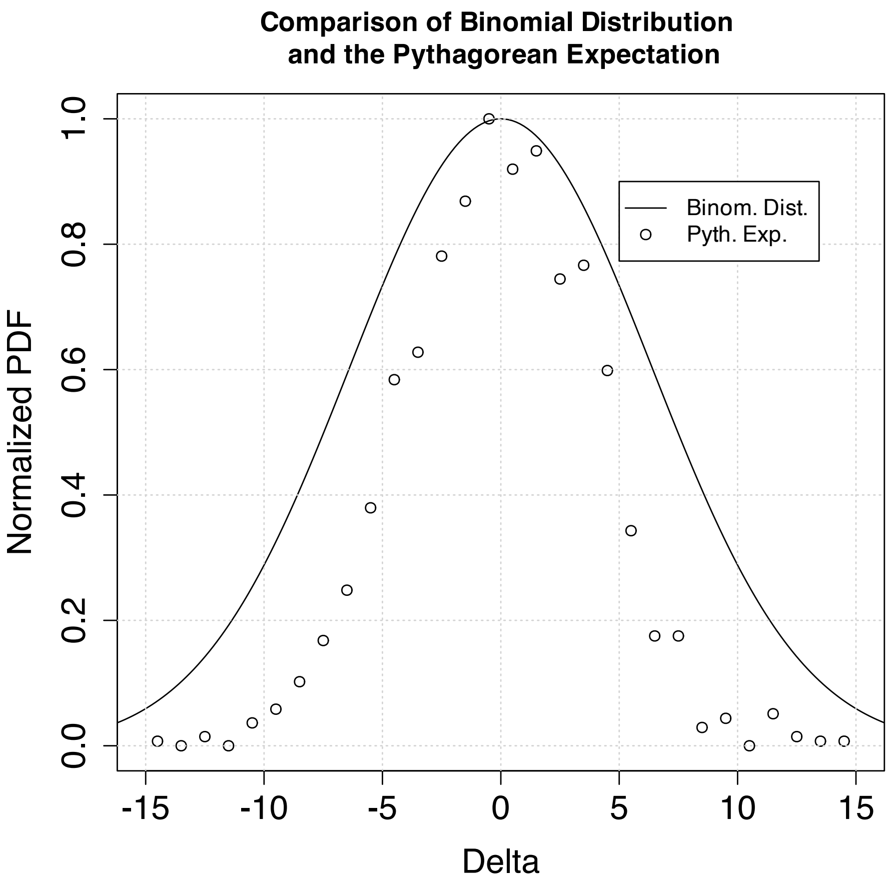

Figure 3: Comparison of the binomial distribution around a team's final record (solid line) versus a normalized histogram predicted by the Pythagorean Expectation (hollow circles) using Retrosheet data of all 1,329 of the 162-game seasons between 1961 and 2023. Note that Pythagorean Expectation has a narrowed variance around the true outcome.

[2] Campbell Gibson. Actual pennant winners versus pythagorean pennant winners, 1901-2020. *SABR Baseball Research Journal*, 50(1):69–74, Spring 2021.

[3] Bill James. *1980 Baseball Abstract.* self-published, Lawrence, KS, 1980.

[4] Stanley Rothman. A new formula to predict a team's winning percentage. *SABR Baseball Research Journal*, 43(2):97–105, Fall 2014.

A Ratios of Binomial Distributions

The direct calculation of a binomial probability can get to be numerically unstable when dealing with large values of n and k, so most programming languages have more efficient and stable ways of computing this. However, most of them are not vectorized to compute them for multiple values of the probability p. Thankfully, you can compute the ratios easily relative to a baseline. Starting with the basic binomial distribution

$$Q(p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{3}$$

you can compute the desired result

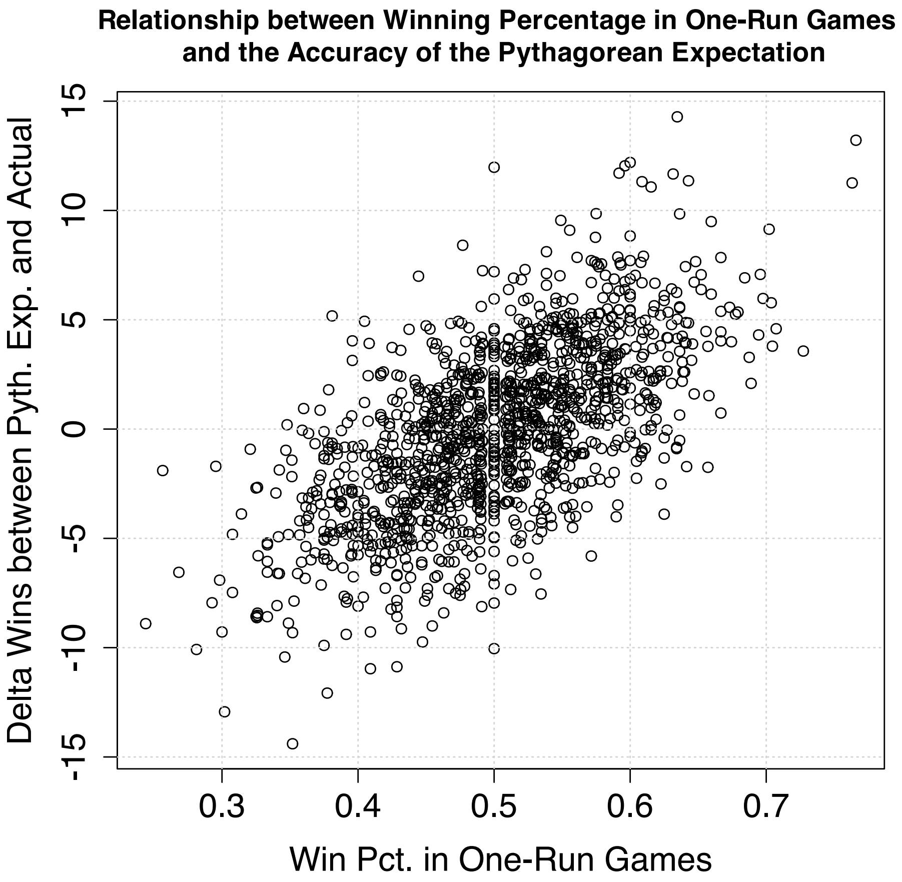

Figure 4: Relationship between winning percentage in one-run games and the difference between the Pythagorean Expectation and teams' actual records

$$Q(p+\delta) = \binom{n}{k}(p+\delta)^k\left[1-(p+\delta)\right]^{n-k} \tag{4}$$

using the ratio of $Q(p+\delta)$ to $Q(p)$

$$\frac{Q(p+\delta)}{Q(p)} = \left(\frac{p+\delta}{p}\right)^k\left(\frac{1-p-\delta}{1-p}\right)^{n-k} = \left(1+\frac{\delta}{p}\right)^k\left(1-\frac{\delta}{1-p}\right)^{n-k} \tag{5}$$

This formula gets interesting when $p = \frac{k}{n}$ (the best *a priori* guess for the center of the distribution), so we can use the basic substitutions $k = pn$ and $n-k = (1-p)n$ to yield

$$\frac{Q(p+\delta)}{Q(p)} = \left[\left(1+\frac{\delta}{p}\right)^p\left(1-\frac{\delta}{1-p}\right)^{1-p}\right]^n \tag{6}$$

The value within the square brackets is a curved function that peaks at 1 when $\delta = 0$. The overall function gets progressively narrower for larger values of $n$as expected. An example of this is shown in Figure XX for values $k = 70$ and $n = 162$ such that $p \approx 0.432$. The two curves show the value of the function within brackets as well as the overall value when $n = 162$.

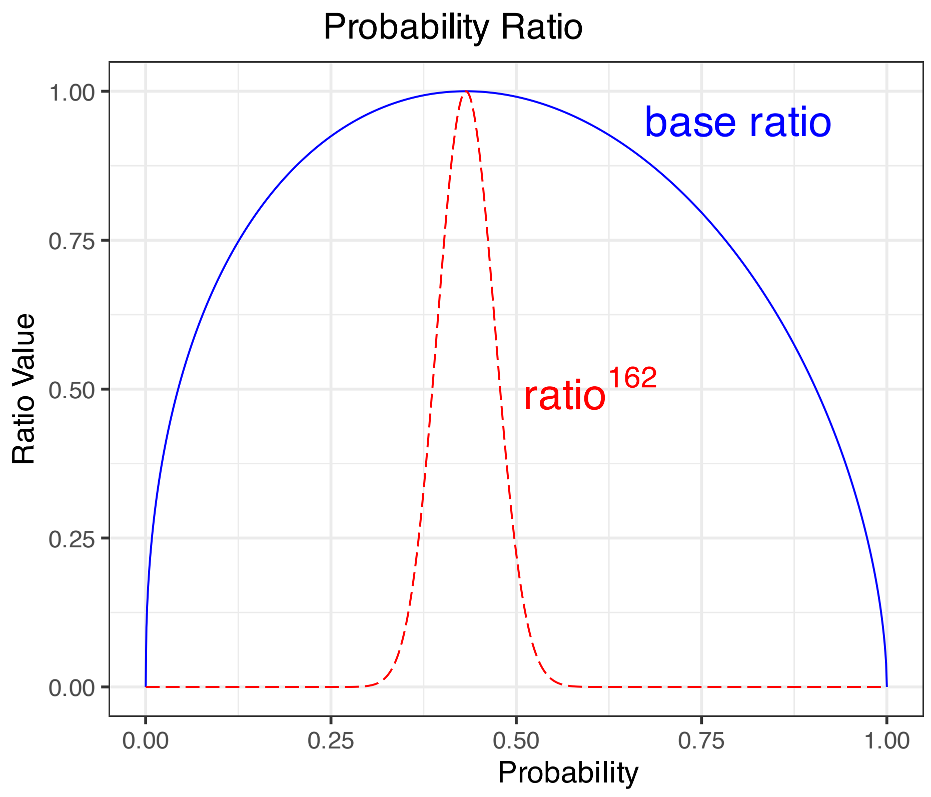

Figure 5: Sample case showing the base and full sample probability ratio when winning 70 games out of 160

B R Code for the Inverse Binomial

The following R language code returns the PDF (probability density function) and CDF (cumulative density function) for $n$successes out of N trials over the space specified by the variable *prob*. Note that the PDF values are correct for the given values of *prob*, but since they represent values on a curve they should be normalized such that the sum of all the PDF values is one when using them to compute averages.

```
# compute the inverse binomial for N-choose-n centered around probabilities in
# the vector probs.  This returns a tibble whose columns are the values of prob,
# the pdf for each value in prob, and the cdf for each value in prob
require(tidyverse)
require(stats)
require(pracma)

invBinom <- function(N,n,prob=linspace(0,1,1001)) {
  # pick a suitable baseline probability to use for scaling
  # (it shouldn't be too close to the either zero or one)
  p0 <- min(c(max(c(0.1,n/N)),.9))
  # compute the baseline binomial probability at p0
  pdf0 <- dbinom(n,N,p0)
  # determine the offsets from p0
  eps <- prob-p0
  # implement (1+eps/p)^n * (1-eps/(1-p))^(N-n) in a more numerically stable way
  ratio <- (1+eps/p0)^(n/N)*(1-eps/(1-p0))^(1-n/N)
  pdf <- pdf0*ratio^N
  # approximate the cdf from the lower and higher edges of each interval
  cdf_low <- cumsum(pdf*diff(c(prob,1))*(N+1))
  cdf_hi <- cumsum(pdf*diff(c(0,prob))*(N+1))
  # return a tibble containing the pdf and cdf for each value in prob
  tibble(prob=prob,pdf=pdf,cdf=pmin(1,(cdf_hi+cdf_low)/2))
}
```